

 免費電子書

學習

# Python Language

Free unaffiliated eBook created from  
**Stack Overflow contributors.**

#python

.....	1
<b>1: Python</b> .....	<b>2</b>
.....	2
.....	2
Python 3.x.....	2
Python 2.x.....	3
Examples.....	3
.....	3
Python.....	3
<b>IDLEPython</b> .....	<b>4</b>
Hello World Python.....	4
<b>Python shell</b> .....	<b>5</b>
.....	6
.....	6
.....	6
.....	6
.....	10
IDLE - Python GUI.....	11
.....	11
.....	12
.....	12
.....	12
.....	12
.....	13
.....	13
.....	14
.....	14
.....	14
.....	15
.....	15
.....	

.....	19
.....	<b>20</b>
.....	20
.....	24
.....	25
- strrepr.....	26
.....	26
STR.....	26
pip.....	27
/.....	<b>27</b>
.....	<b>27</b>
.....	<b>27</b>
Python 2.7.x3.x.....	28
<b>2: * args** kwargs.....</b>	<b>31</b>
.....	31
<b>h11.....</b>	<b>31</b>
<b>h12.....</b>	<b>31</b>
<b>h13.....</b>	<b>31</b>
Examples.....	32
* args.....	32
** kwargs.....	32
* args.....	32
** kwargs.....	33
* args.....	33
.....	34
kwarg.....	34
** kwargs.....	34
<b>3: __name__.....</b>	<b>36</b>
.....	36
.....	36

Examples.....	36
__name__ == '__main__':.....	36
1.....	36
2.....	36
function_class_or_module . __ name__.....	36
.....	37
<b>4: 2to3.....</b>	<b>39</b>
.....	39
.....	39
.....	39
Examples.....	39
.....	39
Unix.....	40
.....	40
Unix.....	40
.....	40
<b>5: ArcPy.....</b>	<b>41</b>
.....	41
Examples.....	41
“”.....	41
createDissolvedGDBgdb.....	41
<b>6: Asyncio.....</b>	<b>42</b>
Examples.....	42
.....	42
.....	43
UVLoop.....	44
.....	44
.....	<b>44</b>
.....	<b>44</b>
Websocket.....	45
asyncio.....	45

<b>7: base64</b>	<b>47</b>
.....	47
.....	47
.....	47
.....	48
Examples	48
Base64	48
Base32	49
Base16	50
ASCII85	50
Base85	51
<b>8: ChemPy - python</b>	<b>52</b>
.....	52
Examples	52
.....	52
.....	52
.....	52
.....	53
.....	53
<b>9: configparser</b>	<b>55</b>
.....	55
.....	55
.....	55
Examples	55
.....	55
.....	55
<b>10: ctypes</b>	<b>57</b>
.....	57
Examples	57
.....	57
.....	57

.....	57
.....	58
ctypes.....	58
ctypes.....	58
ctypes.....	59
.....	59
<b>11: Deque.....</b>	<b>61</b>
.....	61
.....	61
.....	61
Examples.....	61
.....	61
deque.....	61
deque.....	61
.....	62
<b>12: dis.....</b>	<b>64</b>
Examples.....	64
dis.....	64
Python.....	64
.....	64
<b>13: Django.....</b>	<b>66</b>
.....	66
Examples.....	66
DjangoHello World.....	66
<b>14: Functools.....</b>	<b>68</b>
Examples.....	68
.....	68
total_ordering.....	68
.....	68
lru_cache.....	69
cmp_to_key.....	69

<b>15: hashlib</b>	<b>70</b>
.....	70
Examples	70
MD5	70
OpenSSL	70
<b>16: Heapq</b>	<b>72</b>
Examples	72
.....	72
.....	72
<b>17: HTML</b>	<b>74</b>
Examples	74
BeautifulSoup	74
BeautifulSoupCSS	74
PyQuery	75
<b>18: ijson</b>	<b>76</b>
.....	76
Examples	76
.....	76
<b>19: IterablesIterators</b>	<b>77</b>
Examples	77
Iterator vs Iterable vs Generator	77
.....	78
.....	78
iterable	78
.....	79
.....	79
<b>20: Itertools</b>	<b>80</b>
.....	80
Examples	80
.....	80
.....	80
itertools.product	81

itertools.count .....	82
itertools.takewhile .....	82
itertools.dropwhile .....	83
.....	84
itertools .....	84
.....	84
itertools.repeat .....	85
.....	85
.....	85
itertools.permutations .....	85
<b>21: JSON .....</b>	<b>87</b>
.....	87
.....	<b>87</b>
.....	87
.....	87
.....	87
.....	87
.....	87
.....	87
.....	88
Examples .....	88
Python dictJSON .....	88
JSONPython dict .....	88
.....	89
.....	89
`load` vs `load` `dump` vs `stacks` .....	89
`json.tool` JSON .....	90
JSON .....	90
.....	<b>90</b>
.....	<b>91</b>
.....	<b>91</b>
JSON .....	91





.....	100
makedirs - .....	100
<b>26: Pandas Transform</b> .....	<b>102</b>
Examples.....	102
.....	102
.....	102
pandas transform.....	102
.....	103
<b>transform</b> .....	<b>103</b>
<b>27: Pickle</b> .....	<b>104</b>
.....	104
.....	104
.....	104
<b>Pickleable</b> .....	<b>104</b>
<b>pickle</b> .....	<b>104</b>
Examples.....	104
Pickle.....	104
.....	104
.....	105
<b>picklebyte</b> .....	<b>105</b>
.....	105
<b>28: pipPyPI</b> .....	<b>107</b>
.....	107
.....	107
.....	107
Examples.....	107
.....	107
.....	107
.....	108
`pip`.....	108
.....	108

Linux.....	108
Windows.....	108
requirements.txt.....	109
virtualenvrequirements.txt.....	109
Pythonpip.....	109
.....	110
.....	110
.....	110
<b>29: PostgreSQL.....</b>	<b>112</b>
Examples.....	112
.....	112
pip.....	112
.....	112
<b>30: py.test.....</b>	<b>113</b>
Examples.....	113
py.test.....	113
.....	113
.....	113
.....	113
.....	113
.....	113
.....	114
py.test.....	115
.....	116
<b>31: pyaudio.....</b>	<b>117</b>
.....	117
.....	117
Examples.....	117
I/O.....	117
I/O.....	118
<b>32: pyautogui.....</b>	<b>120</b>
.....	120

Examples.....	120
.....	120
.....	120
ScreenShot.....	120
<b>33: pygame</b> .....	<b>121</b>
.....	121
.....	121
.....	121
Examples.....	121
pygame.....	121
Pygame.....	121
.....	121
.....	122
.....	122
<b>34: Pyglet</b> .....	<b>123</b>
.....	123
Examples.....	123
PygletHello World.....	123
Pyglet.....	123
Pyglet.....	123
PygletOpenGL.....	123
PygletOpenGL.....	124
<b>35: PyInstaller - Python</b> .....	<b>125</b>
.....	125
.....	125
Examples.....	125
.....	125
Pyinstaller.....	125
.....	126
.....	126
.....	126
.....	126

<b>36: Python HTTP Server</b> .....	<b>127</b>
Examples.....	127
HTTP.....	127
.....	127
SimpleHTTPServerAPI.....	129
BaseHTTPRequestHandlerGETPOSTPUT.....	130
<b>37: Python Lex-Yacc</b> .....	<b>132</b>
.....	132
.....	132
Examples.....	132
PLY.....	132
“HelloWorld” PLY - .....	132
1Lex.....	134
.....	<b>135</b>
h21.....	135
h22.....	135
h23.....	136
h24.....	136
h25.....	136
h26.....	136
h27.....	136
h28.....	137
h29.....	137
h210.....	137
2Yacc.....	137
.....	<b>138</b>
h211.....	139
<b>38: Python</b> .....	<b>140</b>
.....	140
Examples.....	140
.....	.....

.....140

.....141

**39: PythonMutable vs ImmutableHashable.....143**

    Examples.....143

        .....143

**Immutables.....143**

    .....143

**Mutables.....144**

    .....144

        .....145

        .....145

**40: Python.....146**

    .....146

    Examples.....146

        Lambda.....146

        .....146

        .....146

        .....146

**41: PythonShell.....147**

    .....147

    Examples.....147

        ssh.....147

**42: python.....148**

    Examples.....148

        .....148

        .....148

**43: Pythonpyserial.....149**

    .....149

    .....149

    .....149

Examples.....	149
.....	149
.....	149
.....	149
<b>44: Python.....</b>	<b>151</b>
Examples.....	151
.....	151
.....	151
.....	152
<b>45: PythonExcel.....</b>	<b>153</b>
Examples.....	153
Excel.....	153
OpenPyXL.....	153
xlsxwriterexcel.....	154
xlrdexcel.....	156
xlsxwriterExcel.....	157
<b>46: Python.....</b>	<b>159</b>
.....	159
.....	159
Examples.....	159
Python.....	159
.....	160
<b>47: Python.....</b>	<b>161</b>
.....	161
Examples.....	161
.....	161
.....	161
.....	161
.....	161
.....	161
.....	161
<b>48: Python.....</b>	<b>163</b>

.....	163
Examples.....	163
SSE.....	163
Asyncio SSE.....	163
<b>49: PythonCurses.....</b>	<b>164</b>
.....	164
Examples.....	164
.....	164
wrapper.....	164
<b>50: Python.....</b>	<b>166</b>
Examples.....	166
.....	166
.....	166
Deque.....	167
.....	167
.....	168
<b>51: Python.....</b>	<b>169</b>
.....	169
Examples.....	169
Python - .....	169
Http.....	169
TCP.....	170
UDP.....	170
Simple HttpServer.....	171
<b>52: Python - virtualenv.....</b>	<b>172</b>
.....	172
Examples.....	172
.....	172
.....	172
Virtualenv.....	172
virtualenv.....	172
<b>53: Python.....</b>	<b>174</b>



.....	174
Examples.....	174
.....	174
.....	175
.....	175
.....	175
.....	176
.....	177
<b>54: setup.py.....</b>	<b>178</b>
.....	178
.....	178
Examples.....	178
setup.py.....	178
python.....	178
setup.py.....	179
.....	179
<b>55: Sqlite3.....</b>	<b>181</b>
Examples.....	181
Sqlite3 - .....	181
.....	181
<b>56: SYS.....</b>	<b>183</b>
.....	183
.....	183
.....	183
Examples.....	183
.....	183
.....	183
.....	183
.....	183
<b>57: tempfile NamedTemporaryFile.....</b>	<b>185</b>
.....	185
Examples.....	185

.....	185
<b>58: Tkinter</b> .....	<b>186</b>
.....	186
.....	186
Examples .....	186
tkinter .....	186
.....	187
.....	<b>187</b>
.....	<b>187</b>
.....	<b>187</b>
<b>59: Unicode</b> .....	<b>189</b>
.....	189
.....	189
Examples .....	189
.....	189
<b>Unicode</b> .....	<b>189</b>
<b>unicode</b> .....	<b>190</b>
/ .....	190
.....	<b>190</b>
.....	<b>191</b>
.....	<b>191</b>
I/O .....	191
<b>60: WebWSGI</b> .....	<b>192</b>
.....	192
Examples .....	192
.....	192
<b>61: Web</b> .....	<b>193</b>
.....	193
.....	193
.....	193
.....	193

Examples.....	194
URL.....	194
URL.....	195
<b>62: "with" Statement.....</b>	<b>196</b>
.....	196
.....	196
.....	196
Examples.....	196
with.....	196
.....	197
.....	197
.....	198
.....	198
.....	199
<b>63: Python 2Python 3.....</b>	<b>200</b>
.....	200
.....	200
Examples.....	200
.....	200
Unicode.....	201
.....	203
Reduce.....	205
xrange.....	205
.....	206
Iterables.....	207
.....	208
.next.....	210
.....	211
.....	211
.....	212
execPython 3.....	212
Python 2hasattr.....	213
.....	

.....	214
.....	214
Python 3 <sup>'''</sup> .....	214
<>`=repr.....	215
/.....	215
Python 3cmp.....	216
.....	217
.....	217
filtermapzip.....	218
/.....	219
.....	220
I / O.....	220
round.....	221
.....	221
round.....	221
.....	222
.....	222
.....	223
.....	223
<b>64: intfloatstrtuplefrozensets.....</b>	<b>225</b>
Examples.....	225
.....	225
.....	225
Frozenset.....	225
<b>65: .....</b>	<b>226</b>
.....	226
Examples.....	226
.....	226
ParentChildren.....	226
C.....	227
PyPar.....	227

<b>66:</b>	<b>228</b>
.....	228
Examples.....	228
.....	228
.....	228
.....	228
<b>67:</b>	<b>230</b>
.....	230
Examples.....	230
.....	230
<b>68: "pip"PyPI</b>	<b>231</b>
.....	231
.....	231
Examples.....	231
.....	231
ImportError.....	232
.....	232
<b>69: AMQPStormRabbitMQ</b>	<b>233</b>
.....	233
Examples.....	233
RabbitMQ.....	233
RabbitMQ.....	234
RabbitMQ.....	234
<b>70: Pandas</b>	<b>237</b>
.....	237
Examples.....	237
Pandas.....	237
<b>71: PythonWindows</b>	<b>239</b>
.....	239
Examples.....	239
Python.....	239

Flask Web.....	239
<b>72: PythonRaspberry PI.....</b>	<b>241</b>
Examples.....	241
- .....	241
<b>73: PythonWeb.....</b>	<b>244</b>
.....	244
.....	244
<b>WebPython.....</b>	<b>244</b>
.....	244
requests.....	244
requests-cache.....	244
scrapy.....	244
selenium.....	244
<b>HTML.....</b>	<b>244</b>
BeautifulSoup.....	244
lxml.....	244
<b>Examples.....</b>	<b>244</b>
lxml.....	244
Web.....	245
Scrapy.....	245
Scrapy.....	246
BeautifulSoup4.....	246
Selenium WebDriver.....	246
urllib.requestWeb.....	247
.....	247
<b>74: Python.....</b>	<b>248</b>
Examples.....	248
Matplotlib.....	248
Seaborn.....	249
Mayavi.....	252
Plotly.....	253
<b>75: ZIP.....</b>	<b>255</b>
.....	

.....255

Examples.....255

    Zip.....255

    Zipfile.....255

    zip.....256

    .....256

**76: .....257**

Examples.....257

.....257

**77: .....258**

.....258

.....258

Examples.....258

.....258

.....258

.....259

.....259

.....259

.....260

.....262

.....262

.....263

.....263

.....263

.....**263**

.....**264**

.....264

**78: .....266**

.....266

.....266

.....266

Examples.....	266
.....	266
.....	266
.....	267
.....	267
.....	268
.....	269
.....	269
.....	<b>269</b>
.....	<b>269</b>
.....	<b>269</b>
.....	<b>269</b>
.....	<b>270</b>
.....	<b>270</b>
<b>79:</b> .....	<b>271</b>
.....	271
.....	271
Examples.....	271
.....	271
.....	272
.....	272
.....	<b>272</b>
<b>Python 23six</b> .....	<b>272</b>
.....	273
.....	273
.....	273
.....	273
.....	274
.....	274
<b>80:</b> .....	<b>276</b>
.....	276



Examples.....	276
PyTesseract.....	276
PyOCR.....	276
<b>81: CLI.....</b>	<b>278</b>
.....	278
.....	278
Examples.....	278
.....	278
argparse.....	278
argparse.....	279
<b>82: .....</b>	<b>281</b>
.....	281
Examples.....	281
math.sqrtmath.sqrt.....	281
**pow.....	281
math.pow.....	282
math.expcmath.exp.....	282
1math.expm1.....	283
cmath.....	283
3pow.....	285
nth-root.....	285
.....	286
<b>83: .....</b>	<b>287</b>
Examples.....	287
py2app.....	287
cx_Freeze.....	288
<b>84: .....</b>	<b>289</b>
.....	289
.....	289
Examples.....	289
“”.....	289
.....	289

.....	289
.....	290
<b>85:</b> .....	<b>291</b>
.....	291
.....	291
.....	291
Examples .....	291
.....	291
.....	<b>292</b>
.....	<b>292</b>
.....	<b>292</b>
.....	<b>293</b>
.....	293
.....	294
.....	296
.....	296
.....	297
.....	298
.....	298
.....	299
<b>86:</b> .....	<b>300</b>
.....	300
.....	300
.....	300
Examples .....	300
.....	300
.....	301
.....	302
- .....	303
.....	303
.....	304

<b>87:</b>	<b>305</b>
Examples	305
.....	305
.....	305
.....	305
.....	305
.....	305
.....	305
.....	306
.....	306
.....	306
.....	307
<b>88:</b>	<b>309</b>
Examples	309
IPython%% timeit	309
timeit	309
timeit	309
line_profiler	309
cProfileProfiler	310
<b>89: Python</b>	<b>311</b>
.....	311
Examples	311
.....	311
PyPI	311
.pypirc	311
testpypi	312
.....	312
PyPI	313
.....	313
.....	313
.....	313
.....	313

90: .....	315
.....	315
.....	315
.....	315
.....	315
.....	316
Examples .....	316
.....	316
.....	317
.....	318
.....	318
.....	<b>318</b>
.....	318
.....	319
.....	<b>319</b>
.....	<b>319</b>
.....	<b>320</b>
.....	320
.....	320
.....	320
.....	321
.....	321
.....	321
Lambda/ .....	321
.....	323
.....	324
.....	325
.....	325
.....	326
.....	326
.....	328
Lambda .....	328

.....	328
<b>91:</b> .....	<b>330</b>
Examples .....	330
.....	330
.....	330
Python .....	330
PYTHONSTARTUP .....	331
.....	331
.....	332
<b>92:</b> .....	<b>334</b>
.....	334
Examples .....	334
.....	334
.....	335
.....	335
.....	336
.....	336
del .....	337
<b>del v</b> .....	<b>337</b>
<b>del v.name</b> .....	<b>337</b>
<b>del v[item]</b> .....	<b>337</b>
<b>del v[a:b]</b> .....	<b>337</b>
.....	338
.....	338
.....	338
.....	339
<b>global vs nonlocal Python 3</b> .....	<b>339</b>
<b>93:</b> .....	<b>341</b>
.....	341
.....	341
.....	341

Examples.....	341
.....	341
.....	342
.....	347
.....	347
.....	348
.....	348
.....	348
.....	348
.....	349
.....	350
.....	350
.....	351
.....	352
<b>94:</b> .....	<b>353</b>
.....	353
Examples.....	353
.....	353
unittest.mock.create_autospec.....	353
test.TestCase.....	354
.....	355
Unittests.....	356
pytest.....	357
<b>95: GIL</b> .....	<b>360</b>
.....	360
<b>GIL</b> .....	<b>360</b>
<b>GIL</b> .....	<b>360</b>
<b>GIL</b> .....	<b>360</b>
<b>GIL</b> .....	<b>360</b>
.....	360
Examples.....	361
Multiprocessing.Pool.....	361

<b>David BeazleyGIL</b> .....	<b>361</b>
Cython nogil.....	362
<b>David BeazleyGIL</b> .....	<b>362</b>
<b>nogilCYTHON</b> .....	<b>362</b>
<b>96:</b> .....	<b>363</b>
.....	363
Examples.....	363
PyDotPlus.....	363
.....	363
PyGraphviz.....	363
<b>97: windowvirtualenvwrapper</b> .....	<b>365</b>
Examples.....	365
windowvirtualenvwrapper.....	365
<b>98:</b> .....	<b>366</b>
.....	366
Examples.....	366
.....	366
<b>99:</b> .....	<b>367</b>
.....	367
.....	367
.....	367
Examples.....	367
mapitertools.imapfuture_builtins.map.....	367
.....	368
.....	368
Map“None”python 2.x.....	370
.....	370
<b>100:</b> .....	<b>373</b>
.....	373
.....	373
Examples.....	373
.....	

.....	374
del.....	375
.....	375
.....	376
.....	376
.....	376
.....	377
<b>101:</b> .....	<b>378</b>
Examples.....	378
inputraw_input.....	378
.....	378
.....	378
.....	379
stdin.....	380
.....	380
<b>102:</b> .....	<b>382</b>
.....	382
.....	382
.....	382
Examples.....	382
ListStack.....	382
.....	383
<b>103:</b> .....	<b>384</b>
Examples.....	384
.....	384
.....	384
<b>104:</b> .....	<b>386</b>
Examples.....	386
.....	386
.....	388
<b>105:</b> .....	<b>389</b>



Examples.....	389
.....	389
.....	389
<b>106:</b> .....	<b>391</b>
.....	391
Examples.....	391
.....	391
.....	392
.....	392
.....	393
.....	393
while.....	395
<b>107:</b> .....	<b>396</b>
.....	396
.....	396
Examples.....	396
UDP.....	396
UDP.....	396
TCP.....	397
TCP.....	397
Linux.....	398
<b>108:</b> .....	<b>400</b>
.....	400
.....	400
Examples.....	400
.....	400
Popen.....	400
.....	<b>400</b>
.....	<b>400</b>
.....	<b>400</b>
.....	<b>401</b>

.....	401
.....	401
.....	401
<b>109:</b> .....	<b>403</b>
.....	403
.....	403
.....	403
Examples .....	403
.....	403
dict .....	403
KeyError .....	403
.....	404
.....	405
.....	<b>405</b>
.....	405
dict .....	405
dict() .....	405
.....	<b>405</b>
.....	406
.....	406
** .....	406
.....	407
<b>Python 3.5+</b> .....	<b>407</b>
<b>Python 3.3+</b> .....	<b>407</b>
<b>Python 2.x3.x</b> .....	<b>407</b>
.....	408
.....	408
.....	408
.....	409
.....	410
<b>110:</b> .....	<b>411</b>
.....	

.....	412
Examples.....	412
.....	412
str.casefold().....	412
str.upper().....	412
str.lower().....	412
str.capitalize().....	412
str.title().....	413
str.swapcase().....	413
str.....	413
.....	413
str.split(sep=None, maxsplit=-1).....	413
str.rsplit(sep=None, maxsplit=-1).....	414
.....	414
str.replace(old, new[, count]).....	414
str.formatf-strings.....	415
.....	416
str.count(sub[, start[, end]]).....	416
.....	416
str.startswith(prefix[, start[, end]]).....	416
str.endswith(prefix[, start[, end]]).....	417
.....	417
str.isalpha.....	417
str.isupper str.islower str.istitle.....	418
str.isdecimal str.isdigit str.isnumeric.....	418
str.isalnum.....	419
str.isspace.....	419
str.translate.....	419
/.....	420
str.strip([chars]).....	420
str.rstrip([chars])str.lstrip([chars]).....	420
.....	421

.....	422
.....	422
string.ascii_letters .....	422
string.ascii_lowercase .....	422
string.ascii_uppercase .....	422
string.digits .....	422
string.hexdigits .....	422
string.octaldigits .....	423
string.punctuation .....	423
string.whitespace .....	423
string.printable .....	423
.....	423
.....	424
strunicode.....	424
.....	425
<b>111:</b> .....	<b>426</b>
.....	426
.....	426
.....	426
Examples.....	426
.....	426
.....	427
f-string.....	428
datetime.....	428
GetitemGetattr.....	429
.....	429
.....	430
.....	430
.....	431
.....	431
.....	432
Python 2.x.....	432
Python 3.2+.....	432

.....	432
<b>112:</b> .....	<b>433</b>
.....	433
.....	433
.....	433
Examples .....	433
.....	433
.....	433
.....	433
.....	434
pycrypto .....	434
pycryptoRSA .....	435
pycryptoRSA .....	436
<b>113: /</b> .....	<b>437</b>
.....	437
.....	437
Examples .....	439
.....	439
.....	440
<b>114:</b> .....	<b>443</b>
Examples .....	443
CHello World .....	443
C Extensions .....	443
Cc ++Boost .....	444
<b>C ++</b> .....	<b>444</b>
<b>115: PythonSQL Server</b> .....	<b>446</b>
Examples .....	446
.....	446
<b>116:</b> .....	<b>447</b>
.....	447
.....	447

Examples.....	447
.....	447
.....	448
.....	448
__all__.....	449
.....	450
.....	450
PEP8.....	450
.....	451
__import__.....	451
.....	451
Python 2.....	451
Python 3.....	452
<b>117:</b> .....	<b>453</b>
.....	453
Examples.....	453
@property.....	453
@property.....	453
gettersetterdeleter.....	454
.....	454
<b>118:</b> .....	<b>457</b>
.....	457
Examples.....	457
.....	457
.....	457
<b>119:</b> .....	<b>459</b>
Examples.....	459
.....	459
.....	459
.....	460
.....	460
.....	460
.....	460
.....	460

<b>120:</b> .....	<b>462</b>
.....	462
Examples.....	462
.....	462
.....	464
.....	465
.....	468
int literals.....	469
.....	469
sys.argv [0].....	470
<b>h14:</b> .....	<b>470</b>
.....	470
GIL.....	471
.....	472
.....	472
Pythonic JSON.....	473
<b>121: CPython</b> .....	<b>474</b>
.....	474
.....	474
Examples.....	475
CPython.....	475
CPython.....	475
<b>122:</b> .....	<b>477</b>
.....	477
Examples.....	477
if / else.....	477
.....	477
.....	478
.....	478
<b>123: CSV</b> .....	<b>480</b>
.....	480

.....	480
.....	480
Examples.....	480
.....	480
CSV.....	481
<b>124:</b> .....	<b>482</b>
.....	482
.....	482
.....	482
Examples.....	482
.....	482
.....	483
.....	<b>483</b>
.....	483
<b>break</b> .....	<b>483</b>
<b>continue</b> .....	<b>484</b>
.....	<b>484</b>
returnbreak.....	485
“else”.....	485
.....	487
.....	487
Loop.....	488
.....	488
.....	489
.....	<b>489</b>
.....	489
“”.....	490
.....	490
<b>125:</b> .....	<b>491</b>
.....	491
Examples.....	491
.....	.....



<b>126:</b>	<b>493</b>
Examples	493
.....	493
.....	493
.....	493
"__main__"	494
<b>127:</b>	<b>495</b>
Examples	495
.....	495
.....	496
<b>128: abc</b>	<b>498</b>
Examples	498
ABCMeta	498
/ABCMeta@abstractmethod	498
<b>129:</b>	<b>500</b>
Examples	500
python	500
<b>130:</b>	<b>502</b>
.....	502
.....	502
Examples	502
AND	502
OR	502
.....	503
.....	503
.....	503
NOT	504
.....	505
<b>131:</b>	<b>507</b>
Examples	507
.....	507

.....	507
maxmin.....	507
.....	508
.....	<b>508</b>
.....	508
.....	509
.....	509
NN.....	511
<b>132:</b> .....	<b>513</b>
Examples.....	513
.....	513
.....	513
<b>133: /</b> .....	<b>515</b>
.....	515
Examples.....	515
.....	515
.....	515
<b>134:</b> .....	<b>516</b>
Examples.....	516
.....	516
.....	517
<b>135:</b> .....	<b>518</b>
.....	518
Examples.....	518
str.indexstr.rindexstr.findstr.rfind.....	518
.....	518
.....	518
.....	518
.....	518
.....	519
.....	519
list.indextuple.index.....	519

dict.....	519
bisect.bisect_left.....	520
.....	520
__content__iter__.....	521
<b>136:</b> .....	<b>523</b>
Examples.....	523
.....	523
Methodcaller.....	523
Itemgetter.....	523
<b>137: XML</b> .....	<b>525</b>
.....	525
Examples.....	525
ElementTree.....	525
XML.....	525
XML.....	526
iterparseXML.....	526
XPathXML.....	527
<b>138:</b> .....	<b>528</b>
.....	528
.....	528
.....	528
.....	528
.....	<b>528</b>
Examples.....	528
.....	528
.....	528
.....	529
.....	529
<b>139:</b> .....	<b>531</b>
.....	531
.....	531
Examples.....	531
collections.Counter.....	531

collections.defaultdict.....	532
collections.OrderedDict.....	533
collections.namedtuple.....	534
collections.deque.....	534
collections.ChainMap.....	536
<b>140:</b> .....	<b>537</b>
Examples.....	537
ceiltrunc.....	537
.....	538
.....	538
.....	538
.....	538
.....	538
.....	538
.....	538
/.....	539
.....	539
.....	540
.....	540
.....	540
NaN <sup>49</sup> .....	540
.....	543
cmath.....	543
<b>141:</b> .....	<b>547</b>
.....	547
.....	547
.....	547
.....	547
Examples.....	547
JSON.....	547
Pickle.....	548
<b>142:</b> .....	<b>549</b>
.....	549
Examples.....	549

MySQLdbMySQL.....	549
SQLite.....	550
<b>SQLite.....</b>	<b>550</b>
.....	550
h212.....	551
Connection.....	551
Cursor.....	552
SQLitePython.....	554
psycopg2PostgreSQL.....	555
.....	555
.....	555
.....	555
Oracle.....	556
.....	557
sqlalchemy.....	558
<b>143: .....</b>	<b>559</b>
Examples.....	559
iterablecollections.Counter.....	559
-scollections.Counter.most_common.....	559
list.counttuple.count.....	559
str.count.....	560
numpy.....	560
<b>144: .....</b>	<b>562</b>
.....	562
.....	562
Examples.....	562
.....	562
.....	563
append.....	563
insert.....	563
extendpython.....	563
fromlist.....	564

remove.....	564
pop.....	564
index.....	564
reversepython.....	564
buffer_info.....	565
count.....	565
tostring.....	565
tolistpython.....	565
fromstringchar.....	565
<b>145: I / O.....</b>	<b>567</b>
.....	567
.....	567
.....	567
.....	567
.....	<b>567</b>
Examples.....	567
.....	567
.....	569
.....	569
.....	570
.....	571
.....	571
.....	572
.....	572
.....	572
mmap.....	572
.....	573
.....	573
<b>146: .....</b>	<b>574</b>
Examples.....	574
.....	574
<b>147: .....</b>	<b>575</b>
.....	

Examples.....	575
.....	575
.....	575
.....	575
.....	576
.....	576
.....	577
.....	579
.....	579
ISO 8601.....	579
.....	580
.....	580
.....	580
ISO 8601.....	581
.....	<b>581</b>
.....	<b>581</b>
.....	<b>581</b>
<b>148:</b> .....	<b>582</b>
Examples.....	582
.....	582
datetime.....	582
datetime.....	582
<b>149:</b> .....	<b>583</b>
Examples.....	583
.....	583
JPEG.....	583
<b>150:</b> .....	<b>584</b>
.....	584
Examples.....	584
Python 2.43.3.....	584
.....	584

<b>151:</b> .....	<b>585</b>
.....	585
.....	585
Examples .....	585
ifelif .....	585
“” .....	585
.....	585
.....	586
.....	586
.....	<b>586</b>
.....	<b>586</b>
.....	<b>587</b>
.....	<b>587</b>
.....	588
cmp .....	588
.....	589
None .....	589
<b>152:</b> .....	<b>591</b>
.....	591
Examples .....	591
.....	591
.....	591
<b>153:</b> .....	<b>593</b>
.....	593
Examples .....	593
os.access .....	593
<b>154:</b> .....	<b>594</b>
.....	594
.....	594
Examples .....	594
.....	594
.....	.....



.....	.596
.....	<b>596</b>
.....	<b>596</b>
.....	.597
.....	.597
.....	<b>597</b>
.....	<b>597</b>
.....	<b>597</b>
.....	.598
.....	.598
.....	.598
.....	.599
.....	.599
.....	.599
.....	.599
.....	.600
.....	.600
<b>155:</b> .....	<b>602</b>
.....	.602
.....	.602
<b>Examples</b> .....	.602
.....	.602
.....	.603
.....	.603
.....	.603
.....	<b>603</b>
.....	<b>604</b>
.....	.604
.....	.605
<b>Python</b> .....	.605

<b>156:</b>	<b>607</b>
.....	607
.....	607
Examples.....	607
.....	607
.....	608
<b>157:</b>	<b>609</b>
.....	609
.....	609
Examples.....	609
.....	609
URL.....	609
HTTP.....	610
.....	610
.....	611
.....	612
.....	<b>612</b>
.....	<b>612</b>
.....	<b>613</b>
<b>158: `exec` `eval`</b>	<b>614</b>
.....	614
.....	614
.....	614
Examples.....	614
exec.....	614
eval.....	614
.....	614
eval.....	615
ast.literal_evalPython.....	615
execevalast.literal_eval.....	615
<b>159: Matplotlib</b>	<b>617</b>

.....	617
Examples.....	617
Matplotlib.....	617
.....	617
MATLAB.....	618
.....	619
XYtwinx.....	621
twinYX.....	622
<b>160:</b> .....	<b>625</b>
Examples.....	625
.....	625
.....	626
<b>161:</b> .....	<b>627</b>
.....	627
.....	627
Examples.....	627
.....	627
next.....	627
.....	627
.....	628
.....	628
.....	630
.....	630
<b>-</b> .....	<b>631</b>
.....	631
.....	632
.....	632
.....	632
.....	633
.....	633
.....	633
<b>162: urllib</b> .....	<b>635</b>
Examples.....	635

HTTP GET .....	635
Python 2 .....	635
Python 3 .....	635
HTTP POST .....	635
Python 2 .....	636
Python 3 .....	636
.....	636
<b>163: WebSockets .....</b>	<b>637</b>
Examples .....	637
aiohttp .....	637
aiohttp .....	637
Websocket .....	638
<b>164: .....</b>	<b>640</b>
.....	640
.....	640
.....	<b>640</b>
Examples .....	640
.....	640
.....	641
.....	641
.....	641
.....	641
.....	643
.....	643
.....	644
.....	644
.....	645
.....	645
<b>165: .....</b>	<b>647</b>
.....	647
.....	647
.....	647
Examples .....	647

.....	647
.....	648
.....	648
__getitem__ __setitem__ __delitem__ .....	648
.....	649
.....	650
.....	650
<b>166:</b> .....	<b>651</b>
.....	651
.....	651
Examples .....	651
.....	651
.....	651
.....	652
MethodsBuiltins .....	653
.....	<b>653</b>
.....	<b>653</b>
.....	<b>653</b>
.....	<b>653</b>
.....	<b>654</b>
.....	<b>654</b>
.....	<b>654</b>
.....	<b>654</b>
.....	655
<b>167:</b> .....	<b>656</b>
Examples .....	656
.....	656
<b>168:</b> .....	<b>657</b>
Examples .....	657
.....	657
.....	657
.....	.....

.....	658
<b>169:</b> .....	<b>659</b>
.....	659
Examples.....	659
IndentationErrors.....	659
<b>IndentationError / SyntaxError</b> .....	<b>659</b>
.....	659
<b>IndentationError / SyntaxErrorunindent</b> .....	<b>659</b>
.....	659
<b>IndentationError</b> .....	<b>660</b>
.....	660
<b>IndentationError</b> .....	<b>660</b>
.....	660
.....	660
TypeErrors.....	660
<b>TypeError[]</b> .....	<b>660</b>
.....	660
<b>TypeError[]'???'???'</b> .....	<b>661</b>
.....	661
<b>TypeError'???' object/</b> .....	<b>661</b>
.....	661
<b>TypeError'???'</b> .....	<b>661</b>
.....	662
NameErrorname'???'.....	662
.....	662
.....	662
import .....	662
PythonLEGB.....	662
.....	663
<b>AssertError</b> .....	<b>663</b>

<b>KeyboardInterrupt</b> .....	<b>663</b>
<b>ZeroDivisionError</b> .....	<b>663</b>
.....	664
<b>170:</b> .....	<b>665</b>
.....	665
.....	665
Examples.....	665
.....	665
<b>virtualenv</b> .....	<b>665</b>
.....	665
.....	665
.....	666
.....	666
.....	666
.....	666
.....	666
.....	666
python.....	667
virtualenvwrapper.....	668
.....	668
.....	668
.....	668
.....	669
Unix / Linuxpython.....	669
virtualenv.....	669
Anaconda.....	670
.....	670
.....	670
.....	670
.....	670
.....	670
<b>171: virtualenvwrapper</b> .....	<b>672</b>
.....	

672

Examples.....672  
    virtualenvwrapper.....672

**172: .....674**

.....674  
.....674  
.....674

Examples.....674  
    .....674  
    .....675

**.....675**

**.....676**

.....676

**.....676**

**.....677**

.....677

**.....677**

**.....677**

**.....678**

.....678

.....679

**173: .....680**

Examples.....680  
    .....680  
    .....680  
    .....680  
    .....680  
    .....680

**174: .....682**

.....682

Examples.....682  
    .....



.....	682
<b>175:</b> .....	<b>684</b>
.....	684
Examples.....	684
Python ZipFile.extractallZIP.....	684
Python TarFile.extractalltarball.....	684
<b>176:</b> .....	<b>685</b>
.....	685
Examples.....	685
argparseHello.....	685
docopt.....	685
argparse.....	686
argv.....	686
argparse.....	687
argparse.add_argument_group.....	688
docoptdocopt_dispatch.....	689
<b>177:</b> .....	<b>690</b>
Examples.....	690
Python.....	690
.....	691
<b>178: Python</b> .....	<b>694</b>
Examples.....	694
.....	694
.....	694
.....	694
.....	694
.....	695
.....	695
<b>179:</b> .....	<b>696</b>
.....	696
Examples.....	696

.....	696
Singleton.....	697
.....	698
<b>180:</b> .....	<b>701</b>
.....	701
.....	701
Examples.....	701
.....	701
.....	701
.....	701
.....	702
.....	702
docstrings.....	702
.....	703
PEP 257.....	703
.....	703
Google Python.....	704
<b>181: PythonJavaScript.....</b>	<b>705</b>
.....	705
Examples.....	705
`in`.....	705
<b>182:</b> .....	<b>706</b>
.....	706
Examples.....	706
.....	706
<b>183:</b> .....	<b>707</b>
Examples.....	707
Python_pdb_.....	707
IPythonpdb.....	708
.....	709
<b>184: CSV.....</b>	<b>710</b>
Examples.....	710

TSV .....	710
.....	710
.....	710
.....	710
<b>185:</b> .....	<b>711</b>
Examples .....	711
/ Dunder .....	711
.....	712
.....	713
.....	713
.....	713
<b>186:</b> .....	<b>716</b>
.....	716
.....	716
Examples .....	716
.....	716
.....	716
.....	717
.....	717
NamedTuple .....	718
.....	718
<b>187: ...</b> .....	<b>719</b>
.....	719
.....	719
.....	719
.....	719
Examples .....	719
1 .....	719
2 .....	720
3 .....	720
4 .....	721
<b>188:</b> .....	<b>723</b>

.....	723
.....	723
Examples.....	724
.....	724
.....	724
<b>189:</b> .....	<b>725</b>
.....	725
Examples.....	725
.....	725
.....	726
.....	727
.....	727
.....	728
<b>190:</b> .....	<b>729</b>
.....	729
.....	729
Examples.....	729
python.....	729
<b>191:</b> .....	<b>731</b>
.....	731
.....	731
.....	731
Examples.....	731
.....	731
.....	732
.....	732
filterfalsefilterfalse.....	732
<b>192:</b> .....	<b>734</b>
.....	734
Examples.....	734
1n.....	734
.....	734

.....	736
.....	737
.....	738
.....	738
<b>193:</b> .....	<b>740</b>
.....	740
.....	740
.....	740
.....	740
Examples.....	740
.....	740
<b>194:</b> .....	<b>742</b>
Examples.....	742
Conda.....	742
<b>195:</b> .....	<b>744</b>
.....	744
Examples.....	744
.....	744
<b>196:</b> .....	<b>748</b>
Examples.....	748
python.....	748
<b>197: GZip</b> .....	<b>749</b>
.....	749
Examples.....	749
GNU zip.....	749
<b>198:</b> .....	<b>750</b>
.....	750
.....	750
.....	750
Examples.....	750
.....	750

reduce.....	751
.....	751
/.....	751
truthy / falsy.....	751
<b>199:</b> .....	<b>753</b>
.....	753
Examples.....	753
.....	753
<b>200:</b> .....	<b>754</b>
.....	754
Examples.....	754
.....	754
.....	754
.....	754
.....	754
randinrandrangerandomuniform.....	754
<b>randint</b> .....	<b>755</b>
<b>randrange</b> .....	<b>755</b>
.....	756
.....	756
.....	756
.....	757
.....	757
.....	758
<b>201:</b> .....	<b>759</b>
Examples.....	759
.....	759
<b>202: Python</b> .....	<b>760</b>
Examples.....	760
IronPython.....	760
.....	760

.....	760
Jython.....	760
.....	760
.....	761
Transcript.....	761
.....	761
<b>HTML.....</b>	<b>761</b>
<b>JavaScriptDOM.....</b>	<b>761</b>
<b>JavaScript.....</b>	<b>762</b>
<b>PythonJavaScript.....</b>	<b>762</b>
.....	763
<b>203: .....</b>	<b>765</b>
Examples.....	765
Pyglet.....	765
WAV.....	765
.....	765
.....	765
pythonffmpeg.....	765
Windows.....	766
<b>204: .....</b>	<b>767</b>
.....	767
Examples.....	767
.....	767
.....	767
.....	768
.....	769
.....	771
.....	772
.....	773
.....	774
.....	.....

.....	776
.....	776
.....	777
.....	778
.....	779
.....	781
<b>205: __str__repr__</b> .....	<b>783</b>
.....	783
.....	<b>783</b>
.....	<b>783</b>
Examples.....	783
.....	783
.....	<b>784</b>
<b>1</b> .....	<b>784</b>
<b>2</b> .....	<b>785</b>
.....	<b>786</b>
.....	<b>786</b>
eval-round-trip__repr__.....	787
<b>206:</b> .....	<b>788</b>
Examples.....	788
Ninja TwistTurtle Graphics.....	788
.....	<b>789</b>



---

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [python-language](#)

It is an unofficial and free Python Language ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Python Language.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to [info@zzzprojects.com](mailto:info@zzzprojects.com)

# 1: Python



Python

- Python
- Python PHPWeb Python Web
- Python

```
if something:  
    x = 1  
else:  
    x = 'this is a string'  
print(x)
```

- ;

```
1 + '1' # raises an error  
1 + int('1') # results with 2
```

- :) Python Python

*The Zen of Python*

Python Python 2.x Python 3.x

PythonPython

Python.orgCPython [IronPython](#) .NETPython [Jython](#) JavaPyPy Python

## Python 3.x

[3.7]	201758
3.6	20161223
3.5	2015913
3.4	2014317
3.3	2012929
3.2	2011-02-20

3.1	2009-06-26
3.0	2008-12-03

## Python 2.x

2.7	2010-07-03
2.6	2008-10-02
2.5	2006-09-19
2.4	2004-11-30
2.3	2003-07-29
2.2	2001-12-21
2.1	2001-11-15
2.0	2000-10-16

## Examples

Python Guido van Rossum 1991. Python

Python

- Python 3.x
- Python 2.x 2020. Python 2 Python 3

Python [Python 3](#) [Python 2](#). Python

## Python

Python Windows python

```
$ python --version
```

Python 3.x 3.0

*Python 3*

```
$ python --version
Python 3.6.0
```

## Python 2.x 2.7

### Python 2

```
$ python --version
Python 2.7.13
```

Python 3\$ python --versionPython 2Python 2. MacOSLinux. \$ python3Python 3.

---

# IDLEPython

---

IDLEPythonPython.

## IDLEHelloWorld

- IDLE.
  - WindowsWindowsAll Programs.
  - Windows 8+IDLE.
  - UnixMac\$ idle python\_file.pyshell.
- shell.

### shell

```
>>>
```

```
>>> print("Hello, World")
```

[Enter].

```
>>> print("Hello, World")
Hello, World
```

## Hello World Python

hello.py

### Python 3.x 3.0

```
print('Hello, World')
```

### Python 2.x 2.6

Python 2Python 3 printimport

```
from __future__ import print_function
```

Python 2\_\_future\_\_Python 3 ◦

Python 2.x 2.7

Python 2◦ Python 3◦

```
print 'Hello, World'
```

hello.py◦

python hello.py Enter◦

```
$ python hello.py  
Hello, World
```

Hello, World◦

hello.py◦ Linux“user”python /home/user/hello.py ◦

---

## Python shell

pythonPython shell◦ PythonREPL“”◦

```
$ python  
Python 2.7.12 (default, Jun 28 2016, 08:46:01)  
[GCC 6.1.1 20160602] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print 'Hello, World'  
Hello, World  
>>>
```

Python 3python3 ◦

```
$ python3  
Python 3.6.0 (default, Jan 13 2017, 00:00:00)  
[GCC 6.1.1 20160602] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print('Hello, World')  
Hello, World  
>>>
```

python -i <file.py>◦

```
$ python -i hello.py  
"Hello World"  
>>>
```

Python shell

```
>>> exit()
```

```
>>> quit()
```

CTRL + D shell

Interpreter shell CTRL + C.

[Python shell](#)

---

Python shell

shell

- python
- Python
- 
- 
- <https://www.python.org/shell/> - Python Python shell
- <https://ideone.com/> -
- <https://repl.it/languages/python3> - IDE. Python
- [https://www.tutorialspoint.com/execute\\_python\\_online.php](https://www.tutorialspoint.com/execute_python_online.php) - UNIX shell
- [http://rextester.com//python3\\_online\\_compiler](http://rextester.com//python3_online_compiler) - IDE

---

Pythonshell

```
$ python -c 'print("Hello, World")'  
Hello, World
```

shell

---

- PyPA Python **PIP** • pip install <the package name> • pip install numpy • Windows pip PATH  
python -m pip install <the package name>

Shell - Pythonshell • Shell Python • shell **IDLE** - GUI **IPython** - •

- .pyshell **IDE PyCharm Jupyter** / • ; •

**Python** Python •

**PEP8** Python • •

## Python

```
<variable name> = <value>
```

## Python

```
# Integer
a = 2
print(a)
# Output: 2

# Integer
b = 9223372036854775807
print(b)
# Output: 9223372036854775807

# Floating point
pi = 3.14
print(pi)
# Output: 3.14

# String
c = 'A'
print(c)
# Output: A

# String
name = 'John Doe'
print(name)
# Output: John Doe

# Boolean
q = True
print(q)
# Output: True

# Empty value or null data type
x = None
print(x)
# Output: None
```

◦ ◦

```
0 = x
=> Output: SyntaxError: can't assign to literal
```

## python

```
import keyword
print(keyword.kwlist)
```

1. ◦

```
x = True # valid
```

```
_y = True    # valid

9x = False  # starts with numeral
=> SyntaxError: invalid syntax

$y = False  # starts with symbol
=> SyntaxError: invalid syntax
```

2. ◦

```
has_0_in_it = "Still Valid"
```

3. ◦

```
x = 9
y = X*5
=>NameError: name 'X' is not defined
```

---

## PythonPython

```
a = 2
print(type(a))
# Output: <type 'int'>

b = 9223372036854775807
print(type(b))
# Output: <type 'int'>

pi = 3.14
print(type(pi))
# Output: <type 'float'>

c = 'A'
print(type(c))
# Output: <type 'str'>

name = 'John Doe'
print(type(name))
# Output: <type 'str'>

q = True
print(type(q))
# Output: <type 'bool'>

x = None
print(type(x))
# Output: <type 'NoneType'>
```

python.◦

= = ◦ what = does ◦

```
a_name = an_object # "a_name" is now a name for the reference to the object "an_object"
```



```
pi = 3.14 pi 3.14 ° °
```

---

° =

```
a, b, c = 1, 2, 3
print(a, b, c)
# Output: 1 2 3

a, b, c = 1, 2
=> Traceback (most recent call last):
=> File "name.py", line N, in <module>
=> a, b, c = 1, 2
=> ValueError: need more than 2 values to unpack

a, b = 1, 2, 3
=> Traceback (most recent call last):
=> File "name.py", line N, in <module>
=> a, b = 1, 2, 3
=> ValueError: too many values to unpack
```

° \_

```
a, b, _ = 1, 2, 3
print(a, b)
# Output: 1, 2
```

° ""

```
a, b, _ = 1,2,3,4
=>Traceback (most recent call last):
=>File "name.py", line N, in <module>
=>a, b, _ = 1,2,3,4
=>ValueError: too many values to unpack (expected 3)
```

°

```
a = b = c = 1
print(a, b, c)
# Output: 1 1 1
```

a bc **1**int° a bc**int**°

```
a = b = c = 1 # all three names a, b and c refer to same int object with value 1
print(a, b, c)
# Output: 1 1 1
b = 2 # b now refers to another int object, one with a value of 2
print(a, b, c)
# Output: 1 2 1 # so output is as expected.
```

list dictint string tuple

```
x = y = [7, 8, 9] # x and y refer to the same list object just created, [7, 8, 9]
```

```
x = [13, 8, 9]      # x now refers to a different list object just created, [13, 8, 9]
print(y)           # y still refers to the list it was first assigned
# Output: [7, 8, 9]
```

◦ ◦

```
x = y = [7, 8, 9]  # x and y are two different names for the same list object just created,
[7, 8, 9]
x[0] = 13          # we are updating the value of the list [7, 8, 9] through one of its
names, x in this case
print(y)           # printing the value of the list using its other name
# Output: [13, 8, 9] # hence, naturally the change is reflected
```

python◦ ◦

```
x = [1, 2, [3, 4, 5], 6, 7] # this is nested list
print x[2]
# Output: [3, 4, 5]
print x[2][1]
# Output: 4
```

Python - =◦

```
a = 2
print(a)
# Output: 2

a = "New value"
print(a)
# Output: New value
```

=◦ int2a astring“”◦

input Python 2.xraw\_input Python 2.xinput

Python 2.x 2.3

```
name = raw_input("What is your name? ")
# Out: What is your name? _
```

Python2input() - PythonPython3eval(input())◦ ◦

Python 3.x 3.0

```
name = input("What is your name? ")
# Out: What is your name? _
```

Python 3◦

◦ ◦

```
name = input("What is your name? ")
# Out: What is your name?
```

“Bob”Entername"Bob"

```
name = input("What is your name? ")
# Out: What is your name? Bob
print(name)
# Out: Bob
```

inputstr◦ strstr

```
x = input("Write a number:")
# Out: Write a number: 10
x / 2
# Out: TypeError: unsupported operand type(s) for /: 'str' and 'int'
float(x) / 2
# Out: 5.0
```

`try / except` ◦ `raw_input` ◦ `int` ◦ `unstable` ◦ `ValueError` ◦

## IDLE - Python GUI

IDLEPython◦ IDLEpython◦ WindowsPython◦

IDLE

- 
- Python shell
- 
- Python
- Python.pyIDLE◦

IDLEF5run Python Shell◦ IDLE◦

◦

- Windowspython◦ "'python' is not recognized"PythonPATH◦ "....."◦ "...". PATHPythonScript  
C:\Python27;C:\Python27\Scripts◦ ◦

Pythonpython.exe◦ python27.exepython27Python◦

Python Launcher for Windows◦ py -[xy]python[xy]Python◦ Python 2<sub>py -2</sub> Python 3<sub>py -3</sub>◦

- **Debian / Ubuntu / MacOS**

pythonPATH◦

Debian / Ubuntu / MacOSpython for Python 2.xpython3 for Python 3.x.

which pythonPython◦

- **Arch Linux**

Arch LinuxPythonPython 3Python 3.xpythonpython3 Python 2.xpython2◦

- **Python 3pythonpython3◦ Python 2python2◦**

---

bool TrueFalse◦ and or not◦

```
x or y    # if x is False then y otherwise x
x and y   # if x is False then x otherwise y
not x     # if x is True then False, otherwise True
```

Python 2.xPython 3.xint◦ boolint TrueFalse

```
issubclass(bool, int) # True
isinstance(True, bool) # True
isinstance(False, bool) # True
```

TrueFalse10

```
True + False == 1 # 1 + 0 == 1
True * True == 1 # 1 * 1 == 1
```

- int

```
a = 2
b = 100
c = 123456789
d = 38563846326424324
```

Python◦

Pythonlongint◦◦

- float ;CPython floatC double◦

```
a = 2.0
b = 100.e0
c = 123456789.e1
```

- complex

```
a = 2 + 1j
b = 100 + 10j
```

< <= >=TypeError

## Python 3.x 3.0

- str **unicode** ◦ 'hello'
- bytes ◦ b'hello'

## Python 2.x 2.7

- str ◦ 'hello'
- bytes str
- unicode **unicode** ◦ u'hello'

## Python setdict ◦

- str bytes unicode
- reversed strreversed

```
a = reversed('hello')
```

- tuple n n >= 0 ◦

```
a = (1, 2, 3)
b = ('a', 1, 'python', (1, 2))
b[2] = 'something else' # returns a TypeError
```

```
::
```

- list n n >= 0

```
a = [1, 2, 3]
b = ['a', 1, 'python', (1, 2), [1, 2]]
b[2] = 'something else' # allowed
```

```
;
```

- set ◦ ◦

```
a = {1, 2, 'a'}
```

- dict ; ◦

```
a = {1: 'one',
     2: 'two'}

b = {'a': [1, 2, 3],
     'b': 'a string'}
```

```
__hash__()__eq__()__hash__()
```

- Hashable◦

- True bool
- False bool **false**
- None ◦
- Ellipsis... Python3 +Python2.7 +◦ numpy""◦
- NotImplemented PythonPython◦

```
a = None # No value will be assigned. Any valid datatype can be assigned later
```

## Python 3.x 3.0

None◦ < <= >= > TypeError ◦

## Python 2.x 2.7

None None < -32 True ◦

python type type ◦

```
a = '123'
print(type(a))
# Out: <class 'str'>
b = 123
print(type(b))
# Out: <class 'int'>
```

isinstance◦ ◦

```
i = 7
if isinstance(i, int):
    i += 1
elif isinstance(i, str):
    i = int(i)
    i += 1
```

type() isinstance() [Pythonisinstancetype](#)

NoneType

```
x = None
if x is None:
    print('Not a surprise, I just defined x as None.')
```

◦

'123' str int◦

```
a = '123'
b = int(a)
```

float'123.456'◦

```
a = '123.456'
b = float(a)
c = int(a)    # ValueError: invalid literal for int() with base 10: '123.456'
d = int(b)    # 123
```

```
a = 'hello'
list(a) # ['h', 'e', 'l', 'l', 'o']
set(a)  # {'o', 'e', 'l', 'h'}
tuple(a) # ('h', 'e', 'l', 'l', 'o')
```

◦

- b'foo bar' Python 3bytes Python 2str
- u'foo bar' Python 3str Python 2unicode
- 'foo bar' str
- r'foo bar'

```
normal = 'foo\nbar' # foo
                    # bar
escaped = 'foo\\nbar' # foo\nbar
raw     = r'foo\nbar' # foo\nbar
```

*mutable* ◦

```
def f(m):
    m.append(3) # adds a number to the list. This is a mutation.

x = [1, 2]
f(x)
x == [1, 2] # False now, since an item was added to the list
```

◦

```
def bar():
    x = (1, 2)
    g(x)
    x == (1, 2) # Will always be True, since no function can change the object (1, 2)
```

x x◦ x◦

◦

- int long float complex

- str
- bytes
- tuple
- frozenset
  
- bytearray
- list
- set
- dict

## Python◦ ◦

```
>>> pow(2,3)    #8
```

## python dir().◦ ◦

```
>>> dir(__builtins__)
[
  'ArithmeticError',
  'AssertionError',
  'AttributeError',
  'BaseException',
  'BufferError',
  'BytesWarning',
  'DeprecationWarning',
  'EOFError',
  'Ellipsis',
  'EnvironmentError',
  'Exception',
  'False',
  'FloatingPointError',
  'FutureWarning',
  'GeneratorExit',
  'IOError',
  'ImportError',
  'ImportWarning',
  'IndentationError',
  'IndexError',
  'KeyError',
  'KeyboardInterrupt',
  'LookupError',
  'MemoryError',
  'NameError',
  'None',
  'NotImplemented',
  'NotImplementedError',
  'OSError',
  'OverflowError',
  'PendingDeprecationWarning',
  'ReferenceError',
  'RuntimeError',
  'RuntimeWarning',
  'StandardError',
  'StopIteration',
  'SyntaxError',
  'SyntaxWarning',
  'SystemError',
  'SystemExit',
```



```
'TabError',
'True',
'TypeError',
'UnboundLocalError',
'UnicodeDecodeError',
'UnicodeEncodeError',
'UnicodeError',
'UnicodeTranslateError',
'UnicodeWarning',
'UserWarning',
'ValueError',
'Warning',
'ZeroDivisionError',
'__debug__',
'__doc__',
'__import__',
'__name__',
'__package__',
'abs',
'all',
'any',
'apply',
'basestring',
'bin',
'bool',
'buffer',
'bytearray',
'bytes',
'callable',
'chr',
'classmethod',
'cmp',
'coerce',
'compile',
'complex',
'copyright',
'credits',
'delattr',
'dict',
'dir',
'divmod',
'enumerate',
'eval',
'execfile',
'exit',
'file',
'filter',
'float',
'format',
'frozenset',
'getattr',
'globals',
'hasattr',
'hash',
'help',
'hex',
'id',
'input',
'int',
'intern',
'isinstance',
```

```
'issubclass',
'iter',
'len',
'license',
'list',
'locals',
'long',
'map',
'max',
'memoryview',
'min',
'next',
'object',
'oct',
'open',
'ord',
'pow',
'print',
'property',
'quit',
'range',
'raw_input',
'reduce',
'reload',
'repr',
'reversed',
'round',
'set',
'setattr',
'slice',
'sorted',
'staticmethod',
'str',
'sum',
'super',
'tuple',
'type',
'unichr',
'unicode',
'vars',
'xrange',
'zip'
]
```

help ◦

```
>>> help(max)
Help on built-in function max in module __builtin__:
max(...)
    max(iterable[, key=func]) -> value
    max(a, b, c, ...[, key=func]) -> value
    With a single iterable argument, return its largest item.
    With two or more arguments, return the largest argument.
```

◦ math◦

```
>>> import math
>>> math.sqrt(16) # 4.0
```

◦

```
>>> import math
>>> dir(math)

['__doc__', '__name__', '__package__', 'acos', 'acosh',
'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign',
'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1',
'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma',
'hypot', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10',
'log1p', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt',
'tan', 'tanh', 'trunc']
```

`__doc__`

```
>>> math.__doc__
'This module is always available. It provides access to the\
nmathematical functions defined by the C standard.'
```

◦ `helloWorld.py`

```
"""This is the module docstring."""

def sayHello():
    """This is the function docstring."""
    return 'Hello World'
```

```
>>> import helloWorld
>>> helloWorld.__doc__
'This is the module docstring.'
>>> helloWorld.sayHello.__doc__
'This is the function docstring.'
```

- `dir`

```
>>> class MyClassObject(object):
...     pass
...
>>> dir(MyClassObject)
['__class__', '__delattr__', '__dict__', '__doc__', '__format__', '__getattr__',
'__hash__', '__init__', '__module__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__']
```

`str` ◦ `print`

```
>>> str(123)    # "123"
```

Python ◦ Python ◦ ◦

Python : ◦ Python `if` ◦ ◦

```
def my_function():    # This is a function definition. Note the colon (:)
```

```
a = 2          # This line belongs to the function because it's indented
return a      # This line also belongs to the same function
print(my_function()) # This line is OUTSIDE the function block
```

```
if a > b:      # If block starts here
    print(a)   # This is part of the if block
else:         # else must be at the same level as if
    print(b)   # This line is part of the else block
```

```
if a > b: print(a)
else: print(b)
```

```
if x > y: y = x
    print(y) # IndentationError: unexpected indent

if x > y: while y != z: y -= 1 # SyntaxError: invalid syntax
```

IndentationError ◦ pass

```
def will_be_implemented_later():
    pass
```

---

4◦

[PEP 8](#) Python◦

Python 3.x 3.0

Python 3◦ Inconsistent use of tabs and spaces in indentation◦

Python 2.x 2.7

Python 2;◦ **8**◦ **4**◦

[PEP 8](#)

-tPython 2◦ -tt◦

""◦ \ t'Tab◦

- **8** - ◦ ◦
- Tab4◦

Python[autopep8](#)[pep8](#) -conformant◦ Python [reindent.py](#)

Python◦ intstr◦

listPython◦ JavaScript◦ PythonPython◦

```
int_list = [1, 2, 3]
string_list = ['abc', 'defghi']
```

```
empty_list = []
```

## Python

```
mixed_list = [1, 'abc', True, 2.34, None]
```

```
nested_list = [['a', 'b', 'c'], [1, 2, 3]]
```

### ◦ Python01

```
names = ['Alice', 'Bob', 'Craig', 'Diana', 'Eric']
print(names[0]) # Alice
print(names[2]) # Craig
```

-1◦

```
print(names[-1]) # Eric
print(names[-4]) # Bob
```

```
names[0] = 'Ann'
print(names)
# Outputs ['Ann', 'Bob', 'Craig', 'Diana', 'Eric']
```

/

L.append(object) L.append(object) None ◦

```
names = ['Alice', 'Bob', 'Craig', 'Diana', 'Eric']
names.append("Sia")
print(names)
# Outputs ['Alice', 'Bob', 'Craig', 'Diana', 'Eric', 'Sia']
```

◦ L.insert(index, object)

```
names.insert(1, "Nikki")
print(names)
# Outputs ['Alice', 'Nikki', 'Bob', 'Craig', 'Diana', 'Eric', 'Sia']
```

L.remove(value) L.remove(value) None

```
names.remove("Bob")
print(names) # Outputs ['Alice', 'Nikki', 'Craig', 'Diana', 'Eric', 'Sia']
```

X◦ ◦

```
name.index("Alice")
0
```

```
len(names)
6
```

```
a = [1, 1, 1, 2, 3, 4]
a.count(1)
3
```

```
a.reverse()
[4, 3, 2, 1, 1, 1]
# or
a[::-1]
[4, 3, 2, 1, 1, 1]
```

```
L.pop([index])L.pop([index])
```

```
names.pop() # Outputs 'Sia'
```

```
for element in my_list:
    print (element)
```

tuple ◦ ◦ IP ◦

```
ip_address = ('10.20.30.40', 8080)
```

◦ Python ◦

```
one_member_tuple = ('Only member',)
```

```
one_member_tuple = 'Only member', # No brackets
```

tuple

```
one_member_tuple = tuple(['Only member'])
```

Python dictionary ◦ ◦ ◦

```
state_capitals = {
    'Arkansas': 'Little Rock',
    'Colorado': 'Denver',
    'California': 'Sacramento',
    'Georgia': 'Atlanta'
}
```

```
ca_capital = state_capitals['California']
```

```
for k in state_capitals.keys():
    print('{} is the capital of {}'.format(state_capitals[k], k))
```

## JSON Python JSON

set • setlist •

setdictionary

```
first_names = {'Adam', 'Beth', 'Charlie'}
```

listset

```
my_list = [1,2,3]
my_set = set(my_list)
```

inset

```
if name in first_names:
    print(name)
```

set •

## defaultdict

defaultdict • defaultdict •

defaultdict **KeyError** • •

```
>>> state_capitals = {
    'Arkansas': 'Little Rock',
    'Colorado': 'Denver',
    'California': 'Sacramento',
    'Georgia': 'Atlanta'
}
```

## python

```
>>> state_capitals['Alabama']
Traceback (most recent call last):

  File "<ipython-input-61-236329695e6f>", line 1, in <module>
    state_capitals['Alabama']

KeyError: 'Alabama'
```

defaultdict • **collections** •

```
>>> from collections import defaultdict
>>> state_capitals = defaultdict(lambda: 'Boston')
```

## Give dict

```
>>> state_capitals['Arkansas'] = 'Little Rock'
>>> state_capitals['California'] = 'Sacramento'
>>> state_capitals['Colorado'] = 'Denver'
>>> state_capitals['Georgia'] = 'Atlanta'
```

## dictpythonBoston

```
>>> state_capitals['Alabama']
'Boston'
```

dictionary

```
>>> state_capitals['Arkansas']
'Little Rock'
```

## Python Python

```
>>> help()
```

```
>>> help(help)
```

```
help> help
```

Help on `_Helper` in module `_sitebuiltins` object:

```
class _Helper(builtins.object)
| Define the builtin 'help'.
|
| This is a wrapper around pydoc.help that provides a helpful message
| when 'help' is typed at the Python interactive prompt.
|
| Calling help() at the Python prompt starts an interactive help session.
| Calling help(thing) prints help for the python object 'thing'.
|
| Methods defined here:
|
| __call__(self, *args, **kwds)
|
| __repr__(self)
|
| -----
| Data descriptors defined here:
|
| __dict__
|     dictionary for instance variables (if defined)
|
| __weakref__
|     list of weak references to the object (if defined)
```

```
help(pymysql.connections)
```



```
>>> help(math)
```

```
>>> import math
>>> help(math)
```

◦  
quit  
◦  
.py◦

```
# hello.py
def say_hello():
    print("Hello!")
```

◦  
◦ **Python lib**◦

```
$ python
>>> import hello
>>> hello.say_hello()
=> "Hello!"
```

```
# greet.py
import hello
hello.say_hello()
```

```
# greet.py
from hello import say_hello
say_hello()
```

```
# greet.py
import hello as ai
ai.say_hello()
```

```
# run_hello.py
if __name__ == '__main__':
    from hello import say_hello
    say_hello()
```

```
$ python run_hello.py
=> "Hello!"
```

python\_\_init\_\_.py°

## - strrepr

°

```
repr(x)x.__repr__() x° eval°
```

```
str(x)x.__str__() ° °
```

---

```
eval()° ° °
```

## STR

° **this**repr(object)str(object)eval()° “”° ‘’°

---

1

```
s = """w'o"w"""
repr(s) # Output: '\w\\\'o"w\''
str(s) # Output: 'w\'o"w'
eval(str(s)) == s # Gives a SyntaxError
eval(repr(s)) == s # Output: True
```

2

```
import datetime
today = datetime.datetime.now()
str(today) # Output: '2016-09-15 06:58:46.915000'
repr(today) # Output: 'datetime.datetime(2016, 9, 15, 6, 58, 46, 915000)'
```

```
class Represent(object):

    def __init__(self, x, y):
        self.x, self.y = x, y

    def __repr__(self):
        return "Represent(x={},y=\"{}\")".format(self.x, self.y)

    def __str__(self):
        return "Representing x as {} and y as {}".format(self.x, self.y)
```

```
r = Represent(1, "Hopper")
print(r) # prints __str__
print(r.__repr__) # prints __repr__: '<bound method Represent.__repr__ of
Represent(x=1,y="Hopper")>'
```

```

rep = r.__repr__() # sets the execution of __repr__ to a new variable
print(rep) # prints 'Represent (x=1,y="Hopper") '
r2 = eval(rep) # evaluates rep
print(r2) # prints __str__ from new object
print(r2 == r) # prints 'False' because they are different objects

```

## pip

pythonPyPI pip python.orgPython 2> = 2.7.9Python 3> = 3.4pip Linux\* nix pip

Python 2Python 3 pipPython 2 pip3Python 3.pipPython 2pip3Python 3



```

$ pip search <query>
# Searches for packages whose name or summary contains <query>

```

## /Python

```

$ pip install [package_name] # latest version of the package
$ pip install [package_name]==x.x.x # specific version of the package
$ pip install '[package_name]>=x.x.x' # minimum version of the package

```

xxx

```

$ pip --proxy http://<server address>:<port> install

```

.

```

$ pip list --outdated

```

```

$ pip install [package_name] --upgrade

```

pip

## pip

- LinuxmacOS X

```

$ pip install -U pip

```

Linuxpipsudo

.

## Windows

```
py -m pip install -U pip
```

```
python -m pip install -U pip
```

pip ◦

## Python 2.7.x3.x.

Python 2.7 Python 3.x ◦

[https://www.python.org/downloads/Python 2.7](https://www.python.org/downloads/Python%202.7) ◦ MSI ◦ ◦

## Python

```
C:\Python27\
```

PATH ◦

PythonC\ Python27PATH

```
C:\Python27\;C:\Python27\Scripts\
```

## Pythoncmd

```
python --version
```

## Python 2.x3.x Side-By-Side

Windows Python 2.x3.x

### 1. MSI Python 2.x.

- Python ◦
- Python<sub>PATH</sub> python Python 2.x.

### 2. Python 3.x.

- Python ◦
- Python<sub>PATH</sub>python Python 3.x. Python 2.x <sub>PATH</sub>PATH ◦
- py launcher ◦

Python 3 Python Python 2.x Python 3.x

```
P:\>py -3
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>>

C:\>py -2
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 Intel] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

## Python pip

```
C:\>py -3 -m pip -V
pip 9.0.1 from C:\Python36\lib\site-packages (python 3.6)

C:\>py -2 -m pip -V
pip 9.0.1 from C:\Python27\lib\site-packages (python 2.7)
```

## LINUX

CentOS Fedora Redhat Enterprise RHEL Ubuntu Python 2.7.

### Linux Python 2.7

```
wget --no-check-certificate https://www.python.org/ftp/python/2.7.X/Python-2.7.X.tgz
tar -xzf Python-2.7.X.tgz
cd Python-2.7.X
./configure
make
sudo make install
```

PATH python. python/root/python-2.7.X export PATH = \$PATH:/root/python-2.7.X

### Python

```
python --version
```

### Ubuntu

Python 3.6 Ubuntu 16.10 17.04 3.6. Ubuntu 16.04

```
sudo apt install build-essential checkinstall
sudo apt install libreadline-gplv2-dev libncursesw5-dev libssl-dev libsqlite3-dev tk-dev
libgdbm-dev libc6-dev libbz2-dev
wget https://www.python.org/ftp/python/3.6.1/Python-3.6.1.tar.xz
tar xvf Python-3.6.1.tar.xz
cd Python-3.6.1/
./configure --enable-optimizations
sudo make altinstall
```

macOS Python 2.7.10 Python.

OS X Python. OS X Python.

### Homebrew

```
/usr/bin/ruby -e "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

## Python 2.7

```
brew install python
```

**Python 3.x** `brew install python3`

**Python** <https://riptutorial.com/zh-TW/python/topic/193/python>

---

## 2: \* args \*\* kwargs

### 1. argskwargs°

```
def func(*args, **kwargs):  
    print(args)  
    print(kwargs)
```

---

```
def func(*a, **b):  
    print(a)  
    print(b)
```

### 2. argskwargs

```
def func(*args1, *args2):  
#   File "<stdin>", line 1  
#       def test(*args1, *args2):  
#           ^  
# SyntaxError: invalid syntax
```

---

```
def test(**kwargs1, **kwargs2):  
#   File "<stdin>", line 1  
#       def test(**kwargs1, **kwargs2):  
#           ^  
# SyntaxError: invalid syntax
```

### 3. \*args ° \*args° Python 3°

```
def func(a, b, *args, x, y):  
    print(a, b, args, x, y)  
  
func(1, 2, 3, 4, x=5, y=6)  
#>>> 1, 2, (3, 4), 5, 6
```

---

```
def func(a, b, *, x, y):  
    print(a, b, x, y)  
  
func(1, 2, x=5, y=6)  
#>>> 1, 2, 5, 6
```

### 4. \*\*kwargs°

```
def test(**kwargs, *args):  
#   File "<stdin>", line 1  
#       def test(**kwargs, *args):  
#           ^
```

```
# SyntaxError: invalid syntax
```

## Examples

### \* args

#### star \*

```
def print_args(farg, *args):
    print("formal arg: %s" % farg)
    for arg in args:
        print("another positional arg: %s" % arg)
```

```
print_args(1, "two", 3)
```

#### fargargs◦

### \*\* kwargs

\*\*

```
def print_kwargs(**kwargs):
    print(kwargs)
```

## Python

```
print_kwargs(a="two", b=3)
# prints: "{a: 'two', b=3}"
```

#### \*\* kwargs◦

```
def example(a, **kw):
    print kw

example(a=2, b=3, c=4) # => {'b': 3, 'c': 4}
```

#### kwargs;kwargs

```
def print_kwargs(**kwargs):
    for key in kwargs:
        print("key = {0}, value = {1}".format(key, kwargs[key]))
```

```
print_kwargs(a="two", b=1)
```

```
print_kwargs(a = "two", b = 1)
key = a, value = "two"
key = b, value = 1
```



## \* args

\*args° \_\_init\_\_

```
class A(object):
    def __init__(self, b, c):
        self.y = b
        self.z = c

class B(A):
    def __init__(self, a, *args, **kwargs):
        super(B, self).__init__(*args, **kwargs)
        self.x = a
```

a°

```
b = B(1, 2, 3)
b.x # 1
b.y # 2
b.z # 3
```

B\_\_init\_\_1, 2, 3° a 1 a == 1°

\*args 1, 2 \*args° args == [2, 3]°

\*argsA\_\_init\_\_° Pythonargs\*""° B\_\_init\_\_A\_\_init\_\_2, 3 A(2, 3)°

xa 1°

## \*\* kwargs

;

```
def test_func(arg1, arg2, arg3): # Usual function with three arguments
    print("arg1: %s" % arg1)
    print("arg2: %s" % arg2)
    print("arg3: %s" % arg3)

# Note that dictionaries are unordered, so we can switch arg2 and arg3. Only the names matter.
kwargs = {"arg3": 3, "arg2": "two"}

# Bind the first argument (ie. arg1) to 1, and use the kwargs dictionary to bind the others
test_var_args_call(1, **kwargs)
```

## \* args

\*

```
def print_args(arg1, arg2):
    print(str(arg1) + str(arg2))

a = [1,2]
```

```

b = tuple([3,4])

print_args(*a)
# 12
print_args(*b)
# 34

```

o

## pythonzip\*

```

a = [1,3,5,7,9]
b = [2,4,6,8,10]

zipped = zip(a,b)
# [(1,2), (3,4), (5,6), (7,8), (9,10)]

zip(*zipped)
# (1,3,5,7,9), (2,4,6,8,10)

```

## Python 3. star \*。 \*。 o

```

def print_args(arg1, *args, keyword_required, keyword_only=True):
    print("first positional arg: {}".format(arg1))
    for arg in args:
        print("another positional arg: {}".format(arg))
    print("keyword_required value: {}".format(keyword_required))
    print("keyword_only value: {}".format(keyword_only))

print(1, 2, 3, 4) # TypeError: print_args() missing 1 required keyword-only argument:
'keyword_required'
print(1, 2, 3, keyword_required=4)
# first positional arg: 1
# another positional arg: 2
# another positional arg: 3
# keyword_required value: 4
# keyword_only value: True

```

## kwarg

```

def foobar(foo=None, bar=None):
    return "{}{}".format(foo, bar)

values = {"foo": "foo", "bar": "bar"}

foobar(**values) # "foobar"

```

## \*\* kwargs

### \*\* kwargs

```

def fun(**kwargs):
    print kwargs.get('value', 0)

```

```
fun()  
# print 0  
fun(value=1)  
# print 1
```

**\* args\*\* kwargs** <https://riptutorial.com/zh-TW/python/topic/2475/--args---kwargs>

## 3: `__name__`

`__name__` `name`

Python `__name__` `'__main__'`

`obj.__name__`

## Examples

`__name__ == '__main__'`

`__name__` `import` `if __name__ == '__main__'`

### module\_1.py

```
import module2.py
```

### module2.py

1

### module2.py

```
print('hello')
```

module1.py `hello`

module2.py `hello`

2

### module2.py

```
if __name__ == '__main__':  
    print('hello')
```

module1.py

module2.py `hello`

## function\_class\_or\_module.`__name__`

`__name__`

```
import os
```

```
class C:
```

```

    pass

def f(x):
    x += 2
    return x

print(f)
# <function f at 0x029976B0>
print(f.__name__)
# f

print(C)
# <class '__main__.C'>
print(C.__name__)
# C

print(os)
# <module 'os' from '/spam/eggs/'>
print(os.__name__)
# os

```

\_\_name\_\_°

```

def f():
    pass

print(f.__name__)
# f - as expected

g = f
print(g.__name__)
# f - even though the variable is named g, the function is still named f

```

```

def enter_exit_info(func):
    def wrapper(*arg, **kw):
        print '-- entering', func.__name__
        res = func(*arg, **kw)
        print '-- exiting', func.__name__
        return res
    return wrapper

```

```

@enter_exit_info
def f(x):
    print 'In:', x
    res = x + 2
    print 'Out:', res
    return res

```

```
a = f(2)
```

```

# Outputs:
# -- entering f
# In: 2
# Out: 4
# -- exiting f

```

logging\_\_name\_\_

```
logger = logging.getLogger(__name__)
```

◦

`__name__` <https://riptutorial.com/zh-TW/python/topic/1223/--name-->

# 4: 2to3

- \$ 2to3 [-options] path / to / file.py.

filename / directory_name	2to3 Python
-f FIX - fix = FIX	; --list-fixes
-j PROCESSES - processes = PROCESSES	23
-x NOFIX - nofix = NOFIX	
-l - list-fixes	
-p - print-function	print()
-v - verbose	
--no-diff	
-w	
-n - nobackups	
-o OUTPUT_DIR - output-dir = OUTPUT_DIR	o -n
-W -	o -o -w
--add= ADD_SUFFIX	o -n - --add-suffix='3'.py3

2to3pythonPython 2.xPython 3.x Python 2.xPython 3.x

2to3lib2to3 o lib2to32to3

## Examples

Python2.x example.py

Python 2.x 2.0

```
def greet(name):  
    print "Hello, {0}!".format(name)  
print "What's your name?"  
name = raw_input()  
greet(name)
```

◦ `raw_input()` Python 3.x `input()` `print()` 2to3 Python 3.x ◦

## Unix

```
$ 2to3 example.py
```

```
> path/to/2to3.py example.py
```

◦

```
RefactoringTool: Skipping implicit fixer: buffer
RefactoringTool: Skipping implicit fixer: idioms
RefactoringTool: Skipping implicit fixer: set_literal
RefactoringTool: Skipping implicit fixer: ws_comma
RefactoringTool: Refactored example.py
--- example.py      (original)
+++ example.py      (refactored)
@@ -1,5 +1,5 @@
     def greet(name):
-        print "Hello, {0}!".format(name)
-print "What's your name?"
-name = raw_input()
+        print("Hello, {0}!".format(name))
+print("What's your name?")
+name = input()
     greet(name)
RefactoringTool: Files that need to be modified:
RefactoringTool: example.py
```

`-W` ◦ `-n` `example.py.bak` ◦

## Unix

```
$ 2to3 -w example.py
```

```
> path/to/2to3.py -w example.py
```

`example.py` Python 2.x Python 3.x ◦

`example.py` Python3.x

### Python 3.x 3.0

```
def greet(name):
    print("Hello, {0}!".format(name))
print("What's your name?")
name = input()
greet(name)
```

2to3 <https://riptutorial.com/zh-TW/python/topic/5320/2to3>



## 5: ArcPy

ArcPyda。

arcpy.da.SearchCursorarcpy.SearchCursor。

arcpy.daArcGIS 10.1 for Desktop。

### Examples

““”

C:\TempTest.gdbTestFCTestField

```
with arcpy.da.SearchCursor(r"C:\Temp\Test.gdb\TestFC", ["TestField"]) as cursor:
    for row in cursor:
        print row[0]
```

### createDissolvedGDBgdb

```
def createDissolvedGDB(workspace, gdbName):
    gdb_name = workspace + "/" + gdbName + ".gdb"

    if(arcpy.Exists(gdb_name):
        arcpy.Delete_management(gdb_name)
        arcpy.CreateFileGDB_management(workspace, gdbName, "")
    else:
        arcpy.CreateFileGDB_management(workspace, gdbName, "")

    return gdb_name
```

ArcPy <https://riptutorial.com/zh-TW/python/topic/4693/arcpy>

# 6: Asyncio

## Examples

Python 3.5+ `asyncio` Python 3.5◦

Python 3.x 3.5

Python 3.5 `async` `await` ◦ `await func()`◦

```
import asyncio

async def main():
    print(await func())

async def func():
    # Do time intensive stuff...
    return "Hello, world!"

if __name__ == "__main__":
    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

Python 3.x 3.3 3.5

Python 3.5 `@asyncio.coroutine` ◦ `yield from func()` `yield from func()`◦

```
import asyncio

@asyncio.coroutine
def main():
    print((yield from func()))

@asyncio.coroutine
def func():
    # Do time intensive stuff..
    return "Hello, world!"

if __name__ == "__main__":
    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

Python 3.x 3.5

```
import asyncio

async def cor1():
    print("cor1 start")
    for i in range(10):
        await asyncio.sleep(1.5)
        print("cor1", i)

async def cor2():
    print("cor2 start")
```

```

for i in range(15):
    await asyncio.sleep(1)
    print("cor2", i)

loop = asyncio.get_event_loop()
cors = asyncio.wait([cor1(), cor2()])
loop.run_until_complete(cors)

```

## Python 3.5+ async / await

asyncioconcurrent.futuresExecutor ◦ run\_in\_executor() Executor Callable ◦ Callable ◦

Executor

```

import asyncio
from concurrent.futures import ThreadPoolExecutor

def func(a, b):
    # Do time intensive stuff...
    return a + b

async def main(loop):
    executor = ThreadPoolExecutor()
    result = await loop.run_in_executor(executor, func, "Hello,", " world!")
    print(result)

if __name__ == "__main__":
    loop = asyncio.get_event_loop()
    loop.run_until_complete(main(loop))

```

“” ExecutorExecutor ◦ Executorset\_default\_executor() ◦

```

import asyncio
from concurrent.futures import ThreadPoolExecutor

def func(a, b):
    # Do time intensive stuff...
    return a + b

async def main(loop):
    # NOTE: Using `None` as the first parameter designates the `default` Executor.
    result = await loop.run_in_executor(None, func, "Hello,", " world!")
    print(result)

if __name__ == "__main__":
    loop = asyncio.get_event_loop()
    loop.set_default_executor(ThreadPoolExecutor())
    loop.run_until_complete(main(loop))

```

concurrent.futuresExecutor ThreadPoolExecutorProcessPoolExecutor ◦ ThreadPoolExecutor5.

ThreadPoolExecutorCPU / O ◦ ProcessPoolExecutor ProcessPoolExecutor ◦ ProcessPoolExecutor ◦ ◦

ExecutorThreadPoolExecutor ◦

# UVLoop

uvloop **libuv** asyncio.AbstractEventLoop **nodejs** ◦ 99 asyncio asyncio.EventLoop ◦ uvloop **Windows** pip  
install uvloop ◦

```
import asyncio
import uvloop

if __name__ == "__main__":
    asyncio.set_event_loop(uvloop.new_event_loop())
    # Do your stuff here ...
```

EventLoopPolicy uvloop ◦

```
import asyncio
import uvloop

if __name__ == "__main__":
    asyncio.set_event_loop_policy(uvloop.EventLoopPolicy())
    loop = asyncio.new_event_loop()
```

---

Event ◦

◦

```
import asyncio

# event trigger function
def trigger(event):
    print('EVENT SET')
    event.set() # wake up coroutines waiting

# event consumers
async def consumer_a(event):
    consumer_name = 'Consumer A'
    print('{} waiting'.format(consumer_name))
    await event.wait()
    print('{} triggered'.format(consumer_name))

async def consumer_b(event):
    consumer_name = 'Consumer B'
    print('{} waiting'.format(consumer_name))
    await event.wait()
    print('{} triggered'.format(consumer_name))

# event
event = asyncio.Event()

# wrap coroutines in one future
main_future = asyncio.wait([consumer_a(event),
                           consumer_b(event)])
```

```
# event loop
event_loop = asyncio.get_event_loop()
event_loop.call_later(0.1, functools.partial(trigger, event)) # trigger event in 0.1 sec

# complete main_future
done, pending = event_loop.run_until_complete(main_future)
```

B

B

A

## Websocket

asyncio `echo websocket` / `websocket` `main`

```
import asyncio
import aiohttp

session = aiohttp.ClientSession() # handles the context manager
class EchoWebsocket:

    async def connect(self):
        self.websocket = await session.ws_connect("wss://echo.websocket.org")

    async def send(self, message):
        self.websocket.send_str(message)

    async def receive(self):
        result = (await self.websocket.receive())
        return result.data

async def main():
    echo = EchoWebsocket()
    await echo.connect()
    await echo.send("Hello World!")
    print(await echo.receive()) # "Hello World!"

if __name__ == '__main__':
    # The main loop
    loop = asyncio.get_event_loop()
    loop.run_until_complete(main())
```

## asyncio

asyncio-GIL

asyncio `asyncio.sleep` `await` `time.sleep` `time.sleep`

`requests` `asyncio` `aiohttp` `asyncio`

- CPU `asyncio` `threadsmultiprocessing`

- IO<sub>asyncio</sub>

Asyncio <https://riptutorial.com/zh-TW/python/topic/1319/asyncio>

# 7: base64

Base 64radix 64ASCII.base64Python。 。 base64。

- base64.b64encodesaltchars = None
- base64.b64decodesaltchars = Nonevalidate = False
- base64.standard\_b64encodeS
- base64.standard\_b64decodeS
- base64.urlsafe\_b64encodeS
- base64.urlsafe\_b64decodeS
- base64.b32encodeS
- base64.b32decodeS
- base64.b16encodeS
- base64.b16decodeS
- base64.a85encodebfpadspaces = Falsewrapcol = 0pad = Falseadobe = False
- base64.a85decodebfpadspaces = Falseadobe = Falseignorechars = b'\t\n\r\n'
- base64.b85encodebpad = False
- base64.b85decodeb

<code>base64.b64encode(s, altchars=None)</code>	
<b>altchars</b>	2+Base64“+”“=”。 。
<code>base64.b64decode(s, altchars=None, validate=False)</code>	
<b>altchars</b>	2+Base64“+”“=”。 。
	valideTrueBase64
<code>base64.standard_b64encode(s)</code>	
<code>base64.standard_b64decode(s)</code>	
<code>base64.urlsafe_b64encode(s)</code>	
<code>base64.urlsafe_b64decode(s)</code>	
<code>b32encode(s)</code>	
<code>b32decode(s)</code>	
<code>base64.b16encode(s)</code>	

<code>base64.b16decode(s)</code>	
<code>base64.a85encode(b, foldspaces=False, wrapcol=0, pad=False, adobe=False)</code>	
<b>b</b>	
<b>foldspaces</b>	foldspacesTrue“y”4。
<b>wrapcol</b>	0
	padTrue4
	adobeTrueAdobe'<'>。 。
<code>base64.a85decode(b, foldspaces=False, adobe=False, ignorechars=b'\t\n\r\v')</code>	
<b>b</b>	
<b>foldspaces</b>	foldspacesTrue“y”4。
	adobeTrueAdobe'<'>。 。
<b>ignorechars</b>	
<code>base64.b85encode(b, pad=False)</code>	
<b>b</b>	
	padTrue4
<code>base64.b85decode(b)</code>	
<b>b</b>	

Python 3.4base64bytesbytes bytearray。。

## Examples

### Base64

base64

```
import base64
```

base64。 Python。 UTF-8Python。



```
s = "Hello World!"
b = s.encode("UTF-8")
```

b'Hello World!'

b°

**Base64**base64.b64encode()

```
import base64
s = "Hello World!"
b = s.encode("UTF-8")
e = base64.b64encode(b)
print(e)
```

b'SGVsbG8gV29ybGQh'

**bytes**° Pythondecode() UTF-8

```
import base64
s = "Hello World!"
b = s.encode("UTF-8")
e = base64.b64encode(b)
s1 = e.decode("UTF-8")
print(s1)
```

SGVsbG8gV29ybGQh

base64.b64decode()

```
import base64
# Creating a string
s = "Hello World!"
# Encoding the string into bytes
b = s.encode("UTF-8")
# Base64 Encode the bytes
e = base64.b64encode(b)
# Decoding the Base64 bytes to string
s1 = e.decode("UTF-8")
# Printing Base64 encoded string
print("Base64 Encoded:", s1)
# Encoding the Base64 encoded string into bytes
b1 = s1.encode("UTF-8")
# Decoding the Base64 bytes
d = base64.b64decode(b1)
# Decoding the bytes to string
s2 = d.decode("UTF-8")
print(s2)
```

Base64 Encoded: SGVsbG8gV29ybGQh  
Hello World!

## Base32

base64Base32° Base64

```

import base64
# Creating a string
s = "Hello World!"
# Encoding the string into bytes
b = s.encode("UTF-8")
# Base32 Encode the bytes
e = base64.b32encode(b)
# Decoding the Base32 bytes to string
s1 = e.decode("UTF-8")
# Printing Base32 encoded string
print("Base32 Encoded:", s1)
# Encoding the Base32 encoded string into bytes
b1 = s1.encode("UTF-8")
# Decoding the Base32 bytes
d = base64.b32decode(b1)
# Decoding the bytes to string
s2 = d.decode("UTF-8")
print(s2)

```

```

Base32 Encoded: JBSWY3DPEBLW64TMMQQQ====
Hello World!

```

## Base16

### base64Base16。 16。 Base64Base32

```

import base64
# Creating a string
s = "Hello World!"
# Encoding the string into bytes
b = s.encode("UTF-8")
# Base16 Encode the bytes
e = base64.b16encode(b)
# Decoding the Base16 bytes to string
s1 = e.decode("UTF-8")
# Printing Base16 encoded string
print("Base16 Encoded:", s1)
# Encoding the Base16 encoded string into bytes
b1 = s1.encode("UTF-8")
# Decoding the Base16 bytes
d = base64.b16decode(b1)
# Decoding the bytes to string
s2 = d.decode("UTF-8")
print(s2)

```

```

Base16 Encoded: 48656C6C6F20576F726C6421
Hello World!

```

## ASCII85

### AdobeASCII85 Base85。 Adobe PDF。 Python 3.4。 base64.a85encode()base64.a85encode()

```

import base64
# Creating a string
s = "Hello World!"

```

```

# Encoding the string into bytes
b = s.encode("UTF-8")
# ASCII85 Encode the bytes
e = base64.a85encode(b)
# Decoding the ASCII85 bytes to string
s1 = e.decode("UTF-8")
# Printing ASCII85 encoded string
print("ASCII85 Encoded:", s1)
# Encoding the ASCII85 encoded string into bytes
b1 = s1.encode("UTF-8")
# Decoding the ASCII85 bytes
d = base64.a85decode(b1)
# Decoding the bytes to string
s2 = d.decode("UTF-8")
print(s2)

```

```

ASCII85 Encoded: 87cURD]i,"Ebo80
Hello World!

```

## Base85

Base64Base32Base16Base85base64.b85encode()base64.b85decode()

```

import base64
# Creating a string
s = "Hello World!"
# Encoding the string into bytes
b = s.encode("UTF-8")
# Base85 Encode the bytes
e = base64.b85encode(b)
# Decoding the Base85 bytes to string
s1 = e.decode("UTF-8")
# Printing Base85 encoded string
print("Base85 Encoded:", s1)
# Encoding the Base85 encoded string into bytes
b1 = s1.encode("UTF-8")
# Decoding the Base85 bytes
d = base64.b85decode(b1)
# Decoding the bytes to string
s2 = d.decode("UTF-8")
print(s2)

```

```

Base85 Encoded: NM&qnZy;Bla% ^NF
Hello World!

```

base64 <https://riptutorial.com/zh-TW/python/topic/8678/base64>

## 8: ChemPy - python

ChemPypython. Python.

### Examples

```
from chempy import Substance
ferricyanide = Substance.from_formula('Fe(CN)63-')
ferricyanide.composition == {0: -3, 26: 1, 6: 6, 7: 6}
True
print(ferricyanide.unicode_name)
Fe(CN)63-
print(ferricyanide.latex_name + ", " + ferricyanide.html_name)
Fe(CN){6}{3-}, Fe(CN)<sub>6</sub><sup>3-</sup>
print('%0.3f' % ferricyanide.mass)
211.955
```

0.

```
from chempy import balance_stoichiometry # Main reaction in NASA's booster rockets:
reac, prod = balance_stoichiometry({'NH4ClO4', 'Al'}, {'Al2O3', 'HCl', 'H2O', 'N2'})
from pprint import pprint
pprint(reac)
{'Al': 10, 'NH4ClO4': 6}
pprint(prod)
{'Al2O3': 5, 'H2O': 9, 'HCl': 6, 'N2': 3}
from chempy import mass_fractions
for fractions in map(mass_fractions, [reac, prod]):
...     pprint({k: '{0:.3g} wt%'.format(v*100) for k, v in fractions.items()})
...
{'Al': '27.7 wt%', 'NH4ClO4': '72.3 wt%'}
{'Al2O3': '52.3 wt%', 'H2O': '16.6 wt%', 'HCl': '22.4 wt%', 'N2': '8.62 wt%'}
```

```
from chempy import Equilibrium
from sympy import symbols
K1, K2, Kw = symbols('K1 K2 Kw')
e1 = Equilibrium({'MnO4-': 1, 'H+': 8, 'e-': 5}, {'Mn2+': 1, 'H2O': 4}, K1)
e2 = Equilibrium({'O2': 1, 'H2O': 2, 'e-': 4}, {'OH-': 4}, K2)
coeff = Equilibrium.eliminate([e1, e2], 'e-')
coeff
[4, -5]
redox = e1*coeff[0] + e2*coeff[1]
print(redox)
20 OH- + 32 H+ + 4 MnO4- = 26 H2O + 4 Mn2+ + 5 O2; K1**4/K2**5
autoprot = Equilibrium({'H2O': 1}, {'H+': 1, 'OH-': 1}, Kw)
n = redox.cancel(autoprot)
n
20
redox2 = redox + n*autoprot
print(redox2)
12 H+ + 4 MnO4- = 4 Mn2+ + 5 O2 + 6 H2O; K1**4*Kw**20/K2**5
```

```
from chempy import Equilibrium
```

```

from chempy.chemistry import Species
water_autop = Equilibrium({'H2O'}, {'H+', 'OH-'}, 10**-14) # unit "molar" assumed
ammonia_prot = Equilibrium({'NH4+'}, {'NH3', 'H+'}, 10**-9.24) # same here
from chempy.equilibria import EqSystem
substances = map(Species.from_formula, 'H2O OH- H+ NH3 NH4+'.split())
eqsys = EqSystem([water_autop, ammonia_prot], substances)
print('\n'.join(map(str, eqsys.rxns))) # "rxns" short for "reactions"
H2O = H+ + OH-; 1e-14
NH4+ = H+ + NH3; 5.75e-10
from collections import defaultdict
init_conc = defaultdict(float, {'H2O': 1, 'NH3': 0.1})
x, sol, sane = eqsys.root(init_conc)
assert sol['success'] and sane
print(sorted(sol.keys())) # see package "pyneqsys" for more info
['fun', 'intermediate_info', 'internal_x_vecs', 'nfev', 'njev', 'success', 'x', 'x_vecs']
print(', '.join('%2g' % v for v in x))
1, 0.0013, 7.6e-12, 0.099, 0.0013

```

```

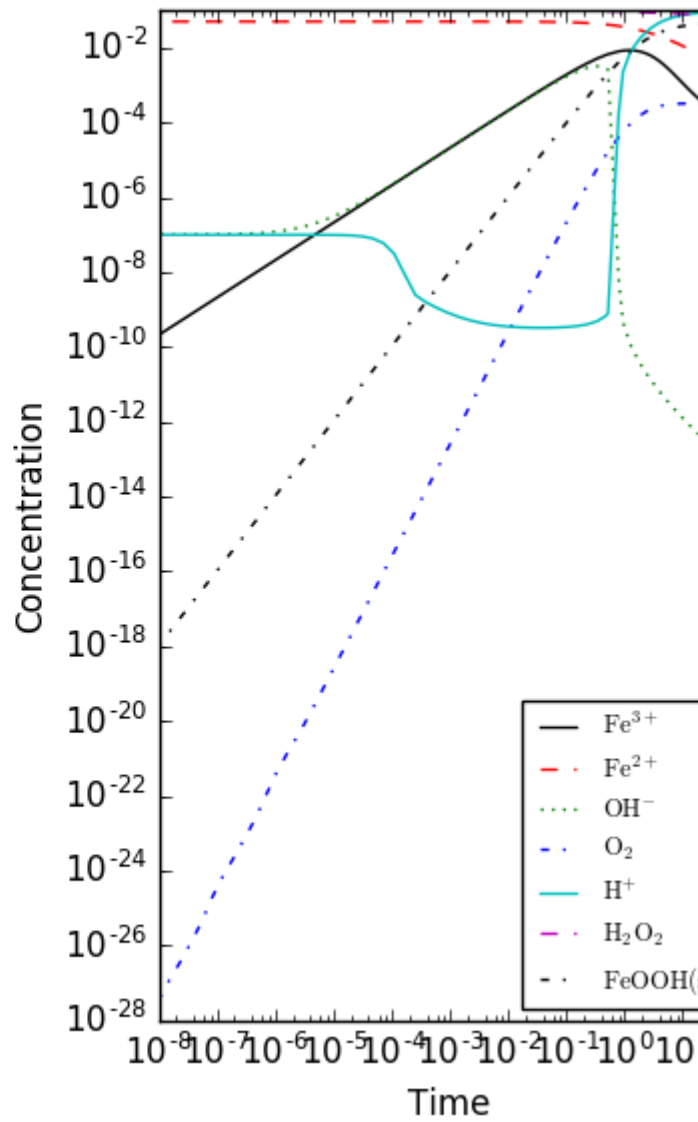
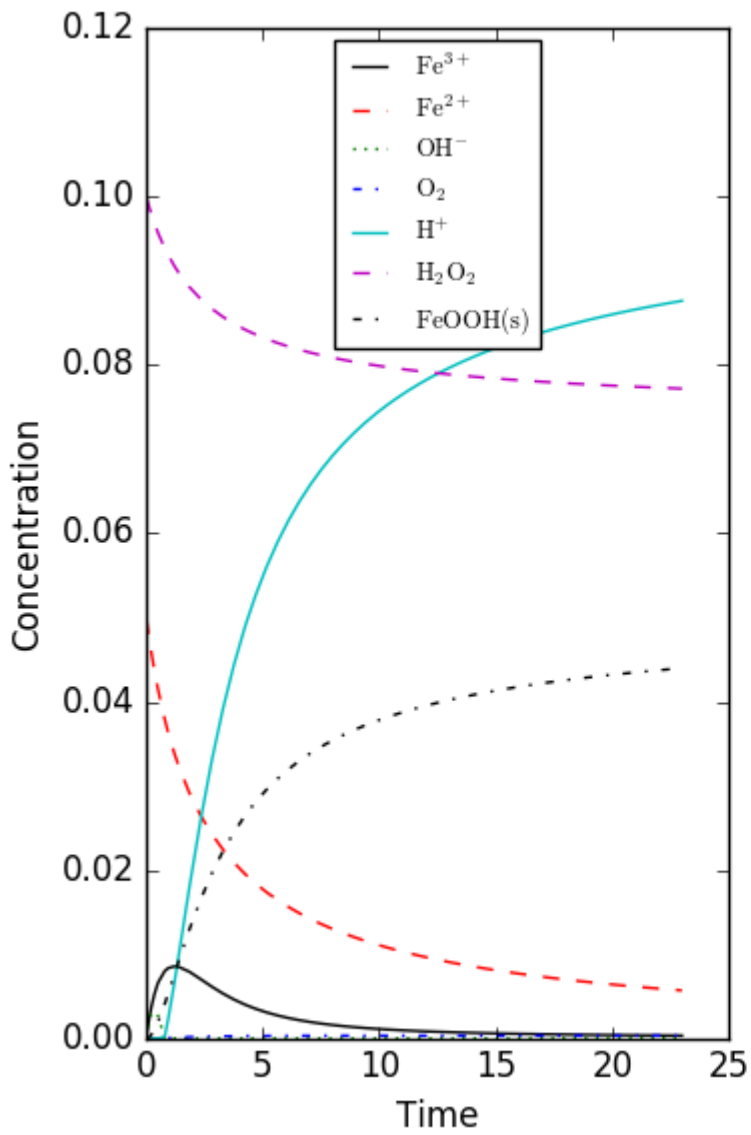
from chempy.electrolytes import ionic_strength
ionic_strength({'Fe+3': 0.050, 'ClO4-': 0.150}) == .3
True

```

```

from chempy import ReactionSystem # The rate constants below are arbitrary
rsys = ReactionSystem.from_string("""2 Fe+2 + H2O2 -> 2 Fe+3 + 2 OH-; 42
2 Fe+3 + H2O2 -> 2 Fe+2 + O2 + 2 H+; 17
H+ + OH- -> H2O; 1e10
H2O -> H+ + OH-; 1e-4
Fe+3 + 2 H2O -> FeOOH(s) + 3 H+; 1
FeOOH(s) + 3 H+ -> Fe+3 + 2 H2O; 2.5""") # "[H2O]" = 1.0 (actually 55.4 at RT)
from chempy.kinetics.ode import get_odesys
odesys, extra = get_odesys(rsys)
from collections import defaultdict
import numpy as np
tout = sorted(np.concatenate((np.linspace(0, 23), np.logspace(-8, 1))))
c0 = defaultdict(float, {'Fe+2': 0.05, 'H2O2': 0.1, 'H2O': 1.0, 'H+': 1e-7, 'OH-': 1e-7})
result = odesys.integrate(tout, c0, atol=1e-12, rtol=1e-14)
import matplotlib.pyplot as plt
_ = plt.subplot(1, 2, 1)
_ = result.plot(names=[k for k in rsys.substances if k != 'H2O'])
_ = plt.legend(loc='best', prop={'size': 9}); _ = plt.xlabel('Time'); _ =
plt.ylabel('Concentration')
_ = plt.subplot(1, 2, 2)
_ = result.plot(names=[k for k in rsys.substances if k != 'H2O'], xscale='log', yscale='log')
_ = plt.legend(loc='best', prop={'size': 9}); _ = plt.xlabel('Time'); _ =
plt.ylabel('Concentration')
_ = plt.tight_layout()
plt.show()

```



ChemPy - python <https://riptutorial.com/zh-TW/python/topic/10625/chempy-----python>

---

# 9: configparser

ConfigParserINI。 Python。

- =
- 
- INI[]

ConfigParser.ConfigParser().get。 eval

## Examples

config.ini

```
[DEFAULT]
debug = True
name = Test
password = password

[FILES]
path = /path/to/file
```

Python

```
from ConfigParser import ConfigParser
config = ConfigParser()

#Load configuration file
config.read("config.ini")

# Access the key "debug" in "DEFAULT" section
config.get("DEFAULT", "debug")
# Return 'True'

# Access the key "path" in "FILES" destion
config.get("FILES", "path")
# Return '/path/to/file'
```

。 configparser。 -

```
import configparser
config = configparser.ConfigParser()
config['settings']={'resolution':'320x240',
                  'color':'blue'}
with open('example.ini', 'w') as configfile:
    config.write(configfile)
```

```
[settings]
resolution = 320x240
color = blue
```

```
settings=config['settings']  
settings['color']='red'
```

**configparser** <https://riptutorial.com/zh-TW/python/topic/9186/configparser>



---

# 10: ctypes

ctypespython

◦

## Examples

libc.toh1

libc.so

```
>>> from ctypes import *
>>> libc = cdll.LoadLibrary('libc.so.6')
>>> libc
<CDLL 'libc.so.6', handle baadf00d at 0xdeadbeef>
```

```
>>> ntohl = libc.ntohl
>>> ntohl
<_FuncPtr object at 0xbaadf00d>
```

```
>>> ntohl(0x6C)
1811939328
>>> hex(_)
'0x6c000000'
```

◦

---

◦ OSError

## OS

```
>>> cdll.LoadLibrary("foobar.so")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/lib/python3.5/ctypes/__init__.py", line 425, in LoadLibrary
    return self._dlltype(name)
  File "/usr/lib/python3.5/ctypes/__init__.py", line 347, in __init__
    self._handle = _dlopen(self._name, mode)
OSError: foobar.so: cannot open shared object file: No such file or directory
```

◦

◦

```
>>> cdll.LoadLibrary("libc.so")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
```

```
File "/usr/lib/python3.5/ctypes/__init__.py", line 425, in LoadLibrary
    return self._dlltype(name)
File "/usr/lib/python3.5/ctypes/__init__.py", line 347, in __init__
    self._handle = _dlopen(self._name, mode)
OSError: /usr/lib/i386-linux-gnu/libc.so: invalid ELF header
```

.so° Linux.dll32python64° °

.so°

AttributeError

```
>>> libc.foo
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/lib/python3.5/ctypes/__init__.py", line 360, in __getattr__
    func = self.__getitem__(name)
  File "/usr/lib/python3.5/ctypes/__init__.py", line 365, in __getitem__
    func = self._FuncPtr((name_or_ordinal, self))
AttributeError: /lib/i386-linux-gnu/libc.so.6: undefined symbol: foo
```

## ctypes

### int

```
>>> obj = ctypes.c_int(12)
>>> obj
c_long(12)
```

obj12°

```
>>> obj.value
12
>>> obj.value = 13
>>> obj
c_long(13)
```

obj

```
>>> sizeof(obj)
4
>>> hex(addressof(obj))
'0xdeadbeef'
```

## ctypes

### C°

```
>>> c_int * 16
<class '__main__.c_long_Array_16'>
```

## 16int◦

```
>>> arr = (c_int * 16)(*range(16))
>>> arr
<__main__.c_long_Array_16 object at 0xbaddcafe>
```

## arr015◦

```
>>> arr[5]
5
>>> arr[5] = 20
>>> arr[5]
20
```

## ctypes

```
>>> sizeof(arr)
64 # sizeof(c_int) * 16
>>> hex(addressof(arr))
'0xc00010ff'
```

## ctypes

### C◦ ctypespython◦

```
>>> def max(x, y):
    return x if x >= y else y
```

### ◦ typeint◦

```
>>> CFUNCTYPE(c_int, c_int, c_int)
<CFunctionType object at 0xdeadbeef>
```

### c\_int c\_int◦

```
>>> CFUNCTYPE(c_int, c_int, c_int)(max)
<CFunctionType object at 0xdeadbeef>
```

### ctypeslibc.ntohl◦

```
>>> libc.ntohl() # garbage in - garbage out
>>> CFUNCTYPE(c_int, c_int)(libc.ntohl)()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: this function takes at least 1 argument (0 given)
```

### libclfind◦

◦ ◦

```
>>> compar_proto = CFUNCTYPE(c_int, POINTER(c_int), POINTER(c_int))
>>> lfind_proto = CFUNCTYPE(c_void_p, c_void_p, c_void_p, POINTER(c_uint), c_uint,
compar_proto)
```

```
>>> key = c_int(12)
>>> arr = (c_int * 16)(*range(16))
>>> nmemb = c_uint(16)
```

```
>>> def compar(x, y):
    return x.contents.value - y.contents.value
```

xyPOINTER(c\_int) ◦

```
>>> lfind = lfind_proto(libc.lfind)
>>> ptr = lfind(byref(key), byref(arr), byref(nmemb), sizeof(c_int), compar_proto(compar))
```

ptrvoid ◦ arrkey None ◦

```
>>> cast(ptr, POINTER(c_int)).contents
c_long(12)
```

ptrarr

```
>>> addressof(arr) + 12 * sizeof(c_int) == ptr
True
```

[ctypes https://riptutorial.com/zh-TW/python/topic/9050/ctypes](https://riptutorial.com/zh-TW/python/topic/9050/ctypes)

# 11: Deque

- dq = deque
- dq = dequeiterabledeque
- dq.appendobject
- dq.appendleftobject
- dq.pop - > object
- dq.popleft - > object
- dq.extenditerabledeque
- dq.extendleftiterable

iterable	◦
maxlen	◦

deque“ ”◦

pop appendextendleft◦ dequeO1◦

## Examples

popleftappendleft

```
from collections import deque

d = deque([1, 2, 3])
p = d.popleft()      # p = 1, d = deque([2, 3])
d.appendleft(5)     # d = deque([5, 2, 3])
```

## deque

maxlen

```
from collections import deque
d = deque(maxlen=3) # only holds 3 items
d.append(1) # deque([1])
d.append(2) # deque([1, 2])
d.append(3) # deque([1, 2, 3])
d.append(4) # deque([2, 3, 4]) (1 is removed because its maxlen is 3)
```

## deque

```
d1 = deque() # deque([]) creating empty deque
```

deque

```
dl = deque([1, 2, 3, 4]) # deque([1, 2, 3, 4])
```

## deque

```
dl.append(5) # deque([1, 2, 3, 4, 5])
```

## deque

```
dl.appendleft(0) # deque([0, 1, 2, 3, 4, 5])
```

## deque

```
dl.extend([6, 7]) # deque([0, 1, 2, 3, 4, 5, 6, 7])
```

```
dl.extendleft([-2, -1]) # deque([-1, -2, 0, 1, 2, 3, 4, 5, 6, 7])
```

```
.pop()
```

```
dl.pop() # 7 => deque([-1, -2, 0, 1, 2, 3, 4, 5, 6])
```

```
.popleft()
```

```
dl.popleft() # -1 deque([-2, 0, 1, 2, 3, 4, 5, 6])
```

```
dl.remove(1) # deque([-2, 0, 2, 3, 4, 5, 6])
```

## deque

```
dl.reverse() # deque([6, 5, 4, 3, 2, 0, -2])
```

## Deque Python◦ queue.Queue◦ Queue◦

```
from collections import deque

def bfs(graph, root):
    distances = {}
    distances[root] = 0
    q = deque([root])
    while q:
        # The oldest seen (but not yet visited) node will be the left most one.
        current = q.popleft()
        for neighbor in graph[current]:
            if neighbor not in distances:
                distances[neighbor] = distances[current] + 1
                # When we see a new node, we add it to the right side of the queue.
                q.append(neighbor)
    return distances
```

```
graph = {1:[2,3], 2:[4], 3:[4,5], 4:[3,5], 5:[]}
```

```
>>> bfs(graph, 1)
{1: 0, 2: 1, 3: 1, 4: 2, 5: 2}

>>> bfs(graph, 3)
{3: 0, 4: 1, 5: 1}
```

Deque <https://riptutorial.com/zh-TW/python/topic/1976/deque>

# 12: dis

## Examples

### dis

```
EXTENDED_ARG = 145 # All opcodes greater than this have 2 operands
HAVE_ARGUMENT = 90 # All opcodes greater than this have at least 1 operands

cmp_op = ('<', '<=', '==', '!=', '>', '>=', 'in', 'not in', 'is', 'is ...
        # A list of comparator id's. The indecies are used as operands in some opcodes

# All opcodes in these lists have the respective types as there operands
hascompare = [107]
hasconst = [100]
hasfree = [135, 136, 137]
hasjabs = [111, 112, 113, 114, 115, 119]
hasjrel = [93, 110, 120, 121, 122, 143]
haslocal = [124, 125, 126]
hasname = [90, 91, 95, 96, 97, 98, 101, 106, 108, 109, 116]

# A map of opcodes to ids
opmap = {'BINARY_ADD': 23, 'BINARY_AND': 64, 'BINARY_DIVIDE': 21, 'BIN...
# A map of ids to opcodes
opname = ['STOP_CODE', 'POP_TOP', 'ROT_TWO', 'ROT_THREE', 'DUP_TOP', '...
```

## Python

Python. PythonPython. disPython.

```
>>> def hello():
...     print "Hello, World"
...
>>> dis.dis(hello)
2          0 LOAD_CONST          1 ('Hello, World')
          3 PRINT_ITEM
          4 PRINT_NEWLINE
          5 LOAD_CONST          0 (None)
          8 RETURN_VALUE
```

Python.

Python. Python.

Python.pycPython.

```
python -m compileall <file>.py
```

```
import dis
import marshal
with open("<file>.pyc", "rb") as code_f:
```



```
code_f.read(8) # Magic number and modification time
code = marshal.load(code_f) # Returns a code object which can be disassembled
dis.dis(code) # Output the disassembly
```

Python<sub>dis</sub> ◦

[dis https://riptutorial.com/zh-TW/python/topic/1763/dis](https://riptutorial.com/zh-TW/python/topic/1763/dis)

# 13: Django

DjangoPython Web。 Web。 。

## Examples

### DjangoHello World

djangoHello World。

django。

```
python -c"import django"  
->django。
```

```
django。  
django-admin startproject HelloWorld
```

HelloWorld。

```
| --helloworld  
| | - init .py  
| | --settings.py  
| | --urls.py  
| | --wsgi.py  
| --manage.py
```

django

PythonWebWeb。 HTML。 views.py。

hello world。 views.py

```
from django.http import HttpResponse  
  
def helloWorld(request):  
    return HttpResponse("Hello World!! Django Welcomes You.")
```

。

- django.httpHttpResponse。
- helloWorld。 。
- HttpRequestrequest。
- ;Django。 helloWorld。
-

HttpResponse。 HttpResponse。

## django

### URL

URLURLconf;

django。

- DjangoURLconf。
- DjangoPythonurlpatterns。 django.conf.urls.urlPython。
- DjangoURLURL。
- DjangoPython。

### URLconf

```
from django.conf.urls import url
from . import views #import the views.py from current directory

urlpatterns = [
    url(r'^helloworld/$', views.helloWorld),
]
```

## django Urls

HelloWorld。

python manage.py runserver

127.0.0.1:8000

127.0.0.1:8000/helloworld/。 “Hello World !! Django”。

Django <https://riptutorial.com/zh-TW/python/topic/8994/django>

# 14: Functools

## Examples

partial ◦

```
>>> from functools import partial
>>> unhex = partial(int, base=16)
>>> unhex.__doc__ = 'Convert base16 string to int'
>>> unhex('callable')
3390155550
```

partial() ◦

```
In [2]: from functools import partial

In [3]: def f(a, b, c, x):
...:     return 1000*a + 100*b + 10*c + x
...:

In [4]: g = partial(f, 1, 1, 1)

In [5]: print g(2)
1112
```

g f a, b, c, x a, b, c, ◦ fg g(2) f ◦

partial; ◦ partial ◦

## total\_ordering

\_\_eq\_\_() \_\_lt\_\_() \_\_le\_\_() \_\_gt\_\_() \_\_le\_\_() \_\_gt\_\_() \_\_ge\_\_() ◦

total\_ordering \_\_eq\_\_() \_\_lt\_\_() \_\_le\_\_() \_\_gt\_\_() \_\_ge\_\_() ◦

```
@total_ordering
class Employee:
    ...

    def __eq__(self, other):
        return ((self.surname, self.name) == (other.surname, other.name))

    def __lt__(self, other):
        return ((self.surname, self.name) < (other.surname, other.name))
```

◦ \_\_lt\_\_() \_\_eq\_\_() \_\_gt\_\_() not \_\_lt\_\_() and not \_\_eq\_\_() ◦

total\_ordering Python 2.7 ◦

## Python 3.x reducefunctools

```
from functools import reduce
def factorial(n):
    return reduce(lambda a, b: (a*b), range(1, n+1))
```

## lru\_cache

@lru\_cache

```
@lru_cache(maxsize=None) # Boundless cache
def fibonacci(n):
    if n < 2:
        return n
    return fibonacci(n-1) + fibonacci(n-2)

>>> fibonacci(15)
```

fibonacci(3) fibonacci LRU fibonacci(3) 230 @lru\_cache

@lru\_cache

- maxsize maxsize LRU
- typed 3.3.3.03

```
>>> fib.cache_info()
CacheInfo(hits=13, misses=16, maxsize=None, cursize=16)
```

@lru\_cache

@lru\_cache Python @lru\_cache 3.2

## cmp\_to\_key

Python

-1,0+1

functools.cmp\_to\_key

```
>>> import functools
>>> import locale
>>> sorted(["A", "S", "F", "D"], key=functools.cmp_to_key(locale.strcoll))
['A', 'D', 'F', 'S']
```

Python

Functools <https://riptutorial.com/zh-TW/python/topic/2492/functools>

# 15: hashlib

hashlib。 FIPSSHA1SHA224SHA256SHA384SHA512。

## Examples

### MD5

。 FIPSSHA1SHA224SHA256SHA384SHA512FIPS 180-2RSAMD5Internet RFC 1321。

。 。 sha1()SHA1。

```
hash.sha1()
```

md5() sha1() sha224() sha256() sha384()sha512() 。

update()。 digest()hexdigest()digest() 。

```
hash.update(arg)
```

arg。 m.updatea; m.updatebm.updatea + b。

```
hash.digest()
```

update。 digest\_sizeASCII。

```
hash.hexdigest()
```

digest。 。

```
>>> import hashlib
>>> m = hashlib.md5()
>>> m.update("Nobody inspects")
>>> m.update(" the spammish repetition")
>>> m.digest()
'\xbbd\x9c\x83\xd\x1e\xa5\xc9\xd9\xde\xc9\xa1\x8d\xf0\xff\xe9'
>>> m.hexdigest()
'bb649c83dd1ea5c9d9dec9a18df0ffe9'
>>> m.digest_size
16
>>> m.block_size
64
```

```
hashlib.md5("Nobody inspects the spammish repetition").hexdigest()
'bb649c83dd1ea5c9d9dec9a18df0ffe9'
```

## OpenSSL

new() **OpenSSL** ◦ new() ◦

new() **OpenSSL**

```
>>> h = hashlib.new('ripemd160')
>>> h.update("Nobody inspects the spammish repetition")
>>> h.hexdigest()
'cc4a5ce1b3df48aec5d22d1f16b894a0b894eccc'
```

**hashlib** <https://riptutorial.com/zh-TW/python/topic/8980/hashlib>

# 16: Heapq

## Examples

heapqnlargest

```
import heapq

numbers = [1, 4, 2, 100, 20, 50, 32, 200, 150, 8]
print(heapq.nlargest(4, numbers)) # [200, 150, 100, 50]
```

nsmallest

```
print(heapq.nsmallest(4, numbers)) # [1, 2, 4, 8]
```

nlargestnsmallest◦ agepeople

```
people = [
    {'firstname': 'John', 'lastname': 'Doe', 'age': 30},
    {'firstname': 'Jane', 'lastname': 'Doe', 'age': 25},
    {'firstname': 'Janie', 'lastname': 'Doe', 'age': 10},
    {'firstname': 'Jane', 'lastname': 'Roe', 'age': 22},
    {'firstname': 'Johnny', 'lastname': 'Doe', 'age': 12},
    {'firstname': 'John', 'lastname': 'Roe', 'age': 45}
]

oldest = heapq.nlargest(2, people, key=lambda s: s['age'])
print(oldest)
# Output: [{'firstname': 'John', 'age': 45, 'lastname': 'Roe'}, {'firstname': 'John', 'age': 30, 'lastname': 'Doe'}]

youngest = heapq.nsmallest(2, people, key=lambda s: s['age'])
print(youngest)
# Output: [{'firstname': 'Janie', 'age': 10, 'lastname': 'Doe'}, {'firstname': 'Johnny', 'age': 12, 'lastname': 'Doe'}]
```

heap heap[0]

```
import heapq

numbers = [10, 4, 2, 100, 20, 50, 32, 200, 150, 8]

heapq.heapify(numbers)
print(numbers)
# Output: [2, 4, 10, 100, 8, 50, 32, 200, 150, 20]

heapq.heappop(numbers) # 2
print(numbers)
# Output: [4, 8, 10, 100, 20, 50, 32, 200, 150]

heapq.heappop(numbers) # 4
print(numbers)
```



```
# Output: [8, 20, 10, 100, 150, 50, 32, 200]
```

Heapq <https://riptutorial.com/zh-TW/python/topic/7489/heapq>

# 17: HTML

## Examples

### BeautifulSoup

#### HTML

```
<div>
  <label>Name:</label>
  John Smith
</div>
```

label“John Smith”。

label.[next\\_sibling](#)

```
from bs4 import BeautifulSoup

data = """
<div>
  <label>Name:</label>
  John Smith
</div>
"""

soup = BeautifulSoup(data, "html.parser")

label = soup.find("label", text="Name:")
print(label.next_sibling.strip())
```

John Smith ◦

### BeautifulSoupCSS

BeautifulSoup [CSS](#) ◦ [select\(\)](#) [select\\_one\(\)](#) ◦

```
from bs4 import BeautifulSoup

data = """
<ul>
  <li class="item">item1</li>
  <li class="item">item2</li>
  <li class="item">item3</li>
</ul>
"""

soup = BeautifulSoup(data, "html.parser")

for item in soup.select("li.item"):
    print(item.get_text())
```

```
item1
item2
item3
```

## PyQuery

pyqueryjquerypython。css。

```
from pyquery import PyQuery

html = """
<h1>Sales</h1>
<table id="table">
<tr>
  <td>Lorem</td>
  <td>46</td>
</tr>
<tr>
  <td>Ipsum</td>
  <td>12</td>
</tr>
<tr>
  <td>Dolor</td>
  <td>27</td>
</tr>
<tr>
  <td>Sit</td>
  <td>90</td>
</tr>
</table>
"""

doc = PyQuery(html)

title = doc('h1').text()

print title

table_data = []

rows = doc('#table > tr')
for row in rows:
    name = PyQuery(row).find('td').eq(0).text()
    value = PyQuery(row).find('td').eq(1).text()

    print "%s\t %s" % (name, value)
```

HTML <https://riptutorial.com/zh-TW/python/topic/1384/html>

---

# 18: ijson

ijson Python JSON。 Python JSON。 C。

## Examples

```
import ijson

def load_json(filename):
    with open(filename, 'r') as fd:
        parser = ijson.parse(fd)
        ret = {'builders': {}}
        for prefix, event, value in parser:
            if (prefix, event) == ('builders', 'map_key'):
                buildername = value
                ret['builders'][buildername] = {}
            elif prefix.endswith('.shortname'):
                ret['builders'][buildername]['shortname'] = value

        return ret

if __name__ == "__main__":
    load_json('allthethings.json')
```

## JSON

ijson <https://riptutorial.com/zh-TW/python/topic/8342/ijson>

# 19: Iterables/Iterators

## Examples

### Iterator vs Iterable vs Generator

**Iterable** ◦ `__iter__` ◦ `__getitem__` ◦ - `IndexError` ◦

Python `str` ◦ `__getitem__` ◦ **Iterable** ◦

**Iterator** ◦ `next(*object*)` ◦ `__next__` ◦ `StopIteration` ◦

**Iterable** ◦ `__iter__` ◦ `__next__` ◦

```
class MyIterable:

    def __iter__(self):

        return self

    def __next__(self):
        #code

#Classic iterable object in older versions of python, __getitem__ is still supported...
class MySequence:

    def __getitem__(self, index):

        if (condition):
            raise IndexError
        return (item)

#Can produce a plain `iterator` instance by using iter(MySequence())
```

`collections` ◦

### Python 2.x 2.3

```
import collections
>>> collections.Iterator()
>>> TypeError: Cant instantiate abstract class Iterator with abstract methods next
```

### Python 3.x 3.0

```
>>> TypeError: Cant instantiate abstract class Iterator with abstract methods __next__
```

### Python 2 Python 3

### Python 2.x 2.3

```
class MyIterable(object): #or collections.Iterator, which I'd recommend....
```

```

.....

def __iter__(self):

    return self

def next(self): #code

__next__ = next

```

```

ex1 = MyIterableClass()
ex2 = MySequence()

for (item) in (ex1): #code
for (item) in (ex2): #code

```

◦ ◦

## Iterable◦ Python

```

[1, 2, 3]      # list, iterate over items
(1, 2, 3)     # tuple
{1, 2, 3}     # set
{1: 2, 3: 4}  # dict, iterate over keys

```

```

def foo(): # foo isn't iterable yet...
    yield 1

res = foo() # ...but res already is

```

```

s = {1, 2, 3}

# get every element in s
for a in s:
    print a # prints 1, then 2, then 3

# copy into list
l1 = list(s) # l1 = [1, 2, 3]

# use list comprehension
l2 = [a * 2 for a in s if a > 2] # l2 = [6]

```

## iterable

```

a, = iterable

def foo():
    yield 1

a, = foo() # a = 1

nums = [1, 2, 3]
a, = nums # ValueError: too many values to unpack

```

```
iter() next() StopIteration
```

```
s = {1, 2} # or list or generator or even iterator
i = iter(s) # get iterator
a = next(i) # a = 1
b = next(i) # b = 2
c = next(i) # raises StopIteration
```

```
def gen():
    yield 1

iterable = gen()
for a in iterable:
    print a

# What was the first item of iterable? No way to get it now.
# Only to get a new iterator
gen()
```

**IterablesIterators** <https://riptutorial.com/zh-TW/python/topic/2343/iterablesiterators>

# 20: Itertools

- import itertools

## Examples

```
lst = [("a", 5, 6), ("b", 2, 4), ("a", 2, 5), ("c", 2, 6)]
```

```
def testGroupBy(lst):  
    groups = itertools.groupby(lst, key=lambda x: x[1])  
    for key, group in groups:  
        print(key, list(group))
```

```
testGroupBy(lst)
```

```
# 5 [('a', 5, 6)]  
# 2 [('b', 2, 4), ('a', 2, 5), ('c', 2, 6)]
```

### ◦ groupby◦

```
lst = [("a", 5, 6), ("b", 2, 4), ("a", 2, 5), ("c", 5, 6)]  
testGroupBy(lst)
```

```
# 5 [('a', 5, 6)]  
# 2 [('b', 2, 4), ('a', 2, 5)]  
# 5 [('c', 5, 6)]
```

### groupby◦ ◦ 525

```
lst = [("a", 5, 6), ("b", 2, 4), ("a", 2, 5), ("c", 2, 6)]  
groups = itertools.groupby(lst, key=lambda x: x[1])  
for key, group in sorted(groups):  
    print(key, list(group))
```

```
# 2 [('c', 2, 6)]  
# 5 []
```

```
groups = itertools.groupby(lst, key=lambda x: x[1])  
for key, group in sorted((key, list(group)) for key, group in groups):  
    print(key, list(group))
```

```
# 2 [('b', 2, 4), ('a', 2, 5), ('c', 2, 6)]  
# 5 [('a', 5, 6)]
```

### Itertools“islice”

```
results = fetch_paged_results() # returns a generator  
limit = 20 # Only want the first 20 results  
for data in itertools.islice(results, limit):  
    print(data)
```



```
def gen():
    n = 0
    while n < 20:
        n += 1
        yield n

for part in gen()[3]:
    print(part)
```

```
Traceback (most recent call last):
  File "gen.py", line 6, in <module>
    for part in gen()[3]:
TypeError: 'generator' object is not subscriptable
```

```
import itertools

def gen():
    n = 0
    while n < 20:
        n += 1
        yield n

for part in itertools.islice(gen(), 3):
    print(part)
```

start stopstep

```
itertools.islice(iterable, 1, 30, 3)
```

## itertools.product

◦

```
for x, y in itertools.product(xrange(10), xrange(10)):
    print x, y
```

```
for x in xrange(10):
    for y in xrange(10):
        print x, y
```

## python\*itertools.product◦

```
its = [xrange(10)] * 2
for x,y in itertools.product(*its):
    print x, y
```

◦

```
>>> from itertools import product
>>> a=[1,2,3,4]
>>> b=['a','b','c']
>>> product(a,b)
```

```
<itertools.product object at 0x0000000002712F78>
>>> for i in product(a,b):
...     print i
...
(1, 'a')
(1, 'b')
(1, 'c')
(2, 'a')
(2, 'b')
(2, 'c')
(3, 'a')
(3, 'b')
(3, 'c')
(4, 'a')
(4, 'b')
(4, 'c')
```

## itertools.count

◦ ...

```
for number in itertools.count():
    if number > 20:
        break
    print(number)
```

```
0
1
2
3
4
5
6
7
8
9
10
```

---

count() startstep

```
for number in itertools.count(start=10, step=4):
    print(number)
    if number > 20:
        break
```

```
10
14
18
22
```

## itertools.takewhile

itertools.takewhileFalse ◦

```
def is_even(x):
    return x % 2 == 0

lst = [0, 2, 4, 12, 18, 13, 14, 22, 23, 44]
result = list(itertools.takewhile(is_even, lst))

print(result)
```

[0, 2, 4, 12, 18] ◦

is\_even13 ◦ takewhileFalse◦

takewhile◦

```
def takewhile(predicate, iterable):
    for x in iterable:
        if predicate(x):
            yield x
        else:
            break
```

takewhiledropwhile◦

result = list(itertools.takewhile(is\_even, lst)) + list(itertools.dropwhile(is\_even, lst))

## itertools.dropwhile

itertools.dropwhileFalse◦

```
def is_even(x):
    return x % 2 == 0

lst = [0, 2, 4, 12, 18, 13, 14, 22, 23, 44]
result = list(itertools.dropwhile(is_even, lst))

print(result)
```

[13, 14, 22, 23, 44] ◦

takewhiledropwhile ◦

is\_even13 ◦ ◦

dropwhile◦

```
def dropwhile(predicate, iterable):
    iterable = iter(iterable)
    for x in iterable:
        if not predicate(x):
```

```

        yield x
    break
for x in iterable:
    yield x

```

`takewhiledropwhile`◦

```
result = list(itertools.takewhile(is_even, lst)) + list(itertools.dropwhile(is_even, lst))
```

`zip()` `itertools.zip_longest` `itertools.zip_longest`◦

```

from itertools import zip_longest
a = [i for i in range(5)] # Length is 5
b = ['a', 'b', 'c', 'd', 'e', 'f', 'g'] # Length is 7
for i in zip_longest(a, b):
    x, y = i # Note that zip longest returns the values as a tuple
    print(x, y)

```

`fillvalue`''

```

for i in zip_longest(a, b, fillvalue='Hogwash!'):
    x, y = i # Note that zip longest returns the values as a tuple
    print(x, y)

```

Python 2.62.7 `itertools.izip_longest` ◦

## Itertools

`itertools.combinations`*k*◦

*k*-◦

```

a = [1,2,3,4,5]
b = list(itertools.combinations(a, 2))
print b

```

```
[(1, 2), (1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5), (3, 4), (3, 5), (4, 5)]
```

*a*

**3**

```

a = [1,2,3,4,5]
b = list(itertools.combinations(a, 3))
print b

```

```

[(1, 2, 3), (1, 2, 4), (1, 2, 5), (1, 3, 4),
 (1, 3, 5), (1, 4, 5), (2, 3, 4), (2, 3, 5),
 (2, 4, 5), (3, 4, 5)]

```

`itertools.chain`◦

```
from itertools import chain
a = (x for x in ['1', '2', '3', '4'])
b = (x for x in ['x', 'y', 'z'])
' '.join(chain(a, b))
```

```
'1 2 3 4 x y z'
```

**classmethod** chain.from\_iterable ◦

```
' '.join(chain.from_iterable([a,b]))
```

chainchain.from\_iterable ◦

## itertools.repeat

n

```
>>> import itertools
>>> for i in itertools.repeat('over-and-over', 3):
...     print(i)
over-and-over
over-and-over
over-and-over
```

## Python 3.x 3.2

accumulate ◦

```
>>> import itertools as it
>>> import operator

>>> list(it.accumulate([1,2,3,4,5]))
[1, 3, 6, 10, 15]

>>> list(it.accumulate([1,2,3,4,5], func=operator.mul))
[1, 2, 6, 24, 120]
```

cycle ◦

```
>>> import itertools as it
>>> it.cycle('ABCD')
A B C D A B C D A B C D ...
```

◦

```
>>> # Iterate over each element in cycle for a fixed range
>>> cycle_iterator = it.cycle('abc123')
>>> [next(cycle_iterator) for i in range(0, 10)]
['a', 'b', 'c', '1', '2', '3', 'a', 'b', 'c', '1']
```

## itertools.permutations

itertools.permutations

```
a = [1,2,3]
list(itertools.permutations(a))
# [(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)]

list(itertools.permutations(a, 2))
[(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)]
```

aset

```
a = [1,2,1]
list(itertools.permutations(a))
# [(1, 2, 1), (1, 1, 2), (2, 1, 1), (2, 1, 1), (1, 1, 2), (1, 2, 1)]

set(itertools.permutations(a))
# {(1, 1, 2), (1, 2, 1), (2, 1, 1)}
```

itertools <https://riptutorial.com/zh-TW/python/topic/1564/itertools>

# 21: JSON

◦

json

JSON	
int	INT

jsonNaN Infinity-InfinityJSON◦

	JSON
intfloatint / float-derived Enums	

NaN Infinity-Infinity allow\_nan=False◦ ValueError◦

◦ functools.partial◦

```
# my_json module

import json
from functools import partial

def serialise_object(obj):
    # Do something to produce json-serialisable data
    return dict_obj

dump = partial(json.dump, default=serialise_object)
dumps = partial(json.dumps, default=serialise_object)
```

jsonobject\_hookparse\_float◦ python ◦

```
# my_json module

import json
```

```

from functools import partial

def deserialise_object(dict_obj):
    # Do something custom
    return obj

def deserialise_float(str_obj):
    # Do something custom
    return obj

load = partial(json.load, object_hook=deserialise_object, parse_float=deserialise_float)
loads = partial(json.loads, object_hook=deserialise_object, parse_float=deserialise_float)

```

json/json.JSONEncoder/json.JSONDecoder ◦ ◦ cls ◦ functools.partial **cls**

```

# my_json module

import json
from functools import partial

class MyEncoder(json.JSONEncoder):
    # Do something custom

class MyDecoder(json.JSONDecoder):
    # Do something custom

dump = partial(json.dump, cls=MyEncoder)
dumps = partial(json.dumps, cls=MyEncoder)
load = partial(json.load, cls=MyDecoder)
loads = partial(json.loads, cls=MyDecoder)

```

## Examples

### Python dictJSON

```

import json
d = {
    'foo': 'bar',
    'alice': 1,
    'wonderland': [1, 2, 3]
}
json.dumps(d)

```

```
'{"wonderland": [1, 2, 3], "foo": "bar", "alice": 1}'
```

### JSONPython dict

```

import json
s = '{"wonderland": [1, 2, 3], "foo": "bar", "alice": 1}'
json.loads(s)

```

```
{u'alice': 1, u'foo': u'bar', u'wonderland': [1, 2, 3]}
```



dJSONfilenamefilename◦

```
import json

d = {
    'foo': 'bar',
    'alice': 1,
    'wonderland': [1, 2, 3]
}

with open(filename, 'w') as f:
    json.dump(d, f)
```

JSONfilenamefilename◦

```
import json

with open(filename, 'r') as f:
    d = json.load(f)
```

## `load` vs `load` `dump` vs `loads`

jsonunicode◦ s◦ StringIO◦

```
import json

data = {"foo": "bar", "baz": []}
json_string = json.dumps(data)
# u'{"foo": "bar", "baz": []}'
json.loads(json_string)
# {"foo": "bar", "baz": []}
```

```
import json

from io import StringIO

json_file = StringIO()
data = {"foo": "bar", "baz": []}
json.dump(data, json_file)
json_file.seek(0) # Seek back to the start of the file before reading
json_file_content = json_file.read()
# u'{"foo": "bar", "baz": []}'
json_file.seek(0) # Seek back to the start of the file before reading
json.load(json_file)
# {"foo": "bar", "baz": []}
```

json◦ ◦ 0

```
import json

json_file_path = './data.json'
data = {"foo": "bar", "baz": []}

with open(json_file_path, 'w') as json_file:
```

```

    json.dump(data, json_file)

with open(json_file_path) as json_file:
    json_file_content = json_file.read()
    # u'{"foo": "bar", "baz": []}'

with open(json_file_path) as json_file:
    json.load(json_file)
    # {"foo": "bar", "baz": []}

```

## jsonjsonpysparkjson-per-line

```

# loading from a file
data = [json.loads(line) for line in open(file_path).splitlines()]

# dumping to a file
with open(file_path, 'w') as json_file:
    for item in data:
        json.dump(item, json_file)
        json_file.write('\n')

```

## `json.tool` JSON

### JSON“foo.json”

```

{"foo": {"bar": {"baz": 1}}}

```

```

$ python -m json.tool foo.json
{
  "foo": {
    "bar": {
      "baz": 1
    }
  }
}

```

### STDOUTBash

```

$ cat foo.json | python -m json.tool

```

## JSON

```

>>> data = {"cats": [{"name": "Tubbs", "color": "white"}, {"name": "Pepper", "color": "black"}]}

```

### JSON

```

>>> print(json.dumps(data))
{"cats": [{"name": "Tubbs", "color": "white"}, {"name": "Pepper", "color": "black"}]}

```

indent

```
>>> print(json.dumps(data, indent=2))
{
  "cats": [
    {
      "name": "Tubbs",
      "color": "white"
    },
    {
      "name": "Pepper",
      "color": "black"
    }
  ]
}
```

◦

```
>>> print(json.dumps(data, sort_keys=True))
{"cats": [{"color": "white", "name": "Tubbs"}, {"color": "black", "name": "Pepper"}]}
```

' , ' : ' '

```
>>> print(json.dumps(data, separators=(',', ':')))
{"cats":[{"name":"Tubbs","color":"white"}, {"name":"Pepper","color":"black"}]}
```

## JSON

```
import json
from datetime import datetime
data = {'datetime': datetime(2016, 9, 26, 4, 44, 0)}
print(json.dumps(data))
```

TypeError: datetime.datetime(2016, 9, 26, 4, 44) is not JSON serializable ◦

### datetime

```
class DatetimeJSONEncoder(json.JSONEncoder):
    def default(self, obj):
        try:
            return obj.isoformat()
        except AttributeError:
            # obj has no isoformat method; let the builtin JSON encoder handle it
            return super(DatetimeJSONEncoder, self).default(obj)
```

json.dumps

```
encoder = DatetimeJSONEncoder()
print(encoder.encode(data))
# prints {"datetime": "2016-09-26T04:44:00"}
```

JSON <https://riptutorial.com/zh-TW/python/topic/272/json>

# 22: kivy - NUIPython

NUINUI。

KivyPython。 ◦

## Examples

kivy

1. **app**
2. ◦
3. **run** ◦

```
from kivy.app import App
from kivy.uix.label import Label

class Test(App):
    def build(self):
        return Label(text='Hello world')

if __name__ == '__main__':
    Test().run()
```

```
from kivy.app import App
```

**app** ◦ your\_installtion\_directory / kivy / app.py

```
from kivy.uix.label import Label
```

**uxLabel** ◦ uxyour\_installation\_directory / kivy / uix / ◦

```
class Test(App):
```

◦ **app** ◦

```
def build(self):
```

**app** ◦ ◦

```
return Label(text='Hello world')
```

◦ **LabelHello world** ◦

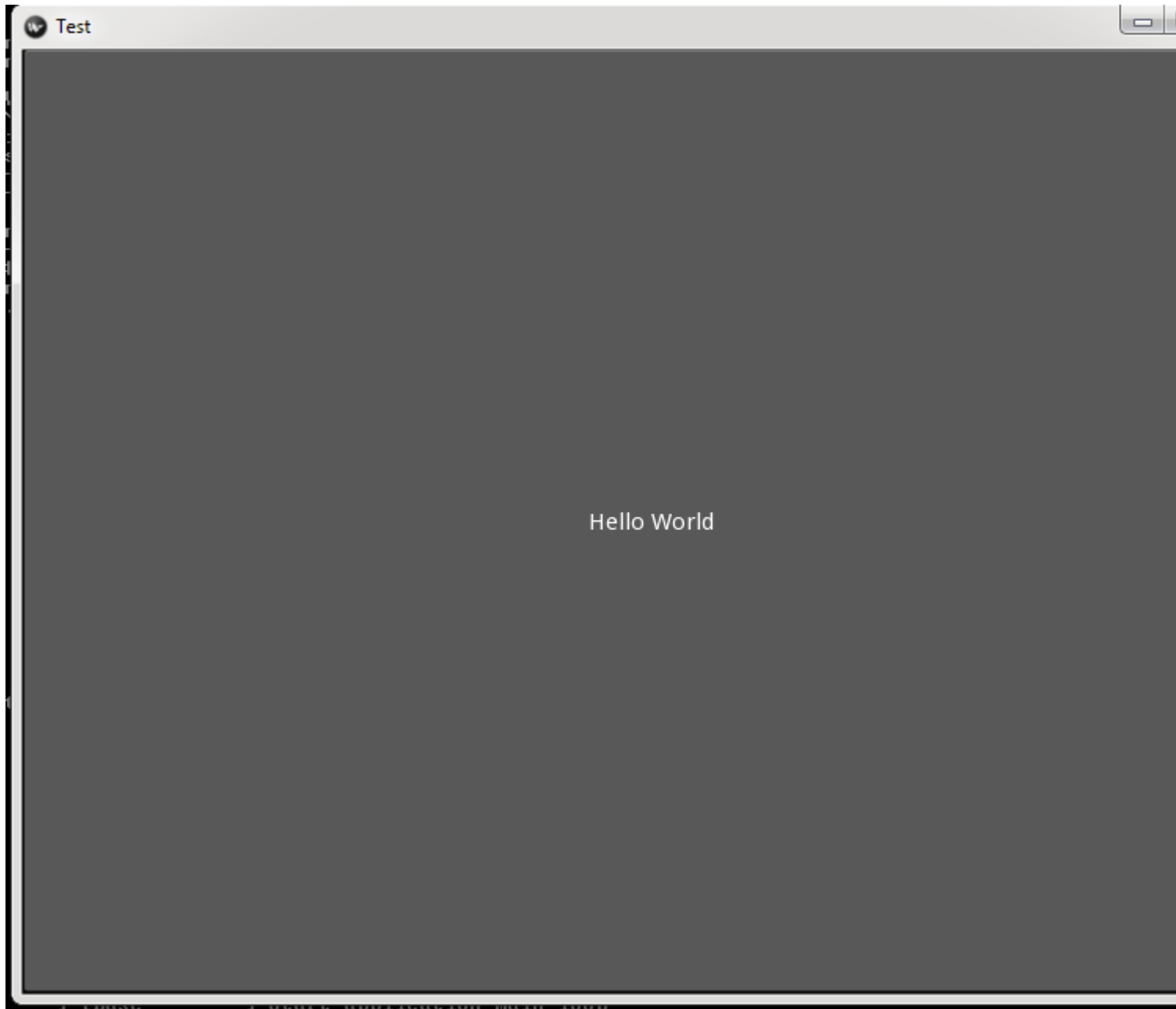
```
if __name__ == '__main__':
```

python。

```
Test().run()
```

Test。 apprun。

。



kivy - NUIPython <https://riptutorial.com/zh-TW/python/topic/10743/kivy---nuipython>

# 23: Neo4jCypherPy2Neo

## Examples

```
from py2neo import authenticate, Graph, Node, Relationship
authenticate("localhost:7474", "neo4j", "<pass>")
graph = Graph()
```

Neo4jlocalhost7474。

graphpythonneo4j。 \_\_init\_\_。

## Neo4j

```
results = News.objects.todays_news()
for r in results:
    article = graph.merge_one("NewsArticle", "news_id", r)
    article.properties["title"] = results[r]['news_title']
    article.properties["timestamp"] = results[r]['news_timestamp']
    article.push()
    [...]
```

graph.merge\_one。

timestampneo4jdate。 '05 -06-1989'

article.push()neo4j。 ◦

## Neo4j

```
results = News.objects.todays_news()
for r in results:
    article = graph.merge_one("NewsArticle", "news_id", r)
    if 'LOCATION' in results[r].keys():
        for loc in results[r]['LOCATION']:
            loc = graph.merge_one("Location", "name", loc)
            try:
                rel = graph.create_unique(Relationship(article, "about_place", loc))
            except Exception, e:
                print e
```

create\_unique。 ◦ ◦

## 1

```
def get_autocomplete(text):
    query = """
    start n = node(*) where n.name =~ '(?i)%s.*' return n.name,labels(n) limit 10;
    """
```

```

query = query % (text)
obj = []
for res in graph.cypher.execute(query):
    # print res[0],res[1]
    obj.append({'name':res[0],'entity_type':res[1]})
return res

```

cypher<sub>textname</sub>◦

## 2

```

def search_news_by_entity(location,timestamp):
    query = """
MATCH (n)-[]->(l)
where l.name='%s' and n.timestamp='%s'
RETURN n.news_id limit 10
"""

    query = query % (location,timestamp)

    news_ids = []
    for res in graph.cypher.execute(query):
        news_ids.append(str(res[0]))

    return news_ids

```

(l) (n) ◦

## Cypher

```

MATCH (n)-[]->(l)
where l.name='Donald Trump'
RETURN n.date,count(*) order by n.date

```

/5◦

```

MATCH (n:NewsArticle)-[]->(l)
where l.name='Donald Trump'
MATCH (n:NewsArticle)-[]->(m)
with m,count(n) as num where num>5
return labels(m)[0],(m.name), num order by num desc limit 10

```

Neo4jCypherPy2Neo <https://riptutorial.com/zh-TW/python/topic/5841/neo4jcypherpy2neo>



---

# 24: os.path

- ◦ Unicode◦
  - os.path.joina\* p
  - os.path.basenamep
  - os.path.dirnamep
  - os.path.splitp
  - os.path.splitextp

## Examples

pythonos

```
import os
os.path.join('a', 'b', 'c')
```

os.path◦

Windows

```
>>> os.path.join('a', 'b', 'c')
'a\b\c'
```

Unix

```
>>> os.path.join('a', 'b', 'c')
'a/b/c'
```

os.path.abspath

```
>>> os.getcwd()
'/Users/rsaftoiu/tmp'
>>> os.path.abspath('foo')
'/Users/rsaftoiu/tmp/foo'
>>> os.path.abspath('../foo')
'/Users/rsaftoiu/foo'
>>> os.path.abspath('/foo')
'/foo'
```

```
>>> p = os.path.join(os.getcwd(), 'foo.txt')
>>> p
'/Users/rsaftoiu/tmp/foo.txt'
>>> os.path.dirname(p)
'/Users/rsaftoiu/tmp'
>>> os.path.basename(p)
'foo.txt'
>>> os.path.split(os.getcwd())
('/Users/rsaftoiu/tmp', 'foo.txt')
```

```
>>> os.path.splitext(os.path.basename(p))
('foo', '.txt')
```

```
os.path.abspath(os.path.join(PATH_TO_GET_THE_PARENT, os.pardir))
```

◦

```
path = '/home/john/temp'
os.path.exists(path)
#this returns false if path doesn't exist or if the path is a broken symbolic link
```

◦

```
dirname = '/home/john/python'
os.path.isdir(dirname)
```

```
filename = dirname + 'main.py'
os.path.isfile(filename)
```

```
symlink = dirname + 'some_sym_link'
os.path.islink(symlink)
```

```
mount_path = '/home'
os.path.ismount(mount_path)
```

**os.path** <https://riptutorial.com/zh-TW/python/topic/1380/os-path>

# 25: os

◦

•

◦ os.path.sep◦	
0700	

## Examples

```
os.mkdir('newdir')
```

mode

```
os.mkdir('newdir', mode=0700)
```

```
os.getcwd()
```

```
print(os.getcwd())
```

os◦

```
os.name
```

## Python 3

- posix
- nt
- ce
- java

[sys.platform](#)

path

```
os.rmdir(path)
```

```
os.remove()◦ OSError
```

## POSIX

◦ os.readlink

```
print(os.readlink(path_to_symlink))
```

```
os.chmod(path, mode)
```

mode◦

## makedirs -

```
└─ dir1
  └─ subdir1
    └─ subdir2
```

dir2subdir1subdir2◦

```
import os

os.makedirs("./dir2/subdir1")
os.makedirs("./dir2/subdir2")
```

```
└─ dir1
  └─ subdir1
    └─ subdir2
└─ dir2
  └─ subdir1
    └─ subdir2
```

dir2subdir1◦

## os.mkdir dir2◦

```
os.mkdir("./dir2/subdir1")
OSError: [Errno 2] No such file or directory: './dir2/subdir1'
```

## os.makedirs◦

```
OSError: [Errno 17] File exists: './dir2/subdir1'
```

◦

```
try:
    os.makedirs("./dir2/subdir1")
except OSError:
    if not os.path.isdir("./dir2/subdir1"):
        raise

try:
    os.makedirs("./dir2/subdir2")
except OSError:
    if not os.path.isdir("./dir2/subdir2"):
        raise
```

os <https://riptutorial.com/zh-TW/python/topic/4127/os>

# 26: Pandas Transform

## Examples

nm

```
orders_df = pd.DataFrame()
orders_df['customer_id'] = [1,1,1,1,1,2,2,3,3,3,3,3]
orders_df['order_id'] = [1,1,1,2,2,3,3,4,5,6,6,6]
orders_df['item'] = ['apples', 'chocolate', 'chocolate', 'coffee', 'coffee', 'apples',
                    'bananas', 'coffee', 'milkshake', 'chocolate', 'strawberry',
                    'strawberry']

# And this is how the dataframe looks like:
print(orders_df)
#   customer_id  order_id      item
# 0           1          1    apples
# 1           1          1  chocolate
# 2           1          1  chocolate
# 3           1          2    coffee
# 4           1          2    coffee
# 5           2          3    apples
# 6           2          3  bananas
# 7           3          4    coffee
# 8           3          5  milkshake
# 9           3          6  chocolate
# 10          3          6  strawberry
# 11          3          6  strawberry
```

◦  
◦

## pandas transform

```
# First, we define the function that will be applied per customer_id
count_number_of_orders = lambda x: len(x.unique())

# And now, we can transform each group using the logic defined above
orders_df['number_of_orders_per_cient'] = (
    orders_df.groupby(['customer_id'])['order_id'] # Create a separate group for each
    .transform(count_number_of_orders)) # Apply the function to each group
customer_id & select the order_id
seperatly

# Inspecting the results ...
print(orders_df)
#   customer_id  order_id      item  number_of_orders_per_cient
# 0           1          1    apples                2
# 1           1          1  chocolate                2
# 2           1          1  chocolate                2
# 3           1          2    coffee                 2
# 4           1          2    coffee                 2
```

```

# 5          2          3      apples          1
# 6          2          3      bananas          1
# 7          3          4        coffee          3
# 8          3          5    milkshake          3
# 9          3          6      chocolate          3
# 10         3          6    strawberry          3
# 11         3          6    strawberry          3

```

## transform

◦ ◦

```

# Create a dummy dataframe
orders_df = pd.DataFrame()
orders_df['customer_id'] = [1,1,1,1,1,2,2,3,3,3,3,3]
orders_df['order_id'] = [1,1,1,2,2,3,3,4,5,6,6,6]
orders_df['item'] = ['apples', 'chocolate', 'chocolate', 'coffee', 'coffee', 'apples',
                    'bananas', 'coffee', 'milkshake', 'chocolate', 'strawberry',
                    'strawberry']

# Let's try to see if the items were ordered more than once in each orders

# First, we define a fuction that will be applied per group
def multiple_items_per_order(_items):
    # Apply .duplicated, which will return True is the item occurs more than once.
    multiple_item_bool = _items.duplicated(keep=False)
    return(multiple_item_bool)

# Then, we transform each group according to the defined function
orders_df['item_duplicated_per_order'] = (                                # Put the results into a new
column                                                                    column
                                orders_df                                # Take the orders dataframe
                                .groupby(['order_id'])['item']          # Create a seperate group for
each order_id & select the item
                                .transform(multiple_items_per_order))    # Apply the defined function to
each group separately

# Inspecting the results ...
print(orders_df)
#   customer_id  order_id      item  item_duplicated_per_order
# 0            1         1    apples                        False
# 1            1         1  chocolate                        True
# 2            1         1  chocolate                        True
# 3            1         2    coffee                         True
# 4            1         2    coffee                         True
# 5            2         3    apples                        False
# 6            2         3    bananas                       False
# 7            3         4    coffee                        False
# 8            3         5  milkshake                       False
# 9            3         6    chocolate                    False
# 10           3         6  strawberry                       True
# 11           3         6  strawberry                       True

```

Pandas Transform <https://riptutorial.com/zh-TW/python/topic/10947/pandas-transform->

---

## 27: Pickle

- pickle.dumpobjectfileprotocol#Tutize
- pickle.loadfile
- pickle.dumpsobjectprotocol
- pickle.loadsbuffer



---

## Pickleable

◦

- None True False
- 
- 
- tuple **S** list **S** set **S**dict
- 
- 
- ◦ `__dict__` `__getstate__()` `__getstate__()` ◦

Python ◦

---

**pickle**

◦ Internet◦

## Examples

### Pickle

picklePython◦ ◦ ◦

dump()load()◦

---

```
import pickle

# An arbitrary collection of objects supported by pickle.
data = {
```



```

'a': [1, 2.0, 3, 4+6j],
'b': ("character string", b"byte string"),
'c': {None, True, False}
}

with open('data.pickle', 'wb') as f:
    # Pickle the 'data' dictionary using the highest protocol available.
    pickle.dump(data, f, pickle.HIGHEST_PROTOCOL)

```

```

import pickle

with open('data.pickle', 'rb') as f:
    # The protocol version used is detected automatically, so we do not
    # have to specify it.
    data = pickle.load(f)

```

## picklebyte

dumpsloadsdumpload ◦

```

serialized_data = pickle.dumps(data, pickle.HIGHEST_PROTOCOL)
# type(serialized_data) is bytes

deserialized_data = pickle.loads(serialized_data)
# deserialized_data == data

```

◦ ◦

\_\_getstate\_\_ \_\_getstate\_\_ ◦ ◦

oposite \_\_setstate\_\_ \_\_getstate\_\_ ◦

```

class A(object):
    def __init__(self, important_data):
        self.important_data = important_data

        # Add data which cannot be pickled:
        self.func = lambda: 7

        # Add data which should never be pickled, because it expires quickly:
        self.is_up_to_date = False

    def __getstate__(self):
        return [self.important_data] # only this is needed

    def __setstate__(self, state):
        self.important_data = state[0]

        self.func = lambda: 7 # just some hard-coded unpicklable function

```

```
self.is_up_to_date = False # even if it was before pickling
```

```
>>> a1 = A('very important')
>>>
>>> s = pickle.dumps(a1) # calls a1.__getstate__()
>>>
>>> a2 = pickle.loads(s) # calls a1.__setstate__(['very important'])
>>> a2
<__main__.A object at 0x0000000002742470>
>>> a2.important_data
'very important'
>>> a2.func()
7
```

[self.important\_data] ◦ \_\_getstate\_\_ ◦ getstate\_\_ ◦ setstate\_\_ ◦ **opposite** ◦ {'important\_data': self.important\_data} ◦

a2pickle.loadsA.\_\_init\_\_ A.\_\_setstate\_\_ ◦ init\_\_ ◦

Pickle <https://riptutorial.com/zh-TW/python/topic/2606/pickle>

# 28: pipPyPI

pipPython Package IndexPython。

- pip <command> [options]<command>
  - 
  - 
  - 
  - 
  - 
  - PyPI
  - Zip
  - pybundles

pip LinuxpythonC WindowsVisual Studio / Visual C ++。

Python	Visual Studio	Visual C ++
2.6 - 3.2	Visual Studio 2008	Visual C ++ 9.0
3.3 - 3.4	Visual Studio 2010	Visual C ++ 10.0
3.5	Visual Studio 2015	Visual C ++ 14.0

[wiki.python.org](http://wiki.python.org)

## Examples

SomePackage

```
$ pip install SomePackage
```

```
$ pip install SomePackage==1.0.4
```

```
$ pip install SomePackage>=1.0.4
```

Linux / Unixsudo

```
$ pip install -r requirements.txt
```

pip install 。

freeze

```
$ pip freeze
```

```
$ pip uninstall SomePackage
```

## ``pip``

```
$ pip list
# example output
docutils (0.9.1)
Jinja2 (2.6)
Pygments (1.5)
Sphinx (1.1.2)
```

```
$ pip list --outdated
# example output
docutils (Current: 0.9.1 Latest: 0.10)
Sphinx (Current: 1.1.2 Latest: 1.1.3)
```

```
$ pip install --upgrade SomePackage
```

SomePackage◦ **pip**◦

## pip

```
$ pip install --upgrade pip
```

## Unix

```
$ python -m pip install --upgrade pip
```

## Windows◦

## Linux

pip◦ **Linux**

```
pip list --outdated --local | grep -v '^-\e' | cut -d = -f 1 | xargs -n1 pip install -U
```

virtualenv◦ pip install -U◦ ◦

## Windows

pip◦ **Windows**

```
for /F "delims= " %i in ('pip list --outdated --local') do pip install -U %i
```

virtualenv pip install -U

## requirements.txt

pip freeze requirements.txt

```
pip freeze > requirements.txt
```

requirements.txt

## virtualenvrequirements.txt

pip freeze requirements.txt

```
pip freeze --local > requirements.txt
```

--local virtualenv

## Python pip

Python 3 Python 2 Python Python 2.3

### Python 2

```
pip install [package]
```

```
pip2 install [package]
```

### Python 3

```
pip3 install [package]
```

### python

```
\path\to\that\python.exe -m pip install some_package # on Windows OR  
/usr/bin/python2.5 -m pip install some_package # on OS-X/Linux
```

OS-X / Linux / Unix python python sudo

Windows python - python -c "import sys; print(sys.path);" python python -c "import sys; print(sys.path);" py -3.5 -c "import sys; print(sys.path);" pip --version pip --version

Windows python 2 python 3 python 3.4 python py

```
py -3 -m pip install -U some_package # Install/Upgrade some_package to the latest python 3  
py -3.3 -m pip install -U some_package # Install/Upgrade some_package to python 3.3 if present  
py -2 -m pip install -U some_package # Install/Upgrade some_package to the latest python 2 -  
64 bit if present
```

```
py -2.7-32 -m pip install -U some_package # Install/Upgrade some_package to python 2.7 - 32 bit if present
```

pythonpython virtualenvvenv python。

pythonPython Package Index。 Windowsvcvarsall.bat。

CC ++PythonPyPWindows。

[Christoph Gohlke](#)。 -cp **NN**-pythonWindows 32python **win64** -win32-64python-win\_amd64 - pythonPython 34-cp **34**-。 pippypi。

pythonWindowspythonpython。

Python 2.7Visual Studio 2008Python 3.33.4Visual Studio 2010Python 3.5+Visual Studio 2015。

- “ [Python 2.7Visual C ++](#) ”Microsoft
- “ [Windows SDK for Windows 7.NET Framework 4](#) ”v7.1Microsoft
- [Visual Studio 2015](#) **CC ++** - **8**。

。

pip - 。

[pypiChristopp](#)pythonpython。 pythonpython。 python - [PIL 6python 3](#)。

bug pypi。 python。

Pip。 V0.98V1.0-rc1 pip installV0.98 - --pip install --pre --pip install --pre --upgrade 。

pipgithub。

1. pip install *path / to / downloaded / file* pip install pip。
2. pip pip install *URL / of / package / repository* - --trusted-host - --client-cert/--proxy。

```
> py -3.5-32 -m venv demo-pip
> demo-pip\Scripts\activate.bat
> python -m pip install -U pip
Collecting pip
  Using cached pip-9.0.1-py2.py3-none-any.whl
Installing collected packages: pip
  Found existing installation: pip 8.1.1
  Uninstalling pip-8.1.1:
    Successfully uninstalled pip-8.1.1
  Successfully installed pip-9.0.1
> pip install git+https://github.com/sphinx-doc/sphinx/
Collecting git+https://github.com/sphinx-doc/sphinx/
  Cloning https://github.com/sphinx-doc/sphinx/ to c:\users\steve-
~1\appdata\local\temp\pip-04yn9hpp-build
Collecting six>=1.5 (from Sphinx==1.7.dev20170506)
  Using cached six-1.10.0-py2.py3-none-any.whl
Collecting Jinja2>=2.3 (from Sphinx==1.7.dev20170506)
```

```

Using cached Jinja2-2.9.6-py2.py3-none-any.whl
Collecting Pygments>=2.0 (from Sphinx==1.7.dev20170506)
Using cached Pygments-2.2.0-py2.py3-none-any.whl
Collecting docutils>=0.11 (from Sphinx==1.7.dev20170506)
Using cached docutils-0.13.1-py3-none-any.whl
Collecting snowballstemmer>=1.1 (from Sphinx==1.7.dev20170506)
Using cached snowballstemmer-1.2.1-py2.py3-none-any.whl
Collecting babel!=2.0,>=1.3 (from Sphinx==1.7.dev20170506)
Using cached Babel-2.4.0-py2.py3-none-any.whl
Collecting alabaster<0.8,>=0.7 (from Sphinx==1.7.dev20170506)
Using cached alabaster-0.7.10-py2.py3-none-any.whl
Collecting imagesize (from Sphinx==1.7.dev20170506)
Using cached imagesize-0.7.1-py2.py3-none-any.whl
Collecting requests>=2.0.0 (from Sphinx==1.7.dev20170506)
Using cached requests-2.13.0-py2.py3-none-any.whl
Collecting typing (from Sphinx==1.7.dev20170506)
Using cached typing-3.6.1.tar.gz
Requirement already satisfied: setuptools in f:\toolbuild\temp\demo-pip\lib\site-packages
(from Sphinx==1.7.dev20170506)
Collecting sphinxcontrib-websupport (from Sphinx==1.7.dev20170506)
Downloading sphinxcontrib_websupport-1.0.0-py2.py3-none-any.whl
Collecting colorama>=0.3.5 (from Sphinx==1.7.dev20170506)
Using cached colorama-0.3.9-py2.py3-none-any.whl
Collecting MarkupSafe>=0.23 (from Jinja2>=2.3->Sphinx==1.7.dev20170506)
Using cached MarkupSafe-1.0.tar.gz
Collecting pytz>=0a (from babel!=2.0,>=1.3->Sphinx==1.7.dev20170506)
Using cached pytz-2017.2-py2.py3-none-any.whl
Collecting sqlalchemy>=0.9 (from sphinxcontrib-websupport->Sphinx==1.7.dev20170506)
Downloading SQLAlchemy-1.1.9.tar.gz (5.2MB)
100% |#####| 5.2MB 220kB/s
Collecting whoosh>=2.0 (from sphinxcontrib-websupport->Sphinx==1.7.dev20170506)
Downloading Whoosh-2.7.4-py2.py3-none-any.whl (468kB)
100% |#####| 471kB 1.1MB/s
Installing collected packages: six, MarkupSafe, Jinja2, Pygments, docutils,
snowballstemmer, pytz, babel, alabaster, imagesize, requests, typing, sqlalchemy, whoosh,
sphinxcontrib-websupport, colorama, Sphinx
Running setup.py install for MarkupSafe ... done
Running setup.py install for typing ... done
Running setup.py install for sqlalchemy ... done
Running setup.py install for Sphinx ... done
Successfully installed Jinja2-2.9.6 MarkupSafe-1.0 Pygments-2.2.0 Sphinx-1.7.dev20170506
alabaster-0.7.10 babel-2.4.0 colorama-0.3.9 docutils-0.13.1 imagesize-0.7.1 pytz-2017.2
requests-2.13.0 six-1.10.0 snowballstemmer-1.2.1 sphinxcontrib-websupport-1.0.0 sqlalchemy-
1.1.9 typing-3.6.1 whoosh-2.7.4

```

## URLgit+

3. git mercurial **DVCS**  

```

pip install ///-requires.txt pip install -r requires.txtpython setup.py
install git git pull origin masterpip uninstall package-namegit checkout

```

**pipPyPI** <https://riptutorial.com/zh-TW/python/topic/1781/pip-pypi>

# 29: PostgreSQL

## Examples

PostgreSQL◦ psycopg2◦

### pip

```
pip install psycopg2
```

my\_databasemy\_table◦

ID
1

psycopg2◦

```
import psycopg2

# Establish a connection to the existing database 'my_database' using
# the user 'my_user' with password 'my_password'
con = psycopg2.connect("host=localhost dbname=my_database user=my_user password=my_password")

# Create a cursor
cur = con.cursor()

# Insert a record into 'my_table'
cur.execute("INSERT INTO my_table(id, first_name, last_name) VALUES (2, 'Jane', 'Doe');")

# Commit the current transaction
con.commit()

# Retrieve all records from 'my_table'
cur.execute("SELECT * FROM my_table;")
results = cur.fetchall()

# Close the database connection
con.close()

# Print the results
print(results)

# OUTPUT: [(1, 'John', 'Doe'), (2, 'Jane', 'Doe')]
```

PostgreSQL <https://riptutorial.com/zh-TW/python/topic/3374/postgresql>



# 30: py.test

## Examples

### py.test

py.test Python `pip`

```
pip install pytest
```

projectroot/module/code.py

```
# projectroot/module/code.py
def add(a, b):
    return a + b
```

projectroot/tests/test\_code.py `test_`

```
# projectroot/tests/test_code.py
from module import code

def test_add():
    assert code.add(1, 2) == 3
```

projectrootpy.test

```
# ensure we have the modules
$ touch tests/__init__.py
$ touch module/__init__.py
$ py.test
===== test session starts
=====
platform darwin -- Python 2.7.10, pytest-2.9.2, py-1.4.31, pluggy-0.3.1
rootdir: /projectroot, inifile:
collected 1 items

tests/test_code.py .

===== 1 passed in 0.01 seconds
=====
```

```
# projectroot/tests/test_code.py
from module import code

def test_add__failing():
    assert code.add(10, 11) == 33
```

```
$ py.test
```

```

===== test session starts
=====
platform darwin -- Python 2.7.10, pytest-2.9.2, py-1.4.31, pluggy-0.3.1
rootdir: /projectroot, inifile:
collected 1 items

tests/test_code.py F

===== FAILURES
=====
_____ test_add_failing

def test_add_failing():
>     assert code.add(10, 11) == 33
E     assert 21 == 33
E     + where 21 = <function add at 0x105d4d6e0>(10, 11)
E     +   where <function add at 0x105d4d6e0> = code.add

tests/test_code.py:5: AssertionError
===== 1 failed in 0.01 seconds
=====

```

◦ ◦ ◦

```

# projectroot/module/stuff.py
class Stuff(object):
    def prep(self):
        self.foo = 1
        self.bar = 2

```

```

# projectroot/tests/test_stuff.py
import pytest
from module import stuff

def test_foo_updates():
    my_stuff = stuff.Stuff()
    my_stuff.prep()
    assert 1 == my_stuff.foo
    my_stuff.foo = 30000
    assert my_stuff.foo == 30000

def test_bar_updates():
    my_stuff = stuff.Stuff()
    my_stuff.prep()
    assert 2 == my_stuff.bar
    my_stuff.bar = 42
    assert 42 == my_stuff.bar

```

Stuff◦ ◦

```

# projectroot/tests/test_stuff.py
import pytest
from module import stuff

```

```

def get_prepped_stuff():
    my_stuff = stuff.Stuff()
    my_stuff.prep()
    return my_stuff

def test_foo_updates():
    my_stuff = get_prepped_stuff()
    assert 1 == my_stuff.foo
    my_stuff.foo = 30000
    assert my_stuff.foo == 30000

def test_bar_updates():
    my_stuff = get_prepped_stuff()
    assert 2 == my_stuff.bar
    my_stuff.bar = 42
    assert 42 == my_stuff.bar

```

```
my_stuff = get_prepped_stuff()
```

## py.test

◦ ◦

```
get_prepped_stuff ◦ ◦ @pytest.fixture ◦ fixture ◦
```

```

@pytest.fixture
def prepped_stuff():
    my_stuff = stuff.Stuff()
    my_stuff.prep()
    return my_stuff

```

◦ ◦ `py.test fixture fixture` ;

```

def test_foo_updates(prepped_stuff):
    my_stuff = prepped_stuff
    assert 1 == my_stuff.foo
    my_stuff.foo = 30000
    assert my_stuff.foo == 30000

def test_bar_updates(prepped_stuff):
    my_stuff = prepped_stuff
    assert 2 == my_stuff.bar
    my_stuff.bar = 42
    assert 42 == my_stuff.bar

```

◦ `my_stuff = prepped_stuff` ◦

```

def test_foo_updates(prepped_stuff):
    assert 1 == prepped_stuff.foo
    prepped_stuff.foo = 30000
    assert prepped_stuff.foo == 30000

```

```
def test_bar_updates(prepped_stuff):
    assert 2 == prepped_stuff.bar
    prepped_stuff.bar = 42
    assert 42 == prepped_stuff.bar
```

.....°

°

## Stuff°

```
# projectroot/module/stuff.py
class Stuff(object):
    def prep(self):
        self.foo = 1
        self.bar = 2

    def finish(self):
        self.foo = 0
        self.bar = 0
```

## fixtures° ° fixture°

```
@pytest.fixture
def prepped_stuff(request): # we need to pass in the request to use finalizers
    my_stuff = stuff.Stuff()
    my_stuff.prep()
    def fin(): # finalizer function
        # do all the cleanup here
        my_stuff.finish()
    request.addfinalizer(fin) # register fin() as a finalizer
    # you can do more setup here if you really want to
    return my_stuff
```

## ° **yield fixture**° returnyieldyield° yield\_fixturepy.test°

```
@pytest.yield_fixture
def prepped_stuff(): # it doesn't need request now!
    # do setup
    my_stuff = stuff.Stuff()
    my_stuff.prep()
    # setup is done, pass control to the test functions
    yield my_stuff
    # do cleanup
    my_stuff.finish()
```

## py.test fixtureyield

py.test <https://riptutorial.com/zh-TW/python/topic/7054/py-test>

# 31: pyaudio

PyAudio / OPortAudioPython. PyAudioPython. PyAudio

1.pyPortAudio / fastaudioPortAudio v18 APIPython.

2.tkSnackTcl / TkPython.

stream\_callback. stream\_callback

1.

2.

3.PortAudiopaAbort.

Stream.readStream.write.

PortAudio

[http://portaudio.com/docs/v19-](http://portaudio.com/docs/v19-doxydocs/portaudio_8h.html#a8a60fb2a5ec9cbade3f54a9c978e2710)

[doxydocs/portaudio\\_8h.html#a8a60fb2a5ec9cbade3f54a9c978e2710](http://portaudio.com/docs/v19-doxydocs/portaudio_8h.html#a8a60fb2a5ec9cbade3f54a9c978e2710)

## Examples

I/O.

```
"""PyAudio Example: Play a wave file (callback version)."""

import pyaudio
import wave
import time
import sys

if len(sys.argv) < 2:
    print("Plays a wave file.\n\nUsage: %s filename.wav" % sys.argv[0])
    sys.exit(-1)

wf = wave.open(sys.argv[1], 'rb')

# instantiate PyAudio (1)
p = pyaudio.PyAudio()

# define callback (2)
def callback(in_data, frame_count, time_info, status):
    data = wf.readframes(frame_count)
    return (data, pyaudio.paContinue)

# open stream using callback (3)
stream = p.open(format=p.get_format_from_width(wf.getsampwidth()),
                channels=wf.getnchannels(),
                rate=wf.getframerate(),
                output=True,
                stream_callback=callback)

# start the stream (4)
stream.start_stream()
```

```

# wait for stream to finish (5)
while stream.is_active():
    time.sleep(0.1)

# stop stream (6)
stream.stop_stream()
stream.close()
wf.close()

# close PyAudio (7)
p.terminate()

```

/PyAudio2。 PyAudio。 callback(<input\_data>, <frame\_count>, <time\_info>, <status\_flag>)  
frame\_count/。

**pyaudio.Stream.start\_stream 4pyaudio.paComplete 。**

5。

**I/O.**

msgstr““PyAudio。 ”“

```

import pyaudio
import wave
import sys

CHUNK = 1024

if len(sys.argv) < 2:
    print("Plays a wave file.\n\nUsage: %s filename.wav" % sys.argv[0])
    sys.exit(-1)

wf = wave.open(sys.argv[1], 'rb')

# instantiate PyAudio (1)
p = pyaudio.PyAudio()

# open stream (2)
stream = p.open(format=p.get_format_from_width(wf.getsampwidth()),
                channels=wf.getnchannels(),
                rate=wf.getframerate(),
                output=True)

# read data
data = wf.readframes(CHUNK)

# play stream (3)
while len(data) > 0:
    stream.write(data)
    data = wf.readframes(CHUNK)

# stop stream (4)
stream.stop_stream()
stream.close()

# close PyAudio (5)

```

```
p.terminate()
```

**PyAudio****PyAudio** 1PyAudioportaudio。

**pyaudio.PyAudio.open** 2。 **pyaudio.Stream**。

**pyaudio.Stream.write****pyaudio.Stream.read**。 3

“ ”**pyaudio.Stream.write****pyaudio.Stream.read**//。 “”

**pyaudio.Stream.stop\_stream**/**pyaudio.Stream.close**。 4

**pyaudio.PyAudio.terminate** 5portaudio

**pyaudio** <https://riptutorial.com/zh-TW/python/topic/10627/pyaudio>

## 32: pyautogui

pyautogui 0.0 Python

### Examples

```
size()          #gave you the size of the screen
position()      #return current position of mouse
moveTo(200,0,duration=1.5)  #move the cursor to (200,0) position with 1.5 second delay

moveRel()      #move the cursor relative to your current position.
click(337,46)   #it will click on the position mention there
dragRel()      #it will drag the mouse relative to position
pyautogui.displayMousePosition()  #gave you the current mouse position but should be done
on terminal.
```

```
typewrite('')  #this will type the string on the screen where current window has focused.
typewrite(['a','b','left','left','X','Y'])
pyautogui.KEYBOARD_KEYS  #get the list of all the keyboard_keys.
pyautogui.hotkey('ctrl','o')  #for the combination of keys to enter.
```

### ScreenShot

```
.screenshot('c:\\path')  #get the screenshot.
.locateOnScreen('c:\\path')  #search that image on screen and get the coordinates for you.
locateCenterOnScreen('c:\\path')  #get the coordinate for the image on screen.
```

pyautogui <https://riptutorial.com/zh-TW/python/topic/9432/pyautogui>



---

# 33: pygame

PygamePython。 <http://www.pygame.org/>。

- `pygame.mixer.init(22050, -16, 2, 4096)`
- `pygame.mixer.pre_init`
- `pygame.mixer.quit`
- `pygame.mixer.get_init`
- `pygame.mixer.stop`
- `pygame.mixer.pause`
- `pygame.mixer.unpause`
- `pygame.mixer.fadeout`
- `pygame.mixer.set_num_channels`
- `pygame.mixer.get_num_channels`
- `pygame.mixer.set_reserved`
- `pygame.mixer.find_channel`
- `pygame.mixer.get_busy`

```
FalseTrue find_channel() TrueFalse
```

## Examples

### pygame

pip

```
pip install pygame
```

conda

```
conda install -c tlatorre pygame=1.9.2
```

[http //www.pygame.org/download.shtml](http://www.pygame.org/download.shtml)

。

<http://www.pygame.org/>

### Pygame

`pygame.mixerpygame。` `mixer`15。

pygame.init()pygame.mixer ◦

◦ ◦ ◦

```
pygame.mixer.init(frequency=22050, size=-16, channels=2, buffer=4096)
```

pygame.mixer.get\_init() TrueFalse ◦ /pygame.mixer.quit() ◦ ◦

pygame.mixer.pause() ◦ pygame.mixer.unpause() ◦ pygame.mixer.fadeout() ◦ ◦

◦ **8** ◦ pygame.mixer.set\_num\_channels() ◦ ◦ ◦

pygame.mixer.get\_channels(count) pygame.mixer.get\_channels(count) ◦ ◦

pygame.mixer.set\_reserved(count) ◦ ◦ ◦

pygame.mixer.find\_channel(force) ◦ ◦ force **False** None ◦ force **true** ◦

[pygame](https://riptutorial.com/zh-TW/python/topic/8761/pygame) <https://riptutorial.com/zh-TW/python/topic/8761/pygame>

# 34: Pyglet

PygletPython。 ◦ [pyglet.org] [1]。 [1]http://pyglet.org

## Examples

### PygletHello World

```
import pyglet
window = pyglet.window.Window()
label = pyglet.text.Label('Hello, world',
                           font_name='Times New Roman',
                           font_size=36,
                           x=window.width//2, y=window.height//2,
                           anchor_x='center', anchor_y='center')

@window.event
def on_draw():
    window.clear()
    label.draw()
pyglet.app.run()
```

### Pyglet

#### Python

#### Python 2

```
pip install pyglet
```

#### Python 3

```
pip3 install pyglet
```

### Pyglet

```
sound = pyglet.media.load(sound.wav)
sound.play()
```

### PygletOpenGL

```
import pyglet
from pyglet.gl import *

win = pyglet.window.Window()

@win.event()
def on_draw():
    #OpenGL goes here. Use OpenGL as normal.
```

```
pyglet.app.run()
```

## PygletOpenGL

```
import pyglet
from pyglet.gl import *

win = pyglet.window.Window()
glClear(GL_COLOR_BUFFER_BIT)

@win.event
def on_draw():
    glBegin(GL_POINTS)
    glVertex2f(x, y) #x is desired distance from left side of window, y is desired distance
    from bottom of window
    #make as many vertexes as you want
    glEnd
```

GL\_POINTSGL\_LINE\_LOOP ◦

Pyglet <https://riptutorial.com/zh-TW/python/topic/8208/pyglet>

# 35: PyInstaller - Python

- `pyinstaller [options] script [script ...] | spec`

PyInstallerpython◦ python◦ numpyDjangoOpenCv◦

- PyinstallerPython 2.7Python 3.3+
- PyinstallerWindowsLinuxMac OS X◦
  - WindowsLinux◦ WindowsPyInstallerWindows

## Examples

Pyinstallerpython◦ pip

```
pip install pyinstaller
```

### Windows

Windows [pywin32](#)[pypiwin32](#)◦ [pippyinstaller](#)◦

### Mac OS X

PyInstallerMac OS XPython 2.7◦ PythonPyQTNumPyMatplotlib◦ [MacPortsHomebrew](#)◦

pip[PyPI](#)◦

[PyInstaller](#)*develop*◦

```
setup.py◦ python setup.py installPyInstaller◦
```

```
pyinstaller◦
```

```
pyinstaller --version◦ pyinstaller◦
```

## Pyinstaller

```
pyinstaller myfile.py
```

Pyinstaller

- **myfile.spec**`myfile.py`
- `myfile.py`**build**
- `myfile.py`**dist**
- 

**dist**

```
pyinstaller◦ ◦
```

'dist \ myfile \ myfile.exe'。

**PyInstaller**myscript.py myscript myscript**windows**myscript.exe ◦  
zip◦

-D--onedir--onedir

pyinstaller myscript.py -D

—————

◦ ◦  
◦ ◦

—————

◦  
/◦

pyinstaller myscript.py -F

-F--onefile ◦ myscript.exe◦

◦ ◦

**PyInstaller - Python** <https://riptutorial.com/zh-TW/python/topic/2289/pyinstaller----python>

# 36: Python HTTP Server

## Examples

### HTTP

#### Python 2.x 2.3

```
python -m SimpleHTTPServer 9000
```

#### Python 3.x 3.0

```
python -m http.server 9000
```

9000◦

8000◦

-msys.path.py◦

### localhostPython

```
import sys
import BaseHTTPServer
from SimpleHTTPServer import SimpleHTTPRequestHandler

HandlerClass = SimpleHTTPRequestHandler
ServerClass = BaseHTTPServer.HTTPServer
Protocol = "HTTP/1.0"

if sys.argv[1:]:
    port = int(sys.argv[1])
else:
    port = 8000
server_address = ('127.0.0.1', port)


HandlerClass.protocol_version = Protocol
httpd = ServerClass(server_address, HandlerClass)


sa = httpd.socket.getsockname()
print "Serving HTTP on", sa[0], "port", sa[1], "..."
httpd.serve_forever()
```

### Documents library

files

Name

 factory.py

 facade.py

## Web

### Python 2.x 2.3

```
import SimpleHTTPServer
import SocketServer

PORT = 8000

handler = SimpleHTTPServer.SimpleHTTPRequestHandler
httpd = SocketServer.TCPServer(("localhost", PORT), handler)
print "Serving files at port {}".format(PORT)
httpd.serve_forever()
```

### Python 3.x 3.0

```
import http.server
import socketserver

PORT = 8000

handler = http.server.SimpleHTTPRequestHandler
httpd = socketserver.TCPServer("", PORT), handler)
print("serving at port", PORT)
httpd.serve_forever()
```

SocketServer◦

SocketServerTCPServerTCP◦ IP◦

SimpleHTTPServerSimpleHTTPRequestHandler◦

◦

## HTTP

### Python 2.x 2.3

python -m SimpleHTTPServer 8000

### Python 3.x 3.0

python -m http.server 8000

'-m"sys.path".py'◦

[localhost8000](http://localhost:8000)



## Directory listing for /

---

- [facade.py](#)
  - [factory.py](#)
  - [server.py](#)
- 

## SimpleHTTPServerAPI

```
python -m SimpleHTTPServer 9000
```

SimpleHTTPServer <https://hg.python.org/cpython/file/2.7/Lib/SimpleHTTPServer.py>

BaseHTTPServer <https://hg.python.org/cpython/file/2.7/Lib/BaseHTTPServer.py>

Python9000SimpleHTTPServer ◦ SimpleHTTPServer

```
def test (HandlerClass = SimpleHTTPRequestHandler,
         ServerClass = BaseHTTPServer.HTTPServer):
    BaseHTTPServer.test (HandlerClass, ServerClass)

if __name__ == '__main__':
    test ()
```

ServerClass ◦ BaseHTTPServer.test

```
def test (HandlerClass = BaseHTTPRequestHandler,
         ServerClass = HTTPServer, protocol="HTTP/1.0"):
    """Test the HTTP request handler class.

    This runs an HTTP server on port 8000 (or the first command line
    argument).

    """

    if sys.argv[1:]:
        port = int(sys.argv[1])
    else:
        port = 8000
    server_address = ('', port)

    HandlerClass.protocol_version = protocol
    httpd = ServerClass (server_address, HandlerClass)

    sa = httpd.socket.getsockname ()
    print "Serving HTTP on", sa[0], "port", sa[1], "..."
    httpd.serve_forever ()
```

◦ ◦ ◦

SocketServer

```

+-----+
| BaseServer |
+-----+
    |
    v
+-----+      +-----+
| TCPServer |----->| UnixStreamServer |
+-----+      +-----+
    |
    v
+-----+      +-----+
| UDPServer |----->| UnixDatagramServer |
+-----+      +-----+

```

<https://hg.python.org/cpython/file/2.7/Lib/BaseHTTPServer.py>

<https://hg.python.org/cpython/file/2.7/Lib/SocketServer.py>。

## BaseHTTPRequestHandlerGETPOSTPUT

```

# from BaseHTTPServer import BaseHTTPRequestHandler, HTTPServer # python2
from http.server import BaseHTTPRequestHandler, HTTPServer # python3
class HandleRequests(BaseHTTPRequestHandler):
    def _set_headers(self):
        self.send_response(200)
        self.send_header('Content-type', 'text/html')
        self.end_headers()

    def do_GET(self):
        self._set_headers()
        self.wfile.write("received get request")

    def do_POST(self):
        '''Reads post request body'''
        self._set_headers()
        content_len = int(self.headers.getheader('content-length', 0))
        post_body = self.rfile.read(content_len)
        self.wfile.write("received post request:<br>{}".format(post_body))

    def do_PUT(self):
        self.do_POST()

host = ''
port = 80
HTTPServer((host, port), HandleRequests).serve_forever()

```

curl

```

$ curl http://localhost/
received get request%

$ curl -X POST http://localhost/
received post request:<br>%

$ curl -X PUT http://localhost/
received post request:<br>%

```

```
$ echo 'hello world' | curl --data-binary @- http://localhost/  
received post request:<br>hello world
```

Python HTTP Server <https://riptutorial.com/zh-TW/python/topic/4247/python-http-server>

# 37: Python Lex-Yacc

PLYlexyaccPython。

- 1.
2. [Github](#)

## Examples

### PLY

python2 / 3PLY

1. 。
2. zip
3. ply-3.10
4. python setup.py install

PLY。 pythonimport ply.lex。

pipPLY。

### “HelloWorld” PLY -

PLY。

```
from ply import lex
import ply.yacc as yacc

tokens = (
    'PLUS',
    'MINUS',
    'TIMES',
    'DIV',
    'LPAREN',
    'RPAREN',
    'NUMBER',
)

t_ignore = ' \t'

t_PLUS = r'\+'
t_MINUS = r'\-'
t_TIMES = r'\*'
t_DIV = r'\/'
t_LPAREN = r'\('
t_RPAREN = r'\)'

def t_NUMBER( t ) :
    r'[0-9]+'
    t.value = int( t.value )
    return t
```

```

def t_newline( t ):
    r'\n+'
    t.lexer.lineno += len( t.value )

def t_error( t ):
    print("Invalid Token:",t.value[0])
    t.lexer.skip( 1 )

lexer = lex.lex()

precedence = (
    ( 'left', 'PLUS', 'MINUS' ),
    ( 'left', 'TIMES', 'DIV' ),
    ( 'nonassoc', 'UMINUS' )
)

def p_add( p ) :
    'expr : expr PLUS expr'
    p[0] = p[1] + p[3]

def p_sub( p ) :
    'expr : expr MINUS expr'
    p[0] = p[1] - p[3]

def p_expr2uminus( p ) :
    'expr : MINUS expr %prec UMINUS'
    p[0] = - p[2]

def p_mult_div( p ) :
    '''expr : expr TIMES expr
    | expr DIV expr'''

    if p[2] == '*' :
        p[0] = p[1] * p[3]
    else :
        if p[3] == 0 :
            print("Can't divide by 0")
            raise ZeroDivisionError('integer division by 0')
        p[0] = p[1] / p[3]

def p_expr2NUM( p ) :
    'expr : NUMBER'
    p[0] = p[1]

def p_parens( p ) :
    'expr : LPAREN expr RPAREN'
    p[0] = p[2]

def p_error( p ) :
    print("Syntax error in input!")

parser = yacc.yacc()

res = parser.parse("-4*-(3-5)") # the input
print(res)

```

calc.py°

-4 \* - (3 - 5)◦

## 1Lex

1 ◦

◦

```
import ply.lex as lex

# List of token names. This is always required
tokens = [
    'NUMBER',
    'PLUS',
    'MINUS',
    'TIMES',
    'DIVIDE',
    'LPAREN',
    'RPAREN',
]

# Regular expression rules for simple tokens
t_PLUS = r'\+'
t_MINUS = r'\-'
t_TIMES = r'\*'
t_DIVIDE = r'\/'
t_LPAREN = r'\('
t_RPAREN = r'\)'

# A regular expression rule with some action code
def t_NUMBER(t):
    r'\d+'
    t.value = int(t.value)
    return t

# Define a rule so we can track line numbers
def t_newline(t):
    r'\n+'
    t.lexer.lineno += len(t.value)

# A string containing ignored characters (spaces and tabs)
t_ignore = ' \t'

# Error handling rule
def t_error(t):
    print("Illegal character '%s'" % t.value[0])
    t.lexer.skip(1)

# Build the lexer
lexer = lex.lex()

# Give the lexer some input
lexer.input(data)

# Tokenize
while True:
    tok = lexer.token()
    if not tok:
        break # No more input
```

```
print(tok)
```

calcllex.py ◦ **Yacc** ◦

---

1. import ply.lex

2. tokens ◦ ◦

```
tokens = [  
    'NUMBER',  
    'PLUS',  
    'MINUS',  
    'TIMES',  
    'DIVIDE',  
    'LPAREN',  
    'RPAREN',  
]
```

tokens ◦

3. ◦ t\_ ◦

- t\_PLUS = r'\+'

- ◦

```
def t_NUMBER(t):  
    r'\d+'  
    t.value = int(t.value)  
    return t
```

doc ◦ LexToken ◦

```
@TOKEN(identifier)          # identifier is a string holding the regex  
def t_ID(t):  
    ...          # actions
```

- LexTokent

1. t.type 'NUMBER' 'PLUS' ◦ t.type t\_ ◦
2. t.value **lexeme**
3. t.lineno t.lineno ◦ t\_newline t\_newline ◦

```
def t_newline(t):  
    r'\n+'  
    t.lexer.lineno += len(t.value)
```

4. t.lexpos ◦

-

- `t_ignore_`.

```
def t_COMMENT(t):
    r'\#.*'
    pass
    # No return value. Token discarded
```

...

```
t_ignore_COMMENT = r'\#.*'
```

◦ ◦

```
t_ignore = "<characters to ignore>"
```

```
t_ignore_COMMENT = r'\#.*'
t_ignore = ' \t' # ignores spaces and tabs
```

- `lex`

- 1. ◦

- 2. ◦

=== ◦

- ◦ `t.type.value`

```
literals = [ '+', '-', '*', '/' ]
```

```
literals = "+-*/"
```

◦ ◦

```
literals = [ '{', '}' ]
```

```
def t_lbrace(t):
    r'\{'
    t.type = '{' # Set token type to the expected literal (ABSOLUTE MUST if this
is a literal)
    return t
```

- `t_error`.

```
# Error handling rule
def t_error(t):
    print("Illegal character '%s'" % t.value[0])
    t.lexer.skip(1) # skip the illegal token (don't process it)
```

`t.lexer.skip(n)` ◦



4. `lexer = lex.lex()` `lexer = lex.lex()` ◦

◦

```
import ply.lex as lex
class MyLexer(object):
    ... # everything relating to token rules and error handling comes here as
usual

    # Build the lexer
    def build(self, **kwargs):
        self.lexer = lex.lex(module=self, **kwargs)

    def test(self, data):
        self.lexer.input(data)
        for token in self.lexer.token():
            print(token)

    # Build the lexer and try it out

m = MyLexer()
m.build() # Build the lexer
m.test("3 + 4") #
```

`lexer.input(data)` **data**

`lexer.token()` ◦

```
for i in lexer:
    print(i)
```

## 2Yacc

### 1 - Context Free GrammarsCFG◦ ◦ LALR◦

```
# Yacc example

import ply.yacc as yacc

# Get the token map from the lexer. This is required.
from calclex import tokens

def p_expression_plus(p):
    'expression : expression PLUS term'
    p[0] = p[1] + p[3]

def p_expression_minus(p):
    'expression : expression MINUS term'
    p[0] = p[1] - p[3]

def p_expression_term(p):
    'expression : term'
    p[0] = p[1]

def p_term_times(p):
```

```

    'term : term TIMES factor'
    p[0] = p[1] * p[3]

def p_term_div(p):
    'term : term DIVIDE factor'
    p[0] = p[1] / p[3]

def p_term_factor(p):
    'term : factor'
    p[0] = p[1]

def p_factor_num(p):
    'factor : NUMBER'
    p[0] = p[1]

def p_factor_expr(p):
    'factor : LPAREN expression RPAREN'
    p[0] = p[2]

# Error rule for syntax errors
def p_error(p):
    print("Syntax error in input!")

# Build the parser
parser = yacc.yacc()

while True:
    try:
        s = raw_input('calc > ')
    except EOFError:
        break
    if not s: continue
    result = parser.parse(s)
    print(result)

```

- ◦ ◦ p◦ p[i]

```

def p_expression_plus(p):
    'expression : expression PLUS term'
    #   ^           ^           ^   ^
    # p[0]         p[1]       p[2] p[3]

    p[0] = p[1] + p[3]

```

- p[i] "p.value" PLUS+
- p[0] None p[-1]p[3] p p[-1]

p\_ ◦

- p\_error(p) yacc / bison yyerror◦
- ◦

```

def p_binary_operators(p):
    '''expression : expression PLUS term
                  | expression MINUS term
    term         : term TIMES factor
                  | term DIVIDE factor'''
    if p[2] == '+':
        p[0] = p[1] + p[3]
    elif p[2] == '-':
        p[0] = p[1] - p[3]
    elif p[2] == '*':
        p[0] = p[1] * p[3]
    elif p[2] == '/':
        p[0] = p[1] / p[3]

```

- 

```

def p_binary_operators(p):
    '''expression : expression '+' term
                  | expression '-' term
    term         : term '*' factor
                  | term '/' factor'''
    if p[2] == '+':
        p[0] = p[1] + p[3]
    elif p[2] == '-':
        p[0] = p[1] - p[3]
    elif p[2] == '*':
        p[0] = p[1] * p[3]
    elif p[2] == '/':
        p[0] = p[1] / p[3]

```

- 

- '''symbol : '''
- start = 'foo' foo◦
- ◦

```

precedence = (
    ('nonassoc', 'LESSTHAN', 'GREATERTHAN'), # Nonassociative operators
    ('left', 'PLUS', 'MINUS'),
    ('left', 'TIMES', 'DIVIDE'),
    ('right', 'UMINUS'), # Unary minus operator
)

```

- nonassoc◦ a < b < ca < b◦

- parser.outyacc◦ /◦

Python Lex-Yacc <https://riptutorial.com/zh-TW/python/topic/10510/python-lex-yacc>

# 38: Python

Python `threading` `multiprocessing` `API` ◦

## Examples

```
from __future__ import print_function
import threading
def counter(count):
    while count > 0:
        print("Count value", count)
        count -= 1
    return

t1 = threading.Thread(target=countdown, args=(10,))
t1.start()
t2 = threading.Thread(target=countdown, args=(20,))
t2.start()
```

Python `CPython` `GIL` `nterpreter` ◦

Python

[David Beazley](#) [YouTube](#) [Python](#)

```
from __future__ import print_function
import multiprocessing

def countdown(count):
    while count > 0:
        print("Count value", count)
        count -= 1
    return

if __name__ == "__main__":
    p1 = multiprocessing.Process(target=countdown, args=(10,))
    p1.start()

    p2 = multiprocessing.Process(target=countdown, args=(20,))
    p2.start()

    p1.join()
    p2.join()
```

◦ Python `VM` `GIL` ◦

`Process.start` `target` `target` ◦ `Process.join` `p1` `p2` ◦

python

- Windows `spawn` ◦
-

unix3.3fork◦

forkPOSIX◦

- unix3.4+fork forkserverspawnmultiprocessing.set\_start\_method◦ forkserverspawn◦

## POSIX fork

forkexecve◦

forkMainThread◦

- MainThreadLock◦ fork◦

pythonmultiprocessingmacOSnumpy / accelerated◦

◦ Queue◦

```
import multiprocessing
import queue
my_Queue=multiprocessing.Queue()
#Creates a queue with an undefined maximum size
#this can be dangerous as the queue becomes increasingly large
#it will take a long time to copy data to/from each read/write thread
```

tryexceptblockempty◦ queue.Empty==Truequeue.Empty==False Iftry'if'◦

```
import multiprocessing
import queue
'''Import necessary Python standard libraries, multiprocessing for classes and queue for the
queue exceptions it provides'''
def Queue_Iftry_Get(get_queue, default=None, use_default=False, func=None, use_func=False):
    '''This global method for the Iftry block is provided for it's reuse and
standard functionality, the if also saves on performance as opposed to catching
the exception, which is expensive.
    It also allows the user to specify a function for the outgoing data to use,
and a default value to return if the function cannot return the value from the queue'''
    if get_queue.empty():
        if use_default:
            return default
    else:
        try:
            value = get_queue.get_nowait()
        except queue.Empty:
            if use_default:
                return default
        else:
            if use_func:
                return func(value)
            else:
                return value
def Queue_Iftry_Put(put_queue, value):
    '''This global method for the Iftry block is provided because of its reuse
and
standard functionality, the If also saves on performance as opposed to catching
the exception, which is expensive.
    Return True if placing value in the queue was successful. Otherwise, false'''
```

```
if put_queue.full():
    return False
else:
    try:
        put_queue.put_nowait(value)
    except queue.Full:
        return False
    else:
        return True
```

Python <https://riptutorial.com/zh-TW/python/topic/3357/python>

# 39: PythonMutable vs ImmutableHashable

## Examples

Python◦ ◦

## Immutable

◦ ◦

frozensets◦

id◦ ◦ id◦ ◦ .....

```
>>> a = 1
>>> id(a)
140128142243264
>>> a += 2
>>> a
3
>>> id(a)
140128142243328
```

13 .....◦ ◦

```
>>> stack = "Overflow"
>>> stack
'Overflow'
>>> id(stack)
140128123955504
>>> stack += " rocks!"
>>> stack
'Overflow rocks!'
```

```
>>> id(stack)
140128123911472
```

◦ stack◦ ◦

```
>>> stack = "Stack"
>>> stackoverflow = stack + "Overflow"
>>> id(stack)
140128069348184
>>> id(stackoverflow)
140128123911480
```

◦

```
s = ""
for i in range(1, 1000):
    s += str(i)
    s += ", "
```

## Mutables

◦ ◦  
◦

id◦

```
>>> b = bytearray(b'Stack')
>>> b
bytearray(b'Stack')
>>> b = bytearray(b'Stack')
>>> id(b)
140128030688288
>>> b += b'Overflow'
>>> b
bytearray(b'StackOverflow')
>>> id(b)
140128030688288
```

*ascii* ◦

**bytearray**id ◦ ◦

◦ ◦

```
>>> c = b
>>> c += b' rocks! '
>>> c
bytearray(b'StackOverflow rocks!')
```

...

```
>>> b
bytearray(b'StackOverflow rocks!')
```

Waiiit.....

```
>>> id(c) == id(b)
True
```

◦ cb◦ cb◦

```
>>> ll = [ [] ]*4 # Create a list of 4 lists to contain our results
>>> ll
```



```
[[], [], [], []]
>>> l1[0].append(23) # Add result 23 to first list
>>> l1
[[23], [23], [23], [23]]
>>> # Oops...
```

◦ ◦ ◦

```
>>> def list_add3(lin):
    lin += [3]
    return lin

>>> a = [1, 2, 3]
>>> b = list_add3(a)
>>> b
[1, 2, 3, 3]
>>> a
[1, 2, 3, 3]
```

lin◦ lina◦ lina◦ lina◦ ab◦

◦

```
>>> def tuple_add3(tin):
    tin += (3,)
    return tin

>>> a = (1, 2, 3)
>>> b = tuple_add3(a)
>>> b
(1, 2, 3, 3)
>>> a
(1, 2, 3)
```

tina◦ ◦ tina◦ tin◦

```
>>> def yoda(prologue, sentence):
    sentence.reverse()
    prologue += " ".join(sentence)
    return prologue

>>> focused = ["You must", "stay focused"]
>>> saying = "Yoda said: "
>>> yoda_sentence = yoda(saying, focused)
```

reverse◦

saying focused

[Python Mutable vs Immutable Hashable](https://riptutorial.com/zh-TW/python/topic/9182/python-mutable-vs-immutable-hashable-) <https://riptutorial.com/zh-TW/python/topic/9182/python-mutable-vs-immutable-hashable->

# 40: Python

- ◦ lambdamapreduce◦

## Examples

### Lambda

lambda◦ lambda◦ ◦ ◦

```
s=lambda x:x*x  
s(2)    =>4
```

Map◦ ◦ ◦

```
name_lengths = map(len, ["Mary", "Isla", "Sam"])  
print(name_lengths)    =>[4, 4, 3]
```

Reduce◦ ◦

◦ ◦

```
total = reduce(lambda a, x: a + x, [0, 1, 2, 3, 4])  
print(total)    =>10
```

◦ True◦

```
arr=[1,2,3,4,5,6]  
[i for i in filter(lambda x:x>4,arr)]    # outputs[5,6]
```

Python <https://riptutorial.com/zh-TW/python/topic/9552/python>

---

# 41: PythonShell



## Examples

### ssh

```
from paramiko import client
ssh = client.SSHClient() # create a new SSHClient object
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy()) #auto-accept unknown host keys
ssh.connect(hostname, username=username, port=port, password=password) #connect with a host
stdin, stdout, stderr = ssh.exec_command(command) # submit a command to ssh
print stdout.channel.recv_exit_status() #tells the status 1 - job failed
```

PythonShell <https://riptutorial.com/zh-TW/python/topic/5709/pythonshell>

# 42: python

## Examples

```
from string import Template

data = dict(item = "candy", price = 8, qty = 2)

# define the template
t = Template("Simon bought $qty $item for $price dollar")
print(t.substitute(data))
```

```
Simon bought 2 candy for 8 dollar
```

\$。

。 priceqty。 。 。

“\$”。

```
from string import Template

class MyOtherTemplate(Template):
    delimiter = "#"

data = dict(id = 1, name = "Ricardo")
t = MyOtherTemplate("My name is #name and I have the id: #id")
print(t.substitute(data))
```

de docs

python <https://riptutorial.com/zh-TW/python/topic/6029/python>

# 43: Pythonpyserial

- ser.read= 1
- ser.readline
- ser.write

GNU / Linux/ dev / ttyUSB0WindowsCOM3。

int9600

50,75,110,134,150,200,300,600,1200,1800,2400,4800,9600,19200,38400,57600,115200

pyserial

## Examples

```
import serial
#Serial takes these two parameters: serial device and baudrate
ser = serial.Serial('/dev/ttyUSB0', 9600)
```

```
import serial
#Serial takes two parameters: serial device and baudrate
ser = serial.Serial('/dev/ttyUSB0', 9600)
```

```
data = ser.read()
```

```
data = ser.read(size=5)
```

。

```
data = ser.readline()
```

。

```
#for python2.7
data = ser.read(ser.inWaiting())

#for python3
ser.read(ser.inWaiting)
```

```
python -m serial.tools.list_ports
```

```
from serial.tools import list_ports
```

```
list_ports.comports() # Outputs list of available serial ports
```

Python shell.

Pythonpyserial <https://riptutorial.com/zh-TW/python/topic/5744/python-pyserial->

# 44: Python

## Examples

except◦

```
try:
    res = get_result()
    res = res[0]
    log('got result: %r' % res)
except:
    if not res:
        res = ''
    print('got exception')
```

3

1. except 5KeyboardInterrupt ◦ ◦
2. except get\_result res◦
3. ◦ get\_result res except res NameError ◦

◦ ◦

```
import traceback

try:
    res = get_result()
except Exception:
    log_exception(traceback.format_exc())
    raise

try:
    res = res[0]
except IndexError:
    res = ''

log('got result: %r' % res)
```

◦ ◦

◦

valueNone

```
def intensive_f(value): # int -> Optional[int]
    # complex, and time-consuming code
    if process_has_failed:
        return None
    return integer_output
```

```
x = 5
```

```

if intensive_f(x) is not None:
    print(intensive_f(x) / 2)
else:
    print(x, "could not be processed")

print(x)

```

intensive\_f ◦ ◦

```

x = 5
result = intensive_f(x)
if result is not None:
    print(result / 2)
else:
    print(x, "could not be processed")

```

## pythonic

```

x = 5
try:
    print(intensive_f(x) / 2)
except TypeError: # The exception raised if None + 1 is attempted
    print(x, "could not be processed")

```

◦ assertAssertionError ◦

◦

```

bird_speeds = get_very_long_dictionary()

if "european swallow" in bird_speeds:
    speed = bird_speeds["european swallow"]
else:
    speed = input("What is the air-speed velocity of an unladen swallow?")

print(speed)

```

```

bird_speeds = get_very_long_dictionary()

try:
    speed = bird_speeds["european swallow"]
except KeyError:
    speed = input("What is the air-speed velocity of an unladen swallow?")

print(speed)

```

◦ ◦

*dict.get(key, default)* ◦

Python <https://riptutorial.com/zh-TW/python/topic/4700/python>



# 45: PythonExcel

## Examples

### Excel

```
import os, sys
from openpyxl import Workbook
from datetime import datetime

dt = datetime.now()
list_values = [
    ["01/01/2016", "05:00:00", 3], \
    ["01/02/2016", "06:00:00", 4], \
    ["01/03/2016", "07:00:00", 5], \
    ["01/04/2016", "08:00:00", 6], \
    ["01/05/2016", "09:00:00", 7]]

# Create a Workbook on Excel:
wb = Workbook()
sheet = wb.active
sheet.title = 'data'

# Print the titles into Excel Workbook:
row = 1
sheet['A'+str(row)] = 'Date'
sheet['B'+str(row)] = 'Hour'
sheet['C'+str(row)] = 'Value'

# Populate with data
for item in list_values:
    row += 1
    sheet['A'+str(row)] = item[0]
    sheet['B'+str(row)] = item[1]
    sheet['C'+str(row)] = item[2]

# Save a file by date:
filename = 'data_' + dt.strftime("%Y%m%d_%I%M%S") + '.xlsx'
wb.save(filename)

# Open the file for the user:
os.chdir(sys.path[0])
os.system('start excel.exe "%s\\%s"' % (sys.path[0], filename, ))
```

## OpenPyXL

### OpenPyXLxlsx/xlsm/xltx/xltm

```
import openpyxl as opx
#To change an existing workbook we located it by referencing its path
workbook = opx.load_workbook(workbook_path)
```

load\_workbook() read\_only True**read\_only**xlsx

```
workbook = opx.load_workbook(workbook_path, read_only=True)
```

```
workbook.worksheets
```

```
first_sheet = workbook.worksheets[0]
```

```
workbook.get_sheet_names()
```

```
sheet = workbook.get_sheet_by_name('Sheet Name')
```

```
sheet.rows
```

```
for row in sheet.rows:  
    print row[0].value
```

```
row[0].value
```

```
#Calling the Workbook() function creates a new book in memory  
wb = opx.Workbook()
```

```
#We can then create a new sheet in the wb  
ws = wb.create_sheet('Sheet Name', 0) #0 refers to the index of the sheet order in the wb
```

**openpyxl** tabColor

```
ws.sheet_properties.tabColor = 'FFC0CB'
```

```
wb.save('filename.xlsx')
```

## xlsxwriterexcel

```
import xlsxwriter  
  
# sample data  
chart_data = [  
    {'name': 'Lorem', 'value': 23},  
    {'name': 'Ipsum', 'value': 48},  
    {'name': 'Dolor', 'value': 15},  
    {'name': 'Sit', 'value': 8},  
    {'name': 'Amet', 'value': 32}  
]  
  
# excel file path  
xls_file = 'chart.xlsx'  
  
# the workbook  
workbook = xlsxwriter.Workbook(xls_file)  
  
# add worksheet to workbook  
worksheet = workbook.add_worksheet()  
  
row_ = 0
```

```

col_ = 0

# write headers
worksheet.write(row_, col_, 'NAME')
col_ += 1
worksheet.write(row_, col_, 'VALUE')
row_ += 1

# write sample data
for item in chart_data:
    col_ = 0
    worksheet.write(row_, col_, item['name'])
    col_ += 1
    worksheet.write(row_, col_, item['value'])
    row_ += 1

# create pie chart
pie_chart = workbook.add_chart({'type': 'pie'})

# add series to pie chart
pie_chart.add_series({
    'name': 'Series Name',
    'categories': '=Sheet1!$A$3:$A$%s' % row_,
    'values': '=Sheet1!$B$3:$B$%s' % row_,
    'marker': {'type': 'circle'}
})

# insert pie chart
worksheet.insert_chart('D2', pie_chart)

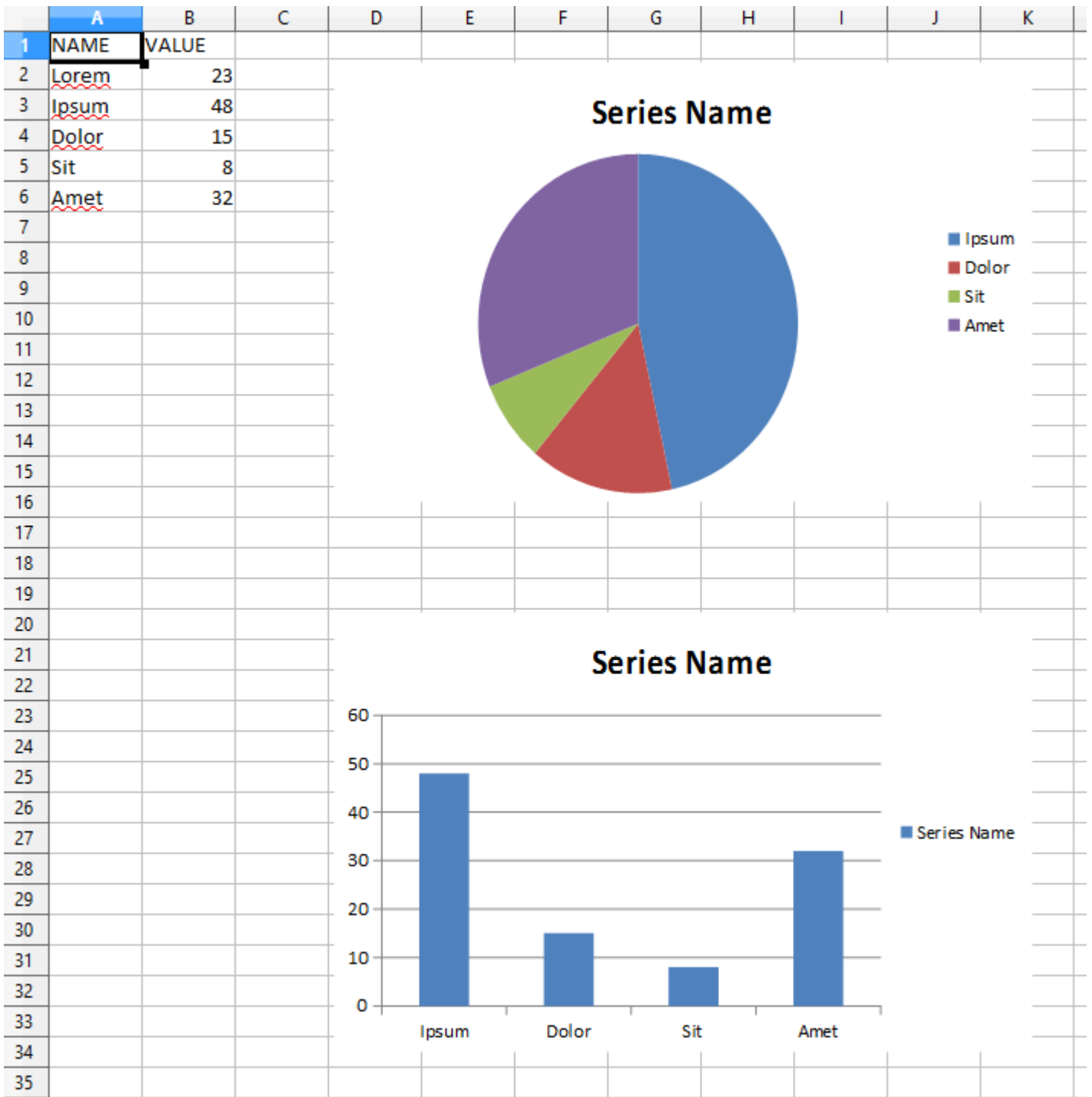
# create column chart
column_chart = workbook.add_chart({'type': 'column'})

# add serie to column chart
column_chart.add_series({
    'name': 'Series Name',
    'categories': '=Sheet1!$A$3:$A$%s' % row_,
    'values': '=Sheet1!$B$3:$B$%s' % row_,
    'marker': {'type': 'circle'}
})

# insert column chart
worksheet.insert_chart('D20', column_chart)

workbook.close()

```



## xlrdexcel

Python xlrdMicrosoft Excel™.

-

```
pip install xlrd
```

pypisetup.py

<https://pypi.python.org/pypi/xlrd>

## Excel -open\_workbookxlrdexcel

```
import xlrd
book=xlrd.open_workbook('sample.xlsx')
```

### excel

```
print book.nsheets
```

```
print book.sheet_names()
```

```
sheet=book.sheet_by_index(1)
```

```
cell = sheet.cell(row,col) #where row=row number and col=column number
print cell.value #to print the cell contents
```

### Excel

```
num_rows=sheet.nrows
num_col=sheet.ncols
```

### Excel

```
sheets = book.sheet_names()
cur_sheet = book.sheet_by_name(sheets[0])
```

## xlsxwriterExcel

```
import xlsxwriter

# create a new file
workbook = xlsxwriter.Workbook('your_file.xlsx')

# add some new formats to be used by the workbook
percent_format = workbook.add_format({'num_format': '0%'})
percent_with_decimal = workbook.add_format({'num_format': '0.0%'})
bold = workbook.add_format({'bold': True})
red_font = workbook.add_format({'font_color': 'red'})
remove_format = workbook.add_format()

# add a new sheet
worksheet = workbook.add_worksheet()

# set the width of column A
worksheet.set_column('A:A', 30, )

# set column B to 20 and include the percent format we created earlier
worksheet.set_column('B:B', 20, percent_format)

# remove formatting from the first row (change in height=None)
worksheet.set_row('0:0', None, remove_format)
```

```
workbook.close()
```

PythonExcel <https://riptutorial.com/zh-TW/python/topic/2986/pythonexcel>

# 46: Python

- `pickle.dumpobjfileprotocol = None*fix_imports = True`
- `pickle.loadfile*fix_imports = Trueencoding = "ASCII"errors = "strict"`

<i>OBJ</i>	obj
	pickler 0 -ASCII 1 -
	filedumpwrite <sub>wb</sub> read <sub>rb</sub>

## Examples

### Python

◦

`pickle`◦

`picklewbrbpickle`◦

```
data={'a':'some_value',
      'b':[9,4,7],
      'c':['some_str','another_str','spam','ham'],
      'd':{'key':'nested_dictionary'},
      }
```

```
import pickle
file=open('filename','wb') #file object in binary write mode
pickle.dump(data,file)    #dump the data in the file object
file.close()              #close the file to write into the file
```

```
import pickle
file=open('filename','rb') #file object in binary read mode
data=pickle.load(file)    #load the data back
file.close()
```

```
>>>data
{'b': [9, 4, 7], 'a': 'some_value', 'd': {'key': 'nested_dictionary'},
 'c': ['some_str', 'another_str', 'spam', 'ham']}
```

- 1.
- 2.
- 3.
- 4.
5. deflambda
- 6.
- 7.

## 8. dictgetstate

```
import pickle
def save(filename,object):
    file=open(filename,'wb')
    pickle.dump(object,file)
    file.close()

def load(filename):
    file=open(filename,'rb')
    object=pickle.load(file)
    file.close()
    return object

>>>list_object=[1,1,2,3,5,8,'a','e','i','o','u']
>>>save(list_file,list_object)
>>>new_list=load(list_file)
>>>new_list
[1, 1, 2, 3, 5, 8, 'a', 'e', 'i', 'o', 'u']
```

Python <https://riptutorial.com/zh-TW/python/topic/7810/python>



# 47: Python

- Python ◦

## Examples

- Python ◦ Intfloatcomplexlong ◦

```
int_num = 10      #int value
float_num = 10.2  #float value
complex_num = 3.14j  #complex value
long_num = 1234567L  #long value
```

- Python ◦

```
a_str = 'Hello World'
print(a_str)      #output will be whole string. Hello World
print(a_str[0])   #output will be first character. H
print(a_str[0:5]) #output will be first five characters. Hello
```

- [] ◦ C ◦

```
list = [123,'abcd',10.2,'d'] #can be a array of any data type or single data type.
list1 = ['hello','world']
print(list)      #will ouput whole list. [123,'abcd',10.2,'d']
print(list[0:2]) #will output first two element of list. [123,'abcd']
print(list1 * 2) #will gave list1 two times. ['hello','world','hello','world']
print(list + list1) #will gave concatenation of both the lists.
[123,'abcd',10.2,'d','hello','world']
```

- [] ◦

```
tuple = (123,'hello')
tuple1 = ('world')
print(tuple)     #will output whole tuple. (123,'hello')
print(tuple[0])  #will output first value. (123)
print(tuple + tuple1) #will output (123,'hello','world')
tuple[1]='update' #this will give you error.
```

- {} ◦

```
dic={'name':'red','age':10}
print(dic)      #will output all the key-value pairs. {'name':'red','age':10}
print(dic['name']) #will output only value with 'name' key. 'red'
print(dic.values()) #will output list of values in dic. ['red',10]
print(dic.keys()) #will output list of keys. ['name','age']
```

1. -

```
basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
print(basket)          # duplicates will be removed
> {'orange', 'banana', 'pear', 'apple'}
a = set('abracadabra')
print(a)              # unique letters in a
> {'a', 'r', 'b', 'c', 'd'}
a.add('z')
print(a)
> {'a', 'c', 'r', 'b', 'z', 'd'}
```

2. - ◦

```
b = frozenset('asdfagsa')
print(b)
> frozenset({'f', 'g', 'd', 'a', 's'})
cities = frozenset(["Frankfurt", "Basel", "Freiburg"])
print(cities)
> frozenset({'Frankfurt', 'Basel', 'Freiburg'})
```

Python <https://riptutorial.com/zh-TW/python/topic/9366/python>

---

# 48: Python

SSEWeb""。 websockets。 SSEwebsocketsSSEwebsockets。 SSEwebsockets/。

## Examples

### SSE

```
@route("/stream")
def stream():
    def event_stream():
        while True:
            if message_to_send:
                yield "data:
                    {}\n\n".format(message_to_send) "

    return Response(event_stream(), mimetype="text/event-stream")
```

### Asyncio SSE

asyncio SSE <https://github.com/brutasse/asyncio-sse>

```
import asyncio
import sse

class Handler(sse.Handler):
    @asyncio.coroutine
    def handle_request(self):
        yield from asyncio.sleep(2)
        self.send('foo')
        yield from asyncio.sleep(2)
        self.send('bar', event='wakeup')

start_server = sse.serve(Handler, 'localhost', 8888)
asyncio.get_event_loop().run_until_complete(start_server)
asyncio.get_event_loop().run_forever()
```

Python <https://riptutorial.com/zh-TW/python/topic/9100/python>

# 49: PythonCurses

CursesPython◦ TUI◦

C'nurses'python

## Examples

```
import curses
import traceback

try:
    # -- Initialize --
    stdscr = curses.initscr()    # initialize curses screen
    curses.noecho()             # turn off auto echoing of keypress on to screen
    curses.cbreak()             # enter break mode where pressing Enter key
                                # after keystroke is not required for it to register
    stdscr.keypad(1)            # enable special Key values such as curses.KEY_LEFT etc

    # -- Perform an action with Screen --
    stdscr.border(0)
    stdscr.addstr(5, 5, 'Hello from Curses!', curses.A_BOLD)
    stdscr.addstr(6, 5, 'Press q to close this screen', curses.A_NORMAL)

    while True:
        # stay in this loop till the user presses 'q'
        ch = stdscr.getch()
        if ch == ord('q'):
            break

    # -- End of user code --

except:
    traceback.print_exc()      # print trace back log of the error

finally:
    # --- Cleanup on exit ---
    stdscr.keypad(0)
    curses.echo()
    curses.nocbreak()
    curses.endwin()
```

wrapper◦

curseswrapper(func, ...)

```
main(scr, *args):
    # -- Perform an action with Screen --
    scr.border(0)
    scr.addstr(5, 5, 'Hello from Curses!', curses.A_BOLD)
    scr.addstr(6, 5, 'Press q to close this screen', curses.A_NORMAL)

    while True:
        # stay in this loop till the user presses 'q'
```

```
ch = scr.getch()
if ch == ord('q'):

curses.wrapper(main)
```

`wrapper``curses``stdscr` `WindowObject``stdscr``func` ◦ `func` `wrapper` ◦

[PythonCurses](https://riptutorial.com/zh-TW/python/topic/5851/pythoncurses) <https://riptutorial.com/zh-TW/python/topic/5851/pythoncurses>

# 50: Python

## Examples

PythonBig-O

```
def list_check(to_check, the_list):  
    for item in the_list:  
        if to_check == item:  
            return True  
    return False
```

o On "O"OOrder

On - n

Ok - k

O1

On

DelOn

On

On

O1

O1

On

Ok

On + k

Ok

On log n

Onk

x inOn

minsmaxsOn

O1

## Deque

o

```
class Deque:
def __init__(self):
    self.items = []

def isEmpty(self):
    return self.items == []

def addFront(self, item):
    self.items.append(item)

def addRear(self, item):
    self.items.insert(0, item)

def removeFront(self):
    return self.items.pop()

def removeRear(self):
    return self.items.pop(0)

def size(self):
    return len(self.items)
```

O1

O1

On

Ok

ExtendleftOk

O1

PopleftO1

On

Ok

xxO1

s - tOlens

stOminlenslentOlens\* lent

s1s2s3...sn ::n-1\* Ollmaxlens1...lensn

s.difference\_updatetOlentOlent\* lens

s.symmetric\_difference\_update(t)

s ^ t

s | t

.....

Python

80/20 802090/10 - 80

“”“”

On log n O(n^2) O(log n) Python O(n)

... Python

3.

1.  $\Theta$  theta.  $\Theta$   $3n^3 + 6n^2 + 6000 = \Theta(n^3)$   $\Theta(n^3)$   $\Theta(n^2)$   $\Theta(n)$   $\Theta(1)$   $\Theta(c)$   $\Theta(0)$   $\Theta(\infty)$   $\Theta(-\infty)$   $\Theta(\text{any constant})$

2. **Big O** Big O.  $O(n^2)$   $O(n)$   $O(1)$   $O(\infty)$   $O(-\infty)$   $O(\text{any constant})$

1.  $\Theta(n^2)$

2.  $\Theta(n)$

Big O.  $O(n) = \{f(n) \geq 0 \leq f(n) \leq c \cdot n\}$

3.  **$\Omega$**  Big O  $\Omega$ .  $\Omega(n)$   $\Omega(1)$   $\Omega(\infty)$   $\Omega(-\infty)$   $\Omega(\text{any constant})$   $\Omega(n)$

Python <https://riptutorial.com/zh-TW/python/topic/9185/python>



# 51: Python

## Python

### Examples

#### Python -

```
import socket

serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
serversocket.bind(('localhost', 8089))
serversocket.listen(5) # become a server socket, maximum 5 connections

while True:
    connection, address = serversocket.accept()
    buf = connection.recv(64)
    if len(buf) > 0:
        print(buf)
    break
```

```
import socket

clientsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
clientsocket.connect(('localhost', 8089))
clientsocket.send('hello')
```

SocketServer.py/.....;

#### Http

httpjavascriptPythonSimpleHTTPServer。 PythonPath。

python 2

```
$ python -m SimpleHTTPServer <portnumber>
```

python 3

```
$ python3 -m http.server <portnumber>
```

8000◦

0.0.0.08000HTTP ...

http://hostipaddress:8000/◦

hostipaddressIP192.168.xx◦

ctrl+c.

## TCP

socketserverTCP◦ echo◦

```
from socketserver import BaseRequestHandler, TCPServer

class EchoHandler(BaseRequestHandler):
    def handle(self):
        print('connection from:', self.client_address)
        while True:
            msg = self.request.recv(8192)
            if not msg:
                break
            self.request.send(msg)

if __name__ == '__main__':
    server = TCPServer(('', 5000), EchoHandler)
    server.serve_forever()
```

```
from socket import socket, AF_INET, SOCK_STREAM
sock = socket(AF_INET, SOCK_STREAM)
sock.connect(('localhost', 5000))
sock.send(b'Monty Python')
sock.recv(8192) # returns b'Monty Python'
```

socketserverTCP◦ ◦ ThreadingTCPServer◦

```
from socketserver import ThreadingTCPServer
...
if __name__ == '__main__':
    server = ThreadingTCPServer(('', 5000), EchoHandler)
    server.serve_forever()
```

## UDP

socketserverUDP◦

```
import time
from socketserver import BaseRequestHandler, UDPServer

class CtimeHandler(BaseRequestHandler):
    def handle(self):
        print('connection from: ', self.client_address)
        # Get message and client socket
        msg, sock = self.request
        resp = time.ctime()
        sock.sendto(resp.encode('ascii'), self.client_address)

if __name__ == '__main__':
    server = UDPServer(('', 5000), CtimeHandler)
    server.serve_forever()
```

```
>>> from socket import socket, AF_INET, SOCK_DGRAM
>>> sock = socket(AF_INET, SOCK_DGRAM)
>>> sick.sendto(b'', ('localhost', 5000))
0
>>> sock.recvfrom(8192)
(b'Wed Aug 15 20:35:08 2012', ('127.0.0.1', 5000))
```

## Simple HttpServer

o

```
from http.server import HTTPServer, CGIHTTPRequestHandler
import webbrowser
import threading

def start_server(path, port=8000):
    '''Start a simple webserver serving path on port'''
    os.chdir(path)
    httpd = HTTPServer(('', port), CGIHTTPRequestHandler)
    httpd.serve_forever()

# Start the server in a new thread
port = 8000
daemon = threading.Thread(name='daemon_server',
                           target=start_server,
                           args=('.', port))
daemon.setDaemon(True) # Set as a daemon so it will be killed once the main thread is dead.
daemon.start()

# Open the web browser
webbrowser.open('http://localhost:{}'.format(port))
```

Python <https://riptutorial.com/zh-TW/python/topic/1309/python>

# 52: Python - virtualenv

“virtualenv”Python。 Python env。 “A2.xxxB2.xxx”。

“virtualenv”Python。

## Examples

pip /apt-getvirtualenv

```
pip install virtualenv
```

```
apt-get install python-virtualenv
```

sudo。

```
$ cd test_proj
```

```
$ virtualenv test_proj
```

```
$ source test_project/bin/activate
```

virtualenv“deactivate”

```
$ deactivate
```

## Virtualenv

virtualenvbineasy\_installeggvirtualenvsite-packages。

```
$ source test_project/bin/activate  
$ pip install flask
```

sudovirtualenv site-packages。 ◦

**virtualenv**

**lsvirtualenv** ◦

**cdvirtualenv** ◦

**cdsitepackages** site-packages。

**lssitepackages** site-packages。



# 53: Python

## HTTP POST Requests Python Requests

### Examples

```
from requests import post

foo = post('http://httpbin.org/post', data = {'key':'value'})
```

#### HTTP POST

```
print(foo.headers)
```

```
{'Content-Length': '439', 'X-Processed-Time': '0.000802993774414', 'X-Powered-By': 'Flask',
'Server': 'meinheld/0.6.1', 'Connection': 'keep-alive', 'Via': '1.1 vegur', 'Access-Control-
Allow-Credentials': 'true', 'Date': 'Sun, 21 May 2017 20:56:05 GMT', 'Access-Control-Allow-
Origin': '*', 'Content-Type': 'application/json'}
```

```
headers = {'Cache-Control': 'max-age=0',
           'Upgrade-Insecure-Requests': '1',
           'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/54.0.2840.99 Safari/537.36',
           'Content-Type': 'application/x-www-form-urlencoded',
           'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8',
           'Referer': 'https://www.groupon.com/signup',
           'Accept-Encoding': 'gzip, deflate, br',
           'Accept-Language': 'es-ES,es;q=0.8'
          }
```

```
foo = post('http://httpbin.org/post', headers=headers, data = {'key':'value'})
```

```
print(foo.encoding)
```

```
'utf-8'
```

```
foo.encoding = 'ISO-8859-1'
```

### SSL

#### SSL

```
foo = post('http://httpbin.org/post', data = {'key':'value'}, verify=False)
```

#### httphttps

```
foo = post('http://httpbin.org/post', data = {'key':'value'}, allow_redirects=False)
```

```
print(foo.url)
```

```
print(foo.history)
```

```
from requests import post

payload = {'key1' : 'value1',
          'key2' : 'value2'
          }

foo = post('http://httpbin.org/post', data=payload)
```

**post**

**data**

**json**

```
from requests import post

payload = {'key1' : 'value1', 'key2' : 'value2'}

foo = post('http://httpbin.org/post', json=payload)
```

**Requests.read()**

```
from requests import post

files = {'file' : open('data.txt', 'rb')}

foo = post('http://http.org/post', files=files)
```

**Filenamecontent\_typeheaders**

```
files = {'file': ('report.xls', open('report.xls', 'rb'), 'application/vnd.ms-excel',
{'Expires': '0'})}

foo = requests.post('http://httpbin.org/post', files=files)
```

**files**

```
multiple_files = [
    ('images', ('foo.png', open('foo.png', 'rb'), 'image/png')),
    ('images', ('bar.png', open('bar.png', 'rb'), 'image/png'))]

foo = post('http://httpbin.org/post', files=multiple_files)
```

```
from requests import post

foo = post('http://httpbin.org/post', data={'data' : 'value'})
print(foo.status_code)
```

```
foo = post('http://httpbin.org/post', data={'data' : 'value'})
print(foo.text)
```

## urllib3 response.HTTPResponse

```
foo = post('http://httpbin.org/post', data={'data' : 'value'})
res = foo.raw

print(res.read())
```

## HTTP

## HTTP

```
from requests import post

foo = post('http://natas0.natas.labs.overthewire.org', auth=('natas0', 'natas0'))
```

```
from requests import post
from requests.auth import HTTPBasicAuth

foo = post('http://natas0.natas.labs.overthewire.org', auth=HTTPBasicAuth('natas0', 'natas0'))
```

## HTTP

## HTTP

```
from requests import post
from requests.auth import HTTPDigestAuth

foo = post('http://natas0.natas.labs.overthewire.org', auth=HTTPDigestAuth('natas0',
'natas0'))
```

## HTTP。AuthBaseRequests

```
from requests.auth import AuthBase
from requests.auth import _basic_auth_str
from requests._internal_utils import to_native_string

class CustomAuth(AuthBase):

    def __init__(self, secret_header, user_agent, username, password):
        # setup any auth-related data here
        self.secret_header = secret_header
        self.user_agent = user_agent
        self.username = username
        self.password = password

    def __call__(self, r):
        # modify and return the request
        r.headers['X-Secret'] = self.secret_header
        r.headers['User-Agent'] = self.user_agent
        r.headers['Authorization'] = _basic_auth_str(self.username, self.password)
```



```
return r
```

```
foo = get('http://test.com/admin', auth=CustomAuth('SecretHeader', 'CustomUserAgent', 'user',  
'password' ))
```

## POST

## HTTP / S

```
from requests import post  
  
proxies = {  
    'http': 'http://192.168.0.128:3128',  
    'https': 'http://192.168.0.127:1080',  
}  
  
foo = requests.post('http://httpbin.org/post', proxies=proxies)
```

## HTTP

```
proxies = {'http': 'http://user:pass@192.168.0.128:312'}  
foo = requests.post('http://httpbin.org/post', proxies=proxies)
```

## SOCKS

`socks` `requests[socks]` `socksHTTPBasicAuth`

```
proxies = {  
    'http': 'socks5://user:pass@host:port',  
    'https': 'socks5://user:pass@host:port'  
}  
  
foo = requests.post('http://httpbin.org/post', proxies=proxies)
```

Python <https://riptutorial.com/zh-TW/python/topic/10021/python>

# 54: setup.py

name	◦
version	◦
packages	Python◦ <code>setuptools.find_packages()</code> ◦
py_modules	Python.py◦

python

◦

## Examples

### setup.py

Distutils◦ ◦

foofoo.py

```
from distutils.core import setup

setup(name='foo',
      version='1.0',
      py_modules=['foo'],
      )
```

setup.py

```
python setup.py sdist
```

sdistUnixtarballWindowsZIPsetup.pyfoo.py.foo-1.0.tar.gz.zipfoo-1.0◦

foofoo-1.0.tar.gz.zipfoo-1.0

```
python setup.py install
```

python

python◦ ◦

greetingshello\_world.py◦

```
greetings/  
  greetings/  
    __init__.py  
    hello_world.py
```

```
python greetings/greetings/hello_world.py
```

```
hello_world.py
```

setup.py setup() scripts

```
from setuptools import setup  
setup(  
    name='greetings',  
    scripts=['hello_world.py']  
)
```

hello\_world.py

```
entry_points={'console_scripts': ['greetings=greetings.hello_world:main']}
```

```
greetings
```

## setup.py

[setuptools\\_scm](#) [Git](#) [Mercurial](#) [Python](#)

```
from setuptools import setup, find_packages  
  
setup(  
    setup_requires=['setuptools_scm'],  
    use_scm_version=True,  
    packages=find_packages(),  
    include_package_data=True,  
)
```

;[SCM](#) find\_packages() use\_scm\_version=True

```
python setup.py install
```

。

```
python setup.py develop
```

## Sphinx fortran

```
cmdclasses = dict()  
  
class BuildSphinx(Command):
```

```
"""Build Sphinx documentation."""

description = 'Build Sphinx documentation'
user_options = []

def initialize_options(self):
    pass

def finalize_options(self):
    pass

def run(self):
    import sphinx
    sphinx.build_main(['setup.py', '-b', 'html', './doc', './doc/_build/html'])
    sphinx.build_main(['setup.py', '-b', 'man', './doc', './doc/_build/man'])

cmdclasses['build_sphinx'] = BuildSphinx

setup(
    ...
    cmdclass=cmdclasses,
)
```

```
initialize_optionsfinalize_optionsrun
```

```
python setup.py build_sphinx
```

[setup.py](https://riptutorial.com/zh-TW/python/topic/1444/setup-py) <https://riptutorial.com/zh-TW/python/topic/1444/setup-py>

# 55: Sqlite3

## Examples

### Sqlite3 - ◦

sqlite3GerhardHäring◦ Connection◦ example.db

```
import sqlite3
conn = sqlite3.connect('example.db')
```

memoryRAM◦ ConnectionCursorexecuteSQL

```
c = conn.cursor()

# Create table
c.execute('''CREATE TABLE stocks
            (date text, trans text, symbol text, qty real, price real)''')

# Insert a row of data
c.execute("INSERT INTO stocks VALUES ('2006-01-05','BUY','RHAT',100,35.14)")

# Save (commit) the changes
conn.commit()

# We can also close the connection if we are done with it.
# Just be sure any changes have been committed or they will be lost.
conn.close()
```

### SQLite3◦

#### select

```
import sqlite3
conn = sqlite3.connect('example.db')
c = conn.cursor()
c.execute("SELECT * from table_name where id=cust_id")
for row in c:
    print row # will be a list
```

#### fetchone

```
print c.fetchone()
```

#### fetchall

```
a=c.fetchall() #which is similar to list(cursor) method used previously
for row in a:
    print row
```

## sqlite3.Error

```
try:
    #SQL Code
except sqlite3.Error as e:
    print "An error occurred:", e.args[0]
```

Sqlite3 <https://riptutorial.com/zh-TW/python/topic/7754/sqlite3>

# 56: SYS

**sys** `sys.argv` `sys.exit()`

◦

- **sys**

```
import sys
```

- **sys**

```
from sys import exit
```

**sys** ◦

## Examples

```
if len(sys.argv) != 4:          # The script name needs to be accounted for as well.
    raise RuntimeError("expected 3 command line arguments")

f = open(sys.argv[1], 'rb')     # Use first command line argument.
start_line = int(sys.argv[2])  # All arguments come as strings, so need to be
end_line = int(sys.argv[3])    # converted explicitly if other types are required.
```

[click](#) ◦

```
# The name of the executed script is at the beginning of the argv list.
print('usage:', sys.argv[0], '<filename> <start> <end>')

# You can use it to generate the path prefix of the executed program
# (as opposed to the current module) to access files relative to that,
# which would be good for assets of a game, for instance.
program_file = sys.argv[0]

import pathlib
program_path = pathlib.Path(program_file).resolve().parent
```

```
# Error messages should not go to standard output, if possible.
print('ERROR: We have no cheese at all.', file=sys.stderr)

try:
    f = open('nonexistent-file.xyz', 'rb')
except OSError as e:
    print(e, file=sys.stderr)
```

```
def main():
    if len(sys.argv) != 4 or '--help' in sys.argv[1:]:
        print('usage: my_program <arg1> <arg2> <arg3>', file=sys.stderr)
```

```
    sys.exit(1)    # use an exit code to signal the program was unsuccessful
process_data()
```

**SYS** <https://riptutorial.com/zh-TW/python/topic/9847/sys>



## 57: tempfile NamedTemporaryFile

PARAM	
	= w + b
	= True
	= "
	= 'tmp'
DIR	tempfiledirname= None
BUFFSIZE	default = -1

### Examples

name◦ unixdelete paramTrue◦

“Hello World”◦ namepath◦ ◦

```
import tempfile

with tempfile.NamedTemporaryFile(delete=False) as t:
    t.write('Hello World!')
    path = t.name
    print path

with open(path) as t:
    print t.read()
```

```
/tmp/tmp6pireJ
Hello World!
```

[tempfile NamedTemporaryFile](https://riptutorial.com/zh-TW/python/topic/3666/tempfile-namedtemporaryfile) <https://riptutorial.com/zh-TW/python/topic/3666/tempfile-namedtemporaryfile>

# 58: Tkinter

TkinterPythonGUI。

tkinterPython 23。 Python 2

```
from Tkinter import * # Capitalized
```

Python 3

```
from tkinter import * # Lowercase
```

Python 23

```
try:
    from Tkinter import *
except ImportError:
    from tkinter import *
```

```
from sys import version_info
if version_info.major == 2:
    from Tkinter import *
elif version_info.major == 3:
    from tkinter import *
```

[tkinter](#)

## Examples

tkinter

tkinterGUI Tk / Tcl GUI Python。 tkinter。

Python 2”。

```
import tkinter as tk

# GUI window is a subclass of the basic tkinter Frame object
class HelloWorldFrame(tk.Frame):
    def __init__(self, master):
        # Call superclass constructor
        tk.Frame.__init__(self, master)
        # Place frame into main window
        self.grid()
        # Create text box with "Hello World" text
        hello = tk.Label(self, text="Hello World! This label can hold strings!")
        # Place text box into frame
        hello.grid(row=0, column=0)
```

```

# Spawn window
if __name__ == "__main__":
    # Create main window object
    root = tk.Tk()
    # Set title of window
    root.title("Hello World!")
    # Instantiate HelloWorldFrame object
    hello_frame = HelloWorldFrame(root)
    # Start GUI
    hello_frame.mainloop()

```

## Tkinter place packgrid ◦

place◦

pack4◦ ◦

grid◦

---

widget.place

- x **X**
- y **Y**
- height
- width

place

```

class PlaceExample(Frame):
    def __init__(self, master):
        Frame.__init__(self, master)
        self.grid()
        top_text=Label(master, text="This is on top at the origin")
        #top_text.pack()
        top_text.place(x=0, y=0, height=50, width=200)
        bottom_right_text=Label(master, text="This is at position 200,400")
        #top_text.pack()
        bottom_right_text.place(x=200, y=400, height=50, width=200)
# Spawn Window
if __name__=="__main__":
    root=Tk()
    place_frame=PlaceExample(root)
    place_frame.mainloop()

```

---

widget.pack

- expand
- fill **NONEXYBOTH**
- side **TOPBOTTOMLEFT**

---

widget.grid

- row
- rowspan **columns1**
- column **0**
- colspan **1**
- sticky **NNEESESWWNW**

◦ ◦

grid

```
from tkinter import *

class GridExample(Frame):
    def __init__(self, master):
        Frame.__init__(self, master)
        self.grid()
        top_text=Label(self, text="This text appears on top left")
        top_text.grid() # Default position 0, 0
        bottom_text=Label(self, text="This text appears on bottom left")
        bottom_text.grid() # Default position 1, 0
        right_text=Label(self, text="This text appears on the right and spans both rows",
                          wraplength=100)
        # Position is 0,1
        # Rowspan means actual position is [0-1],1
        right_text.grid(row=0, column=1, rowspan=2)

# Spawn Window
if __name__=="__main__":
    root=Tk()
    grid_frame=GridExample(root)
    grid_frame.mainloop()
```

**packgrid**

**Tkinter** <https://riptutorial.com/zh-TW/python/topic/7574/tkinter>

# 59: Unicode

- `str.encodeencodingerrors='strict'`
- `bytes.decodeencodingerrors='strict'`
- `=`

```
'ascii' 'utf8'...  
'replace' 'ignore'.....
```

## Examples

### Python 3 `str`/`unicode`/`bytes`◦

```
type("f") == type(u"f") # True, <class 'str'>  
type(b"f") # <class 'bytes'>
```

### Python 2 `unicode`“u”◦

```
type("f") == type(b"f") # True, <type 'str'>  
type(u"f") # <type 'unicode'>
```

# Unicode

`.encode(encoding)` `Unicode`◦

### Python 3

```
>>> "£13.55".encode('utf8')  
b'\xc2\xa313.55'  
>>> "£13.55".encode('utf16')  
b'\xff\xfe\xa3\x001\x003\x00.\x005\x005\x00'
```

### Python 2

`py2``sys.getdefaultencoding()` == `'ascii'``py3``utf-8`◦

```
>>> print type(u"£13.55".encode('utf8'))  
<type 'str'>  
>>> print u"£13.55".encode('utf8')  
SyntaxError: Non-ASCII character '\xc2' in...  
  
# with encoding set inside a file
```

```
# -*- coding: utf-8 -*-
>>> print u"£13.55".encode('utf8')
Tú13.55
```

## “UnicodeEncodeError”

```
>>> "£13.55".encode('ascii')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeEncodeError: 'ascii' codec can't encode character '\xa3' in position 0: ordinal not in range(128)
```

---

# unicode

.decode(encoding) **unicode**

## unicode

```
>>> b'\xc2\xa313.55'.decode('utf8')
'£13.55'
```

UnicodeDecodeError

```
>>> b'\xc2\xa313.55'.decode('utf16')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/csaftoiu/csaftoiu-github/yahoo-groups-backup/.virtualenv/bin/./lib/python3.5/encodings/utf_16.py", line 16, in decode
    return codecs.utf_16_decode(input, errors, True)
UnicodeDecodeError: 'utf-16-le' codec can't decode byte 0x35 in position 6: truncated data
```

/

.encode.decode

'strict' . .

```
>>> "£13.55".encode('ascii', errors='replace')
b'?13.55'
>>> "£13.55".encode('ascii', errors='ignore')
b'13.55'
>>> "£13.55".encode('ascii', errors='namereplace')
b'\N{POUND SIGN}13.55'
>>> "£13.55".encode('ascii', errors='xmlcharrefreplace')
b'&#163;13.55'
>>> "£13.55".encode('ascii', errors='backslashreplace')
```

```
b'\\xa313.55'
```

```
>>> b = "£13.55".encode('utf8')
>>> b.decode('ascii', errors='replace')
'◆◆13.55'
>>> b.decode('ascii', errors='ignore')
'13.55'
>>> b.decode('ascii', errors='backslashreplace')
'\\xc2\\xa313.55'
```

unicode。

## I/O.

'r' 'w' ◦ default 'utf8' ◦

```
open(fn, mode='r') # opens file for reading in utf8
open(fn, mode='r', encoding='utf16') # opens file for reading utf16

# ERROR: cannot write bytes when a string is expected:
open("foo.txt", "w").write(b"foo")
```

'rb' 'wb' ◦ ◦

```
open(fn, mode='wb') # open file for writing bytes

# ERROR: cannot write string when bytes is expected:
open(fn, mode='wb').write("hi")
```

Unicode <https://riptutorial.com/zh-TW/python/topic/1216/unicode>

# 60: WebWSGI

start\_response

## Examples

'application'。。

```
import os, sys

def run(application):
    environ['wsgi.input']      = sys.stdin
    environ['wsgi.errors']     = sys.stderr

    headers_set = []
    headers_sent = []

    def write(data):
        """
        Writes header data from 'start_response()' as well as body data from 'response'
        to system standard output.
        """
        if not headers_set:
            raise AssertionError("write() before start_response()")

        elif not headers_sent:
            status, response_headers = headers_sent[:] = headers_set
            sys.stdout.write('Status: %s\r\n' % status)
            for header in response_headers:
                sys.stdout.write('%s: %s\r\n' % header)
            sys.stdout.write('\r\n')

        sys.stdout.write(data)
        sys.stdout.flush()

    def start_response(status, response_headers):
        """ Sets headers for the response returned by this server. """
        if headers_set:
            raise AssertionError("Headers already set!")

        headers_set[:] = [status, response_headers]
        return write

    # This is the most important piece of the 'server object'
    # Our result will be generated by the 'application' given to this method as a parameter
    result = application(environ, start_response)
    try:
        for data in result:
            if data:
                write(data)          # Body isn't empty send its data to 'write()'
            if not headers_sent:
                write('')           # Body is empty, send empty string to 'write()'
    
```

WebWSGI <https://riptutorial.com/zh-TW/python/topic/5315/web-wsgi->



# 61: Web

## PythonwebbrowserWeb. webbrowser.

- `webbrowser.open(url, new=0, autoraise=False)`
- `webbrowser.open_new(url)`
- `webbrowser.open_new_tab(url)`
- `webbrowser.get(usage=None)`
- `webbrowser.register(name, constructor, instance=None)`

<code>webbrowser.open()</code>	
	WebURL
0URL12	
autoraise	True
<code>webbrowser.open_new()</code>	
	WebURL
<code>webbrowser.open_new_tab()</code>	
	WebURL
<code>webbrowser.get()</code>	
<code>webbrowser.register()</code>	
	<code>webbrowser.get()</code> Web

- `webbrowser.get()`

'mozilla'	Mozilla('mozilla')
'firefox'	Mozilla('mozilla')
'netscape'	Mozilla('netscape')
'galeon'	Galeon('galeon')
'epiphany'	Galeon('epiphany')
'skipstone'	BackgroundBrowser('skipstone')
'kfmclient'	Konqueror()
'konqueror'	Konqueror()

'kfm'	Konqueror()
'mosaic'	BackgroundBrowser('mosaic')
'opera'	Opera()
'grail'	Grail()
'links'	GenericBrowser('links')
'elinks'	Elinks('elinks')
'lynx'	GenericBrowser('lynx')
'w3m'	GenericBrowser('w3m')
'windows-default'	WindowsDefault
'macosx'	MacOSX('default')
'safari'	MacOSX('safari')
'google-chrome'	Chrome('google-chrome')
'chrome'	Chrome('chrome')
'chromium'	Chromium('chromium')
'chromium-browser'	Chromium('chromium-browser')

## Examples

### URL

URL.webbrowser.open()

```
import webbrowser
webbrowser.open("http://stackoverflow.com")
```

URL.URL.open

- url - WebURL []
- new - 012 [0]
- autoraise - True [False]

newautoraise°

Webbrowser.open\_newURL

```
import webbrowser
webbrowser.open_new("http://stackoverflow.com")
```

URL.open\_new\_tab

```
import webbrowser
webbrowser.open_new_tab("http://stackoverflow.com")
```

## URL

**webbrowser**register()get()◦ get◦

```
import webbrowser
ff_path = webbrowser.get("C:/Program Files/Mozilla Firefox/firefox.exe")
ff = webbrowser.get(ff_path)
ff.open("http://stackoverflow.com/")
```

```
import webbrowser
ff_path = webbrowser.get("C:/Program Files/Mozilla Firefox/firefox.exe")
ff = webbrowser.get(ff_path)
webbrowser.register('firefox', None, ff)
# Now to refer to use Firefox in the future you can use this
webbrowser.get('firefox').open("https://stackoverflow.com/")
```

Web <https://riptutorial.com/zh-TW/python/topic/8676/web>

# 62: “with” Statement

Python ◦ Python ◦

- “context\_manager”“context\_manager”“”\*

PEP 343 ◦ try ... finally ◦

PEP `__exit__()` `__enter__()` `__exit__()` with ◦

with ◦

```
with EXPR as VAR:  
    BLOCK
```

```
mgr = (EXPR)  
exit = type(mgr).__exit__ # Not calling it yet  
value = type(mgr).__enter__(mgr)  
exc = True  
try:  
    try:  
        VAR = value # Only if "as VAR" is present  
        BLOCK  
    except:  
        # The exceptional case is handled here  
        exc = False  
        if not exit(mgr, *sys.exc_info()):  
            raise  
        # The exception is swallowed if exit() returns true  
finally:  
    # The normal and non-local-goto cases are handled here  
    if exc:  
        exit(mgr, None, None, None)
```

## Examples

with

◦ with ◦

◦

```
open_file = open(filename)  
with open_file:  
    file_contents = open_file.read()  
  
# the open_file object has automatically been closed.
```

as

```
with open(filename) as open_file:
    file_contents = open_file.read()

# the open_file object has automatically been closed.
```

## exit. / Python 2.5.

◦ with

with

```
with database_connection as cursor:
    cursor.execute(sql_query)
```

```
with open(filename) as open_file:
    file_contents = open_file.read()
```

\_\_enter\_\_()\_\_exit\_\_()

```
class AContextManager():

    def __enter__(self):
        print("Entered")
        # optionally return an object
        return "A-instance"

    def __exit__(self, exc_type, exc_value, traceback):
        print("Exited" + (" (with an exception)" if exc_type else ""))
        # return True if you want to suppress the exception
```

exc\_type exc\_value tracebacks sys.exc\_info() ◦ None ◦

\_\_exit\_\_ True \_\_exit\_\_ ◦

```
with AContextManager() as a:
    print("a is %r" % a)
# Entered
# a is 'A-instance'
# Exited

with AContextManager() as a:
    print("a is %d" % a)
# Entered
# Exited (with an exception)
# Traceback (most recent call last):
#   File "<stdin>", line 2, in <module>
# TypeError: %d format: a number is required, not str
```

with \_\_exit\_\_ ◦

\_\_exit\_\_

```
class MyContextManager:
    def __enter__(self):
```

```
    return self

def __exit__(self):
    print('something')
```

### `contextlib.contextmanager`

```
import contextlib

@contextlib.contextmanager
def context_manager(num):
    print('Enter')
    yield num + 1
    print('Exit')

with context_manager(2) as cm:
    # the following instructions are run when the 'yield' point of the context
    # manager is reached.
    # 'cm' will have the value that was yielded
    print('Right in the middle with cm = {}'.format(cm))
```

```
Enter
Right in the middle with cm = 3
Exit
```

- `yield_enter_with`/`yield_exit`

`try..except..finally` -with-

```
@contextlib.contextmanager
def error_handling_context_manager(num):
    print("Enter")
    try:
        yield num + 1
    except ZeroDivisionError:
        print("Caught error")
    finally:
        print("Cleaning up")
    print("Exit")

with error_handling_context_manager(-1) as cm:
    print("Dividing by cm = {}".format(cm))
    print(2 / cm)
```

```
Enter
Dividing by cm = 0
Caught error
Cleaning up
Exit
```

```
with open(input_path) as input_file, open(output_path, 'w') as output_file:

    # do something with both files.

    # e.g. copy the contents of input_file into output_file
```

```
for line in input_file:
    output_file.write(line + '\n')
```

```
with open(input_path) as input_file:
    with open(output_path, 'w') as output_file:
        for line in input_file:
            output_file.write(line + '\n')
```

```
class File():
    def __init__(self, filename, mode):
        self.filename = filename
        self.mode = mode

    def __enter__(self):
        self.open_file = open(self.filename, self.mode)
        return self.open_file

    def __exit__(self, *args):
        self.open_file.close()
```

`__init__()` ◦ `__enter__()` `__exit__()` ◦

`__enter__` `__exit__` **with** `with` ◦

## File

```
for _ in range(10000):
    with File('foo.txt', 'w') as f:
        f.write('foo')
```

“with” Statement <https://riptutorial.com/zh-TW/python/topic/928/--with-statement->

# 63: Python 2Python 3

Python 2008Python 3 Python 2Python 3

Python2.7Python 23.6Python 3 3.33.4

Python 2.7Python1.x2.xPythonPython CPython CPython2020

[Python Enhancement Proposal 373](#) 2016625Python 22020 2020Python2

Python 3 Python 3

Python 3.0Python 3Python 2.6Python 3Python 2 PythonPython 2Python 3

```
from __future__ import print_function
# other imports and instructions go after __future__
print('Hello world')
```

`__future__` [Python](#)

[2to3](#)PythonPython 2.xPython 3.x[Python](#)

[six](#)Python 2/3

- 
- / unicode
- 

Python 2Python 3

## Examples

Python 2 [print](#)

Python 2.x 2.7

```
print "Hello World"
print # print a newline
print "No newline", # add trailing comma to remove newline
print >>sys.stderr, "Error" # print to stderr
print("hello") # print "hello", since ("hello") == "hello"
print() # print an empty tuple "()"
print 1, 2, 3 # print space-separated arguments: "1 2 3"
print(1, 2, 3) # print tuple "(1, 2, 3)"
```

Python 3 [print\(\)](#)

Python 3.x 3.0



```
print "Hello World"          # SyntaxError
print("Hello World")
print()                      # print a newline (must use parentheses)
print("No newline", end="")  # end specifies what to append (defaults to newline)
print("Error", file=sys.stderr) # file specifies the output buffer
print("Comma", "separated", "output", sep=",") # sep specifies the separator
print("A", "B", "C", sep="") # null string for sep: prints as ABC
print("Flush this", flush=True) # flush the output buffer, added in Python 3.3
print(1, 2, 3)              # print space-separated arguments: "1 2 3"
print((1, 2, 3))           # print tuple "(1, 2, 3)"
```

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

sep◦

```
print('foo', 'bar', sep='~') # out: foo~bar
print('foo', 'bar', sep='.') # out: foo.bar
```

endprint◦

```
print('foo', 'bar', end='!') # out: foo bar!
```

```
print('foo', end='~')
print('bar')
# out: foo~bar
```

[print Python 2.6](#);print

```
from __future__ import print_function
```

Python 3flush◦

[PEP 3105](#) ◦

## Unicode

Python 2.x 2.7

Python 2 [str str unicode](#) ◦

Python 2 [str](#)◦

◦

```
s = 'Cafe' # type(s) == str
```

[uUnicode](#)

```
s = u'Café' # type(s) == unicode
b = 'Lorem ipsum' # type(b) == str
```

## Unicode

```
from __future__ import unicode_literals

s = 'Café' # type(s) == unicode
b = 'Lorem ipsum' # type(b) == unicode
```

## Unicode

```
isinstance(s, basestring)
```

## Python 3.x 3.0

### Python 3 `str`Unicode

```
s = 'Cafe' # type(s) == str
s = 'Café' # type(s) == str (note the accented trailing e)
```

### Python 3 `bytes` “blob” `bencode`

```
# Or, if you really need a byte string:
s = b'Cafe' # type(s) == bytes
s = 'Café'.encode() # type(s) == bytes
```

```
isinstance(s, str)
```

## Python 3.x 3.3

### `u`Python 2Python 3 `Unicode``u`

```
u'Cafe' == 'Cafe'
```

### Python 2Unicode`ur`

```
>>> ur'Café'
File "<stdin>", line 1
  ur'Café'
    ^
SyntaxError: invalid syntax
```

`encode` Python 3 text `str` `encode``bytes` `UTF-8`

`decode``bytes`Unicode

```
>>> b.decode()
'Café'
```

## Python 2.x 2.6

## Python 2.3 bytes unicode Python 2 Python 3 Unicode

```
from __future__ import unicode_literals
print(repr("hi"))
# u'hi'
```

## Python 3.x 3.0

## Python 3 int

```
b"abc"[0] == 97
```

## 11

```
b"abc"[0:1] == b"a"
```

## Python 3 unicode Python 2

```
# -*- coding: utf8 -*-
print("Hi, my name is Łukasz Langa.")
print(u"Hi, my name is Łukasz Langa."[::-1])
print("Hi, my name is Łukasz Langa."[::-1])

# Output in Python 2
# Hi, my name is Łukasz Langa.
# .agnaL zsakuŁ si eman ym ,iH
# .agnaL zsaku◆◆ si eman ym ,iH

# Output in Python 3
# Hi, my name is Łukasz Langa.
# .agnaL zsakuŁ si eman ym ,iH
# .agnaL zsakuŁ si eman ym ,iH
```

## / Python 3 Python 2

## Python 3x / y \_\_truediv\_\_ Python 2

	Python 2	Python 3
3 / 2	1	1.5
2 / 3	0	0.6666666666666666
-3 / 2	-2	-1.5

## Python 2.2 Python 2.7 Python 3

## Python 2 float Python 2 02/32 / 3.02.0 / 32.0/3.00.6666666666666666

	Python 2	Python 3
3.0 / 2.0	1.5	1.5
2 / 3.0	0.6666666666666666	0.6666666666666666
-3.0 / 2	-1.5	-1.5

**floor division** // ◦ //\_\_floordiv\_\_ ◦

	Python 2	Python 3
3 // 2	1	1
2 // 3	0	0
-3 // 2	-2	-2
3.0 // 2.0	1.0	1.0
2.0 // 3	0.0	0.0
-3 // 2.0	-2.0	-2.0

operator

```
from operator import truediv, floordiv
assert truediv(10, 8) == 1.25          # equivalent to `/` in Python 3
assert floordiv(10, 8) == 1           # equivalent to `//`
```

◦ /◦ from \_\_future\_\_ import division

```
# needs to be the first statement in a module
from __future__ import division
```

	Python 2	Python 3
3 / 2	1.5	1.5
2 / 3	0.6666666666666666	0.6666666666666666
-3 / 2	-1.5	-1.5

from \_\_future\_\_ import division/float/float

Python -3 / 2 == -1 ◦ ◦

**float** from \_\_future\_\_ import division/float 3 / 2.0 == 1.5 ◦ ◦ average = sum(items) / len(items)

float◦ floatint◦ Python 3 `3 / 2 == 1` True◦

Python 3 [PEP 238](#) ◦

---

*Simple Math* ◦

## Reduce

Python 2 [reduce](#) [functools](#) 2.6 Python 3 [reduce](#) [functools](#) ◦ [reduce](#) Python2 Python3

```
reduce(function_to_reduce, list_to_reduce)
```

◦ operator [truediv](#) ◦

Python 2.x

Python 2.x 2.3

```
>>> my_list = [1, 2, 3, 4, 5]
>>> import operator
>>> reduce(operator.truediv, my_list)
0.008333333333333333
```

Python 3.x

Python 3.x 3.0

```
>>> my_list = [1, 2, 3, 4, 5]
>>> import operator, functools
>>> functools.reduce(operator.truediv, my_list)
0.008333333333333333
```

```
from functools import reduce
```

## xrange

Python 2 [range](#) [xrange](#) [xrange](#) [index](#) [count](#)

Python 2.x 2.3

```
print(range(1, 10))
# Out: [1, 2, 3, 4, 5, 6, 7, 8, 9]

print(isinstance(range(1, 10), list))
# Out: True

print(xrange(1, 10))
# Out: xrange(1, 10)

print(isinstance(xrange(1, 10), xrange))
# Out: True
```

Python 3 [xrange](#) [range](#) ◦ [xrange](#)

## Python 3.x 3.0

```
print(range(1, 10))
# Out: range(1, 10)

print(isinstance(range(1, 10), range))
# Out: True

# print(xrange(1, 10))
# The output will be:
#Traceback (most recent call last):
# File "<stdin>", line 1, in <module>
#NameError: name 'xrange' is not defined
```

## Python 3.2 range indexcount

```
print(range(1, 10)[3:7])
# Out: range(3, 7)
print(range(1, 10).count(5))
# Out: 1
print(range(1, 10).index(7))
# Out: 6
```

## Python 2.x 2.3

```
# range(1000000000000000000)
# The output would be:
# Traceback (most recent call last):
# File "<stdin>", line 1, in <module>
# MemoryError

print(xrange(1000000000000000000))
# Out: xrange(1000000000000000000)
```

## Python 3. Python 3list() rangelist()

## Python 3.x 3.0

```
print(list(range(1, 10)))
# Out: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

## Python 2.xPython 3.x**builtinsfuture**

## Python 2.x 2.0

```
#forward-compatible
from builtins import range

for i in range(10**8):
    pass
```

## Python 3.x 3.0

```
#backward-compatible
```

```
from past.builtins import xrange

for i in xrange(10**8):
    pass
```

future.range Python indexcount Python 3.2+.

## Iterables

### Python 3.x 3.0

#### Python 3.0 . . . list

```
first, second, *tail, last = [1, 2, 3, 4, 5]
print(first)
# Out: 1
print(second)
# Out: 2
print(tail)
# Out: [3, 4]
print(last)
# Out: 5
```

\*variable variable. . .

```
first, second, *tail, last = [1, 2, 3, 4]
print(tail)
# Out: [3]

first, second, *tail, last = [1, 2, 3]
print(tail)
# Out: []
print(last)
# Out: 3
```

str

```
begin, *tail = "Hello"
print(begin)
# Out: 'H'
print(tail)
# Out: ['e', 'l', 'l', 'o']
```

date; \_year

```
person = ('John', 'Doe', (10, 16, 2016))
*_, (*_, year_of_birth) = person
print(year_of_birth)
# Out: 2016
```

\*\* -

```
*head, *tail = [1, 2]
```

```
# Out: SyntaxError: two starred expressions in assignment
```

## Python 3.x 3.5

◦ **\*\*\* Python 3.5** ◦

```
{*range(4), 4, *(5, 6, 7)}  
# Out: {0, 1, 2, 3, 4, 5, 6, 7}
```

## Python 2.x 2.0

### iterable

```
iterable = [1, 2, 3, 4, 5]  
print(iterable)  
# Out: [1, 2, 3, 4, 5]  
print(*iterable)  
# Out: 1 2 3 4 5
```

## Python 3.x 3.5

\*\* **PEP 448**

```
tail = {'y': 2, 'z': 3}  
{'x': 1, **tail}  
# Out: {'x': 1, 'y': 2, 'z': 3}
```

◦

```
dict1 = {'x': 1, 'y': 1}  
dict2 = {'y': 2, 'z': 3}  
{**dict1, **dict2}  
# Out: {'x': 1, 'y': 2, 'z': 3}
```

## Python 3.x 3.0

## Python 3◦ Python 3

```
# Works in Python 2, but syntax error in Python 3:  
map(lambda (x, y): x + y, zip(range(5), range(5)))  
# Same is true for non-lambdas:  
def example((x, y)):  
    pass  
  
# Works in both Python 2 and Python 3:  
map(lambda x: x[0] + x[1], zip(range(5), range(5)))  
# And non-lambdas, too:  
def working_example(x_y):  
    x, y = x_y  
    pass
```

**PEP 3113** ◦



Python2, raiseexcept

## Python 2.x 2.3

```
try:
    raise IOError, "input/output error"
except IOError, exc:
    print exc
```

## Python 3, as

```
try:
    raise IOError("input/output error")
except IOError as exc:
    print(exc)
```

## Python 3Python 2.6

---

## Python 3.x 3.0

## Python 3

```
try:
    file = open('database.db')
except FileNotFoundError as e:
    raise DatabaseError('Cannot open {}'.format(e))
```

exceptDatabaseError\_\_cause\_\_

```
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
FileNotFoundError
```

The above exception was the direct cause of the following exception:

```
Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
DatabaseError('Cannot open database.db')
```

except

```
try:
    file = open('database.db')
except FileNotFoundError as e:
    raise DatabaseError('Cannot open {}'.format(e))
```

```
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
FileNotFoundError
```

During handling of the above exception, another exception occurred:

```
Traceback (most recent call last):
```

```
File "<stdin>", line 4, in <module>
DatabaseError('Cannot open database.db')
```

## Python 2.x 2.0

### Python 2.x;except。

```
import sys
import traceback

try:
    funcWithError()
except:
    sys_vers = getattr(sys, 'version_info', (0,))
    if sys_vers < (3, 0):
        traceback.print_exc()
        raise Exception("new exception")
```

## Python 3.x 3.3

```
""" raise from None """ raise from None
```

```
try:
    file = open('database.db')
except FileNotFoundError as e:
    raise DatabaseError('Cannot open {}') from None
```

```
Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
DatabaseError('Cannot open database.db')
```

## Python 23

```
import six
try:
    file = open('database.db')
except FileNotFoundError as e:
    six.raise_from(DatabaseError('Cannot open {}'), None)
```

## .next

### Python 2<sub>next</sub>

### Python 2.x 2.3

```
g = (i for i in range(0, 3))
g.next() # Yields 0
g.next() # Yields 1
g.next() # Yields 2
```

Python 3 .next.\_\_next\_\_ “”.nextAttributeError。 Python 2Python 3<sub>next</sub> 。

## Python 3.x 3.0

```
g = (i for i in range(0, 3))
next(g) # Yields 0
next(g) # Yields 1
next(g) # Yields 2
```

## 2.6.

### Python 2.x 2.3

◦ ◦ None ◦ intstr tuplelist

```
[1, 2] > 'foo'
# Out: False
(1, 2) > 'foo'
# Out: True
[1, 2] > (1, 2)
# Out: False
100 < [1, 'x'] < 'xyz' < (1, 'x')
# Out: True
```

```
l = [7, 'x', (1, 2), [5, 6], 5, 8.0, 'y', 1.2, [7, 8], 'z']
sorted(l)
# Out: [1.2, 5, 7, 8.0, [5, 6], [7, 8], 'x', 'y', 'z', (1, 2)]
```

### Python 3.x 3.0

```
1 < 1.5
# Out: True

[1, 2] > 'foo'
# TypeError: unorderable types: list() > str()
(1, 2) > 'foo'
# TypeError: unorderable types: tuple() > str()
[1, 2] > (1, 2)
# TypeError: unorderable types: list() > tuple()
```

### Python 3

```
>>> list = [1, 'hello', [3, 4], {'python': 2}, 'stackoverflow', 8, {'python': 3}, [5, 6]]
>>> sorted(list, key=str)
# Out: [1, 8, [3, 4], [5, 6], 'hello', 'stackoverflow', {'python': 2}, {'python': 3}]
```

strkey◦ [ ' {0-9◦

### Python 2raw\_input

### Python 2.x 2.3

```
user_input = raw_input()
```

### Python 3input◦

### Python 3.x 3.0

```
user_input = input()
```

Python 2 `input()` Python 3 `eval(input())`

## Python

```
try:  
    input = raw_input  
except NameError:  
    pass
```

Python 3 Python 2 `has_key` `has_key` `iter*view*` `d.has_key(key)` `key in d` `key in d`

Python 2 `keys` `values` `items` Python 3;

- `len`
- 

◦

Python 2.7 Python 3; `viewkeys` `viewvalues` `viewitems` Python 2 Python 3

- `d.keys()` Python 2 `d.keys()` `d.values()` `d.items()` `list(d.keys())` `list(d.values())` `list(d.items())`
- `d.iterkeys()` `d.itervalues()` `d.iteritems()` `iter(d.keys())` `iter(d)` ; `iter(d.values())` `iter(d.items())`
- Python 2.7 `d.viewkeys()` `d.viewvalues()` `d.viewitems()` `d.keys()` `d.values()` `d.items()` ◦

Python 2 ◦

```
d = {'a': 0, 'b': 1, 'c': 2, '!': 3}  
for key in d.keys():  
    if key.isalpha():  
        del d[key]
```

Python 3 `keys` Python 3 `RuntimeError: dictionary changed size during iteration` `RuntimeError: dictionary changed size during iteration` ◦ `for key in list(d)` ◦

`next()` ;; Python 2 `d.iterkeys()` `d.itervalues()` `d.iteritems()` `iter(d)` `iter(d.values())` Python 3 `iter(d.values())` `iter(d.items())` ◦

## exec Python 3

Python 2 `exec code [in globals[, locals]]` Python 3 `exec exec(code, [, globals[, locals]])`

Python 2 `SyntaxError` ◦

`print statement` `__future__ import` ◦ `from __future__ import exec_function` Python 2 `exec` Python 3 `exec` ◦

Python 2.x 2.3

```
exec 'code'
exec 'code' in global_vars
exec 'code' in global_vars, local_vars
```

## Python 3.x 3.0

```
exec('code')
exec('code', global_vars)
exec('code', global_vars, local_vars)
```

## Python 2Python 3.

## Python 2hasattr

### Python 2 hasattrFalse .

```
class A(object):
    @property
    def get(self):
        raise IOError

class B(object):
    @property
    def get(self):
        return 'get in b'

a = A()
b = B()

print 'a hasattr get: ', hasattr(a, 'get')
# output False in Python 2 (fixed, True in Python 3)
print 'b hasattr get', hasattr(b, 'get')
# output True in Python 2 and Python 3
```

## Python3. Python 2

```
try:
    a.get
except AttributeError:
    print("no get property!")
```

### getattr

```
p = getattr(a, "get", None)
if p is not None:
    print(p)
else:
    print("no get property!")
```

<code>_winreg</code>	WinReg
<code>ConfigParser</code>	<code>configparser</code>

copy_reg	copyreg
SocketServer	SocketServer
_markupbase	markupbase
	reprlib
test.test_support	test.support
Tkinter	Tkinter
tkFileDialog	tkinter.filedialog
urllib / urllib2	urlliburllib.parseurllib.errorurllib.responseurllib.requesturllib.robotparser

◦ tkinterurllib◦

Python 2.x3.xfuturePython 2.xPython 3.x◦

## Python 2

```
>>> 0755 # only Python 2
```

```
0o755 # both Python 2 and Python 3
```

## Python 3“”◦

Python 3.x ;pythonobject ◦ objectclass

## Python 3.x 3.0

```
class X: pass
class Y(object): pass
```

mroobject

## Python 3.x 3.0

```
>>> X.__mro__
(__main__.X, object)
```

```
>>> Y.__mro__
(__main__.Y, object)
```

Python 2.x;object◦ objectclass

## Python 2.x 2.3

```
class X: pass
class Y(object): pass
```

`Y.__mro__` Python 3.x

## Python 2.x 2.3

```
>>> Y.__mro__
(<class '__main__.Y'>, <type 'object'>)
```

Y class Y(object): pass ◦ X `__mro__AttributeError` ◦

## Python object

```
class mycls(object):
    """I am fully compatible with Python 2/3"""
```

`__metaclass__` type object

```
__metaclass__ = type

class mycls:
    """I am also fully compatible with Python 2/3"""
```

## <> `=repr

Python 2 <> !=; `foo` repr(foo) ◦

## Python 2.x 2.7

```
>>> 1 <> 2
True
>>> 1 <> 1
False
>>> foo = 'hello world'
>>> repr(foo)
'hello world'
>>> `foo`
'hello world'
```

## Python 3.x 3.0

```
>>> 1 <> 2
File "<stdin>", line 1
  1 <> 2
    ^
SyntaxError: invalid syntax
>>> `foo`
File "<stdin>", line 1
  `foo`
    ^
SyntaxError: invalid syntax
```

/

## Python 2.x 2.7

```
"1deadbeef3".decode('hex')
# Out: '\x1d\xea\xdb\xee\xf3'
'\x1d\xea\xdb\xee\xf3'.encode('hex')
# Out: 1deadbeef3
```

## Python 3.x 3.0

```
"1deadbeef3".decode('hex')
# Traceback (most recent call last):
#   File "<stdin>", line 1, in <module>
# AttributeError: 'str' object has no attribute 'decode'

b"1deadbeef3".decode('hex')
# Traceback (most recent call last):
#   File "<stdin>", line 1, in <module>
# LookupError: 'hex' is not a text encoding; use codecs.decode() to handle arbitrary codecs

'\x1d\xea\xdb\xee\xf3'.encode('hex')
# Traceback (most recent call last):
#   File "<stdin>", line 1, in <module>
# LookupError: 'hex' is not a text encoding; use codecs.encode() to handle arbitrary codecs

b'\x1d\xea\xdb\xee\xf3'.encode('hex')
# Traceback (most recent call last):
#   File "<stdin>", line 1, in <module>
# AttributeError: 'bytes' object has no attribute 'encode'
```

### codecs

```
import codecs
codecs.decode('1deadbeef4', 'hex')
# Out: b'\x1d\xea\xdb\xee\xf4'
codecs.encode(b'\x1d\xea\xdb\xee\xf4', 'hex')
# Out: b'1deadbeef4'
```

### codecs.encodebytes◦ strdecodeASCII

```
codecs.encode(b'\x1d\xea\xdb\xee\xff', 'hex').decode('ascii')
# Out: '1deadbeeff'
```

## Python 3cmp

### Python 3 `cmp__cmp__`◦

`cmp()__cmp__()◦ __lt__() __eq__() __hash__()__hash__()◦ cmp() (a > b) - (a < b) cmp(a, b)◦`

`cmpkey` keyword only◦

`functools.cmp_to_key(func)` `cmp` style`key` style



◦ `sorted()` `min()` `max()` `heapq.nlargest()` `heapq.nsmallest()` `itertools.groupby()` ◦

Python 2◦

## Python 2.x 2.3

```
x = 'hello world!'
vowels = [x for x in 'AEIOU']

print (vowels)
# Out: ['A', 'E', 'I', 'O', 'U']
print(x)
# Out: 'U'
```

## Python 3.x 3.0

```
x = 'hello world!'
vowels = [x for x in 'AEIOU']

print (vowels)
# Out: ['A', 'E', 'I', 'O', 'U']
print(x)
# Out: 'hello world!'
```

Python 2 `xhello world!U x`◦

Python 3 `xhello world!` ◦

2.5PythonPython 3Python 2.7Python 2◦

Python 2Python 3for

```
x = 'hello world!'
vowels = []
for x in 'AEIOU':
    vowels.append(x)
print(x)
# Out: 'U'
```

`map()` `iterable`◦ Python 2 `map`◦ Python 3 `map`*map* ◦

```
# Python 2.X
>>> map(str, [1, 2, 3, 4, 5])
['1', '2', '3', '4', '5']
>>> type(_)
>>> <class 'list'>

# Python 3.X
>>> map(str, [1, 2, 3, 4, 5])
<map object at 0x*>
>>> type(_)
<class 'map'>

# We need to apply map again because we "consumed" the previous map....
>>> map(str, [1, 2, 3, 4, 5])
>>> list(_)
```

```
['1', '2', '3', '4', '5']
```

Python 2<sup>None</sup> Python 3<sup>◦</sup>

Python 2.x 2.3

```
>>> map(None, [0, 1, 2, 3, 0, 4])
[0, 1, 2, 3, 0, 4]
```

Python 3.x 3.0

```
>>> list(map(None, [0, 1, 2, 3, 0, 5]))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'NoneType' object is not callable
```

Python 2<sup>None map itertools.izip\_longest</sup> Python 3<sup>◦</sup>

Python 2

Python 2.x 2.3

```
>>> map(None, [1, 2, 3], [1, 2], [1, 2, 3, 4, 5])
[(1, 1, 1), (2, 2, 2), (3, None, 3), (None, None, 4), (None, None, 5)]
```

Python 3

Python 3.x 3.0

```
>>> list(map(lambda x, y, z: (x, y, z), [1, 2, 3], [1, 2], [1, 2, 3, 4, 5]))
[(1, 1, 1), (2, 2, 2)]

# to obtain the same padding as in Python 2 use zip_longest from itertools
>>> import itertools
>>> list(itertools.zip_longest([1, 2, 3], [1, 2], [1, 2, 3, 4, 5]))
[(1, 1, 1), (2, 2, 2), (3, None, 3), (None, None, 4), (None, None, 5)]
```

Python 2/3<sup>map</sup> ◦ map(str, [1, 2, 3, 4, 5])

```
>>> [str(i) for i in [1, 2, 3, 4, 5]]
['1', '2', '3', '4', '5']
```

filtermapzip

Python 2.x 2.7

Python 2 <sup>filter mapzip</sup> ◦ mapzipfilter

```
>>> s = filter(lambda x: x.isalpha(), 'alb2c3')
>>> s
'abc'
>>> s = map(lambda x: x * x, [0, 1, 2])
```

```
>>> s
[0, 1, 4]
>>> s = zip([0, 1, 2], [3, 4, 5])
>>> s
[(0, 3), (1, 4), (2, 5)]
```

## Python 3.x 3.0

### Python 3 [filter](#) [map](#) [zip](#)

```
>>> it = filter(lambda x: x.isalpha(), 'alb2c3')
>>> it
<filter object at 0x00000098A55C2518>
>>> ''.join(it)
'abc'
>>> it = map(lambda x: x * x, [0, 1, 2])
>>> it
<map object at 0x000000E0763C2D30>
>>> list(it)
[0, 1, 4]
>>> it = zip([0, 1, 2], [3, 4, 5])
>>> it
<zip object at 0x000000E0763C52C8>
>>> list(it)
[(0, 3), (1, 4), (2, 5)]
```

### Python 2 [itertools.izip](#) Python 3 [zip](#) [izip](#) Python 3

/

### Python 3 [PEP 404](#) Python 2 `from ... import *`

### Python 2 Python 3

- `from __future__ import absolute_import`
- 

## Python 2

```
import foo
```

### `import foo`

```
from .moduleY import spam
from .moduleY import spam as ham
from . import moduleY
from ..subpackage1 import moduleY
from ..subpackage2.moduleZ import eggs
from ..moduleA import foo
from ...package import bar
from ...sys import path
```

..

shapes ◦

```
shapes
├── __init__.py
│
├── circle.py
│
├── square.py
│
└── triangle.py
```

circle.py square.py triangle.py util.py ◦

```
from . import util # use util.PI, util.sq(x), etc
```

```
from .util import * #use PI, sq(x), etc to call functions
```

. ◦

shapes

```
shapes
├── __init__.py
│
├── circle
│   ├── __init__.py
│   └── circle.py
│
├── square
│   ├── __init__.py
│   └── square.py
│
├── triangle
│   ├── __init__.py
│   └── triangle.py
│
└── util.py
```

### 3util.py

```
from .. import util # use util.PI, util.sq(x), etc
```

```
from ..util import * # use PI, sq(x), etc to call functions
```

.. ◦ .S ◦

## I/O.

file3.X open ◦

I/OioStringIO

```
import io
assert io.open is open # the builtin is an alias
buffer = io.StringIO()
buffer.write('hello, ') # returns number of characters written
buffer.write('world!\n')
buffer.getvalue() # 'hello, world!\n'
```

```
with open('data.txt') as f:
    first_line = next(f)
    assert type(first_line) is str
with open('data.bin', 'rb') as f:
    first_kb = f.read(1024)
    assert type(first_kb) is bytes
```

`locale.getpreferredencoding(False)` `locale.getpreferredencoding(False)` ◦ `encoding`

```
with open('old_japanese_poetry.txt', 'shift_jis') as text:
    haiku = text.read()
```

## round

Python 2 [round\(\)](#) ◦

Python 2.x 2.7

```
round(1.5) # Out: 2.0
round(0.5) # Out: 1.0
round(-0.5) # Out: -1.0
round(-1.5) # Out: -2.0
```

Python 3 [round\(\)](#) ◦

Python 3.x 3.0

```
round(1.5) # Out: 2
round(0.5) # Out: 0
round(-0.5) # Out: 0
round(-1.5) # Out: -2
```

round [half even](#) `round(2.5)` **23.0** ◦

◦

[IEEE 754.NET](#) ◦

◦ ◦ ◦

---

## round

`round()` Python 2.7 `float`

## Python 2.x 2.7

```
round(4.8)
# 5.0
```

## Python 3.0`int` ◦

## Python 3.x 3.0

```
round(4.8)
# 5
```

## Python 2 `True False None` ◦ ◦

## Python 2.x 2.0

```
True, False = False, True
True # False
False # True
```

## Python 2.4`None` ◦

## Python 2.x 2.4

```
None = None # SyntaxError: cannot assign to None
```

## Python 3 `True False None` ◦

## Python 3.x 3.0

```
True, False = False, True # SyntaxError: can't assign to keyword
None = None # SyntaxError: can't assign to keyword
```

## Python 2`None`

## Python 2.x 2.3

```
hi = sys.stdout.write('hello world\n')
# Out: hello world
type(hi)
# Out: <type 'NoneType'>
```

## Python 3

## Python 3.x 3.0

```
import sys

char_count = sys.stdout.write('hello world \n')
# Out: hello world 
char_count
# Out: 14
```

```

byte_count = sys.stdout.buffer.write(b'hello world \xf0\x9f\x90\x8d\n')
# Out: hello world []
byte_count
# Out: 17

```

## Python 2C ssize\_tlongL° 32Python

### Python 2.x 2.7

```

>>> 2**31
2147483648L
>>> type(2**31)
<type 'long'>
>>> 2**30
1073741824
>>> type(2**30)
<type 'int'>
>>> 2**31 - 1 # 2**31 is long and long - int is long
2147483647L

```

### Python 3long;int °

### Python 3.x 3.0

```

2**1024
# Output:
179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021...

print(-(2**1024))
# Output: -
179769313486231590772930519078902473361797697894230657273430081157732675805500963132708477322407536021...

type(2**1024)
# Output: <class 'int'>

```

### Python 2.x 2.7

### Python 2\_\_nonzero\_\_° True°

```

class MyClass:
    def __nonzero__(self):
        return False

my_instance = MyClass()
print bool(MyClass) # True
print bool(my_instance) # False

```

### Python 3.x 3.0

### Python 3\_\_bool\_\_\_\_nonzero\_\_

```

class MyClass:
    def __bool__(self):
        return False

```

```
my_instance = MyClass()
print(bool(MyClass))    # True
print(bool(my_instance)) # False
```

Python 2Python 3 <https://riptutorial.com/zh-TW/python/topic/809/python-2python-3>



---

# 64: intfloatstrtuplefrozensets

## Examples

```
foo = "bar"  
foo[0] = "c" # Error
```

◦

```
foo = ("bar", 1, "Hello!",)  
foo[1] = 2 # ERROR!!
```

◦ ◦

## Frozenset

```
foo = frozenset(["bar", 1, "Hello!"])  
foo[2] = 7 # ERROR  
foo.add(3) # ERROR
```

- frozensets◦

[intfloatstrtuplefrozensets](https://riptutorial.com/zh-TW/python/topic/4806/-int-float-str-tuple-frozensets-) <https://riptutorial.com/zh-TW/python/topic/4806/-int-float-str-tuple-frozensets->

# 65:

GILpython。 IOCPU。 CPUmultiprocessing。

GILCPythonPythonPyPy。 [JythonIronPythonGIL](#)。

## Examples

```
import multiprocessing

def fib(n):
    """computing the Fibonacci in an inefficient way
    was chosen to slow down the CPU."""
    if n <= 2:
        return 1
    else:
        return fib(n-1)+fib(n-2)
p = multiprocessing.Pool()
print(p.map(fib, [38, 37, 36, 35, 34, 33]))

# Out: [39088169, 24157817, 14930352, 9227465, 5702887, 3524578]
```

fib**1.8**。

Python 2.2+

## ParentChildren

### child.py

```
import time

def main():
    print "starting work"
    time.sleep(1)
    print "work work work work work"
    time.sleep(1)
    print "done working"

if __name__ == '__main__':
    main()
```

### parent.py

```
import os

def main():
    for i in range(5):
        os.system("python child.py &")

if __name__ == '__main__':
    main()
```

HTTP//。 **child.py**。 Redis。

## C

CPythonCGIL。

```
#include "Python.h"
...
PyObject *pyfunc(PyObject *self, PyObject *args) {
    ...
    Py_BEGIN_ALLOW_THREADS
    // Threaded C code
    ...
    Py_END_ALLOW_THREADS
    ...
}
```

## PyPar

PyParMPIPython。 PyPar<https://github.com/daleroberts/pypar>

```
import pypar as pp

ncpus = pp.size()
rank = pp.rank()
node = pp.get_processor_name()

print 'I am rank %d of %d on node %s' % (rank, ncpus, node)

if rank == 0:
    msh = 'P0'
    pp.send(msg, destination=1)
    msg = pp.receive(source=rank-1)
    print 'Processor 0 received message "%s" from rank %d' % (msg, rank-1)
else:
    source = rank-1
    destination = (rank+1) % ncpus
    msg = pp.receive(source)
    msg = msg + 'P' + str(rank)
    pypar.send(msg, destination)
pp.finalize()
```

<https://riptutorial.com/zh-TW/python/topic/542/>

## 66:

- packfmtv1v2...
- unpackfmtbuffer

## Examples

```
from struct import pack

print(pack('I3c', 123, b'a', b'b', b'c')) # b'\x00\x00\x00abc'
```

```
from struct import unpack

print(unpack('I3c', b'\x00\x00\x00abc')) # (123, b'a', b'b', b'c')
```

“**struct**”pythonpython。

pack。。

```
import struct
import sys
print "Native byteorder: ", sys.byteorder
# If no byteorder is specified, native byteorder is used
buffer = struct.pack("ihb", 3, 4, 5)
print "Byte chunk: ", repr(buffer)
print "Byte chunk unpacked: ", struct.unpack("ihb", buffer)
# Last element as unsigned short instead of unsigned char ( 2 Bytes)
buffer = struct.pack("ihh", 3, 4, 5)
print "Byte chunk: ", repr(buffer)
```

```
'\ x03 \ x00 \ x00 \ x00 \ x04 \ x00 \ x05'3,4,5'\ x03 \ x00 \ x00 \ x00 \ x04 \ x00 \ X05 \
X00'
```

。

```
import struct
# If no byteorder is specified, native byteorder is used
buffer = struct.pack("hhh", 3, 4, 5)
print "Byte chunk native byte order: ", repr(buffer)
buffer = struct.pack("!hhh", 3, 4, 5)
print "Byte chunk network byte order: ", repr(buffer)
```

```
'\ x03 \ x00 \ x04 \ x00 \ x05 \ x00'
```

```
'\ x00 \ x03 \ x00 \ x04 \ x00 \ x05'
```

。

```
import struct
```

```
from ctypes import create_string_buffer
bufferVar = create_string_buffer(8)
bufferVar2 = create_string_buffer(8)
# We use a buffer that has already been created
# provide format, buffer, offset and data
struct.pack_into("hhh", bufferVar, 0, 3, 4, 5)
print "Byte chunk: ", repr(bufferVar.raw)
struct.pack_into("hhh", bufferVar2, 2, 3, 4, 5)
print "Byte chunk: ", repr(bufferVar2.raw)
```

'\x03\x00\x04\x00\x05\x00\x00\x00'

'\x00\x00\x03\x00\x04\x00\x05\x00'

<https://riptutorial.com/zh-TW/python/topic/2978/>

# 67:

Python . . . .

## Examples

	ns	
	ns__main__	
	ns__main__	
	ns	
	ns	
exec	ns	
eval()		ns
execfile() execfile()		ns
input()		ns

[https://riptutorial.com/zh-TW/python/topic/10741/-](https://riptutorial.com/zh-TW/python/topic/10741/)

# 68: “pip”PyPI

pythonpip◦ ImportError◦ Windows Python\_root/Scripts/pip.exePython\_root/Scripts/pip.exe  
\_\_main\_\_.pypipmain◦ pippip◦ pip pipPyPI Package Manager

- pip◦ <function | attribute | class>function
  - ◦ ◦ shellbashzshfish◦
  - check\_isolated
    - param args {list}
    - {boolean}
  - create\_main\_parser
    - {pip.baseparser.ConfigOptionParser}
  - =
    - param args {list}
    - {integer}0
  - parseopts
    - param args {list}
  - get\_installed\_distributions
    - {list}
  - get\_similar\_commands
    - ◦
    - {string}
    - {boolean}
  - get\_summaries=
    - ◦
  - get\_prog
    - {string}
  - dist\_is\_editableDIST
    - 
    - param dist {object}
    - {boolean}
  - commands\_dict
    - {}

## Examples

```
import pip

command = 'install'
parameter = 'selenium'
second_param = 'numpy' # You can give as many package names as needed
switch = '--upgrade'

pip.main([command, parameter, second_param, switch])
```

pip.main(['freeze'])pip.main(['freeze', '', ''])◦

/"1".

```
import pip

installed = pip.get_installed_distributions()
list = []
for i in installed:
    list.append(i.key)

pip.main(['install']+list+['--upgrade'])
```

o

```
for i in installed:
    pip.main(['install']+i.key+['--upgrade'])
```

## ImportError

python.

```
if __name__ == '__main__':
    try:
        import requests
    except ImportError:
        print("To use this module you need 'requests' module")
        t = input('Install requests? y/n: ')
        if t == 'y':
            import pip
            pip.main(['install', 'requests'])
            import requests
            import os
            import sys
            pass
        else:
            import os
            import sys
            print('Some functionality can be unavailable.')
else:
    import requests
    import os
    import sys
```

3.43.6. .whlwheel. o scikit\_learn-0.18.1-cp36-cp36m-win\_amd64.whl [package\_name] - [version] - [python interpreter] - [python-interpreter] - [Operating System] .whl. python. pipkjhfkjdf.whl is not a valid wheel filename. o

.whl7-zip. - o site-packages

"pip"PyPI <https://riptutorial.com/zh-TW/python/topic/10730/-pip--pypi>



# 69: AMQPStormRabbitMQ

## AMQPStormpipipip

```
pip install amqpstorm
```

## Examples

### RabbitMQ

◦

```
from amqpstorm import Connection
```

◦ `start_consumingto_tuple`

◦ `RabbitMQ`

```
def on_message(message):
    """This function is called on message received.

    :param message: Delivered message.
    :return:
    """
    print("Message:", message.body)

    # Acknowledge that we handled the message without any issues.
    message.ack()

    # Reject the message.
    # message.reject()

    # Reject the message, and put it back in the queue.
    # message.reject(requeue=True)
```

`RabbitMQ`

```
connection = Connection('127.0.0.1', 'guest', 'guest')
```

◦ ◦

```
channel = connection.channel()
```

`RabbitMQ` ◦ `on_message` ◦

`RabbitMQ` ◦ `simple_queue` `RabbitMQ` ◦

```
channel.basic.consume(callback=on_message, queue='simple_queue', no_ack=False)
```

## IORabbitMQ。

```
channel.start_consuming(to_tuple=False)
```

## RabbitMQ

。

```
from amqpstorm import Connection
from amqpstorm import Message
```

## RabbitMQ。

```
connection = Connection('127.0.0.1', 'guest', 'guest')
```

。。

```
channel = connection.channel()
```

。

```
# Message Properties.
properties = {
    'content_type': 'text/plain',
    'headers': {'key': 'value'}
}

# Create the message.
message = Message.create(channel=channel, body='Hello World!', properties=properties)
```

publishrouting\_key。 simple\_queue。

```
message.publish(routing_key='simple_queue')
```

## RabbitMQ

。 ;confirm delivery delivery\_modedurable。 RabbitMQ。

。

```
channel.queue.bind(exchange='amq.direct', routing_key='hello', queue='hello')
```

。

```
delay_channel.queue.declare(queue='hello_delay', durable=True, arguments={
    'x-message-ttl': 5000,
    'x-dead-letter-exchange': 'amq.direct',
    'x-dead-letter-routing-key': 'hello'
```

```
})
```

- [x-message-ttl](#) -

- 

- [X-](#)

- 

- [X-](#)

hello\_delayhelloExchange◦

Pika◦

```
delay_channel.basic.publish(exchange='',
                             routing_key='hello_delay',
                             body='test',
                             properties={'delivery_mod': 2})
```

RabbitMQ◦

Overview					Messages			Messag	
Name	Exclusive	Parameters	Policy	Status	Ready	Unacked	Total	incoming	deliv
hello		D		Idle	1	0	1		
hello_delay		TTL DLX DLK D		Idle	0	0	0	0.00/s	

◦

```
from amqpstorm import Connection

connection = Connection('127.0.0.1', 'guest', 'guest')

# Create normal 'Hello World' type channel.
channel = connection.channel()
channel.confirm_deliveries()
channel.queue.declare(queue='hello', durable=True)

# We need to bind this channel to an exchange, that will be used to transfer
# messages from our delay queue.
channel.queue.bind(exchange='amq.direct', routing_key='hello', queue='hello')

# Create our delay channel.
delay_channel = connection.channel()
delay_channel.confirm_deliveries()

# This is where we declare the delay, and routing for our delay channel.
delay_channel.queue.declare(queue='hello_delay', durable=True, arguments={
    'x-message-ttl': 5000, # Delay until the message is transferred in milliseconds.
    'x-dead-letter-exchange': 'amq.direct', # Exchange used to transfer the message from A to
    B.
    'x-dead-letter-routing-key': 'hello' # Name of the queue we want the message transferred
```

```
to.  
})  
  
delay_channel.basic.publish(exchange='',  
                             routing_key='hello_delay',  
                             body='test',  
                             properties={'delivery_mode': 2})  
  
print("[x] Sent")
```

AMQPStormRabbitMQ <https://riptutorial.com/zh-TW/python/topic/3373/amqpstormrabbitmq>

---

# 70: Pandas

pandas.

## Examples

### Pandas

```
# Print the working directory
import os
print os.getcwd()
# C:\Python27\Scripts

# Set the working directory
os.chdir('C:/Users/general1/Documents/simple Python files')
print os.getcwd()
# C:\Users\general1\Documents\simple Python files

# load pandas
import pandas as pd

# read a csv data file named 'small_dataset.csv' containing 4 lines and 3 variables
my_data = pd.read_csv("small_dataset.csv")
my_data
#      x  y  z
# 0    1  2  3
# 1    4  5  6
# 2    7  8  9
# 3   10 11 12

my_data.shape      # number of rows and columns in data set
# (4, 3)

my_data.shape[0]   # number of rows in data set
# 4

my_data.shape[1]   # number of columns in data set
# 3

# Python uses 0-based indexing. The first row or column in a data set is located
# at position 0. In R the first row or column in a data set is located
# at position 1.

# Select the first two rows
my_data[0:2]
#      x  y  z
#0    1  2  3
#1    4  5  6

# Select the second and third rows
my_data[1:3]
#      x  y  z
# 1    4  5  6
# 2    7  8  9

# Select the third row
```

```
my_data[2:3]
#   x  y  z
#2  7  8  9

# Select the first two elements of the first column
my_data.iloc[0:2, 0:1]
#   x
# 0  1
# 1  4

# Select the first element of the variables y and z
my_data.loc[0, ['y', 'z']]
# y    2
# z    3

# Select the first three elements of the variables y and z
my_data.loc[0:2, ['y', 'z']]
#   y  z
# 0  2  3
# 1  5  6
# 2  8  9

# Write the first three elements of the variables y and z
# to an external file. Here index = 0 means do not write row names.

my_data2 = my_data.loc[0:2, ['y', 'z']]

my_data2.to_csv('my.output.csv', index = 0)
```

**Pandas** <https://riptutorial.com/zh-TW/python/topic/8854/pandas->

# 71: PythonWindows

WindowsUI。 WindowsPowershell""。 Web。 WindowsPython。

## Examples

### Python

[pywin32](#) Python for Windows。 Python。

```
import win32serviceutil
import win32service
import win32event
import servicemanager
import socket

class AppServerSvc (win32serviceutil.ServiceFramework):
    _svc_name_ = "TestService"
    _svc_display_name_ = "Test Service"

    def __init__(self, args):
        win32serviceutil.ServiceFramework.__init__(self, args)
        self.hWaitStop = win32event.CreateEvent (None, 0, 0, None)
        socket.setdefaulttimeout (60)

    def SvcStop(self):
        self.ReportServiceStatus (win32service.SERVICE_STOP_PENDING)
        win32event.SetEvent (self.hWaitStop)

    def SvcDoRun(self):
        servicemanager.LogMsg (servicemanager.EVENTLOG_INFORMATION_TYPE,
                               servicemanager.PYS_SERVICE_STARTED,
                               (self._svc_name_, ''))
        self.main()

    def main(self):
        pass

if __name__ == '__main__':
    win32serviceutil.HandleCommandLine (AppServerSvc)
```

- main。
- ◦ GUI。 Python

```
nssm install MyServiceName c:\python27\python.exe c:\temp\myscript.py
```

my\_script.pymain。 PythonPython。

Windows Server Resource Kit。

## Flask Web

◦ main()run()◦ WSGIRequestHandler◦

```
import win32serviceutil
import win32service
import win32event
import servicemanager
from multiprocessing import Process

from app import app

class Service(win32serviceutil.ServiceFramework):
    _svc_name_ = "TestService"
    _svc_display_name_ = "Test Service"
    _svc_description_ = "Tests Python service framework by receiving and echoing messages over a named pipe"

    def __init__(self, *args):
        super().__init__(*args)

    def SvcStop(self):
        self.ReportServiceStatus(win32service.SERVICE_STOP_PENDING)
        self.process.terminate()
        self.ReportServiceStatus(win32service.SERVICE_STOPPED)

    def SvcDoRun(self):
        self.process = Process(target=self.main)
        self.process.start()
        self.process.run()

    def main(self):
        app.run()

if __name__ == '__main__':
    win32serviceutil.HandleCommandLine(Service)
```

<http://stackoverflow.com/a/25130524/318488>

[PythonWindows https://riptutorial.com/zh-TW/python/topic/9065/pythonwindows](https://riptutorial.com/zh-TW/python/topic/9065/pythonwindows)



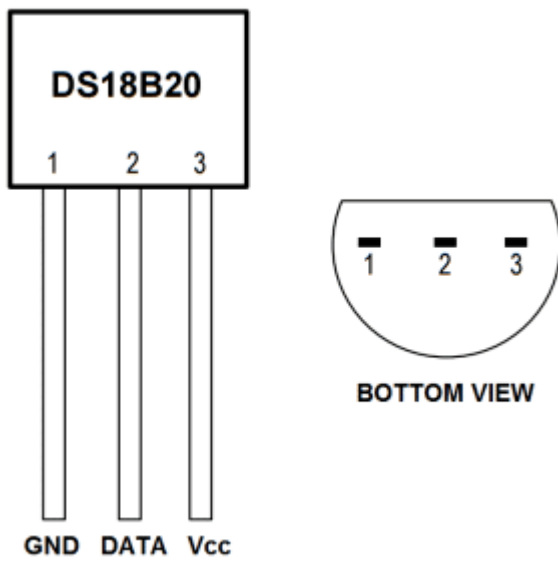
# 72: PythonRaspberry Pi

## Examples

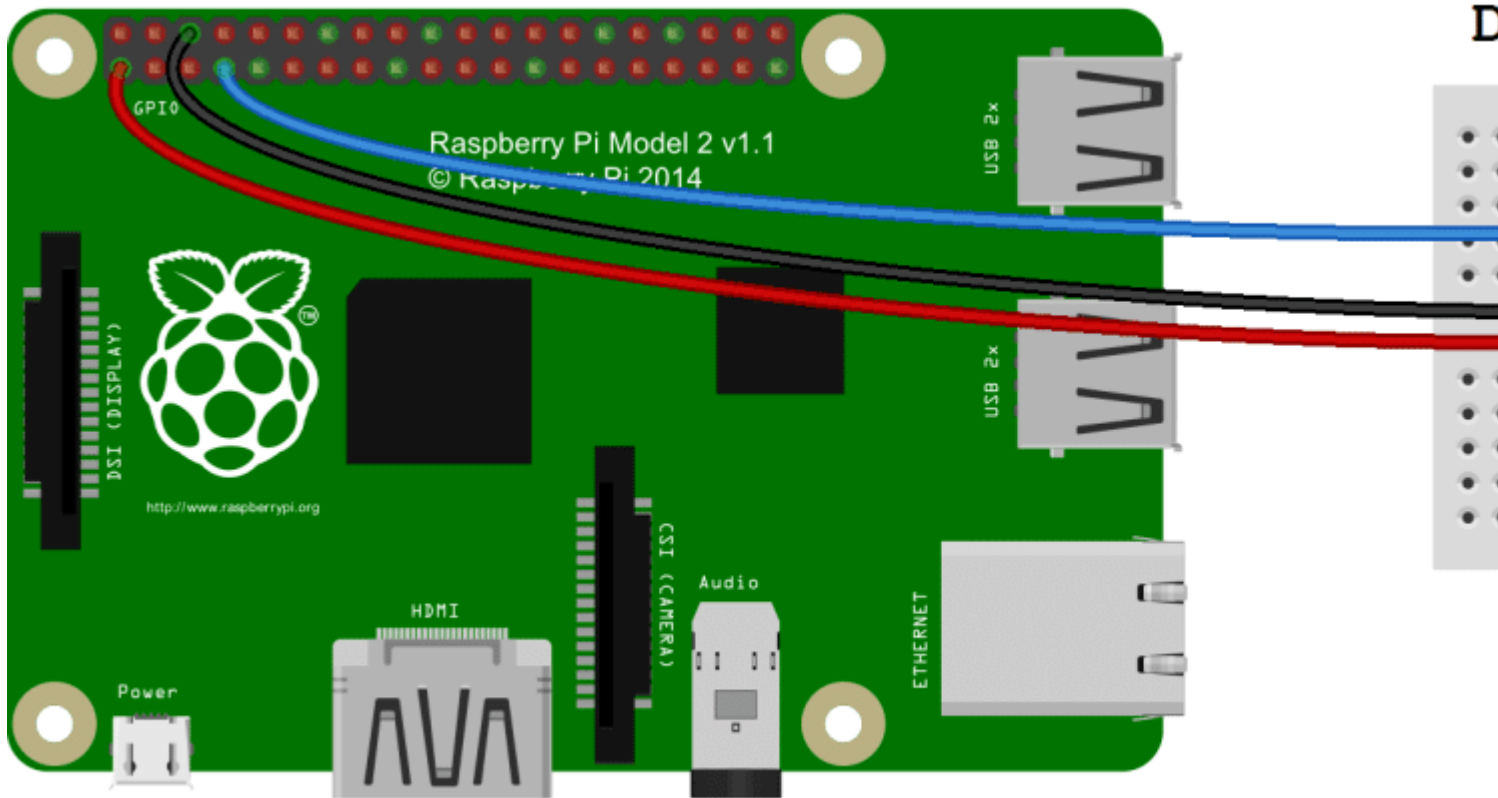
-

DS18B20Raspberry pi

**DS18B20Raspberry pi**



1. VCC
2. GND
- 3.



## R14.7k

1. **Vcc**Raspberry pi5v3.3vPIN01,02,04,17。
2. **Gnd**Raspberry piGndPIN06,09,14,20,25。
3. PIN07

## RPi

4. puttylinux / unixRaspberry pi。

5. /boot/config.txt。

```
nano /boot/config.txt
```

6. dtoverlay=w1-gpio。

7. Raspberry pi `sudo reboot` 。

8. Raspberry pi`sudo modprobe g1-gpio`

9. `sudo modprobe w1-therm`

10. / `sys / bus / w1 / devices` `cd /sys/bus/w1/devices`

11. 28 - \*\*\*\*\*。

12. `cd 28-*****`

### 13. w1-slave CRC. cat w1-slave .

#### python

```
import glob
import time

RATE = 30
sensor_dirs = glob.glob("/sys/bus/w1/devices/28*")

if len(sensor_dirs) != 0:
    while True:
        time.sleep(RATE)
        for directories in sensor_dirs:
            temperature_file = open(directories + "/w1_slave")
            # Reading the files
            text = temperature_file.read()
            temperature_file.close()
            # Split the text with new lines (\n) and select the second line.
            second_line = text.split("\n")[1]
            # Split the line into words, and select the 10th word
            temperature_data = second_line.split(" ")[9]
            # We will read after ignoring first two character.
            temperature = float(temperature_data[2:])
            # Now normalise the temperature by dividing 1000.
            temperature = temperature / 1000
            print 'Address : '+str(directories.split('/')[-1])+', Temperature : '+str(temperature)
```

python. RATE.

#### GPIO

1. [ [https://www.element14.com/community/servlet/JiveServlet/previewBody/73950-102-11-339300/pi3\\_gpio.png](https://www.element14.com/community/servlet/JiveServlet/previewBody/73950-102-11-339300/pi3_gpio.png)] [3]

PythonRaspberry PI <https://riptutorial.com/zh-TW/python/topic/10735/pythonraspberry-pi>

---

# 73: PythonWeb

“” ° ° °

---

## WebPython

[requests](#)

HTTP°

[requests-cache](#)

requests ;° ° .....°

[scrapy](#)

Webrequests°

[selenium](#)

Selenium WebDriverPython° requestsHTTP° requestsJavaScript°

## HTML

[BeautifulSoup](#)

HTMLXMLPythonHTML Parser [html5lib](#) [lxml](#)[lxml.html](#)

[lxml](#)

HTMLXML° CSSXPathHTML°

## Examples

### lxml

```
# For Python 2 compatibility.
from __future__ import print_function

import lxml.html
import requests

def main():
    r = requests.get("https://httpbin.org")
    html_source = r.text
    root_element = lxml.html.fromstring(html_source)
```

```

# Note root_element.xpath() gives a *list* of results.
# XPath specifies a path to the element we want.
page_title = root_element.xpath('/html/head/title/text()')[0]
print(page_title)

if __name__ == '__main__':
    main()

```

## Web

### cookie requests.Session TCP

```

import requests

with requests.Session() as session:
    # all requests through session now have User-Agent header set
    session.headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36'}

    # set cookies
    session.get('http://httpbin.org/cookies/set?key=value')

    # get cookies
    response = session.get('http://httpbin.org/cookies')
    print(response.text)

```

## Scrapy

### Scrapy

```
scrapy startproject projectName
```

### StackOverflow

```

import scrapy

class StackOverflowSpider(scrapy.Spider):
    name = 'stackoverflow' # each spider has a unique name
    start_urls = ['http://stackoverflow.com/questions?sort=votes'] # the parsing starts from
a specific set of urls

    def parse(self, response): # for each request this generator yields, its response is sent
to parse_question
        for href in response.css('.question-summary h3 a::attr(href)'): # do some scraping
stuff using css selectors to find question urls
            full_url = response.urljoin(href.extract())
            yield scrapy.Request(full_url, callback=self.parse_question)

    def parse_question(self, response):
        yield {
            'title': response.css('h1 a::text').extract_first(),
            'votes': response.css('.question .vote-count-post::text').extract_first(),
            'body': response.css('.question .post-text').extract_first(),
            'tags': response.css('.question .post-tag::text').extract(),
            'link': response.url,

```

```
}
```

```
projectName\spiders - projectName\spiders\stackoverflow_spider.py
```

o

```
scrapy crawl stackoverflow
```

## Scrapy

Scrapy "Scrapy/VERSION (+http://scrapy.org)" o **settings.py** o

```
#USER_AGENT = 'projectName (+http://www.yourdomain.com)'
```

```
USER_AGENT = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.103 Safari/537.36'
```

## BeautifulSoup4

```
from bs4 import BeautifulSoup
import requests

# Use the requests module to obtain a page
res = requests.get('https://www.codechef.com/problems/easy')

# Create a BeautifulSoup object
page = BeautifulSoup(res.text, 'lxml') # the text field contains the source of the page

# Now use a CSS selector in order to get the table containing the list of problems
datatable_tags = page.select('table.dataTable') # The problems are in the <table> tag,
# with class "dataTable"

# We extract the first tag from the list, since that's what we desire
datatable = datatable_tags[0]
# Now since we want problem names, they are contained in <b> tags, which are
# directly nested under <a> tags
prob_tags = datatable.select('a > b')
prob_names = [tag.getText().strip() for tag in prob_tags]

print prob_names
```

## Selenium WebDriver

o o SeleniumWeb o

```
from selenium import webdriver

browser = webdriver.Firefox() # launch firefox browser

browser.get('http://stackoverflow.com/questions?sort=votes') # load url

title = browser.find_element_by_css_selector('h1').text # page title (first h1 element)
```

```

questions = browser.find_elements_by_css_selector('.question-summary') # question list

for question in questions: # iterate over questions
    question_title = question.find_element_by_css_selector('.summary h3 a').text
    question_excerpt = question.find_element_by_css_selector('.summary .excerpt').text
    question_vote = question.find_element_by_css_selector('.stats .vote .votes .vote-count-
post').text

    print "%s\n%s\n%s\n%s votes\n-----\n" % (question_title, question_excerpt,
question_vote)

```

◦ `cookieJavaScript`◦

## urllib.requestWeb

urllib.requestWeb

```

from urllib.request import urlopen

response = urlopen('http://stackoverflow.com/questions?sort=votes')
data = response.read()

# The received bytes should usually be decoded according the response's character set
encoding = response.info().get_content_charset()
html = data.decode(encoding)

```

Python 2◦

```

from subprocess import Popen, PIPE
from lxml import etree
from io import StringIO

```

```

user_agent = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/55.0.2883.95 Safari/537.36'
url = 'http://stackoverflow.com'
get = Popen(['curl', '-s', '-A', user_agent, url], stdout=PIPE)
result = get.stdout.read().decode('utf8')

```

-s

-A

```

tree = etree.parse(StringIO(result), etree.HTMLParser())
divs = tree.xpath('//div')

```

PythonWeb <https://riptutorial.com/zh-TW/python/topic/1792/pythonweb>

# 74: Python

## Examples

### Matplotlib

[Matplotlib Python](#) ◦

[matplotlib SO Docs](#) ◦

### Matplotlib

- [matplotlib pyplot MATLAB](#) ◦
- [matplotlib](#) ◦ ◦

pyplot

```
import matplotlib.pyplot as plt

# Generate some data for plotting.
x = [0, 1, 2, 3, 4, 5, 6]
y = [i**2 for i in x]

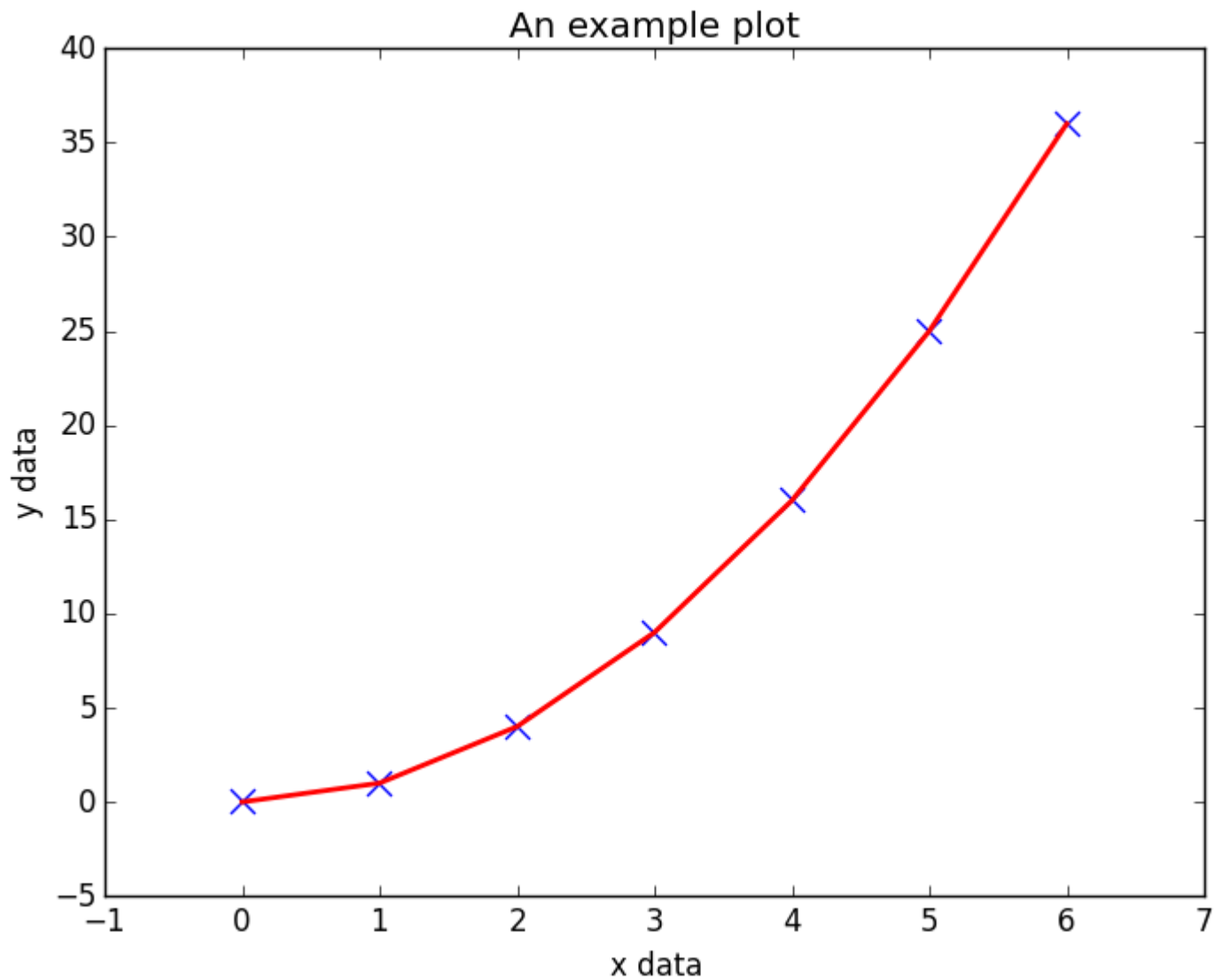
# Plot the data x, y with some keyword arguments that control the plot style.
# Use two different plot commands to plot both points (scatter) and a line (plot).

plt.scatter(x, y, c='blue', marker='x', s=100) # Create blue markers of shape "x" and size 100
plt.plot(x, y, color='red', linewidth=2) # Create a red line with linewidth 2.

# Add some text to the axes and a title.
plt.xlabel('x data')
plt.ylabel('y data')
plt.title('An example plot')

# Generate the plot and show to the user.
plt.show()
```





```
matplotlib.pyplot plt.show() plt.show(block=True)◦ plt.show(block=True)◦
```

## Seaborn

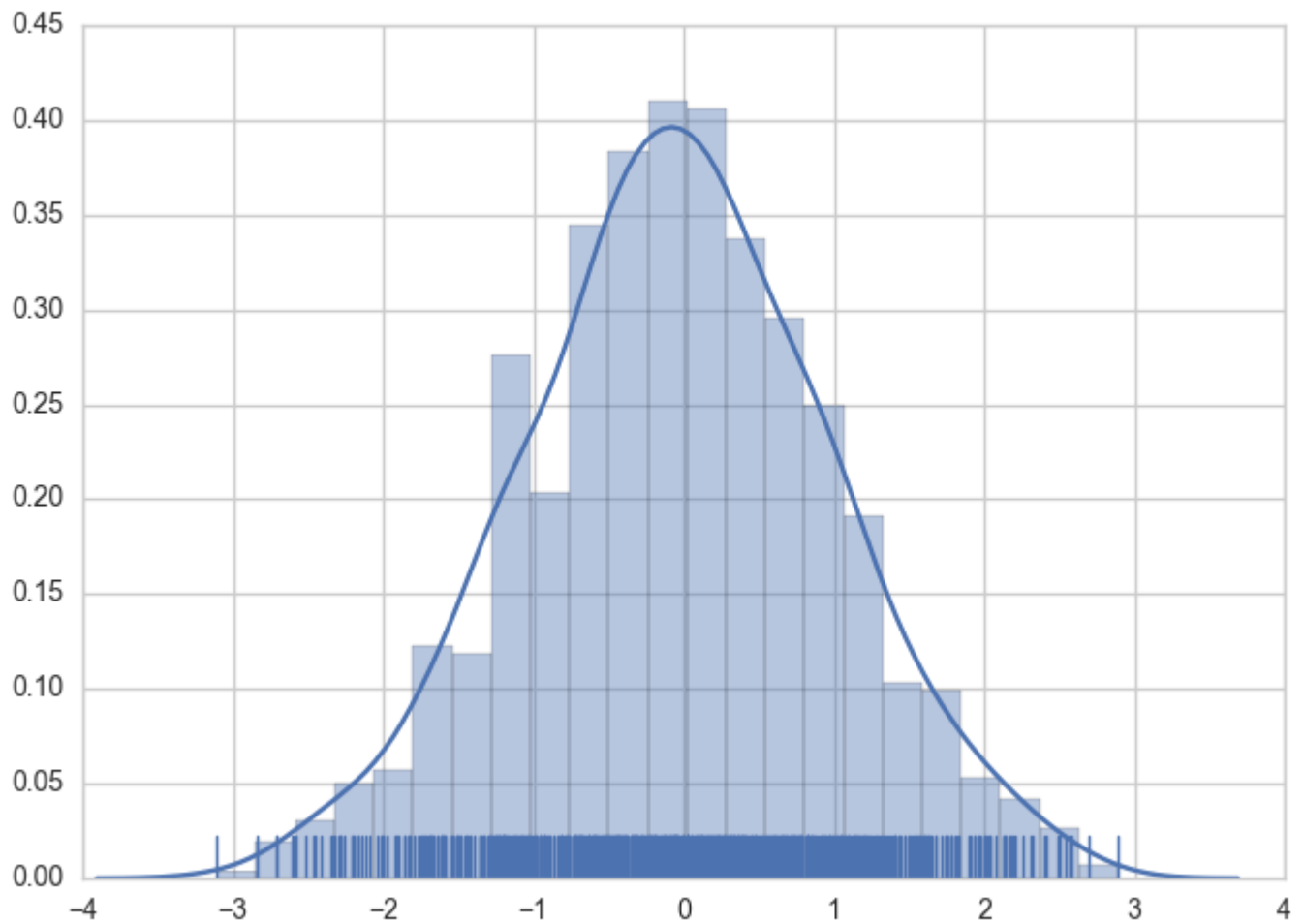
[SeabornMatplotlib](#)◦ ◦ [SeabornAPI](#)◦

Seaborn◦ ◦

```
import numpy as np # numpy used to create data from plotting
import seaborn as sns # common form of importing seaborn

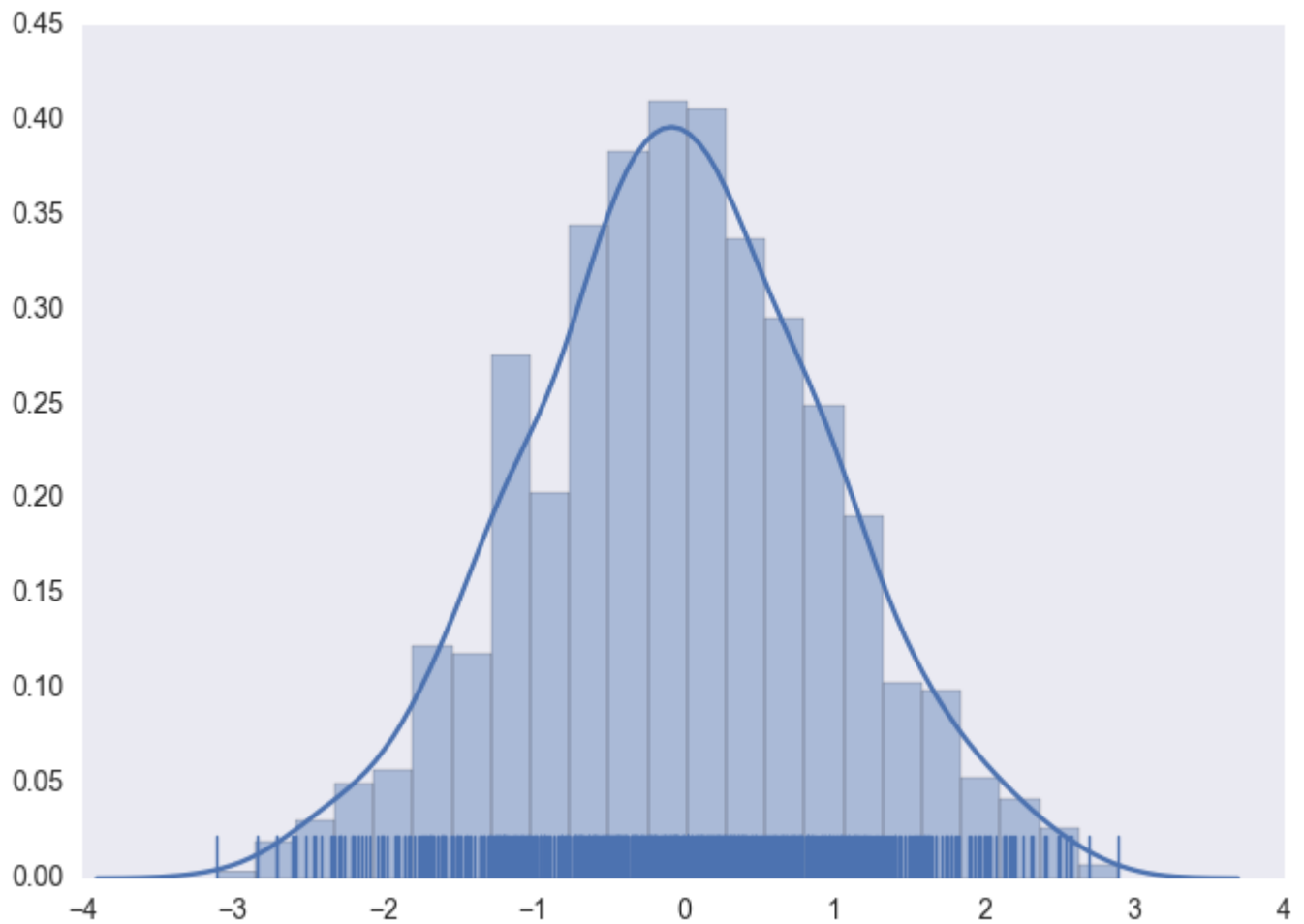
# Generate normally distributed data
data = np.random.randn(1000)

# Plot a histogram with both a rugplot and kde graph superimposed
sns.distplot(data, kde=True, rug=True)
```



o

```
# Using previously created imports and data.  
  
# Use a dark background with no grid.  
sns.set_style('dark')  
# Create the plot again  
sns.distplot(data, kde=True, rug=True)
```

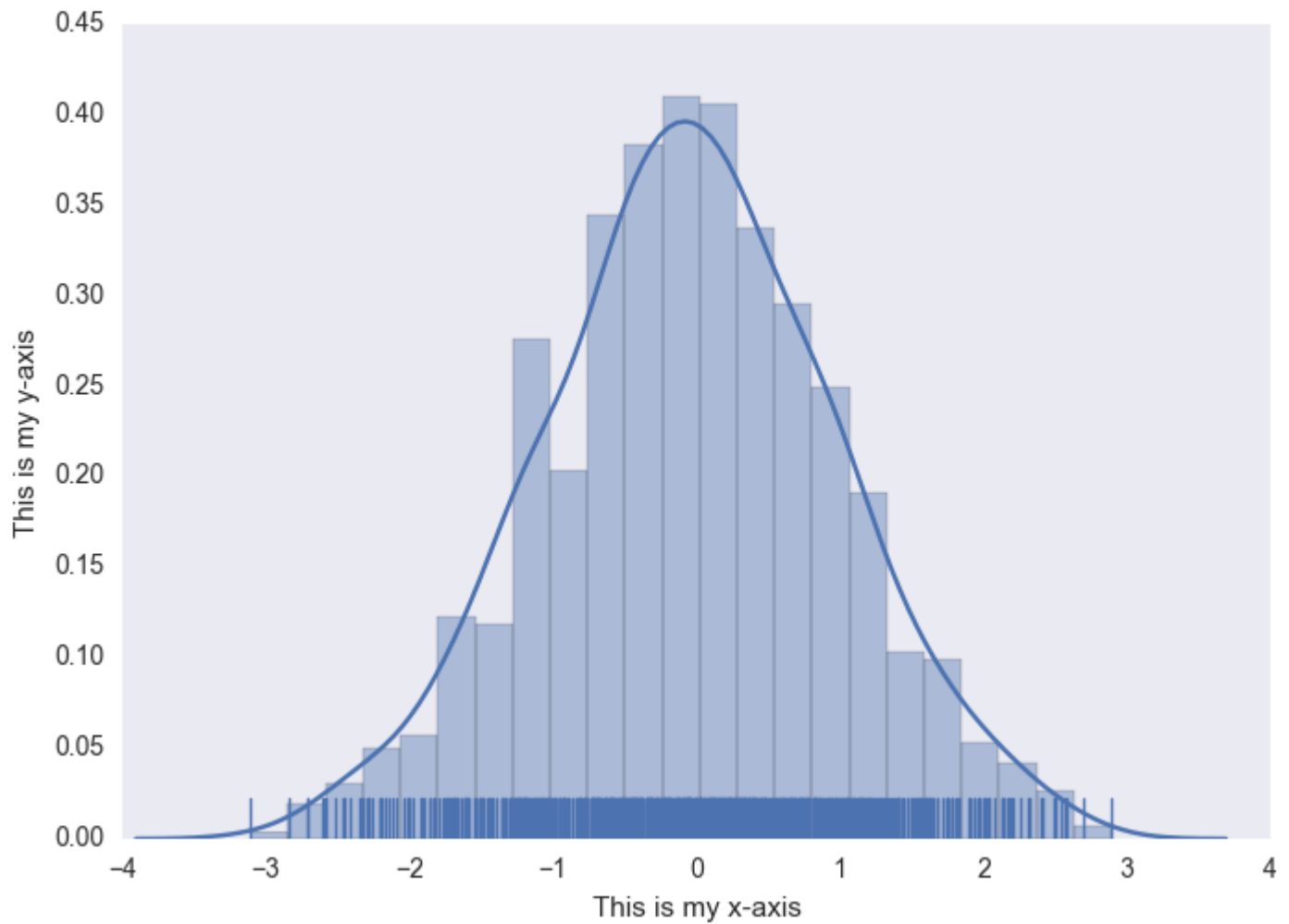


matplotlibSeaborn. ◦

```
# Using previously created data and style

# Access to matplotlib commands
import matplotlib.pyplot as plt

# Previously created plot.
sns.distplot(data, kde=True, rug=True)
# Set the axis labels.
plt.xlabel('This is my x-axis')
plt.ylabel('This is my y-axis')
```



## Mayavi

[MayaVI](#)。 [VTK](#)。 [VTK MayaVI](#)。 [MatplotlibVTK](#)。

## MayaVIPython 2.7xPython 3-x Python 3

。

## MayaVI。

```
# Author: Gael Varoquaux <gael.varoquaux@normalesup.org>
# Copyright (c) 2007, Enthought, Inc.
# License: BSD Style.

from numpy import sin, cos, mgrid, pi, sqrt
from mayavi import mlab

mlab.figure(fgcolor=(0, 0, 0), bgcolor=(1, 1, 1))
u, v = mgrid[- 0.035:pi:0.01, - 0.035:pi:0.01]

X = 2 / 3. * (cos(u) * cos(2 * v)
            + sqrt(2) * sin(u) * cos(v)) * cos(u) / (sqrt(2) -
```

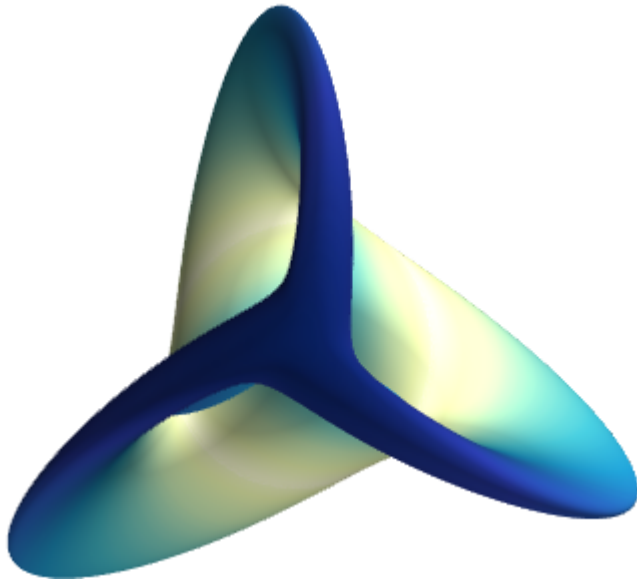
```

sin(2 * u) * sin(3 * v))
Y = 2 / 3. * (cos(u) * sin(2 * v) -
             sqrt(2) * sin(u) * sin(v)) * cos(u) / (sqrt(2)
             - sin(2 * u) * sin(3 * v))
Z = -sqrt(2) * cos(u) * cos(u) / (sqrt(2) - sin(2 * u) * sin(3 * v))
S = sin(u)

mlab.mesh(X, Y, Z, scalars=S, colormap='YlGnBu', )

# Nice view from the front
mlab.view(.0, - 5.0, 4)
mlab.show()

```



## Plotly

[Plotly](#) ◦ [Plotly Python](#) ◦ [R](#) ◦ [JavaScript](#) ◦ [Julia](#) ◦ [MATLAB](#) ◦ [Web](#) ◦

[plotly](#) ◦ [Jupyter](#) ◦

◦ ◦

250 ◦ ◦ ◦

```

import plotly.graph_objs as go
import plotly as ply

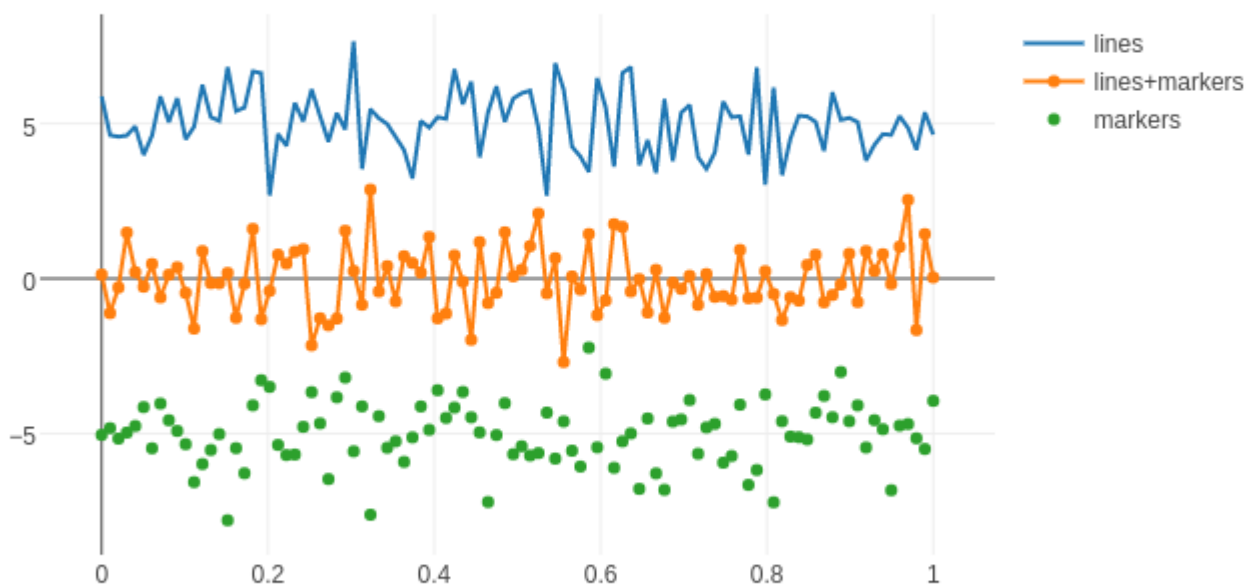
# Create random data with numpy
import numpy as np

N = 100
random_x = np.linspace(0, 1, N)
random_y0 = np.random.randn(N)+5
random_y1 = np.random.randn(N)
random_y2 = np.random.randn(N)-5

# Create traces
trace0 = go.Scatter(
    x = random_x,

```

```
y = random_y0,  
mode = 'lines',  
name = 'lines'  
)  
trace1 = go.Scatter(  
    x = random_x,  
    y = random_y1,  
    mode = 'lines+markers',  
    name = 'lines+markers'  
)  
trace2 = go.Scatter(  
    x = random_x,  
    y = random_y2,  
    mode = 'markers',  
    name = 'markers'  
)  
data = [trace0, trace1, trace2]  
  
ply.offline.plot(data, filename='line-mode')
```



Python <https://riptutorial.com/zh-TW/python/topic/2388/python>

# 75: ZIP

- zipfile
- class zipfile. ZipFile filemode='r'compression = ZIP\_STOREDallowZip64 = True

ZIPzipfile.BadZipFile ◦

Python 2.7zipfile.BadZipfile Python 3.2+

## Examples

### Zip

zipfile◦

```
import zipfile
filename = 'zipfile.zip'
```

zipzip ◦

```
zip = zipfile.ZipFile(filename)
print(zip)
# <zipfile.ZipFile object at 0x0000000002E51A90>
zip.close()
```

Python 2.73.2Python 3with context manager. “”

```
with zipfile.ZipFile(filename, 'r') as z:
    print(zip)
# <zipfile.ZipFile object at 0x0000000002E51A90>
```

### Zipfile

zipfile◦ printdirstdout

```
with zipfile.ZipFile(filename) as zip:
    zip.printdir()

# Out:
# File Name                Modified                Size
# pyexpat.pyd              2016-06-25 22:13:34    157336
# python.exe               2016-06-25 22:13:34     39576
# python3.dll              2016-06-25 22:13:34     51864
# python35.dll             2016-06-25 22:13:34   3127960
# etc.
```

namelistnamelist ◦

```
with zipfile.ZipFile(filename) as zip:
    print(zip.namelist())

# Out: ['pyexpat.pyd', 'python.exe', 'python3.dll', 'python35.dll', ... etc. ...]
```

infolistnamelist ZipInfo

```
with zipfile.ZipFile(filename) as zip:
    info = zip.infolist()
    print(zip[0].filename)
    print(zip[0].date_time)
    print(info[0].file_size)

# Out: pyexpat.pyd
# Out: (2016, 6, 25, 22, 13, 34)
# Out: 157336
```

## zip

### zip

```
import zipfile
with zipfile.ZipFile('zipfile.zip','r') as zfile:
    zfile.extractall('path')
```

```
import zipfile
f=open('zipfile.zip','rb')
zfile=zipfile.ZipFile(f)
for cont in zfile.namelist():
    zfile.extract(cont,path)
```

### zipfile

```
import zipfile
new_arch=zipfile.ZipFile("filename.zip",mode="w")
```

### write

```
new_arch.write('filename.txt','filename_in_archive.txt') #first parameter is filename and
second parameter is filename in archive by default filename will taken if not provided
new_arch.close()
```

### writestr

```
str_bytes="string buffer"
new_arch.writestr('filename_string_in_archive.txt',str_bytes)
new_arch.close()
```

**ZIP** <https://riptutorial.com/zh-TW/python/topic/3728/zip>



---

# 76:

## Examples

```
def func(myList):  
    for item in myList:  
        print(item)
```

```
func([1,2,3,5,7])
```

```
1  
2  
3  
5  
7
```

```
aList = ['a','b','c','d']  
func(aList)
```

```
a  
b  
c  
d
```

<https://riptutorial.com/zh-TW/python/topic/7744/>

# 77:

◦ ;◦ Python◦ BaseException◦

- 
- #re-raise
- #Python 3 -
- NonePython 3 -
- 
- [[]]
- 
- 

## Examples

◦

```
def even_the_odds(odds):
    if odds % 2 != 1:
        raise ValueError("Did not get an odd number")

    return odds + 1
```

try...except:◦

```
try:
    x = 5 / 0
except ZeroDivisionError as e:
    # `e` is the exception object
    print("Got a divide by zero! The exception was:", e)
    # handle exceptional case
    x = 0
finally:
    print "The END"
    # it runs no matter what execute.
```

- ZeroDivisionError - ◦

ZeroDivisionErrorArithmeticError

```
>>> ZeroDivisionError.__bases__
(<class 'ArithmeticError'>,)
```

ZeroDivisionError

```
try:
    5 / 0
except ArithmeticError:
    print("Got arithmetic error")
```

◦

tryfinally◦

```
resource = allocate_some_expensive_resource()
try:
    do_stuff(resource)
except SomeException as e:
    log_error(e)
    raise # re-raise the error
finally:
    free_expensive_resource(resource)
```

with◦

◦ ◦

raise◦

```
try:
    5 / 0
except ZeroDivisionError:
    print("Got an error")
    raise
```

◦ ◦

```
try:
    5 / 0
except ZeroDivisionError as e:
    raise ZeroDivisionError("Got an error", e)
```

raiseraise◦

**Python 3** raise - from

```
raise ZeroDivisionError("Got an error") from e
```

◦ IOError◦

**Python 3.x 3.0**

```
>>> try:
    5 / 0
except ZeroDivisionError as e:
    raise ValueError("Division failed") from e
```

```
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
ZeroDivisionError: division by zero
```

The above exception was the direct cause of the following exception:

```
Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
ValueError: Division failed
```

o

IOErrorOSErrorEnvironmentError o IOErrorOSError o EnvironmentErrorIOErrorOSError o

## Python 2.x 2.3

```
BaseException
+-- SystemExit
+-- KeyboardInterrupt
+-- GeneratorExit
+-- Exception
    +-- StopIteration
    +-- StandardError
        | +-- BufferError
        | +-- ArithmeticError
        | | +-- FloatingPointError
        | | +-- OverflowError
        | | +-- ZeroDivisionError
        | +-- AssertionError
        | +-- AttributeError
        | +-- EnvironmentError
        | | +-- IOError
        | | +-- OSError
        | | +-- WindowsError (Windows)
        | | +-- VMSError (VMS)
        | +-- EOFError
        | +-- ImportError
        | +-- LookupError
        | | +-- IndexError
        | | +-- KeyError
        | +-- MemoryError
        | +-- NameError
        | | +-- UnboundLocalError
        | +-- ReferenceError
        | +-- RuntimeError
        | | +-- NotImplementedError
        | +-- SyntaxError
        | | +-- IndentationError
        | | +-- TabError
        | +-- SystemError
        | +-- TypeError
        | +-- ValueError
        | | +-- UnicodeError
        | | +-- UnicodeDecodeError
        | | +-- UnicodeEncodeError
        | | +-- UnicodeTranslateError
    +-- Warning
        +-- DeprecationWarning
        +-- PendingDeprecationWarning
        +-- RuntimeWarning
        +-- SyntaxWarning
        +-- UserWarning
        +-- FutureWarning
    +-- ImportWarning
```

```
+-- UnicodeWarning
+-- BytesWarning
```

## Python 3.x 3.0

```
BaseException
+-- SystemExit
+-- KeyboardInterrupt
+-- GeneratorExit
+-- Exception
    +-- StopIteration
    +-- StopAsyncIteration
    +-- ArithmeticError
    |   +-- FloatingPointError
    |   +-- OverflowError
    |   +-- ZeroDivisionError
    +-- AssertionError
    +-- AttributeError
    +-- BufferError
    +-- EOFError
    +-- ImportError
    +-- LookupError
    |   +-- IndexError
    |   +-- KeyError
    +-- MemoryError
    +-- NameError
    |   +-- UnboundLocalError
    +-- OSError
    |   +-- BlockingIOError
    |   +-- ChildProcessError
    |   +-- ConnectionError
    |   |   +-- BrokenPipeError
    |   |   +-- ConnectionAbortedError
    |   |   +-- ConnectionRefusedError
    |   |   +-- ConnectionResetError
    |   +-- FileExistsError
    |   +-- FileNotFoundError
    |   +-- InterruptedError
    |   +-- IsADirectoryError
    |   +-- NotADirectoryError
    |   +-- PermissionError
    |   +-- ProcessLookupError
    |   +-- TimeoutError
    +-- ReferenceError
    +-- RuntimeError
    |   +-- NotImplementedError
    |   +-- RecursionError
    +-- SyntaxError
    |   +-- IndentationError
    |   +-- TabError
    +-- SystemError
    +-- TypeError
    +-- ValueError
    |   +-- UnicodeError
    |   |   +-- UnicodeDecodeError
    |   |   +-- UnicodeEncodeError
    |   |   +-- UnicodeTranslateError
    +-- Warning
    |   +-- DeprecationWarning
    |   +-- PendingDeprecationWarning
```

```
+-- RuntimeError
+-- SyntaxWarning
+-- UserWarning
+-- FutureWarning
+-- ImportError
+-- UnicodeWarning
+-- BytesWarning
+-- ResourceWarning
```

BaseException Python ◦ Python raise Python ◦

```
>>> def failing_function():
...     raise ValueError('Example error!')
>>> failing_function()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in failing_function
ValueError: Example error!
```

'Example error!' ValueError 'Example error!' failing\_function() ◦

```
>>> try:
...     failing_function()
... except ValueError:
...     print('Handled the error')
Handled the error
```

except...

```
>>> try:
...     failing_function()
... except ValueError as e:
...     print('Caught exception', repr(e))
Caught exception ValueError('Example error!',)
```

Python Python <https://docs.python.org/3.5/library/exceptions.html> ◦ ◦

Exception

```
class FooException(Exception):
    pass
try:
    raise FooException("insert description here")
except FooException:
    print("A FooException was raised.")
```

```
class NegativeError(ValueError):
    pass

def foo(x):
    # function that only accepts positive values of x
    if x < 0:
        raise NegativeError("Cannot process negative numbers")
    ... # rest of function body
try:
```

```

    result = foo(int(input("Enter a positive integer: "))) # raw_input in Python 2.x
except NegativeError:
    print("You entered a negative number!")
else:
    print("The result was " + str(result))

```

Exception

```

try:
    very_difficult_function()
except Exception:
    # log / try to reconnect / exit gracefully
finally:
    print "The END"
    # it runs no matter what execute.

```

BaseExceptionException

```

try:
    even_more_difficult_function()
except:
    pass # do whatever needed

```

◦ SystemExit KeyboardInterrupt MemoryError ◦ ◦ ◦

“bug” ◦ ◦

◦

◦

◦ KeyErrorAttributeError◦

```

try:
    d = {}
    a = d[1]
    b = d.non_existing_field
except (KeyError, AttributeError) as e:
    print("A KeyError or an AttributeError exception has been caught.")

```

◦ KeyErrorAttributeError ◦

```

try:
    d = {}
    a = d[1]
    b = d.non_existing_field
except KeyError as e:
    print("A KeyError has occurred. Exception message:", e)
except AttributeError as e:
    print("An AttributeError has occurred. Exception message:", e)

```

input ◦ ◦ try / except

## Python 3.x 3.0

```
while True:
    try:
        nb = int(input('Enter a number: '))
        break
    except ValueError:
        print('This is not a number, try again.')
```

Python 2.x raw\_input ; input Python 2.x ◦ input2 + 2 ◦

ValueError ◦ except ◦ break ◦ nb ◦

range(n) d d[i] ◦

```
d = [{7: 3}, {25: 9}, {38: 5}]

for i in range(len(d)):
    do_stuff(i)
    try:
        dic = d[i]
        i += dic[i]
    except KeyError:
        i += 1
```

KeyError ◦

try else ◦ ◦

```
try:
    data = {1: 'one', 2: 'two'}
    print(data[1])
except KeyError as e:
    print('key not found')
else:
    raise ValueError()
# Output: one
# Output: ValueError
```

else: if else elif ◦ if else:

```
try:
    ...
except ...:
    ...
else:
    if ...:
        ...
    elif ...:
        ...
    else:
```



...

<https://riptutorial.com/zh-TW/python/topic/1788/>

# 78:

◦ Python value = 1, 2, 3 ◦

- 1a""#a
- 
- 1a 1◦ (1)◦
- 1,2,331,2,3

◦

◦ ◦

## Examples

```
x = (1, 2, 3)
x[0] # 1
x[1] # 2
x[2] # 3
x[3] # IndexError: tuple index out of range
```

-1

```
x[-1] # 3
x[-2] # 2
x[-3] # 1
x[-4] # IndexError: tuple index out of range
```

```
print(x[:-1]) # (1, 2)
print(x[-1:]) # (3,)
print(x[1:3]) # (2, 3)
```

Python listtuple◦

```
>>> t = (1, 4, 9)
>>> t[0] = 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

list.append.extend◦ +=

```
>>> t = (1, 2)
>>> q = t
>>> t += (3, 4)
>>> t
```

```
(1, 2, 3, 4)
>>> q
(1, 2)
```

lists ◦ ◦

```
>>> t = (1, 2, 3, [1, 2, 3])
(1, 2, 3, [1, 2, 3])
>>> t[3] += [4, 5]
```

```
TypeError: 'tuple' object does not support item assignment
>>> t
(1, 2, 3, [1, 2, 3, 4, 5])
```

+="" - "" ◦

◦

```
hash( (1, 2) ) # ok
hash( ([], {"hello"}) ) # not ok, since lists and sets are not hashable
```

setdict ◦

```
{ (1, 2) } # ok
{ ([], {"hello"}) } # not ok
```

```
t = 'a', 'b', 'c', 'd', 'e'
```

```
t = ('a', 'b', 'c', 'd', 'e')
```

```
t0 = ()
type(t0) # <type 'tuple'>
```

```
t1 = 'a',
type(t1) # <type 'tuple'>
```

```
t2 = ('a')
type(t2) # <type 'str'>
```

◦

```
t2 = ('a',)
type(t2) # <type 'tuple'>
```

PEP8 ◦ PEP8

```
t2 = ('a',) # PEP8-compliant
```

```
t2 = 'a',          # this notation is not recommended by PEP8
t2 = ('a', )      # this notation is not recommended by PEP8
```

tuple ◦

```
t = tuple('lupins')
print(t)          # ('l', 'u', 'p', 'i', 'n', 's')
t = tuple(range(3))
print(t)          # (0, 1, 2)
```

Allen B. Downey “Think Python”。

Python ◦

```
a = 1, 2, 3  # a is the tuple (1, 2, 3)
```

```
a = (1, 2, 3) # a is the tuple (1, 2, 3)
```

◦ a = 1, 2, 3 ◦

◦ Python

```
a = 1 # a is the value 1
a = 1, # a is the tuple (1,)
```

```
a = (1,) # a is the tuple (1,)
a = (1) # a is the value 1 and not a tuple
```

```
# unpacking AKA multiple assignment
x, y, z = (1, 2, 3)
# x == 1
# y == 2
# z == 3
```

—

```
a = 1, 2, 3, 4
_, x, y, _ = a
# x == 2
# y == 3
```

```
x, = 1, # x is the value 1
x = 1, # x is the tuple (1,)
```

Python 3\*[catch-all](#)iterables

Python 3.x 3.0

```
first, *more, last = (1, 2, 3, 4, 5)
# first == 1
```

```
# more == [2, 3, 4]
# last == 5
```

```
colors = "red", "green", "blue"
rev = colors[::-1]
# rev: ("blue", "green", "red")
colors = rev
# colors: ("blue", "green", "red")
```

## reverse

```
rev = tuple(reversed(colors))
# rev: ("blue", "green", "red")
colors = rev
# colors: ("blue", "green", "red")
```

---

## python

- 
- 
- 

“” 0

```
tuple1 = ('a', 'b', 'c', 'd', 'e')
tuple2 = ('1', '2', '3')
tuple3 = ('a', 'b', 'c', 'd', 'e')

cmp(tuple1, tuple2)
Out: 1

cmp(tuple2, tuple1)
Out: -1

cmp(tuple1, tuple3)
Out: 0
```

---

len

```
len(tuple1)
Out: 5
```

---

max

```
max(tuple1)
Out: 'e'

max(tuple2)
Out: '3'
```

---

## min

```
min(tuple1)  
Out: 'a'
```

```
min(tuple2)  
Out: '1'
```

---

## tuple

```
list = [1,2,3,4,5]  
tuple(list)  
Out: (1, 2, 3, 4, 5)
```

---

+

```
tuple1 + tuple2  
Out: ('a', 'b', 'c', 'd', 'e', '1', '2', '3')
```

<https://riptutorial.com/zh-TW/python/topic/927/>

# 79:

type Python ◦

- 
- 
- Python 3.6 `__init_subclass__()` ◦

## Examples

type meta ◦ / ◦

```
Dummy = type('OtherDummy', (), dict(x=1))
Dummy.__class__ # <type 'type'>
Dummy().__class__.__class__ # <type 'type'>
```

type ◦

```
class mytype(type):
    def __init__(cls, name, bases, dict):
        # call the base initializer
        type.__init__(cls, name, bases, dict)

        # perform custom initialization...
        cls.__custom_attribute__ = 2
```

mytype type ◦

```
MyDummy = mytype('MyDummy', (), dict(x=2))
MyDummy.__class__ # <class '__main__.mytype'>
MyDummy().__class__.__class__ # <class '__main__.mytype'>
MyDummy.__custom_attribute__ # 2
```

class ◦

```
>>> class Foo(object):
...     pass

>>> type(Foo)
type
```

object object type ◦ Python 2 Python 3

## Python 2.x 2.7

`__metaclass__` ◦

```
class MyDummy(object):
    __metaclass__ = mytype
```

```
type(MyDummy) # <class '__main__.mytype'>
```

## Python 3.x 3.0

metaclass°

```
class MyDummy(metaclass=mytype):  
    pass  
type(MyDummy) # <class '__main__.mytype'>
```

metaclass° class MyDummy(metaclass=mytype, x=2)x=2mytype°

python°

/° python singleton °

```
class SingletonType(type):  
    def __call__(cls, *args, **kwargs):  
        try:  
            return cls.__instance  
        except AttributeError:  
            cls.__instance = super(SingletonType, cls).__call__(*args, **kwargs)  
            return cls.__instance
```

## Python 2.x 2.7

```
class MySingleton(object):  
    __metaclass__ = SingletonType
```

## Python 3.x 3.0

```
class MySingleton(metaclass=SingletonType):  
    pass
```

```
MySingleton() is MySingleton() # True, only one instantiation occurs
```

---

## Python 2.x 2.7

```
class MyClass(object):  
    __metaclass__ = SomeMetaClass
```

## Python 3.x 3.0

```
class MyClass(metaclass=SomeMetaClass):  
    pass
```

---

# Python 23<sub>six</sub>



```
import six

class MyClass(six.with_metaclass(SomeMetaclass)):
    pass
```

◦ ◦

```
class VerboseMetaclass(type):

    def __new__(cls, class_name, class_parents, class_dict):
        print("Creating class ", class_name)
        new_class = super().__new__(cls, class_name, class_parents, class_dict)
        return new_class
```

```
class Spam(metaclass=VerboseMetaclass):
    def eggs(self):
        print("[insert example string here]")
s = Spam()
s.eggs()
```

```
Creating class Spam
[insert example string here]
```

Python◦ ◦

Xtype(x)

```
>>> type(5)
<type 'int'>
>>> type(str)
<type 'type'>
>>> type([1, 2, 3])
<type 'list'>

>>> class C(object):
...     pass
...
>>> type(C)
<type 'type'>
```

python`type`◦ `type`◦ ◦

Python `type`◦

```
class SimplestMetaclass(type):
    pass

class MyClass(object):
    __metaclass__ = SimplestMetaclass
```

MyClassSimplestMetaclass

```
>>> type(MyClass)
<class '__main__.SimplestMetaclass'>
```

\_\_new\_\_ type \_\_new\_\_

```
class AnotherMetaclass(type):
    def __new__(cls, name, parents, dct):
        # cls is this class
        # name is the name of the class to be created
        # parents is the list of the class's parent classes
        # dct is the list of class's attributes (methods, static variables)

        # here all of the attributes can be modified before creating the class, e.g.

        dct['x'] = 8 # now the class will have a static variable x = 8

        # return value is the new class. super will take care of that
        return super(AnotherMetaclass, cls).__new__(cls, name, parents, dct)
```

## Python◦

```
>>> type(1)
int
```

### 1int◦

```
>>> class Foo(object):
...     pass
... 
```

```
>>> bar = Foo()
```

bar

```
>>> type(bar)
Foo
```

barFoo◦ Foo

```
>>> type(Foo)
type
```

Footype◦ type

```
>>> type(type)
type
```

◦

- Python
-

- type

Python“hackable”。

<https://riptutorial.com/zh-TW/python/topic/286/>

---

# 80:

- pythonOCR◦

## Examples

### PyTesseract

PyTesseractOCRpython◦

### PyTesseract

```
try:
    import Image
except ImportError:
    from PIL import Image

import pytesseract

#Basic OCR
print(pytesseract.image_to_string(Image.open('test.png')))

#In French
print(pytesseract.image_to_string(Image.open('test-european.jpg'), lang='fra'))
```

PyTesseract◦

### PyOCR

PyOCR ◦

PyTesseract◦

```
from PIL import Image
import sys

import pyocr
import pyocr.builders

tools = pyocr.get_available_tools()
# The tools are returned in the recommended order of usage
tool = tools[0]

langs = tool.get_available_languages()
lang = langs[0]
# Note that languages are NOT sorted in any way. Please refer
# to the system locale settings for the default language
# to use.
```

```
txt = tool.image_to_string(
    Image.open('test.png'),
```

```

    lang=lang,
    builder=pyocr.builders.TextBuilder()
)
# txt is a Python string

word_boxes = tool.image_to_string(
    Image.open('test.png'),
    lang="eng",
    builder=pyocr.builders.WordBoxBuilder()
)
# list of box objects. For each box object:
#   box.content is the word in the box
#   box.position is its position on the page (in pixels)
#
# Beware that some OCR tools (Tesseract for instance)
# may return empty boxes

line_and_word_boxes = tool.image_to_string(
    Image.open('test.png'), lang="fra",
    builder=pyocr.builders.LineBoxBuilder()
)
# list of line objects. For each line object:
#   line.word_boxes is a list of word boxes (the individual words in the line)
#   line.content is the whole text of the line
#   line.position is the position of the whole line on the page (in pixels)
#
# Beware that some OCR tools (Tesseract for instance)
# may return empty boxes

# Digits - Only Tesseract (not 'libtesseract' yet !)
digits = tool.image_to_string(
    Image.open('test-digits.png'),
    lang=lang,
    builder=pyocr.tesseract.DigitBuilder()
)
# digits is a python string

```

<https://riptutorial.com/zh-TW/python/topic/9302/>

# 81: CLI

hgsvn ""

o

hgsvn

```
usage: sub <command>

commands:

  status - show status
  list   - print list
```

## Examples

```
"""
usage: sub <command>

commands:

  status - show status
  list   - print list
"""

import sys

def check():
    print("status")
    return 0

if sys.argv[1:] == ['status']:
    sys.exit(check())
elif sys.argv[1:] == ['list']:
    print("list")
else:
    print(__doc__.strip())
```

```
usage: sub <command>

commands:

  status - show status
  list   - print list
```

- deps
- 
- 

## argparse

```

import argparse
import sys

def check():
    print("status")
    return 0

parser = argparse.ArgumentParser(prog="sub", add_help=False)
subparser = parser.add_subparsers(dest="cmd")

subparser.add_parser('status', help='show status')
subparser.add_parser('list', help='print list')

# hack to show help when no arguments supplied
if len(sys.argv) == 1:
    parser.print_help()
    sys.exit(0)

args = parser.parse_args()

if args.cmd == 'list':
    print('list')
elif args.cmd == 'status':
    sys.exit(check())

```

```

usage: sub {status,list} ...

positional arguments:
  {status,list}
    status      show status
    list        print list

```

- Python
- 

## argparse

<http://www.riptutorial.com/python/example/25282/argparse--default-help-formatter->

```

import argparse
import sys

class CustomHelpFormatter(argparse.HelpFormatter):
    def _format_action(self, action):
        if type(action) == argparse._SubParsersAction:
            # inject new class variable for subcommand formatting
            subactions = action._get_subactions()
            invocations = [self._format_action_invocation(a) for a in subactions]
            self._subcommand_max_length = max(len(i) for i in invocations)

        if type(action) == argparse._SubParsersAction._ChoicesPseudoAction:
            # format subcommand help line
            subcommand = self._format_action_invocation(action) # type: str
            width = self._subcommand_max_length
            help_text = ""
            if action.help:
                help_text = self._expand_help(action)

```

```

        return "  {:{width}} -  {}\n".format(subcommand, help_text, width=width)

    elif type(action) == argparse._SubParsersAction:
        # process subcommand help section
        msg = '\n'
        for subaction in action._get_subactions():
            msg += self._format_action(subaction)
        return msg
    else:
        return super(CustomHelpFormatter, self)._format_action(action)

def check():
    print("status")
    return 0

parser = argparse.ArgumentParser(usage="sub <command>", add_help=False,
                                formatter_class=CustomHelpFormatter)

subparser = parser.add_subparsers(dest="cmd")
subparser.add_parser('status', help='show status')
subparser.add_parser('list', help='print list')

# custom help message
parser._positionals.title = "commands"

# hack to show help when no arguments supplied
if len(sys.argv) == 1:
    parser.print_help()
    sys.exit(0)

args = parser.parse_args()

if args.cmd == 'list':
    print('list')
elif args.cmd == 'status':
    sys.exit(check())

```

```
usage: sub <command>
```

```
commands:
```

```
  status - show status
```

```
  list   - print list
```

CLI <https://riptutorial.com/zh-TW/python/topic/7701/cli>



# 82:

- `value1 ** value2`
- `powvalue1value2 [value3]`
- `value1 .__ pow __value2 [value3]`
- `2 .__ rpow __1`
- `operator.powvalue1value2`
- `operator .__ pow __value1value2`
- `math.powvalue1value2`
- `math.sqrt`
- `math.exp`
- `cmath.exp`
- `math.expm1`

## Examples

### `math.sqrt`/`cmath.sqrt`

`math`/`math.sqrt()` `float` `float`

```
import math

math.sqrt(9) # 3.0
math.sqrt(11.11) # 3.3331666624997918
math.sqrt(Decimal('6.25')) # 2.5
```

`complex` `math.sqrt()` `ValueError`

```
math.sqrt(-10)
```

### ValueError

`math.sqrt(x)`/`math.pow(x, 0.5)`/`x ** 0.5` `math.pow(x, 0.5)` ◦ `cmath`/`math+bi` ◦ `.sqrt()`

```
import cmath

cmath.sqrt(4) # 2+0j
cmath.sqrt(-4) # 2j
```

$j j - 1$  ◦  $a + bi$   $a + bj$  ◦  $a^2 + 0j^2 + 0j^2$  ◦  $b^2$   $0. b^2 j^2$  ◦  $2j^2 j^0 + 2j^2$  ◦

### **\*\***`pow`

`pow` -function\*\*

```
2 ** 3 # 8
pow(2, 3) # 8
```

## Python 2.x `**` ;

- **Base** `int` **exponent** `int < 0`

```
2 ** -3
# Out: 0.125 (result is a float)
```

- **Python 3.x**
- **Python 2.2.0** `ValueError` ◦

- **Base** `int < 0` `float < 0` **exponent** `float != int`

```
(-2) ** (0.5) # also (-2.) ** (0.5)
# Out: (8.659560562354934e-17+1.4142135623730951j) (result is complex)
```

- **python 3.0.0** `ValueError` ◦

`operator** operator`

```
import operator
operator.pow(4, 2) # 16
operator.__pow__(4, 3) # 64
```

`__pow__`

```
val1, val2 = 4, 2
val1.__pow__(val2) # 16
val2.__rpow__(val1) # 16
# in-place power operation isn't supported by immutable classes like int, float, complex:
# val1.__ipow__(val2)
```

## math.pow

**math -module** `math.pow()` ◦ `pow()**float`

```
import math
math.pow(2, 2) # 4.0
math.pow(-2., 2) # 4.0
```

```
math.pow(2, 2+0j)
```

**TypeError** `complexfloat`

```
math.pow(-2, 0.5)
```

**ValueError**

## math.exp `cmath.exp`

```
math module Euler e = math.exp(1)
math pow(2, e) ** math.exp(1) math.exp(1)
```

```
import math

math.e ** 2 # 7.3890560989306495
math.exp(2) # 7.38905609893065

import cmath
cmath.e ** 2 # 7.3890560989306495
cmath.exp(2) # (7.38905609893065+0j)
```

math.e

```
print(math.e ** 10) # 22026.465794806703
print(math.exp(10)) # 22026.465794806718
print(cmath.exp(10).real) # 22026.465794806718
# difference starts here -----^
```

## 1 math.expml

```
math expml(x) x math.e ** x - 1 math.exp(x) cmath.exp(x)
```

```
import math

print(math.e ** 1e-3 - 1) # 0.0010005001667083846
print(math.exp(1e-3) - 1) # 0.0010005001667083846
print(math.expml(1e-3)) # 0.0010005001667083417
# -----^
```

X

```
print(math.e ** 1e-15 - 1) # 1.1102230246251565e-15
print(math.exp(1e-15) - 1) # 1.1102230246251565e-15
print(math.expml(1e-15)) # 1.0000000000000007e-15
# -----^
```

o 1

```
def planks_law(lambda_, T):
    from scipy.constants import h, k, c # If no scipy installed hardcode these!
    return 2 * h * c ** 2 / (lambda_ ** 5 * math.expml(h * c / (lambda_ * k * T)))

def planks_law_naive(lambda_, T):
    from scipy.constants import h, k, c # If no scipy installed hardcode these!
    return 2 * h * c ** 2 / (lambda_ ** 5 * (math.e ** (h * c / (lambda_ * k * T)) - 1))

planks_law(100, 5000) # 4.139080074896474e-19
planks_law_naive(100, 5000) # 4.139080073488451e-19
# -----^

planks_law(1000, 5000) # 4.139080128493406e-23
planks_law_naive(1000, 5000) # 4.139080233183142e-23
# -----^
```

## cmath

```
class Integer(object):
    def __init__(self, value):
        self.value = int(value) # Cast to an integer

    def __repr__(self):
        return '{cls}({val})'.format(cls=self.__class__.__name__,
                                     val=self.value)

    def __pow__(self, other, modulo=None):
        if modulo is None:
            print('Using __pow__')
            return self.__class__(self.value ** other)
        else:
            print('Using __pow__ with modulo')
            return self.__class__(pow(self.value, other, modulo))

    def __float__(self):
        print('Using __float__')
        return float(self.value)

    def __complex__(self):
        print('Using __complex__')
        return complex(self.value, 0)
```

pow\*\*\_\_pow\_\_

```
Integer(2) ** 2           # Integer(4)
# Prints: Using __pow__
Integer(2) ** 2.5         # Integer(5)
# Prints: Using __pow__
pow(Integer(2), 0.5)      # Integer(1)
# Prints: Using __pow__
operator.pow(Integer(2), 3) # Integer(8)
# Prints: Using __pow__
operator.__pow__(Integer(3), 3) # Integer(27)
# Prints: Using __pow__
```

\_\_pow\_\_() builtin- pow()

```
pow(Integer(2), 3, 4)     # Integer(0)
# Prints: Using __pow__ with modulo
Integer(2).__pow__(3, 4) # Integer(0)
# Prints: Using __pow__ with modulo
```

mathfloat

```
import math

math.pow(Integer(2), 0.5) # 1.4142135623730951
# Prints: Using __float__
```

cmath -functions complex complexfloat

```
import cmath

cmath.exp(Integer(2))      # (7.38905609893065+0j)
# Prints: Using __complex__

del Integer.__complex__   # Deleting __complex__ method - instances cannot be cast to complex

cmath.exp(Integer(2))      # (7.38905609893065+0j)
# Prints: Using __float__
```

```
__float__() - mathcmath
```

```
del Integer.__float__    # Deleting __complex__ method

math.sqrt(Integer(2))    # also cmath.exp(Integer(2))
```

## TypeErrorfloat

### pow

`pow(a, b, c)`  $a^b \bmod c$

```
pow(3, 4, 17)    # 13

# equivalent unoptimized expression:
3 ** 4 % 17      # 13

# steps:
3 ** 4           # 81
81 % 17         # 13
```

- int
- int >= 0
- int != 0

## python 3.x

### pow

```
def modular_inverse(x, p):
    """Find a such as a·x ≡ 1 (mod p), assuming p is prime."""
    return pow(x, p-2, p)

[modular_inverse(x, 13) for x in range(1,13)]
# Out: [1, 7, 9, 10, 8, 11, 2, 5, 3, 4, 6, 12]
```

## nth-root

`math.sqrt ** n`

- 31/3

```
>>> x = 3
```

```
>>> y = x ** 3
>>> y
27
>>> z = y ** (1.0 / 3)
>>> z
3.0
>>> z == x
True
```

## PythonPythonn°

```
x = 2 ** 100
cube = x ** 3
root = cube ** (1.0 / 3)
```

## OverflowErrorlong intfloat

n°

```
def nth_root(x, n):
    # Start with some reasonable bounds around the nth root.
    upper_bound = 1
    while upper_bound ** n <= x:
        upper_bound *= 2
    lower_bound = upper_bound // 2
    # Keep searching for a better result as long as the bounds make sense.
    while lower_bound < upper_bound:
        mid = (lower_bound + upper_bound) // 2
        mid_nth = mid ** n
        if lower_bound < mid and mid_nth < x:
            lower_bound = mid
        elif upper_bound > mid and mid_nth > x:
            upper_bound = mid
        else:
            # Found perfect nth root.
            return mid
    return mid + 1

x = 2 ** 100
cube = x ** 3
root = nth_root(cube, 3)
x == root
# True
```

<https://riptutorial.com/zh-TW/python/topic/347/>

# 83:

## Examples

### py2app

#### py2app◦

```
sudo easy_install -U py2app
```

#### pip

```
pip install py2app
```

### python

```
py2applet --make-setup MyApplication.py
```

```
"""
This is a setup.py script generated by py2applet

Usage:
    python setup.py py2app
"""

from setuptools import setup

APP = ['test.py']
DATA_FILES = []
OPTIONS = {'argv_emulation': True}

setup(
    app=APP,
    data_files=DATA_FILES,
    options={'py2app': OPTIONS},
    setup_requires=['py2app'],
)
```

### .icns

```
DATA_FILES = ['myInsertedImage.jpg']
OPTIONS = {'argv_emulation': True, 'iconfile': 'myCoolIcon.icns'}
```

```
python setup.py py2app
```

### dist◦

```
optimize (-O)          optimization level: -O1 for "python -O", -O2 for
                        "python -OO", and -O0 to disable [default: -O0]
```

```
includes (-i)          comma-separated list of modules to include
packages (-p)         comma-separated list of packages to include
extension             Bundle extension [default:.app for app, .plugin for
                     plugin]
extra-scripts         comma-separated list of additional scripts to include
                     in an application or plugin.
```

## cx\_Freeze

### cx\_Freeze

```
python setup.py build
sudo python setup.py install
```

### python“setup.py”

```
application_title = "My Application" # Use your own application name
main_python_file = "my_script.py" # Your python script

import sys

from cx_Freeze import setup, Executable

base = None
if sys.platform == "win32":
    base = "Win32GUI"

includes = ["atexit", "re"]

setup(
    name = application_title,
    version = "0.1",
    description = "Your Description",
    options = {"build_exe" : {"includes" : includes }},
    executables = [Executable(main_python_file, base = base)])
```

### setup.py

```
python setup.py bdist_mac
```

### El CapitanrootSIP。

<https://riptutorial.com/zh-TW/python/topic/2026/>



## 84:

- `a [startend]itemsend-1`
- `a [start]items`
- `a [end]1`
- `a []`
- `a []`
- 
- `lst[::-1]`
- `startend°`

```
a[-1]    # last item in the array
a[-2:]   # last two items in the array
a[:-2]   # everything except the last two items
```

## Examples

““”

```
lst = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']

lst[::2]
# Output: ['a', 'c', 'e', 'g']

lst[::3]
# Output: ['a', 'd', 'g']
```

```
lst = ['a', 'b', 'c', 'd', 'e']

lst[2:4]
# Output: ['c', 'd']

lst[2:]
# Output: ['c', 'd', 'e']

lst[:4]
# Output: ['a', 'b', 'c', 'd']
```

```
a = [1, 2, 3, 4, 5]

# steps through the list backwards (step=-1)
b = a[::-1]

# built-in list method to reverse 'a'
a.reverse()

if a == b:
    print(True)

print(b)
```

```
# Output:
# True
# [5, 4, 3, 2, 1]
```

```
def shift_list(array, s):
    """Shifts the elements of a list to the left or right.

    Args:
        array - the list to shift
        s - the amount to shift the list ('+': right-shift, '-': left-shift)

    Returns:
        shifted_array - the shifted list
    """
    # calculate actual shift amount (e.g., 11 --> 1 if length of the array is 5)
    s %= len(array)

    # reverse the shift direction to be more intuitive
    s *= -1

    # shift array with list slicing
    shifted_array = array[s:] + array[:s]

    return shifted_array

my_array = [1, 2, 3, 4, 5]

# negative numbers
shift_list(my_array, -7)
>>> [3, 4, 5, 1, 2]

# no shift on numbers equal to the size of the array
shift_list(my_array, 5)
>>> [1, 2, 3, 4, 5]

# works on positive numbers
shift_list(my_array, 3)
>>> [3, 4, 5, 1, 2]
```

<https://riptutorial.com/zh-TW/python/topic/1494/-->

# 85:

Python

- `[x for x in 1,2,3]` list comprehension `[2,3,4]`
- `x + 1 for x in 1,2,3` generator `2,3,4`
- `[x for x in 1,2,3 if x2 == 0]` #list comprehension with filter `give [2]`
- `[x + 1 if x2 == 0 else x for x in 1,2,3]` #list comprehension with ternary
- `[x + 1 if x2 == 0 else x for x in range(3,4) if x > 0]` #list comprehension with ternary and filtering
- `{x for x in 1,2,2,3}` set comprehension `{1,2,3}`
- `{kvkv['a'1'b'2]}` #dict comprehension `{'a'1'b'2}` python 2.7+3.0+
- `[xy[1,2]x [10,20]][11,21,12,22]`
- `[xy[1,2,3]xx > 2][3,4,5]y1st`
- `[xy[1,2,3]x[3,4,5]yx > 2]2`
- `[x for x in xrange(10) if x2 == 0]`

o o

- 
- 
- [lambda](#)
- o `[print(x) for x in range(10)]`

## Examples

[iterable](#) list

```
[ <expression> for <element> in <iterable> ]
```

'if'

```
[ <expression> for <element> in <iterable> if <condition> ]
```

<condition> [true](#) <iterable> <element><element><expression> o o - o

list

```
squares = [x * x for x in (1, 2, 3, 4)]  
# squares: [1, 4, 9, 16]
```

for(1, 2, 3, 4)x o x \* x list o list squares o

for

```
squares = []  
for x in (1, 2, 3, 4):
```

```
squares.append(x * x)
# squares: [1, 4, 9, 16]
```

```
# Get a list of uppercase characters from a string
[s.upper() for s in "Hello World"]
# ['H', 'E', 'L', 'L', 'O', ' ', 'W', 'O', 'R', 'L', 'D']

# Strip off any commas from the end of strings in a list
[w.strip(',') for w in ['these,', 'words,', 'mostly', 'have,commas,']]
# ['these', 'words', 'mostly', 'have,commas']

# Organize letters in words more reasonably - in an alphabetical order
sentence = "Beautiful is better than ugly"
["".join(sorted(word, key = lambda x: x.lower())) for word in sentence.split()]
# ['aBefiltuu', 'is', 'beertt', 'ahnt', 'gluy']
```

---

**List comprehension** ◦ **if / else** for for

```
# create a list of characters in apple, replacing non vowels with '*'
# Ex - 'apple' --> ['a', '*', '*', '*', 'e']

[x for x in 'apple' if x in 'aeiou' else '*']
#SyntaxError: invalid syntax

# When using if/else together use them before the loop
[x if x in 'aeiou' else '*' for x in 'apple']
#['a', '*', '*', '*', 'e']
```

◦ for...in if ◦

---

[... for x in ... for y in ...] ◦ for

```
def foo(i):
    return i, i + 0.5

for i in range(3):
    for x in foo(i):
        yield str(x)
```

```
[str(x)
    for i in range(3)
    for x in foo(i)
]
```

```
[str(x) for i in range(3) for x in foo(i)]
```

---

None ◦

◦ ◦ ◦

`list.sort()` None ◦

```
[x.sort() for x in [[2, 1], [4, 3], [0, 1]]]
# [None, None, None]
```

`sorted()` list

```
[sorted(x) for x in [[2, 1], [4, 3], [0, 1]]]
# [[1, 2], [3, 4], [0, 1]]
```

## I/O for Python 3

```
[print(x) for x in (1, 2, 3)]
```

```
for x in (1, 2, 3):
    print(x)
```

◦ `random.randrange()` ◦ `next()` ◦

```
from random import randrange
[randrange(1, 7) for _ in range(10)]
# [2, 3, 2, 1, 1, 5, 2, 4, 3, 5]
```

---

◦

```
[
    x for x
    in 'foo'
    if x not in 'bar'
]
```

◦

## Python 2.x 2.7

```
{x: x * x for x in (1, 2, 3, 4)}
# Out: {1: 1, 2: 4, 3: 9, 4: 16}
```

```
dict((x, x * x) for x in (1, 2, 3, 4))
# Out: {1: 1, 2: 4, 3: 9, 4: 16}
```

---

dictdict◦

## Python 2.x 2.7

```
{name: len(name) for name in ('Stack', 'Overflow', 'Exchange') if len(name) > 6}
# Out: {'Exchange': 8, 'Overflow': 8}
```

◦

```
dict((name, len(name)) for name in ('Stack', 'Overflow', 'Exchange') if len(name) > 6)
# Out: {'Exchange': 8, 'Overflow': 8}
```

---

## Python 2.x 2.7

```
initial_dict = {'x': 1, 'y': 2}
{key: value for key, value in initial_dict.items() if key == 'x'}
# Out: {'x': 1}
```

---

## *hashabledict*

```
my_dict = {1: 'a', 2: 'b', 3: 'c'}
```

- `swapped = {v: k for k, v in my_dict.items()}`
- `swapped = dict((v, k) for k, v in my_dict.iteritems())`
- `swapped = dict(zip(my_dict.values(), my_dict))`
- `swapped = dict(zip(my_dict.values(), my_dict.keys()))`
- `swapped = dict(map(reversed, my_dict.items()))`

```
print(swapped)
# Out: {a: 1, b: 2, c: 3}
```

---

## Python 2.x 2.3

[itertools](#): `izip` `imap` ◦

◦

```
dict1 = {'w': 1, 'x': 1}
dict2 = {'x': 2, 'y': 2, 'z': 2}

{k: v for d in [dict1, dict2] for k, v in d.items()}
# Out: {'w': 1, 'x': 2, 'y': 2, 'z': 2}
```

[PEP 448](#) ◦

---

## Python 3.x 3.5

```
{**dict1, **dict2}
# Out: {'w': 1, 'x': 2, 'y': 2, 'z': 2}
```

Python 3.02.7+2.0 ◦ `<2.7dict()` ◦

◦ ; ◦

```
# list comprehension
[x**2 for x in range(10)]
# Output: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

## Python 2.x 2.4

```
# generator comprehension
(x**2 for x in xrange(10))
# Output: <generator object <genexpr> at 0x11b4b7c80>
```

- list comprehension `list` generator comprehension `generator` ◦
- generator `next` ◦

`xrange` ◦ `xrange` `python 2` ◦ `python 3` `range` ◦ [xrange](#) ◦

---

## Python 2.x 2.4

```
g = (x**2 for x in xrange(10))
print(g[0])
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'generator' object has no attribute '__getitem__'
```

```
g.next() # 0
g.next() # 1
g.next() # 4
...
g.next() # 81

g.next() # Throws StopIteration Exception
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

## Python 3.x 3.0

`g.next()` `next(g)` `xrange` `range` `Iterator.next()` `xrange()` `Python 3` ◦

---

```
for i in [x**2 for x in range(10)]:
    print(i)
```

```
"""
Out:
0
1
```

```
4
...
81
"""
```

## Python 2.x 2.4

```
for i in (x**2 for x in xrange(10)):
    print(i)
```

```
"""
Out:
0
1
4
.
.
.
81
"""
```

o

```
for square in (x**2 for x in range(1000000)):
    #do something
```

## iterable◦ get\_objects() API◦◦◦

```
def get_objects():
    """Gets objects from an API one by one"""
    while True:
        yield get_next_item()

def object_matches_pattern(obj):
    # perform potentially complex calculation
    return matches_pattern

def right_item_exists():
    items = (object_matched_pattern(each) for each in get_objects())
    for item in items:
        if item.is_the_right_one:

            return True
    return False
```

o

## Python 2.x 2.7

```
# A set containing every value in range(5):
{x for x in range(5)}
# Out: {0, 1, 2, 3, 4}

# A set of even numbers between 1 and 10:
```



```
{x for x in range(1, 11) if x % 2 == 0}
# Out: {2, 4, 6, 8, 10}

# Unique alphabetic characters in a string of text:
text = "When in the Course of human events it becomes necessary for one people..."
{ch.lower() for ch in text if ch.isalpha()}
# Out: set(['a', 'c', 'b', 'e', 'f', 'i', 'h', 'm', 'l', 'o',
#          'n', 'p', 's', 'r', 'u', 't', 'w', 'v', 'y'])
```

◦ ◦

python 2.7+2.0 Python 2.2 Python 2.6 set()

## Python 2.x 2.2

```
set(x for x in range(5))
# Out: {0, 1, 2, 3, 4}
```

```
>>> def f(x):
...     import time
...     time.sleep(.1)          # Simulate expensive function
...     return x**2

>>> [f(x) for x in range(1000) if f(x) > 10]
[16, 25, 36, ...]
```

f(x) 1000 x if f(x) f()

x

```
>>> [v for v in (f(x) for x in range(1000)) if v > 10]
[16, 25, 36, ...]
```

```
>>> [v for v in map(f, range(1000)) if v > 10]
[16, 25, 36, ...]
```

v ◦ v iterable

```
>>> [v for x in range(1000) for v in [f(x)] if v > 10]
[16, 25, 36, ...]
```

◦

```
>>> def process_prime_numbers(iterable):
...     for x in iterable:
...         if is_prime(x):
...             yield f(x)
...
>>> [x for x in process_prime_numbers(range(1000)) if x > 10]
[11, 13, 17, 19, ...]
```

```
l = [[1, 2, 3], [4, 5, 6], [7], [8, 9]]
```

```
reduce(lambda x, y: x+y, l)
sum(l, [])
list(itertools.chain(*l))
```

o

```
[item for sublist in l for item in sublist]
```

L+OL ^ 2 - o LIL-1IL-2;lx1LI \*L \*\* 2/ 2o

o

for

```
[x + y for x, y in [(1, 2), (3, 4), (5, 6)]]
# Out: [3, 7, 11]

[x + y for x, y in zip([1, 3, 5], [2, 4, 6])]
# Out: [3, 7, 11]
```

for

```
for x, y in [(1,2), (3,4), (5,6)]:
    print(x+y)
# 3
# 7
# 11
```

```
[x, y for x, y in [(1, 2), (3, 4), (5, 6)]]
# SyntaxError: invalid syntax

[(x, y) for x, y in [(1, 2), (3, 4), (5, 6)]]
# Out: [(1, 2), (3, 4), (5, 6)]
```

```
# Count the numbers in `range(1000)` that are even and contain the digit `9`:
print (sum(
    1 for x in range(1000)
    if x % 2 == 0 and
    '9' in str(x)
))
# Out: 95
```

1. range(1000) o
2. if o
3. 11 o
- 4.

1°

1 sum° °

- **List Comprehension**`map()`◦

```
# Convert a list of strings to integers.
items = ["1","2","3","4"]
[int(item) for item in items]
# Out: [1, 2, 3, 4]

# Convert a list of strings to float.
items = ["1","2","3","4"]
map(float, items)
# Out:[1.0, 2.0, 3.0, 4.0]
```

<https://riptutorial.com/zh-TW/python/topic/196/>

# 86:

110 `[i ** 2 for i in range(1,11)]` `irange` `for`.

- `[i for i in range(10)]` #basic list comprehension
- `[i for i in xrange(10)]`python 2.x
- `[i for i in range(20)if i2 == 0]` #with filter with
- `[xy[1,2,3]x[3,4,5]]y`
- `[i if i > 6 else 0 for i in range(10)]` #ternary expression
- `[i if i > 4 else 0 for i in range(20)if i2 == 0]` #with filter and threenary expression
- `[3,4,5][[x + y[1,2,3]x]` #nested list comprehension

[PEP 202](#)Python 2.0.

## Examples

if

```
[<expression> for <element> in <iterable> if <condition>]
```

```
<iterable><element> <iterable> ;<condition>True <expression> <element>
```

```
[x for x in range(10) if x % 2 == 0]  
# Out: [0, 2, 4, 6, 8]
```

```
even_numbers = []  
for x in range(10):  
    if x % 2 == 0:  
        even_numbers.append(x)  
  
print(even_numbers)  
# Out: [0, 2, 4, 6, 8]
```

```
[e for x in y if c] exlist(filter(lambda x: c, map(lambda x: e, y))).
```

2. .

```
... if ... else ... <expression>
```

```
[x if x % 2 == 0 else None for x in range(10)]  
# Out: [0, None, 2, None, 4, None, 6, None, 8, None]
```

```
<value-if-condition-is-true> if <condition> else <value-if-condition-is-false>
```

```
[2 * (x if x % 2 == 0 else -1) + 1 for x in range(10)]
# Out: [1, -1, 5, -1, 9, -1, 13, -1, 17, -1]
```

## Python 2.7 xrange

```
[2 * (x if x % 2 == 0 else -1) + 1 for x in xrange(10)]
# Out: [1, -1, 5, -1, 9, -1, 13, -1, 17, -1]
```

```
numbers = []
for x in range(10):
    if x % 2 == 0:
        temp = x
    else:
        temp = -1
    numbers.append(2 * temp + 1)
print(numbers)
# Out: [1, -1, 5, -1, 9, -1, 13, -1, 17, -1]
```

if

```
[x if x > 2 else '*' for x in range(10) if x % 2 == 0]
# Out: ['*', '*', 4, 6, 8]
```

```
[x if (x > 2 and x % 2 == 0) else '*' for x in range(10)]
# Out: ['*', '*', '*', '*', 4, '*', 6, '*', 8, '*']
```

。

## List Comprehensions

```
[ expression for target1 in iterable1 [if condition1]
    for target2 in iterable2 [if condition2]...
    for targetN in iterableN [if conditionN] ]
```

for

```
data = [[1, 2], [3, 4], [5, 6]]
output = []
for each_list in data:
    for element in each_list:
        output.append(element)
print(output)
# Out: [1, 2, 3, 4, 5, 6]
```

for

```
data = [[1, 2], [3, 4], [5, 6]]
output = [element for each_list in data for element in each_list]
print(output)
# Out: [1, 2, 3, 4, 5, 6]
```

◦

◦

```
In [1]: data = [[1,2],[3,4],[5,6]]
In [2]: def f():
...:     output=[]
...:     for each_list in data:
...:         for element in each_list:
...:             output.append(element)
...:     return output
In [3]: timeit f()
1000000 loops, best of 3: 1.37 µs per loop
In [4]: timeit [inner for outer in data for inner in outer]
1000000 loops, best of 3: 632 ns per loop
```

**140ns** ◦

if **S**for

```
data = [[1], [2, 3], [4, 5]]
output = [element for each_list in data
          if len(each_list) == 2
          for element in each_list
          if element != 5]
print(output)
# Out: [2, 3, 4]
```

**for** ◦ **2/** ◦ ◦

filtermap ◦ [Guido Van Rossum 2005](#)

```
filter(P, S)[x for x in S if P(x)] x==42 lambdalambda ◦ map(F, S)map(F, S)[F(x) for
x in S] map(F, S) [F(x) for x in S] ◦ ◦
```

“*pythonic*”python linters ◦

```
filter(lambda x: x % 2 == 0, range(10)) # even numbers < 10
map(lambda x: 2*x, range(10)) # multiply each number by two
reduce(lambda x,y: x+y, range(10)) # sum of all elements in list
```

filtermap ;**lambda** - ◦

```
# Filter:
# P(x) = x % 2 == 0
# S = range(10)
[x for x in range(10) if x % 2 == 0]

# Map
# F(x) = 2*x
# S = range(10)
```

```
[2*x for x in range(10)]
```

◦ ;◦ **MapFilter**◦

```
# Map & Filter
filtered = filter(lambda x: x % 2 == 0, range(10))
results = map(lambda x: 2*x, filtered)

# List comprehension
results = [2*x for x in range(10) if x % 2 == 0]
```

■

- `map(F, S) == [F(x) for x in S]`
- `filter(P, S) == [x for x in S if P(x)]`

*FPbool*

◦ ◦

```
#List Comprehension with nested loop
[x + y for x in [1, 2, 3] for y in [3, 4, 5]]
#Out: [4, 5, 6, 5, 6, 7, 6, 7, 8]

#Nested List Comprehension
[[x + y for x in [1, 2, 3]] for y in [3, 4, 5]]
#Out: [[4, 5, 6], [5, 6, 7], [6, 7, 8]]
```

```
l = []
for y in [3, 4, 5]:
    temp = []
    for x in [1, 2, 3]:
        temp.append(x + y)
    l.append(temp)
```

◦

```
matrix = [[1,2,3],
          [4,5,6],
          [7,8,9]]

[[row[i] for row in matrix] for i in range(len(matrix))]
# [[1, 4, 7], [2, 5, 8], [3, 6, 9]]
```

forfor◦

```
[[[i + j + k for k in 'cd'] for j in 'ab'] for i in '12']
# Out: [[['1ac', '1ad'], ['1bc', '1bd']], [['2ac', '2ad'], ['2bc', '2bd']]]
```

`zip()`

```
>>> list_1 = [1, 2, 3, 4]
>>> list_2 = ['a', 'b', 'c', 'd']
>>> list_3 = ['6', '7', '8', '9']

# Two lists
>>> [(i, j) for i, j in zip(list_1, list_2)]
[(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd')]

# Three lists
>>> [(i, j, k) for i, j, k in zip(list_1, list_2, list_3)]
[(1, 'a', '6'), (2, 'b', '7'), (3, 'c', '8'), (4, 'd', '9')]

# so on ...
```

<https://riptutorial.com/zh-TW/python/topic/5265/>



# 87:

## Examples

### “Iterable

```
a, b = (1, 2)
print(a)
# Prints: 1
print(b)
# Prints: 2
```

```
a, b, c = [1]
# Raises: ValueError: not enough values to unpack (expected 3, got 1)
```

### Python 3.x 3.0

```
head, *tail = [1, 2, 3, 4, 5]
```

```
print(head)
# Prints: 1
print(tail)
# Prints: [2, 3, 4, 5]
```

```
l = [1, 2, 3, 4, 5]
head = l[0]
tail = l[1:]
```

```
a, b, *other, z = [1, 2, 3, 4, 5]
print(a, b, z, other)
# Prints: 1 2 5 [3, 4]
```

\_° \_°

```
a, _ = [1, 2]
print(a)
# Prints: 1
a, _, c = (1, 2, 3)
print(a)
# Prints: 1
print(c)
# Prints: 3
```

### Python 3.x 3.0

\*\_

```
a, *_ = [1, 2, 3, 4, 5]
print(a)
```

```
# Prints: 1
```

◦

```
a, *_ , b = [1, 2, 3, 4, 5]
print(a, b)
# Prints: 1 5
```

```
a, _, b, _, c, *_ = [1, 2, 3, 4, 5, 6]
print(a, b, c)
# Prints: 1 3 5
```

```
def fun1(arg1, arg2, arg3):
    return (arg1, arg2, arg3)
```

```
fun1(1, 2, 3)
```

```
def fun2(arg1='a', arg2='b', arg3='c'):
    return (arg1, arg2, arg3)
```

```
fun2(1)           → (1, b, c)
fun2(1, 2)        → (1, 2, c)
fun2(arg2=2, arg3=3) → (a, 2, 3)
...
```

listdict◦

```
l = [1, 2, 3]
```

\*

```
fun1(*l)
# Returns: (1, 2, 3)
fun1(*['w', 't', 'f'])
# Returns: ('w', 't', 'f')
```

```
fun1(*['oops'])
# Raises: TypeError: fun1() missing 2 required positional arguments: 'arg2' and 'arg3'
```

◦ **\*\*Python**dict

```
d = {
    'arg1': 1,
    'arg2': 2,
    'arg3': 3
}
fun1(**d)
# Returns: (1, 2, 3)
```

```
fun1(**{'arg1':1, 'arg2':2})
# Raises: TypeError: fun1() missing 1 required positional argument: 'arg3'
fun1(**{'arg1':1, 'arg2':2, 'arg3':3, 'arg4':4})
# Raises: TypeError: fun1() got an unexpected keyword argument 'arg4'
```

```
fun2(**d)
# Returns: (1, 2, 3)
```

```
fun2(**{'arg2': 2})
# Returns: ('a', 2, 'c')
```

```
fun2(**{'arg1':1, 'arg2':2, 'arg3':3, 'arg4':4})
# Raises: TypeError: fun2() got an unexpected keyword argument 'arg4'
```

```
def fun3(arg1, arg2='b', arg3='c')
    return (arg1, arg2, arg3)
```

## iterable

```
fun3(*[1])
# Returns: (1, 'b', 'c')
fun3(*[1,2,3])
# Returns: (1, 2, 3)
```

```
fun3(**{'arg1':1})
# Returns: (1, 'b', 'c')
fun3(**{'arg1':1, 'arg2':2, 'arg3':3})
# Returns: (1, 2, 3)
```

```
fun3(*[1,2], **{'arg3':3})
# Returns: (1,2,3)
```

```
fun3(*[1,2], **{'arg2':42, 'arg3':3})
# Raises: TypeError: fun3() got multiple values for argument 'arg2'
```

“”

```
def fun1(*args, **kwargs):
    print(args, kwargs)
```

\*args\*\*kwargstupledict

```
fun1(1,2,3)
# Prints: (1, 2, 3) {}
fun1(a=1, b=2, c=3)
# Prints: () {'a': 1, 'b': 2, 'c': 3}
fun1('x', 'y', 'z', a=1, b=2, c=3)
# Prints: ('x', 'y', 'z') {'a': 1, 'b': 2, 'c': 3}
```

## Python

```
class MyString(str):
    def __init__(self, *args, **kwarg):
        print('Constructing MyString')
        super(MyString, self).__init__(*args, **kwarg)
```

<https://riptutorial.com/zh-TW/python/topic/4282/-->

# 88:

## Examples

### IPython%% timeit

```
In [1]: import string

In [2]: %%timeit s=""; long_list=list(string.ascii_letters)*50
....: for substring in long_list:
....:     s+=substring
....:
1000 loops, best of 3: 570 us per loop

In [3]: %%timeit long_list=list(string.ascii_letters)*50
....: s="".join(long_list)
....:
100000 loops, best of 3: 16.1 us per loop
```

```
In [4]: %timeit for i in range(100000):pass
100 loops, best of 3: 2.82 ms per loop

In [5]: %timeit for i in list(range(100000)):pass
100 loops, best of 3: 3.95 ms per loop
```

### timeit

```
>>> import timeit
>>> timeit.timeit('list(itertools.repeat("a", 100))', 'import itertools', number = 1000000)
10.997665435877963
>>> timeit.timeit('["a"]*100', number = 1000000)
7.118789926862576
```

### timeit

```
python -m timeit "'-'.join(str(n) for n in range(100))"
10000 loops, best of 3: 29.2 usec per loop

python -m timeit "'-'.join(map(str, range(100)))"
100000 loops, best of 3: 19.4 usec per loop
```

### line\_profiler

#### @profile

```
import requests

@profile
def slow_func():
    s = requests.session()
```

```

html=s.get("https://en.wikipedia.org/").text
sum([pow(ord(x),3.1) for x in list(html)])

for i in range(50):
    slow_func()

```

## kernprof

```

$ kernprof -lv so6.py

Wrote profile results to so6.py.lprof
Timer unit: 4.27654e-07 s

Total time: 22.6427 s
File: so6.py
Function: slow_func at line 4

```

Line #	Hits	Time	Per Hit	% Time	Line Contents
4					@profile
5					def slow_func():
6	50	20729	414.6	0.0	s = requests.session()
7	50	47618627	952372.5	89.9	
8	50	5306958	106139.2	10.0	sum([pow(ord(x),3.1) for x in list(html)])

o

## cProfileProfiler

PythoncProfile。timeit。

- ncalls
- tottime
- percall。tottimencalls
- cumtime。。
- percall **cumtime**
- filename:lineno(function)

## cProfiler

```
$ python -m cProfile main.py
```

```
$ python -m cProfile -s time main.py
```

<https://riptutorial.com/zh-TW/python/topic/3818/>

---

# 89: Python

[pypa](#) [setup.py](#) ◦

## Examples

[setup.py](#) ◦

```
+-- package_name
|           |
|           +-- __init__.py
|
+-- setup.py
```

```
__init__.pydef foo(): return 100 ◦
```

[setup.py](#)

```
from setuptools import setup

setup(
    name='package_name',           # package name
    version='0.1',                 # version
    description='Package Description', # short description
    url='http://example.com',      # package URL
    install_requires=[],           # list of packages this package depends
                                   # on.
    packages=['package_name'],     # List of module names that installing
                                   # this package will provide.
)
```

[virtualenv](#) Python

```
$ virtualenv .virtualenv
...
$ source .virtualenv/bin/activate
$ python setup.py install
running install
...
Installed .../package_name-0.1-....egg
...
$ python
>>> import package_name
>>> package_name.foo()
100
```

## PyPI

[setup.py](#) [PyPI](#) ◦

---

# .pypirc

◦ ◦

```
# .pypirc file

[distutils]
index-servers =
  pypi
  pypitest

[pypi]
repository=https://pypi.python.org/pypi
username=your_username
password=your_password

[pypitest]
repository=https://testpypi.python.org/pypi
username=your_username
password=your_password
```

twine ◦

```
$ pip install twine
```

---

# testpypi

PyPI testpypi ◦ ◦ ◦

testpypi ◦ ◦

```
$ python setup.py register -r pypitest
```

```
$ twine upload dist/* -r pypitest
```

◦

---

◦ testpypiPyPI pip install◦

```
# Using virtualenv
$ mkdir testenv
$ cd testenv
$ virtualenv .virtualenv
...
$ source .virtualenv/bin/activate
# Test from testpypi
(.virtualenv) pip install --verbose --extra-index-url https://testpypi.python.org/pypi
package_name
```



```
...
# Or test from PyPI
(.virtualenv) $ pip install package_name
...

(.virtualenv) $ python
Python 3.5.1 (default, Jan 27 2016, 19:16:39)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import package_name
>>> package_name.foo()
100
```

- [PyPIAPI](#) ◦ ◦ [testpypi](#)

---

## PyPI

twine

```
$ pip install twine
```

[PyPI](#) ◦

```
$ python setup.py register -r pypi
$ twine upload dist/*
```

- 
- 

- 
- [PyPiStructuredText](#) ◦

README.rst ◦ [PyPi](#) ◦

setup.cfg

```
[metadata]
description-file = README.rst
```

[MarkdownPyPi](#) ◦

---

LICENSE.txt [OpenSource](#) ◦

[TL; DR](#) ◦

[\\_\\_main\\_\\_.py](#) ◦

`__main__.py`package\_name°

```
python -m package_name
```

`__main__.py`

```
pythonpackage_name.__main__;package_name.__main__; 'package_name'°
```

Python <https://riptutorial.com/zh-TW/python/topic/1381/python>

# 90:

Python ◦ ◦ Python ◦

Python `print()` `input()` `len()` ◦ - ◦

- `def function_name arg1... argN* argskw1kw2 = default... **kwargs`
- `lambda arg1... argN* argskw1kw2 = default... **kwargs expression`

<code>arg1 ... argN</code>	
<code>* ARGS</code>	
<code>KW1...KWN</code>	
<code>**kwargs</code>	

5

- ```
def f():
    print(20)
y = f
y()
# Output: 20
```

- ```
def f(a, b, y):
    def inner_add(a, b):          # inner_add is hidden from outer code
        return a + b
    return inner_add(a, b)**y
```

- ```
def f(y):
    def nth_power(x):
        return x ** y
    return nth_power           # returns a function

squareOf = f(2)                # function that returns the square of a number
cubeOf = f(3)                  # function that returns the cube of a number
squareOf(3)                    # Output: 9
cubeOf(2)                      # Output: 8
```

- ```
def a(x, y):
    print(x, y)
def b(fun, str):                # b has two arguments: a function and a string
    fun('Hello', str)
b(a, 'Sophia')                 # Output: Hello Sophia
```

- [Closure](#)

```
def outer_fun(name):
    def inner_fun(): # the variable name is available to the inner function
        return "Hello "+ name + "!"
    return inner_fun
greet = outer_fun("Sophia")
print(greet()) # Output: Hello Sophia!
```

- <https://www.thecodeship.com/patterns/guide-to-python-function-decorators/>

## Examples

def python◦

```
def function_name(parameters):
    statement(s)
```

*function\_name* *function\_name* ◦ ◦

*parameters* ◦ ◦

*statement(s)* - - ◦ ◦

Hello

```
def greet():
    print("Hello")
```

`greet()`

```
greet()
# Out: Hello
```

```
def greet_two(greeting):
    print(greeting)
```

`greet_two()`

```
greet_two("Howdy")
# Out: Howdy
```

```
def greet_two(greeting="Howdy"):
    print(greeting)
```

```
greet_two()
# Out: Howdy
```

◦ **Python** `return` `return` ◦

```
def many_types(x):
```

```
    if x < 0:
        return "Hello!"
    else:
        return 0

print(many_types(1))
print(many_types(-1))

# Output:
0
Hello!
```

## Python

### returnNone

```
def do_nothing():
    pass

print(do_nothing())
# Out: None
```

◦ pass - ◦ ◦ ◦

return

```
def give_me_five():
    return 5

print(give_me_five()) # Print the returned value
# Out: 5
```

```
num = give_me_five()
print(num) # Print the saved returned value
# Out: 5
```

```
print(give_me_five() + 10)
# Out: 15
```

return

```
def give_me_another_five():
    return 5
    print('This statement will not be printed. Ever.')

print(give_me_another_five())
# Out: 5
```

return

```
def give_me_two_fives():
    return 5, 5 # Returns two 5

first, second = give_me_two_fives()
```

```
print(first)
# Out: 5
print(second)
# Out: 5
```

returnNone ◦ returnNone ◦

```
def divide(dividend, divisor): # The names of the function and its arguments
    # The arguments are available by name in the body of the function
    print(dividend / divisor)
```

◦ ◦

```
divide(10, 2)
# output: 5
```

```
divide(divisor=2, dividend=10)
# output: 5
```

## = argument-name

```
def make(action='nothing'):
    return action
```

## 3

```
make("fun")
# Out: fun

make(action="sleep")
# Out: sleep

# The argument is optional so the function will use the default value if the argument is
# not passed in.
make()
# Out: nothing
```

---

list dict set◦ ◦ ◦

```
def func(value1, value2, optionalvalue=10):
    return '{0} {1} {2}'.format(value1, value2, optionalvalue1)
```

```
print(func(1, 'a', 100))
# Out: 1 a 100

print(func('abc', 14))
# abc 14 10
```

◦

```
print(func('This', optionalvalue='StackOverflow Documentation', value2='is'))
# Out: This is StackOverflow Documentation
```

★

```
def func(*args):
    # args will be a tuple containing all values that are passed in
    for i in args:
        print(i)

func(1, 2, 3) # Calling it with 3 arguments
# Out: 1
#      2
#      3

list_of_arg_values = [1, 2, 3]
func(*list_of_arg_values) # Calling it with list of values, * expands the list
# Out: 1
#      2
#      3

func() # Calling it without arguments
# No Output
```

argsfunc(\*args=[1, 2, 3])◦

func(\*args=[1, 2, 3])TypeError ◦

Iterable func(\*my\_stuff) ◦

\*args args[0]

★

```
def func(**kwargs):
    # kwargs will be a dictionary containing the names as keys and the values as values
    for name, value in kwargs.items():
        print(name, value)

func(value1=1, value2=2, value3=3) # Calling it with 3 arguments
# Out: value1 1
#      value2 2
#      value3 3

func() # Calling it without arguments
# No Out put

my_dict = {'foo': 1, 'bar': 2}
func(**my_dict) # Calling it with a dictionary
# Out: foo 1
#      bar 2
```

func(1, 2, 3)TypeError ◦

kwargspython ◦ args['value1']value1 ◦ KeyError ◦

◦

/◦ ◦

\*arg◦ ◦

◦ ◦

\*\*kwargs◦ ◦

```
#      |-positional-|-optional-|---keyword-only--|-optional-|
def func(arg1, arg2=10 , *args, kwarg1, kwarg2=2, **kwargs):
    pass
```

- arg1 TypeError ◦ func(10) func(arg1=10) ◦
- kwarg1 **keyword-argument** func(kwarg1=10) ◦
- arg2kwarg2◦ arg1 kwarg1 ◦
- \*args◦ arg1arg2\*args func(1, 1, 1, 1) ◦
- \*\*kwargs◦ arg1 arg2 kwarg1kwarg2 ◦ func(kwarg3=10) ◦
- **Python 3**◦ **Python 3.5**math.isclosedef math.isclose (a, b, \*, rel\_tol=1e-09, abs\_tol=0.0)

◦

Python 2.x◦ kwargs

```
def func(arg1, arg2=10, **kwargs):
    try:
        kwarg1 = kwargs.pop("kwarg1")
    except KeyError:
        raise TypeError("missing required keyword-only argument: 'kwarg1'")

    kwarg2 = kwargs.pop("kwarg2", 2)
    # function body ...
```

argskwargs ◦

**noneone** \*args**none** \*\*kwargs◦ \*args \*\*kwargs◦ ◦

**\*\*\*argskwargs**◦

```
def fn(**kwargs):
    print(kwargs)
    f1(**kwargs)

def f1(**kwargs):
    print(len(kwargs))

fn(a=1, b=2)
# Out:
# {'a': 1, 'b': 2}
# 2
```



- 
- `__defaults__`◦

```
def f(a, b=42, c=[]):
    pass

print(f.__defaults__)
# Out: (42, [])
```

## Argument ;◦ ◦ ◦ ◦

```
def append(elem, to=[]):
    to.append(elem)      # This call to append() mutates the default variable "to"
    return to

append(1)
# Out: [1]

append(2) # Appends it to the internally stored list
# Out: [1, 2]

append(3, []) # Using a new created list gives the expected result
# Out: [3]

# Calling it again without argument will append to the internally stored list again
append(4)
# Out: [1, 2, 4]
```

## IDEPyCharm◦

- 
- None None ◦

```
def append(elem, to=None):
    if to is None:
        to = []

    to.append(elem)
    return to
```

## Lambda/

- lambda◦ ◦

```
def greeting():
    return "Hello"
```

```
print(greeting())
```

```
Hello
```

## lambda

```
greet_me = lambda: "Hello"
```

lambdas ◦ ◦

```
greet_meHello ◦ lambda return ◦ :◦
```

```
print(greet_me())
```

```
Hello
```

lambda

```
strip_and_upper_case = lambda s: s.strip().upper()
```

```
strip_and_upper_case(" Hello ")
```

```
HELLO
```

/◦

```
greeting = lambda x, *args, **kwargs: print(x, args, kwargs)
greeting('hello', 'world', world='world')
```

```
hello ('world',) {'world': 'world'}
```

lambda **S**sorted filtermap ◦

```
sorted([" foo ", " bAR", "BaZ "], key=lambda s: s.strip().upper())
# Out:
# [' bAR', 'BaZ ', ' foo ']
```

```
sorted([" foo ", " bAR", "BaZ "], key=lambda s: s.strip())
# Out:
# ['BaZ ', ' bAR', ' foo ']
```

map

```
sorted(map(lambda s: s.strip().upper(), [" foo ", " bAR", "BaZ "]))
# Out:
# ['BAR', 'BAZ', 'FOO']
```

```
sorted(map(lambda s: s.strip(), [" foo ", " bAR", "BaZ "]))
# Out:
# ['BaZ', 'bAR', 'foo']
```

```
my_list = [3, -4, -2, 5, 1, 7]
sorted(my_list, key=lambda x: abs(x))
# Out:
# [1, -2, 3, -4, 5, 7]

list(filter(lambda x: x>0, my_list))
# Out:
# [3, 5, 1, 7]

list(map(lambda x: abs(x), my_list))
# Out:
[3, 4, 2, 5, 1, 7]
```

---

## lambda/◦

```
def foo(msg):
    print(msg)

greet = lambda x = "hello world": foo(x)
greet()
```

```
hello world
```

lambda◦

---

## PEP-8 Pythonlambda

### deflambda◦

```
def f(x): return 2*x
```

```
f = lambda x: 2*x
```

f<lambda> ◦ ◦ lambda def◦

- **argument** ;
- **parameter** ◦

## Python //◦

- ◦

```
def foo(x):          # here x is the parameter
    x[0] = 9         # This mutates the list labelled by both x and y
    print(x)

y = [4, 5, 6]
foo(y)              # call foo with y as argument
# Out: [9, 5, 6]   # list labelled by x has been mutated
print(y)
```

```
# Out: [9, 5, 6] # list labelled by y has been mutated too
```

- ◦

```
def foo(x): # here x is the parameter, when we call foo(y) we assign y to x
    x[0] = 9 # This mutates the list labelled by both x and y
    x = [1, 2, 3] # x is now labeling a different list (y is unaffected)
    x[2] = 8 # This mutates x's list, not y's list

y = [4, 5, 6] # y is the argument, x is the parameter
foo(y) # Pretend that we wrote "x = y", then go to line 1
y
# Out: [9, 5, 6]
```

## Python ◦

- ◦ ◦
- ◦ ◦

```
x = [3, 1, 9]
y = x
x.append(5) # Mutates the list labelled by x and y, both x and y are bound to [3, 1, 9]
x.sort() # Mutates the list labelled by x and y (in-place sorting)
x = x + [4] # Does not mutate the list (makes a copy for x only, not y)
z = x # z is x ([1, 3, 9, 4])
x += [6] # Mutates the list labelled by both x and z (uses the extend function).
x = sorted(x) # Does not mutate the list (makes a copy for x only).
x
# Out: [1, 3, 4, 5, 6, 9]
y
# Out: [1, 3, 5, 9]
z
# Out: [1, 3, 5, 9, 4, 6]
```

## Python◦ makeIncincx◦ makeIncx◦

```
def makeInc(x):
    def inc(y):
        # x is "attached" in the definition of inc
        return y + x

    return inc

incOne = makeInc(1)
incFive = makeInc(5)

incOne(5) # returns 6
incFive(5) # returns 10
```

```
def makeInc(x):
    def inc(y):
        # incrementing x is not allowed
        x += y
        return x
```

```
return inc

incOne = makeInc(1)
incOne(5) # UnboundLocalError: local variable 'x' referenced before assignment
```

### Python 3 `nonlocal` [Nonlocal Variables](#) ◦

#### Python 3.x 3.0

```
def makeInc(x):
    def inc(y):
        nonlocal x
        # now assigning a value to x is allowed
        x += y
        return x

    return inc

incOne = makeInc(1)
incOne(5) # returns 6
```

### ◦ `factorial(n) = n*(n-1)*(n-2)*...*3*2*1` [factorial](#) ◦

```
def factorial(n):
    #n here should be an integer
    if n == 0:
        return 1
    else:
        return n*factorial(n-1)
```

```
factorial(0)
#out 1
factorial(1)
#out 1
factorial(2)
#out 2
factorial(3)
#out 6
```

### ◦ `return factorial(n-1)` ◦

### [lambda](#)

```
factorial = lambda n: 1 if n == 0 else n*factorial(n-1)
```

◦

### Python ◦ `RuntimeError`

```
def cursing(depth):
    try:
        cursing(depth + 1) # actually, re-cursing
    except RuntimeError as RE:
```

```
print('I recursed {} times!'.format(depth))

cursing(0)
# Out: I recursed 1083 times!
```

```
sys.setrecursionlimit(limit) sys.getrecursionlimit()
```

```
sys.setrecursionlimit(2000)
cursing(0)
# Out: I recursed 1997 times!
```

## Python 3.5 RecursionError RuntimeError

### python

```
def fibonacci(n):
    def step(a,b):
        return b, a+b
    a, b = 0, 1
    for i in range(n):
        a, b = step(a, b)
    return a
```

```
def make_adder(n):
    def adder(x):
        return n + x
    return adder
add5 = make_adder(5)
add6 = make_adder(6)
add5(10)
#Out: 15
add6(10)
#Out: 16
```

```
def repeatedly_apply(func, n, x):
    for i in range(n):
        x = func(x)
    return x
```

```
repeatedly_apply(add5, 5, 1)
#Out: 26
```

### argskwargs

```
def unpacking(a, b, c=45, d=60, *args, **kwargs):
    print(a, b, c, d, args, kwargs)
```

```
>>> unpacking(1, 2)
1 2 45 60 () {}
>>> unpacking(1, 2, 3, 4)
1 2 3 4 () {}
>>> unpacking(1, 2, c=3, d=4)
1 2 3 4 () {}
>>> unpacking(1, 2, d=4, c=3)
1 2 3 4 () {}
```

```

>>> pair = (3,)
>>> unpacking(1, 2, *pair, d=4)
1 2 3 4 () {}
>>> unpacking(1, 2, d=4, *pair)
1 2 3 4 () {}
>>> unpacking(1, 2, *pair, c=3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unpacking() got multiple values for argument 'c'
>>> unpacking(1, 2, c=3, *pair)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unpacking() got multiple values for argument 'c'

>>> args_list = [3]
>>> unpacking(1, 2, *args_list, d=4)
1 2 3 4 () {}
>>> unpacking(1, 2, d=4, *args_list)
1 2 3 4 () {}
>>> unpacking(1, 2, c=3, *args_list)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unpacking() got multiple values for argument 'c'
>>> unpacking(1, 2, *args_list, c=3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unpacking() got multiple values for argument 'c'

>>> pair = (3, 4)
>>> unpacking(1, 2, *pair)
1 2 3 4 () {}
>>> unpacking(1, 2, 3, 4, *pair)
1 2 3 4 (3, 4) {}
>>> unpacking(1, 2, d=4, *pair)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unpacking() got multiple values for argument 'd'
>>> unpacking(1, 2, *pair, d=4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unpacking() got multiple values for argument 'd'

>>> args_list = [3, 4]
>>> unpacking(1, 2, *args_list)
1 2 3 4 () {}
>>> unpacking(1, 2, 3, 4, *args_list)
1 2 3 4 (3, 4) {}
>>> unpacking(1, 2, d=4, *args_list)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unpacking() got multiple values for argument 'd'
>>> unpacking(1, 2, *args_list, d=4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unpacking() got multiple values for argument 'd'

```

```

>>> arg_dict = {'c':3, 'd':4}
>>> unpacking(1, 2, **arg_dict)
1 2 3 4 () {}
>>> arg_dict = {'d':4, 'c':3}
>>> unpacking(1, 2, **arg_dict)
1 2 3 4 () {}
>>> arg_dict = {'c':3, 'd':4, 'not_a_parameter': 75}
>>> unpacking(1, 2, **arg_dict)
1 2 3 4 () {'not_a_parameter': 75}

>>> unpacking(1, 2, *pair, **arg_dict)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unpacking() got multiple values for argument 'd'
>>> unpacking(1, 2, 3, 4, **arg_dict)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unpacking() got multiple values for argument 'd'

# Positional arguments take priority over any other form of argument passing
>>> unpacking(1, 2, **arg_dict, c=3)
1 2 3 4 () {'not_a_parameter': 75}
>>> unpacking(1, 2, 3, **arg_dict, c=3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unpacking() got multiple values for argument 'c'

```

o

```

def f(*a, b):
    pass

f(1, 2, 3)
# TypeError: f() missing 1 required keyword-only argument: 'b'

```

## Python 3o

```

def f(a, b, *, c):
    pass

f(1, 2, 3)
# TypeError: f() takes 2 positional arguments but 3 were given
f(1, 2, c=3)
# No error

```

## Lambda

lambdao -

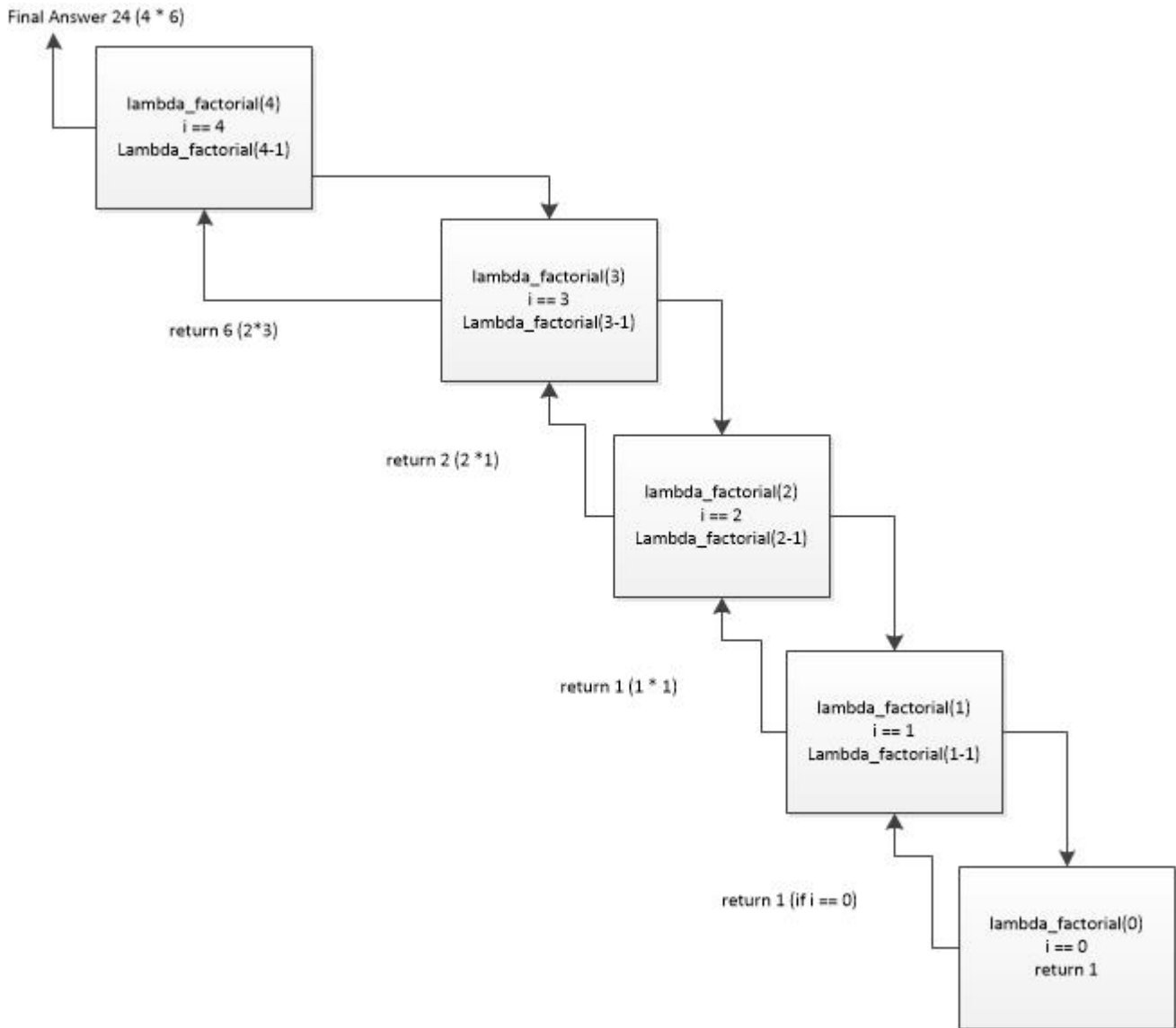
```

lambda_factorial = lambda i:1 if i==0 else i*lambda_factorial(i-1)
print(lambda_factorial(4)) # 4 * 3 * 2 * 1 = 12 * 2 = 24

```

lambda410 i \*lambda-1 i-1o 0 return 1o





<https://riptutorial.com/zh-TW/python/topic/228/>

# 91:

## Examples

helpPythonPython◦

```
>>> help()
```

```
Welcome to Python 3.4's help utility!
```

```
If this is your first time using Python, you should definitely check out the tutorial on the Internet at http://docs.python.org/3.4/tutorial/.
```

```
Enter the name of any module, keyword, or topic to get help on writing Python programs and using Python modules. To quit this help utility and return to the interpreter, just type "quit".
```

```
To get a list of available modules, keywords, symbols, or topics, type "modules", "keywords", "symbols", or "topics". Each module also comes with a one-line summary of what it does; to list the modules whose name or summary contain a given string such as "spam", type "modules spam".
```

— ◦

```
>>> 2 + 2
```

```
4
```

```
>>> _
```

```
4
```

```
>>> _ + 6
```

```
10
```

python◦ for◦ \_◦

```
>>> "Hello, {}".format("World")
```

```
'Hello, World'
```

```
>>> _
```

```
'Hello, World'
```

```
>>> def wontchangethings():
```

```
...     pass
```

```
>>> _
```

```
'Hello, World'
```

```
>>> 27 / 0
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
ZeroDivisionError: division by zero
```

```
>>> _
```

```
'Hello, World'
```

python◦ ◦

## Python

## Python Windows Python

```
$ py
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:44:40) [MSC v.1600 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

python2python3

## PATH Python

## PYTHONSTARTUP

Python PYTHONSTARTUP Python

## PYTHONSTARTUP

```
print("Welcome!")
```

## Python

```
$ py
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:44:40) [MSC v.1600 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
Welcome!
>>>
```

## Python py --help Python 3.4

Python Launcher

```
usage: py [ launcher-arguments ] [ python-arguments ] script [ script-arguments ]
```

Launcher arguments:

```
-2      : Launch the latest Python 2.x version
-3      : Launch the latest Python 3.x version
-X.Y    : Launch the specified Python version
-X.Y-32: Launch the specified 32bit Python version
```

The following help text is from Python:

```
usage: G:\Python34\python.exe [option] ... [-c cmd | -m mod | file | -] [arg] ...
```

Options and arguments (and corresponding environment variables):

```
-b      : issue warnings about str(bytes_instance), str(bytearray_instance)
         and comparing bytes/bytearray with str. (-bb: issue errors)
-B      : don't write .py[co] files on import; also PYTHONDONTWRITEBYTECODE=x
-c cmd  : program passed in as string (terminates option list)
-d      : debug output from parser; also PYTHONDEBUG=x
-E      : ignore PYTHON* environment variables (such as PYTHONPATH)
-h      : print this help message and exit (also --help)
-i      : inspect interactively after running script; forces a prompt even
         if stdin does not appear to be a terminal; also PYTHONINSPECT=x
-I      : isolate Python from the user's environment (implies -E and -s)
```

```

-m mod : run library module as a script (terminates option list)
-O      : optimize generated bytecode slightly; also PYTHONOPTIMIZE=x
-OO     : remove doc-strings in addition to the -O optimizations
-q      : don't print version and copyright messages on interactive startup
-s      : don't add user site directory to sys.path; also PYTHONNOUSERSITE
-S      : don't imply 'import site' on initialization
-u      : unbuffered binary stdout and stderr, stdin always buffered;
         also PYTHONUNBUFFERED=x
         see man page for details on internal buffering relating to '-u'
-v      : verbose (trace import statements); also PYTHONVERBOSE=x
         can be supplied multiple times to increase verbosity
-V      : print the Python version number and exit (also --version)
-W arg  : warning control; arg is action:message:category:module:lineno
         also PYTHONWARNINGS=arg
-x      : skip first line of source, allowing use of non-Unix forms of #!cmd
-X opt  : set implementation-specific option
file    : program read from script file
-       : program read from stdin (default; interactive mode if a tty)
arg ... : arguments passed to program in sys.argv[1:]

```

Other environment variables:

```

PYTHONSTARTUP: file executed on interactive startup (no default)
PYTHONPATH   : ';'-separated list of directories prefixed to the
               default module search path. The result is sys.path.
PYTHONHOME   : alternate <prefix> directory (or <prefix>;<exec_prefix>).
               The default module search path uses <prefix>\lib.
PYTHONCASEOK : ignore case in 'import' statements (Windows).
PYTHONIOENCODING: Encoding[:errors] used for stdin/stdout/stderr.
PYTHONFAULTHANDLER: dump the Python traceback on fatal errors.
PYTHONHASHSEED: if this variable is set to 'random', a random value is used
                 to seed the hashes of str, bytes and datetime objects. It can also be
                 set to an integer in the range [0,4294967295] to get hash values with a
                 predictable seed.

```

Python`help` ◦

`helpdocstring` ◦

```

>>> help(print)
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep:  string inserted between values, default a space.
    end:  string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.

```

`helpdocstring` ◦

```

>>> x = 2
>>> help(x)
Help on int object:

class int(object)

```

```
| int(x=0) -> integer
| int(x, base=10) -> integer
|
| Convert a number or string to an integer, or return 0 if no arguments
| are given. If x is a number, return x.__int__(). For floating point
| numbers, this truncates towards zero.
|
| If x is not a number or if base is given, then x must be a string,
| bytes, or bytearray instance representing an integer literal in the
| given base. The literal can be preceded by '+' or '-' and be surrounded
| by whitespace. The base defaults to 10. Valid bases are 0 and 2-36.
| Base 0 means to interpret the base from the string as an integer literal.
| >>> int('0b100', base=0)
| 4
|
| Methods defined here:
|
| __abs__(self, /)
|     abs(self)
|
| __add__(self, value, /)
|     Return self+value...
```

<https://riptutorial.com/zh-TW/python/topic/2473/-->

## 92:

- abc
- ab
- x =x
- xy=xy
- x +=x。 “op =”
- del xx
- for x in something#binds x
- xx
- Ex Exceptionex

## Examples

Python。 。 。

```
x = 'Hi'

def read_x():
    print(x)    # x is just referenced, therefore assumed global

read_x()       # prints Hi

def read_y():
    print(y)    # here y is just referenced, therefore assumed global

read_y()       # NameError: global name 'y' is not defined

def read_y():
    y = 'Hey'   # y appears in an assignment, therefore it's local
    print(y)    # will find the local y

read_y()       # prints Hey

def read_x_local_fail():
    if False:
        x = 'Hey' # x appears in an assignment, therefore it's local
    print(x)     # will look for the _local_ z, which is not assigned, and will not be found

read_x_local_fail() # UnboundLocalError: local variable 'x' referenced before assignment
```

```
x = 'Hi'

def change_local_x():
    x = 'Bye'
    print(x)
change_local_x() # prints Bye
print(x)        # prints Hi
```

global

```
x = 'Hi'
```

```
def change_global_x():
    global x
    x = 'Bye'
    print(x)

change_global_x() # prints Bye
print(x) # prints Bye
```

global◦ ◦

x

1. global x x
2. nonlocal x x
3. x = 5for x in range(3)x
4. x

```
def foo():
    a = 5
    print(a) # ok

print(a) # NameError: name 'a' is not defined
```

except

```
def foo():
    if True:
        a = 5
    print(a) # ok

b = 3
def bar():
    if False:
        b = 5
    print(b) # UnboundLocalError: local variable 'b' referenced before assignment
```

fora += 5

## Python 3.x 3.0

Python 3nonlocal◦ nonlocal◦ [PEP 3104](#)◦ ◦

```
def counter():
    num = 0
    def incrementer():
        num += 1
        return num
    return incrementer
```

## UnboundLocalErrornum◦

```
def counter():
    num = 0
```

```

def incremter():
    nonlocal num
    num += 1
    return num
return incremter

c = counter()
c() # = 1
c() # = 2
c() # = 3

```

nonlocal ◦ counter nonlocal ◦ SyntaxError ◦ nonlocal ◦

◦

```

x = 5
x += 7
for x in iterable: pass

```

- x5 ◦ x ◦ "" ◦

◦ **lambdas** ◦

```

a = 'global'

class Fred:
    a = 'class' # class scope
    b = (a for i in range(10)) # function scope
    c = [a for i in range(10)] # function scope
    d = a # class scope
    e = lambda: a # function scope
    f = lambda a=a: a # default argument uses class scope

    @staticmethod # or @classmethod, or regular instance method
    def g(): # function scope
        return a

print(Fred.a) # class
print(next(Fred.b)) # global
print(Fred.c[0]) # class in Python 2, global in Python 3
print(Fred.d) # class
print(Fred.e()) # global
print(Fred.f()) # class
print(Fred.g()) # global

```

b ce class ◦

---

## PEP 227

◦ ◦ ◦

Python

; - ◦



```
class A:
    a = 42
    b = list(a + i for i in range(10))
```

---

Martijn Pieters.

## del

◦

---

**del v**

vdel v◦

```
x = 5
print(x) # out: 5
del x
print(x) # NameError: name 'f' is not defined
```

```
del nonlocal global del vv◦ v del vnonlocal vglobal v◦
```

◦ ◦

---

**del v.name**

v.\_\_delattr\_\_(name)◦

name◦

```
class A:
    pass

a = A()
a.x = 7
print(a.x) # out: 7
del a.x
print(a.x) # error: AttributeError: 'A' object has no attribute 'x'
```

---

**del v[item]**

v.\_\_delitem\_\_(item)◦

itemv◦

```
x = {'a': 1, 'b': 2}
del x['a']
print(x) # out: {'b': 2}
print(x['a']) # error: KeyError: 'a'
```

`del v[a:b]`

`v.__delslice__(a, b)`

-

```
x = [0, 1, 2, 3, 4]
del x[1:3]
print(x) # out: [0, 3, 4]
```

`#del`

**Python**

◦

```
foo = 1 # global

def func():
    bar = 2 # local
    print(foo) # prints variable foo from global scope
    print(bar) # prints variable bar from local scope
```

◦ `locals()``globals()`◦

```
foo = 1

def func():
    bar = 2
    print(globals().keys()) # prints all variable names in global scope
    print(locals().keys()) # prints all variable names in local scope
```

```
foo = 1

def func():
    foo = 2 # creates a new variable foo in local scope, global foo is not affected

    print(foo) # prints 2

    # global variable foo still exists, unchanged:
    print(globals()['foo']) # prints 1
    print(locals()['foo']) # prints 2
```

global

```
foo = 1

def func():
    global foo
    foo = 2 # this modifies the global foo, rather than creating a local variable
```

o

```
foo = 1

def func():
    # This function has a local variable foo, because it is defined down below.
    # So, foo is local from this point. Global foo is hidden.

    print(foo) # raises UnboundLocalError, because local foo is not yet initialized
    foo = 7
    print(foo)
```

```
foo = 1

def func():
    # In this function, foo is a global variable from the beginning

    foo = 7 # global foo is modified

    print(foo) # 7
    print(globals()['foo']) # 7

    global foo # this could be anywhere within the function
    print(foo) # 7
```

---

o o

```
foo = 1

def f1():
    bar = 1

    def f2():
        baz = 2
        # here, foo is a global variable, baz is a local variable
        # bar is not in either scope
        print(locals().keys()) # ['baz']
        print('bar' in locals()) # False
        print('bar' in globals()) # False

    def f3():
        baz = 3
        print(bar) # bar from f1 is referenced so it enters local scope of f3 (closure)
        print(locals().keys()) # ['bar', 'baz']
        print('bar' in locals()) # True
        print('bar' in globals()) # False

    def f4():
        bar = 4 # a new local bar which hides bar from local scope of f1
        baz = 4
        print(bar)
        print(locals().keys()) # ['bar', 'baz']
        print('bar' in locals()) # True
        print('bar' in globals()) # False
```

# global VS nonlocal Python 3

◦

global◦

```
foo = 0 # global foo

def f1():
    foo = 1 # a new foo local in f1

    def f2():
        foo = 2 # a new foo local in f2

        def f3():
            foo = 3 # a new foo local in f3
            print(foo) # 3
            foo = 30 # modifies local foo in f3 only

        def f4():
            global foo
            print(foo) # 0
            foo = 100 # modifies global foo
```

Python 3nonlocal ◦

nonlocalPython

◦

Python 3.x 3.0

```
def f1():

    def f2():
        foo = 2 # a new foo local in f2

        def f3():
            nonlocal foo # foo from f2, which is the nearest enclosing scope
            print(foo) # 2
            foo = 20 # modifies foo from f2!
```

<https://riptutorial.com/zh-TW/python/topic/263/>

# 93:

Python List Python ◦ ◦ ◦ ◦

- [...]
- []

list iterable Python ◦ set tuple dictionary

list Python C ++ vector<void\*> Java ArrayList<Object> ◦ ◦

◦ list ◦

◦

## Examples

Python ◦

```
lst = [1, 2, 3, 4]
lst[0] # 1
lst[1] # 2
```

IndexError ◦

```
lst[4] # IndexError: list index out of range
```

◦

```
lst[-1] # 4
lst[-2] # 3
lst[-5] # IndexError: list index out of range
```

```
lst[len(lst)-1] # 4
```

lst[start:end:step] ◦ start end-1 ◦ start end step 1

```
lst[1:] # [2, 3, 4]
lst[:3] # [1, 2, 3]
lst[::2] # [1, 3]
lst[::-1] # [4, 3, 2, 1]
lst[-1:0:-1] # [4, 3, 2]
lst[5:8] # [] since starting index is greater than length of lst, returns empty list
lst[1:10] # [2, 3, 4] same as omitting ending index
```

```
lst[::-1] # [4, 3, 2, 1]
```

◦

```
lst[3:1:-1] # [4, 3]
```

```
reversed(lst)[0:2] # 0 = 1 -1  
                  # 2 = 3 -1
```

1.

slice\_\_getitem\_\_() ◦ Python ◦

```
data = 'chandan purohit    22 2000' #assuming data fields of fixed length  
name_slice = slice(0,19)  
age_slice = slice(19,21)  
salary_slice = slice(22,None)  
  
#now we can have more readable slices  
print(data[name_slice]) #chandan purohit  
print(data[age_slice]) #'22'  
print(data[salary_slice]) #'2000'
```

\_\_getitem\_\_ ◦

a

```
a = [1, 2, 3, 4, 5]
```

1. append(value) - ◦

```
# Append values 6, 7, and 7 to the list  
a.append(6)  
a.append(7)  
a.append(7)  
# a: [1, 2, 3, 4, 5, 6, 7, 7]  
  
# Append another list  
b = [8, 9]  
a.append(b)  
# a: [1, 2, 3, 4, 5, 6, 7, 7, [8, 9]]  
  
# Append an element of a different type, as list elements do not need to have the same  
type  
my_string = "hello world"  
a.append(my_string)  
# a: [1, 2, 3, 4, 5, 6, 7, 7, [8, 9], "hello world"]
```

append() ◦ ◦

```
# Appending a list to another list  
a = [1, 2, 3, 4, 5, 6, 7, 7]  
b = [8, 9]  
a.append(b)  
# a: [1, 2, 3, 4, 5, 6, 7, 7, [8, 9]]
```

```
a[8]
# Returns: [8,9]
```

## 2. extend(enumerable) - ◦

```
a = [1, 2, 3, 4, 5, 6, 7, 7]
b = [8, 9, 10]

# Extend list by appending all elements from b
a.extend(b)
# a: [1, 2, 3, 4, 5, 6, 7, 7, 8, 9, 10]

# Extend list with elements from a non-list enumerable:
a.extend(range(3))
# a: [1, 2, 3, 4, 5, 6, 7, 7, 8, 9, 10, 0, 1, 2]
```

+◦

```
a = [1, 2, 3, 4, 5, 6] + [7, 7] + b
# a: [1, 2, 3, 4, 5, 6, 7, 7, 8, 9, 10]
```

## 3. index(value, [startIndex]) - ◦ ValueError ◦

```
a.index(7)
# Returns: 6

a.index(49) # ValueError, because 49 is not in a.

a.index(7, 7)
# Returns: 7

a.index(7, 8) # ValueError, because there is no 7 starting at index 8
```

## 4. insert(index, value) - indexvalue ◦ index ◦

```
a.insert(0, 0) # insert 0 at position 0
a.insert(2, 5) # insert 5 at position 2
# a: [0, 1, 5, 2, 3, 4, 5, 6, 7, 7, 8, 9, 10]
```

## 5. pop([index]) - index◦ ◦

```
a.pop(2)
# Returns: 5
# a: [0, 1, 2, 3, 4, 5, 6, 7, 7, 8, 9, 10]
a.pop(8)
# Returns: 7
# a: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# With no argument:
a.pop()
# Returns: 10
# a: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

## 6. remove(value)

- ◦ ValueError ◦

```
a.remove(0)
a.remove(9)
# a: [1, 2, 3, 4, 5, 6, 7, 8]
a.remove(10)
# ValueError, because 10 is not in a
```

7. reverse() - None ◦

```
a.reverse()
# a: [8, 7, 6, 5, 4, 3, 2, 1]
```

◦

8. count(value) - ◦

```
a.count(7)
# Returns: 2
```

9. sort() - None ◦

```
a.sort()
# a = [1, 2, 3, 4, 5, 6, 7, 8]
# Sorts the list in numerical order
```

sort(reverse=True)◦

```
a.sort(reverse=True)
# a = [8, 7, 6, 5, 4, 3, 2, 1]
```

key

```
import datetime

class Person(object):
    def __init__(self, name, birthday, height):
        self.name = name
        self.birthday = birthday
        self.height = height

    def __repr__(self):
        return self.name

l = [Person("John Cena", datetime.date(1992, 9, 12), 175),
     Person("Chuck Norris", datetime.date(1990, 8, 28), 180),
     Person("Jon Skeet", datetime.date(1991, 7, 6), 185)]

l.sort(key=lambda item: item.name)
# l: [Chuck Norris, John Cena, Jon Skeet]

l.sort(key=lambda item: item.birthday)
# l: [Chuck Norris, Jon Skeet, John Cena]
```



```
l.sort(key=lambda item: item.height)
# l: [John Cena, Chuck Norris, Jon Skeet]
```

## dicts

```
import datetime

l = [{'name': 'John Cena', 'birthday': datetime.date(1992, 9, 12), 'height': 175},
     {'name': 'Chuck Norris', 'birthday': datetime.date(1990, 8, 28), 'height': 180},
     {'name': 'Jon Skeet', 'birthday': datetime.date(1991, 7, 6), 'height': 185}]

l.sort(key=lambda item: item['name'])
# l: [Chuck Norris, John Cena, Jon Skeet]

l.sort(key=lambda item: item['birthday'])
# l: [Chuck Norris, Jon Skeet, John Cena]

l.sort(key=lambda item: item['height'])
# l: [John Cena, Chuck Norris, Jon Skeet]
```

```
import datetime

l = [{'name': 'John Cena', 'birthday': datetime.date(1992, 9, 12), 'size': {'height': 175,
                                  'weight': 100}},
     {'name': 'Chuck Norris', 'birthday': datetime.date(1990, 8, 28), 'size': {'height': 180,
                                       'weight': 90}},
     {'name': 'Jon Skeet', 'birthday': datetime.date(1991, 7, 6), 'size': {'height': 185,
                                   'weight': 110}}]

l.sort(key=lambda item: item['size']['height'])
# l: [John Cena, Chuck Norris, Jon Skeet]
```

## attrgetteritemgetter

attrgetteritemgetter° °

```
from operator import itemgetter, attrgetter

people = [{'name': 'chandan', 'age': 20, 'salary': 2000},
          {'name': 'chetan', 'age': 18, 'salary': 5000},
          {'name': 'guru', 'age': 30, 'salary': 3000}]
by_age = itemgetter('age')
by_salary = itemgetter('salary')

people.sort(key=by_age) #in-place sorting by age
people.sort(key=by_salary) #in-place sorting by salary
```

itemgetter° °

```
list_of_tuples = [(1,2), (3,4), (5,0)]
list_of_tuples.sort(key=itemgetter(1))
print(list_of_tuples) #[ (5, 0), (1, 2), (3, 4) ]
```

attrgetter

```

persons = [Person("John Cena", datetime.date(1992, 9, 12), 175),
           Person("Chuck Norris", datetime.date(1990, 8, 28), 180),
           Person("Jon Skeet", datetime.date(1991, 7, 6), 185)] #reusing Person class from
above example

person.sort(key=attrgetter('name')) #sort by name
by_birthday = attrgetter('birthday')
person.sort(key=by_birthday) #sort by birthday

```

## 10. clear() -

```

a.clear()
# a = []

```

## 11. - o

```

b = ["blah"] * 3
# b = ["blah", "blah", "blah"]
b = [1, 3, 5] * 5
# [1, 3, 5, 1, 3, 5, 1, 3, 5, 1, 3, 5, 1, 3, 5]

```

- o

## 12. - del

```

a = list(range(10))
del a[::2]
# a = [1, 3, 5, 7, 9]
del a[-1]
# a = [1, 3, 5, 7]
del a[:]
# a = []

```

## 13. "=" o o o o

```

b = a
a.append(6)
# b: [1, 2, 3, 4, 5, 6]

```

o

```

new_list = old_list[:]

```

### list

```

new_list = list(old_list)

```

### copy.copy

```

import copy
new_list = copy.copy(old_list) #inserts references to the objects found in the original.

```

listold\_list。

copy.deepcopy

```
import copy
new_list = copy.deepcopy(old_list) #inserts copies of the objects found in the original.
```

。

## Python 3.x 3.0

copy() -

```
aa = a.copy()
# aa = [1, 2, 3, 4, 5]
```

len()。

```
len(['one', 'two']) # returns 2

len(['one', [2, 3], 'four']) # returns 3, not 4
```

len()。

len()。

len()O(1)。

## Python for

```
my_list = ['foo', 'bar', 'baz']
for item in my_list:
    print(item)
```

```
# Output: foo
# Output: bar
# Output: baz
```

```
for (index, item) in enumerate(my_list):
    print('The item in position {} is: {}'.format(index, item))
```

```
# Output: The item in position 0 is: foo
# Output: The item in position 1 is: bar
# Output: The item in position 2 is: baz
```

```
for i in range(0, len(my_list)):
    print(my_list[i])
```

```
#output:
>>>
foo
bar
baz
```

```
for item in my_list:
    if item == 'foo':
        del my_list[0]
    print(item)

# Output: foo
# Output: baz
```

bar°

Python° in°

```
lst = ['test', 'twest', 'tweast', 'treast']

'test' in lst
# Out: True

'toast' in lst
# Out: False
```

in° listset set °

```
slst = set(lst)
'test' in slst
# Out: True
```

reversed

```
In [3]: rev = reversed(numbers)

In [4]: rev
Out[4]: [9, 8, 7, 6, 5, 4, 3, 2, 1]
```

“”°

[reverse](#) °

-1

```
In [1]: numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

In [2]: numbers[::-1]
Out[2]: [9, 8, 7, 6, 5, 4, 3, 2, 1]
```

False len(lst) == 0 lst not lst

```
lst = []
if not lst:
    print("list is empty")

# Output: list is empty
```

1.

```
list1list2
```

```
merged = list1 + list2
```

## 2. zip ȳ

```
alist = ['a1', 'a2', 'a3']  
blist = ['b1', 'b2', 'b3']  
  
for a, b in zip(alist, blist):  
    print(a, b)
```

```
# Output:  
# a1 b1  
# a2 b2  
# a3 b3
```

```
alist = ['a1', 'a2', 'a3']  
blist = ['b1', 'b2', 'b3', 'b4']  
for a, b in zip(alist, blist):  
    print(a, b)
```

```
# Output:  
# a1 b1  
# a2 b2  
# a3 b3
```

```
alist = []  
len(list(zip(alist, blist)))
```

```
# Output:  
# 0
```

Noneitertools.zip\_longest Python 2itertools.izip\_longest

```
alist = ['a1', 'a2', 'a3']  
blist = ['b1']  
clist = ['c1', 'c2', 'c3', 'c4']  
  
for a,b,c in itertools.zip_longest(alist, blist, clist):  
    print(a, b, c)
```

```
# Output:  
# a1 b1 c1  
# a2 None c2  
# a3 None c3  
# None None c4
```

3. 

```
alist = [123, 'xyz', 'zara', 'abc']  
alist.insert(3, [2009])  
print("Final List :", alist)
```

```
Final List : [123, 'xyz', 'zara', 2009, 'abc']
```

all()

## iterableTrue

```
nums = [1, 1, 0, 1]
all(nums)
# False
chars = ['a', 'b', 'c', 'd']
all(chars)
# True
```

## any() iterableTrue

```
nums = [1, 1, 0, 1]
any(nums)
# True
vals = [None, None, None, False]
any(vals)
# False
```

◦

```
vals = [1, 2, 3, 4]
any(val > 12 for val in vals)
# False
any((val * 2) > 6 for val in vals)
# True
```

## set ◦ listlist()list()

```
names = ["aixk", "duke", "edik", "tofp", "duke"]
list(set(names))
# Out: ['duke', 'tofp', 'aixk', 'edik']
```

◦

## OrderedDict

```
import collections
>>> collections.OrderedDict.fromkeys(names).keys()
# Out: ['aixk', 'duke', 'edik', 'tofp']
```

```
alist = [[1,2],[3,4]], [[5,6,7],[8,9,10], [12, 13, 14]]
```

```
print(alist[0][0][1])
#2
#Accesses second element in the first list in the first list
```

```
print(alist[1][1][2])
#10
#Accesses the third element in the second list in the second list
```

```
alist[0][0].append(11)
print(alist[0][0][2])
```

```
#11
#Appends 11 to the end of the first list in the first list
```

## for

```
for row in alist: #One way to loop through nested lists
    for col in row:
        print(col)
#[1, 2, 11]
#[3, 4]
#[5, 6, 7]
#[8, 9, 10]
#[12, 13, 14]
```

```
[col for row in alist for col in row]
#[[1, 2, 11], [3, 4], [5, 6, 7], [8, 9, 10], [12, 13, 14]]
```

```
alist[1].insert(2, 15)
#Inserts 15 into the third position in the second list
```

## for◦

```
for row in range(len(alist)): #A less Pythonic way to loop through lists
    for col in range(len(alist[row])):
        print(alist[row][col])

#[1, 2, 11]
#[3, 4]
#[5, 6, 7]
#[8, 9, 10]
#15
#[12, 13, 14]
```

```
print(alist[1][1:])
#[[8, 9, 10], 15, [12, 13, 14]]
#Slices still work
```

```
print(alist)
#[[[1, 2, 11], [3, 4]], [[5, 6, 7], [8, 9, 10], 15, [12, 13, 14]]]
```

◦ ◦

```
[1, 10, 100] < [2, 10, 100]
# True, because 1 < 2
[1, 10, 100] < [1, 10, 100]
# False, because the lists are equal
[1, 10, 100] <= [1, 10, 100]
# True, because the lists are equal
[1, 10, 100] < [1, 10, 101]
# True, because 100 < 101
[1, 10, 100] < [0, 10, 100]
# False, because 0 < 1
```

o

```
[1, 10] < [1, 10, 100]  
# True
```

None

```
my_list = [None] * 10  
my_list = ['test'] * 10
```

```
>>> my_list=[{1}] * 10  
>>> print(my_list)  
[{1}, {1}, {1}, {1}, {1}, {1}, {1}, {1}, {1}, {1}]  
>>> my_list[0].add(2)  
>>> print(my_list)  
[{1, 2}, {1, 2}, {1, 2}, {1, 2}, {1, 2}, {1, 2}, {1, 2}, {1, 2}, {1, 2}, {1, 2}]
```

```
my_list=[{1} for _ in range(10)]
```

<https://riptutorial.com/zh-TW/python/topic/209/>



# 94:

Python ◦ unittest ◦ py.test ◦ nosetest ◦ python ◦

## Examples

◦ ◦

```
class WrongInputException(Exception):
    pass
```

◦

```
def convert2number(random_input):
    try:
        my_input = int(random_input)
    except ValueError:
        raise WrongInputException("Expected an integer!")
    return my_input
```

assertRaises ◦ assertRaises

1. ◦ ◦

2. with ◦ ◦ **with self.assertRaisesWrongInputExceptionconvert2number“not a number”**

```
import unittest

class ExceptionTestCase(unittest.TestCase):

    def test_wrong_input_string(self):
        self.assertRaises(WrongInputException, convert2number, "not a number")

    def test_correct_input(self):
        try:
            result = convert2number("56")
            self.assertIsInstance(result, int)
        except WrongInputException:
            self.fail()
```

◦ ◦ ◦ fail ◦

## unittest.mock.create\_autospec

create\_autospec ◦ ◦

custom\_math.pymultiply

```
def multiply(a, b):
    return a * b
```

process\_math.pymultiples\_of

```
from custom_math import multiply

def multiples_of(integer, *args, num_multiples=0, **kwargs):
    """
    :rtype: list
    """
    multiples = []

    for x in range(1, num_multiples + 1):
        """
        Passing in args and kwargs here will only raise TypeError if values were
        passed to multiples_of function, otherwise they are ignored. This way we can
        test that multiples_of is used correctly. This is here for an illustration
        of how create_autospec works. Not recommended for production code.
        """
        multiple = multiply(integer, x, *args, **kwargs)
        multiples.append(multiple)

    return multiples
```

multiplymultiples\_of ◦ Pythonunittestpytestnose

```
from unittest.mock import create_autospec
import unittest

# we import the entire module so we can mock out multiply
import custom_math
custom_math.multiply = create_autospec(custom_math.multiply)
from process_math import multiples_of

class TestCustomMath(unittest.TestCase):
    def test_multiples_of(self):
        multiples = multiples_of(3, num_multiples=1)
        custom_math.multiply.assert_called_with(3, 1)

    def test_multiples_of_with_bad_inputs(self):
        with self.assertRaises(TypeError) as e:
            multiples_of(1, "extra arg", num_multiples=1) # this should raise a TypeError
```

## test.TestCase

◦ setUp◦ tearDown◦ ◦ **TestCases**super setUptearDown◦ TestCasessetUptearDown

```
import unittest

class SomeTest(unittest.TestCase):
    def setUp(self):
        super(SomeTest, self).setUp()
        self.mock_data = [1,2,3,4,5]

    def test(self):
        self.assertEqual(len(self.mock_data), 5)
```

```

def tearDown(self):
    super(SomeTest, self).tearDown()
    self.mock_data = []

if __name__ == '__main__':
    unittest.main()

```

python2.7 +addCleanup° setUptearDownsetUpaddCleanup°

```

import unittest
import some_module

class SomeOtherTest(unittest.TestCase):
    def setUp(self):
        super(SomeOtherTest, self).setUp()

        # Replace `some_module.method` with a `mock.Mock`
        my_patch = mock.patch.object(some_module, 'method')
        my_patch.start()

        # When the test finishes running, put the original method back.
        self.addCleanup(my_patch.stop)

```

tearDownsuper°

unittest°

```

def division_function(dividend, divisor):
    return dividend / divisor

class MyTestCase(unittest.TestCase):
    def test_using_context_manager(self):
        with self.assertRaises(ZeroDivisionError):
            x = division_function(1, 0)

```

° °

°

```

class MyTestCase(unittest.TestCase):
    def test_using_context_manager(self):
        with self.assertRaises(ZeroDivisionError) as ex:
            x = division_function(1, 0)

        self.assertEqual(ex.message, 'integer division or modulo by zero')

```

```

def division_function(dividend, divisor):
    """
    Dividing two numbers.

    :type dividend: int

```

```

:type divisor: int

:raises: ZeroDivisionError if divisor is zero (0).
:rtype: int
"""
return dividend / divisor

class MyTestCase(unittest.TestCase):
    def test_passing_function(self):
        self.assertRaises(ZeroDivisionError, division_function, 1, 0)

```

◦ ◦

## Unittests

### Python [assert](#) Python◦

assertTrue

```

import unittest

class SimplisticTest(unittest.TestCase):
    def test_basic(self):
        self.assertTrue(1 + 1 == 2)

```

```

self.assertTrue(1 + 1 == 3)

```

◦

assertTrue◦

```

self.assertEqual(1 + 1, 3)

```

```

=====
FAIL: test (__main__.TruthTest)
-----

Traceback (most recent call last):
  File "stuff.py", line 6, in test
    self.assertTrue(1 + 1 == 3)
AssertionError: False is not true

```

```

=====
FAIL: test (__main__.TruthTest)
-----

```

```
Traceback (most recent call last):
```

```
File "stuff.py", line 6, in test
```

```
self.assertEqual(1 + 1, 3)
```

```
AssertionError: 2 != 3
```

- 
- ◦ 1 + 1 == 2 assertEquals assertTrue ◦ a is None assertIsNone assertEquals ◦
- assertEquals assertNotEqual assertIsNone assertIsNotNone ◦ ◦

## pytest

### pytest

```
pip install pytest
```

```
mkdir tests  
touch tests/test_docker.py
```

docker\_something/helpers.py

```
from subprocess import Popen, PIPE  
# this Popen is monkeypatched with the fixture `all_popens`  
  
def copy_file_to_docker(src, dest):  
    try:  
        result = Popen(['docker', 'cp', src, 'something_cont:{}'.format(dest)], stdout=PIPE,  
                        stderr=PIPE)  
        err = result.stderr.read()  
        if err:  
            raise Exception(err)  
    except Exception as e:  
        print(e)  
    return result  
  
def docker_exec_something(something_file_string):  
    fl = Popen(["docker", "exec", "-i", "something_cont", "something"], stdin=PIPE,  
              stdout=PIPE, stderr=PIPE)  
    fl.stdin.write(something_file_string)  
    fl.stdin.close()  
    err = fl.stderr.read()  
    fl.stderr.close()  
    if err:  
        print(err)  
        exit()  
    result = fl.stdout.read()  
    print(result)
```

test\_docker.py

```

import os
from tempfile import NamedTemporaryFile
import pytest
from subprocess import Popen, PIPE

from docker_something import helpers
copy_file_to_docker = helpers.copy_file_to_docker
docker_exec_something = helpers.docker_exec_something

```

test\_docker.pytest\_docker.py

```

class MockBytes():
    '''Used to collect bytes
    '''
    all_read = []
    all_write = []
    all_close = []

    def read(self, *args, **kwargs):
        # print('read', args, kwargs, dir(self))
        self.all_read.append((self, args, kwargs))

    def write(self, *args, **kwargs):
        # print('wrote', args, kwargs)
        self.all_write.append((self, args, kwargs))

    def close(self, *args, **kwargs):
        # print('closed', self, args, kwargs)
        self.all_close.append((self, args, kwargs))

    def get_all_mock_bytes(self):
        return self.all_read, self.all_write, self.all_close

```

test\_docker.py**pytest**

```

@pytest.fixture
def all_popens(monkeypatch):
    '''This fixture overrides / mocks the builtin Popen
    and replaces stdin, stdout, stderr with a MockBytes object

    note: monkeypatch is magically imported
    '''
    all_popens = []

    class MockPopen(object):
        def __init__(self, args, stdout=None, stdin=None, stderr=None):
            all_popens.append(self)
            self.args = args
            self.byte_collection = MockBytes()
            self.stdin = self.byte_collection
            self.stdout = self.byte_collection
            self.stderr = self.byte_collection
            pass
    monkeypatch.setattr(helpers, 'Popen', MockPopen)

    return all_popens

```

test\_docker.pytest\_ test\_docker.py

```

def test_docker_install():
    p = Popen(['which', 'docker'], stdout=PIPE, stderr=PIPE)
    result = p.stdout.read()
    assert 'bin/docker' in result

def test_copy_file_to_docker(all_popen):
    result = copy_file_to_docker('asdf', 'asdf')
    collected_popen = all_popen.pop()
    mock_read, mock_write, mock_close = collected_popen.byte_collection.get_all_mock_bytes()
    assert mock_read
    assert result.args == ['docker', 'cp', 'asdf', 'something_cont:asdf']

def test_docker_exec_something(all_popen):

    docker_exec_something(something_file_string)

    collected_popen = all_popen.pop()
    mock_read, mock_write, mock_close = collected_popen.byte_collection.get_all_mock_bytes()
    assert len(mock_read) == 3
    something_template_stdin = mock_write[0][1][0]
    these = [os.environ['USER'], os.environ['password_prod'], 'table_name_here', 'test_vdm',
'col_a', 'col_b', '/tmp/test.tsv']
    assert all([x in something_template_stdin for x in these])

```

```

py.test -k test_docker_install tests
py.test -k test_copy_file_to_docker tests
py.test -k test_docker_exec_something tests

```

teststests

```

py.test -k test_ tests

```

<https://riptutorial.com/zh-TW/python/topic/631/>

---

# 95: GIL

---

## GIL

1992PythonGILCPython python GILPythonPython GIL

GILpython

o

GILPython pythonGIL PythonGIL

**GIL** CPythonPyPyCythonnogilGIL

**GIL** JythonIronPython

---

## GIL

GIL GIL CPython3.2python 3.2GIL dabeaz.com

I/OCPythonGIL numpyGIL

---

## GIL

GILGIL o

- - GIL CPythongarbageGIL ;GILpython.org wikiGIL
- GIL -
- 

---

## GIL

GILpythonpython python GIL

---

<https://wiki.python.org/moin/GlobalInterpreterLock> -

<http://programmers.stackexchange.com/questions/186889/why-was-python-written-with-the-gil> -

<http://www.dabeaz.com/python/UnderstandingGIL.pdf> - GIL

<http://www.dabeaz.com/GIL/gilvis/index.html> - GIL



<http://jeffknupp.com/blog/2012/03/31/pythons-hardest-problem/> - GIL

<https://jeffknupp.com/blog/2013/06/30/pythons-hardest-problem-revisited/> - GIL

## Examples

### Multiprocessing.Pool

Python“。 。 ”Pool。

[David BeazleyGIL multiprocessing.Pool](#)

---

## David BeazleyGIL

```
from threading import Thread
import time
def countdown(n):
    while n > 0:
        n -= 1

COUNT = 10000000

t1 = Thread(target=countdown, args=(COUNT/2,))
t2 = Thread(target=countdown, args=(COUNT/2,))
start = time.time()
t1.start();t2.start()
t1.join();t2.join()
end = time.time()
print end-start
```

### multiprocessing.Pool

```
import multiprocessing
import time
def countdown(n):
    while n > 0:
        n -= 1

COUNT = 10000000

start = time.time()
with multiprocessing.Pool as pool:
    pool.map(countdown, [COUNT/2, COUNT/2])

    pool.close()
    pool.join()

end = time.time()
print(end-start)
```

。 GIL。 multiprocessing.Pool。 。 。 。

with。 。

## Cython nogil

Cythonpython。 GIL。

David BeazleyGIL nogil

---

# David BeazleyGIL

```
from threading import Thread
import time
def countdown(n):
    while n > 0:
        n -= 1

COUNT = 10000000

t1 = Thread(target=countdown, args=(COUNT/2,))
t2 = Thread(target=countdown, args=(COUNT/2,))
start = time.time()
t1.start();t2.start()
t1.join();t2.join()
end = time.time()
print end-start
```

---

# nogilCYTHON

```
from threading import Thread
import time
def countdown(n):
    while n > 0:
        n -= 1

COUNT = 10000000

with nogil:
    t1 = Thread(target=countdown, args=(COUNT/2,))
    t2 = Thread(target=countdown, args=(COUNT/2,))
    start = time.time()
    t1.start();t2.start()
    t1.join();t2.join()

end = time.time()
print end-start
```

cython。 python

PythonGILPython。 Cython。

GIL <https://riptutorial.com/zh-TW/python/topic/4061/-gil->

# 96:

python

## Examples

### PyDotPlus

PyDotPlus pydot Graphviz Dot Python.

```
pip install pydotplus
```

```
pip install https://github.com/carlos-jenkins/pydotplus/archive/master.zip
```

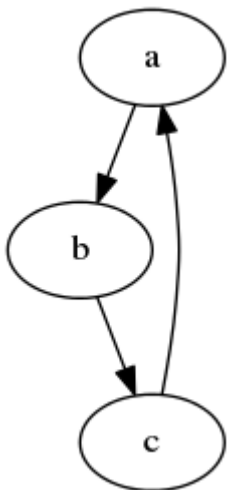
DOT

- DOT. Dot. demo.dot

```
digraph demo1 {a -> b -> c; c -> a; }
```

```
import pydotplus
graph_a = pydotplus.graph_from_dot_file('demo.dot')
graph_a.write_svg('test.svg') # generate graph in svg.
```

svg



### PyGraphviz

<http://pypi.python.org/pypi/pygraphviz> Python PyGraphviz

```
pip install pygraphviz
```

Python.

github.com

```
pip install git://github.com/pygraphviz/pygraphviz.git#egg=pygraphviz
```

<http://pypi.python.org/pypi/pygraphviz> PythonPyGraphviz

```
easy_install pygraphviz
```

Python。

DOT

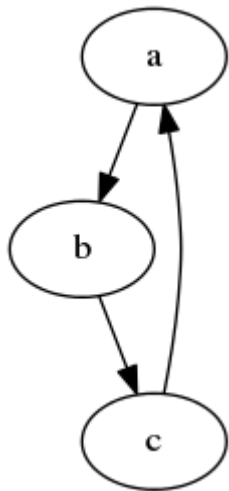
- DOT。 Dot。 demo.dot

```
digraph demo1 {a -> b -> c; c -> a; }
```

- 。

```
import pygraphviz as pgv
G = pgv.AGraph("demo.dot")
G.draw('test', format='svg', prog='dot')
```

svg



<https://riptutorial.com/zh-TW/python/topic/9483/>

# 97: windowvirtualenvwrapper

## Examples

### windowvirtualenvwrapper

ABC.ABpython 3。 Cpython 2.7。

- python

1pip python -m pip install -U pip

2“virtualenvwrapper-win”windows power shell

```
pip install virtualenvwrapper-win
```

3virtualenv mkvirtualenv python\_3.5

4

```
workon < environment name>
```

### virtualenvwrapper

```
mkvirtualenv <name>
```

Create a new virtualenv environment named <name>. The environment will be created in WORKON\_HOME.

```
lsvirtualenv
```

List all of the environments stored in WORKON\_HOME.

```
rmvirtualenv <name>
```

Remove the environment <name>. Uses folder\_delete.bat.

```
workon [<name>]
```

If <name> is specified, activate the environment named <name> (change the working virtualenv to <name>). If a project directory has been defined, we will change into it. If no argument is specified, list the available environments. One can pass additional option -c after virtualenv name to cd to virtualenv directory if no projectdir is set.

```
deactivate
```

Deactivate the working virtualenv and switch back to the default system Python.

```
add2virtualenv <full or relative path>
```

If a virtualenv environment is active, appends <path> to virtualenv\_path\_extensions.pth inside the environment's site-packages, which effectively adds <path> to the environment's PYTHONPATH. If a virtualenv environment is not active, appends <path> to virtualenv\_path\_extensions.pth inside the default Python's site-packages. If <path> doesn't exist, it will be created.

windowvirtualenvwrapper <https://riptutorial.com/zh-TW/python/topic/9984/windows-virtualenvwrapper>

---

# 98:

Python“return”。

## Examples

var1

```
def func(params):
    for value in params:
        print ('Got value {}'.format(value))

        if value == 1:
            # Returns from function as soon as value is 1
            print (">>>> Got 1")
            return

        print ("Still looping")

    return "Couldn't find 1"

func([5, 3, 1, 2, 8, 9])
```

```
Got value 5
Still looping
Got value 3
Still looping
Got value 1
>>>> Got 1
```

<https://riptutorial.com/zh-TW/python/topic/10883/>

# 99:

- `mapfunctioniterable` [\* `additional_iterables`]
- `future_builtins.mapfunctioniterable` [\* `additional_iterables`]
- `itertools.imapfunctioniterable` [\* `additional_iterables`]

	iterable
* <code>additional_iterables</code>	

`map`[comprehensions](#)

```
list(map(abs, [-1,-2,-3])) # [1, 2, 3]
[abs(i) for i in [-1,-2,-3]] # [1, 2, 3]
```

`iterables`[zip](#)

```
import operator
alist = [1,2,3]
list(map(operator.add, alist, alist)) # [2, 4, 6]
[i + j for i, j in zip(alist, alist)] # [2, 4, 6]
```

`map`°

## Examples

`map`[itertools.imap](#)[future\\_builtins.map](#)

`map`Python° [map\(\)](#) `iterable`

```
names = ['Fred', 'Wilma', 'Barney']
```

Python 3.x 3.0

```
map(len, names) # map in Python 3.x is a class; its instances are iterable
# Out: <map object at 0x00000198B32E2CF8>
```

`future_builtins`Python 3<sub>map</sub>

Python 2.x 2.6

```
from future_builtins import map # contains a Python 3.x compatible map()
map(len, names) # see below
# Out: <itertools.imap instance at 0x3eb0a20>
```

## Python 2 `itertools` `imap`

### Python 2.x 2.3

```
map(len, names) # map() returns a list
# Out: [4, 5, 6]

from itertools import imap
imap(len, names) # itertools.imap() returns a generator
# Out: <itertools.imap at 0x405ea20>
```

### list Python 23

```
list(map(len, names))
# Out: [4, 5, 6]
```

`map()`

```
[len(item) for item in names] # equivalent to Python 2.x map()
# Out: [4, 5, 6]
```

```
(len(item) for item in names) # equivalent to Python 3.x map()
# Out: <generator object <genexpr> at 0x00000195888D5FC0>
```

```
list(map(abs, (1, -1, 2, -2, 3, -3))) # the call to `list` is unnecessary in 2.x
# Out: [1, 1, 2, 2, 3, 3]
```

```
map(lambda x:x*2, [1, 2, 3, 4, 5])
# Out: [2, 4, 6, 8, 10]
```

```
def to_percent(num):
    return num * 100

list(map(to_percent, [0.95, 0.75, 1.01, 0.1]))
# Out: [95.0, 75.0, 101.0, 10.0]
```

```
from functools import partial
from operator import mul

rate = 0.9 # fictitious exchange rate, 1 dollar = 0.9 euros
dollars = {'under_my_bed': 1000,
           'jeans': 45,
           'bank': 5000}

sum(map(partial(mul, rate), dollars.values()))
# Out: 5440.5
```

`functools.partialmaplambda`

`i`

```
def average(*args):
```



```

    return float(sum(args)) / len(args) # cast to float - only mandatory for python 2.x

measurement1 = [100, 111, 99, 97]
measurement2 = [102, 117, 91, 102]
measurement3 = [104, 102, 95, 101]

list(map(average, measurement1, measurement2, measurement3))
# Out: [102.0, 110.0, 95.0, 100.0]

```

## pythoniterable<sub>map</sub>

- ```

def median_of_three(a, b, c):
    return sorted((a, b, c))[1]

list(map(median_of_three, measurement1, measurement2))

```

**TypeError: median\_of\_three() takes 3 positional arguments but 4 were given**

```
list(map(median_of_three, measurement1, measurement2, measurement3, measurement3))
```

**TypeError: median\_of\_three() takes 3 positional arguments but 4 were given**

## Python 2.x 2.0.1

- map **iterable** iterables **None**

```

import operator

measurement1 = [100, 111, 99, 97]
measurement2 = [102, 117]

# Calculate difference between elements
list(map(operator.sub, measurement1, measurement2))

```

**TypeError: 'int' object is not iterable**

- itertools.imap future\_builtins.map **iterable**

```

import operator
from itertools import imap

measurement1 = [100, 111, 99, 97]
measurement2 = [102, 117]

# Calculate difference between elements
list(imap(operator.sub, measurement1, measurement2))
# Out: [-2, -6]
list(imap(operator.sub, measurement2, measurement1))
# Out: [2, 6]

```

## Python 3.x 3.0.0

- import operator

```

measurement1 = [100, 111, 99, 97]
measurement2 = [102, 117]

# Calculate difference between elements
list(map(operator.sub, measurement1, measurement2))
# Out: [-2, -6]
list(map(operator.sub, measurement2, measurement1))
# Out: [2, 6]

```

## Map“None”python 2.x

```

from itertools import imap
from future_builtins import map as fmap # Different name to highlight differences

image = [[1, 2, 3],
         [4, 5, 6],
         [7, 8, 9]]

list(map(None, *image))
# Out: [(1, 4, 7), (2, 5, 8), (3, 6, 9)]
list(fmap(None, *image))
# Out: [(1, 4, 7), (2, 5, 8), (3, 6, 9)]
list(imap(None, *image))
# Out: [(1, 4, 7), (2, 5, 8), (3, 6, 9)]

image2 = [[1, 2, 3],
         [4, 5],
         [7, 8, 9]]
list(map(None, *image2))
# Out: [(1, 4, 7), (2, 5, 8), (3, None, 9)] # Fill missing values with None
list(fmap(None, *image2))
# Out: [(1, 4, 7), (2, 5, 8)] # ignore columns with missing values
list(imap(None, *image2))
# Out: [(1, 4, 7), (2, 5, 8)] # dito

```

## Python 3.x 3.0.0

```
list(map(None, *image))
```

**TypeError'NoneType'**

```

def conv_to_list(*args):
    return list(args)

list(map(conv_to_list, *image))
# Out: [[1, 4, 7], [2, 5, 8], [3, 6, 9]]

```

map'import'. print5importprintimport pprint. pprint

iterable. ◦

1

```
insects = ['fly', 'ant', 'beetle', 'cankerworm']
```

```
f = lambda x: x + ' is an insect'
print(list(map(f, insects))) # the function defined by f is executed on each item of the
iterable insects
```

```
['fly is an insect', 'ant is an insect', 'beetle is an insect', 'cankerworm is an insect']
```

## 2

```
print(list(map(len, insects))) # the len function is executed each item in the insect list
```

```
[3, 3, 6, 10]
```

o o

```
carnivores = ['lion', 'tiger', 'leopard', 'arctic fox']
herbivores = ['african buffalo', 'moose', 'okapi', 'parakeet']
omnivores = ['chicken', 'dove', 'mouse', 'pig']

def animals(w, x, y, z):
    return '{0}, {1}, {2}, and {3} ARE ALL ANIMALS'.format(w.title(), x, y, z)
```

## 3

```
# Too many arguments
# observe here that map is trying to pass one item each from each of the four iterables to
len. This leads len to complain that
# it is being fed too many arguments
print(list(map(len, insects, carnivores, herbivores, omnivores)))
```

```
TypeError: len() takes exactly one argument (4 given)
```

## 4

```
# Too few arguments
# observe here that map is suppose to execute animal on individual elements of insects one-by-
one. But animals complain when
# it only gets one argument, whereas it was expecting four.
print(list(map(animals, insects)))
```

```
TypeError: animals() missing 3 required positional arguments: 'x', 'y', and 'z'
```

## 5

```
# here map supplies w, x, y, z with one value from across the list
import pprint
pprint.pprint(list(map(animals, insects, carnivores, herbivores, omnivores)))
```

```
['Fly, lion, african buffalo, and chicken ARE ALL ANIMALS',
 'Ant, tiger, moose, and dove ARE ALL ANIMALS',
 'Beetle, leopard, okapi, and mouse ARE ALL ANIMALS',
```

```
'Cankerworm, arctic fox, parakeet, and pig ARE ALL ANIMALS']
```

<https://riptutorial.com/zh-TW/python/topic/333/>

# 100:

Python3.5 a = myobject myobject 0

Python del del refcount refcount

Python a = object cpython refcount 0 GC

1960 John McCarthy Lisp Python **Generational Garbage Collection**

Python Python 0 "1" 0 Python 30; GC1 GC2 "2" GCed GC GC " " "60

GC 25 GC " "; GC GC Martin von Löwis 2008 " " " " "

GC

```
/* Break reference cycles by clearing the containers involved. This is
 * tricky business as the lists can be changing and we don't know which
 * objects may be freed. It is possible I screwed something up here.
 */
static void
delete_garbage(PyGC_Head *collectable, PyGC_Head *old)
```

; GC 0 Python \_\_del\_\_ Python \_\_del\_\_ \_\_del\_\_ \_\_del\_\_

```
class A(object):
    def __init__(self, b=None):
        self.b = b

    def __del__(self):
        print("We're deleting an instance of A containing:", self.b)

class B(object):
    def __init__(self, a=None):
        self.a = a

    def __del__(self):
        print("We're deleting an instance of B containing:", self.a)
```

ABA; A \_\_del\_\_ A B \_\_del\_\_ A \_\_del\_\_ \_\_del\_\_ GCed GC \_\_del\_\_; Python GCing \_\_del\_\_

CPython GC \_\_del\_\_

```
/* list of uncollectable objects */
static PyObject *garbage = NULL;
```

## Examples

Python

◦ ◦

◦ ◦

```
>>> import gc; gc.disable() # disable garbage collector
>>> class Track:
    def __init__(self):
        print("Initialized")
    def __del__(self):
        print("Destructed")
>>> def foo():
    Track()
    # destructed immediately since no longer has any references
    print("---")
    t = Track()
    # variable is referenced, so it's not destructed yet
    print("---")
    # variable is destructed when function exits
>>> foo()
Initialized
Destructed
---
Initialized
---
Destructed
```

```
>>> def bar():
    return Track()
>>> t = bar()
Initialized
>>> another_t = t # assign another reference
>>> print("...")
...
>>> t = None # not destructed yet - another_t still refers to it
>>> another_t = None # final reference gone, object is destructed
Destructed
```

◦ **ABBAAB.AB1**◦

```
>>> import gc; gc.disable() # disable garbage collector
>>> class Track:
    def __init__(self):
        print("Initialized")
    def __del__(self):
        print("Destructed")
>>> A = Track()
Initialized
>>> B = Track()
Initialized
>>> A.other = B
>>> B.other = A
>>> del A; del B # objects are not destructed due to reference cycle
>>> gc.collect() # trigger collection
Destructed
Destructed
4
```

## ◦ ABC.....AZAZ

```
>>> objs = [Track() for _ in range(10)]
Initialized
Initialized
Initialized
Initialized
Initialized
Initialized
Initialized
Initialized
Initialized
Initialized
Initialized
>>> for i in range(len(objs)-1):
...     objs[i].other = objs[i + 1]
...
>>> objs[-1].other = objs[0] # complete the cycle
>>> del objs                 # no one can refer to objs now - still not destructed
>>> gc.collect()
Destructed
Destructed
Destructed
Destructed
Destructed
Destructed
Destructed
Destructed
Destructed
Destructed
Destructed
20
```

## del

del v del v[item] del [i:j] del v[item] del v.name ◦ ◦

```
>>> import gc
>>> gc.disable() # disable garbage collector
>>> class Track:
...     def __init__(self):
...         print("Initialized")
...     def __del__(self):
...         print("Destructed")
>>> def bar():
...     return Track()
>>> t = bar()
Initialized
>>> another_t = t # assign another reference
>>> print("...")
...
>>> del t         # not destructed yet - another_t still refers to it
>>> del another_t # final reference gone, object is destructed
Destructed
```

## ◦ ;-5256Python

```
>>> import sys
>>> sys.getrefcount(1)
```

```
797
>>> a = 1
>>> b = 1
>>> sys.getrefcount(1)
799
```

## ab1 Python

```
>>> a = 999999999
>>> sys.getrefcount(999999999)
3
>>> b = 999999999
>>> sys.getrefcount(999999999)
3
```

999999999ab

```
>>> import sys
>>> a = object()
>>> sys.getrefcount(a)
2
>>> b = a
>>> sys.getrefcount(a)
3
>>> del b
>>> sys.getrefcount(a)
2
```

## Python 230

ctypes

Python

## Python 3.x 3.0

```
import ctypes
deallocated = 12345
ctypes.pythonapi._Py_Dealloc(ctypes.py_object(deallocated))
```

## Python 2.x 2.3

```
import ctypes, sys
deallocated = 12345
(ctypes.c_char * sys.getsizeof(deallocated)).from_address(id(deallocated))[:4] = '\x00' * 4
```

Python - ...

None - Fatal Python error: deallocating NoneFatal Python error: deallocating None

o o

o Python - **generation0** o - o



```
import gc
gc.set_threshold(1000, 100, 10) # Values are just for demonstration purpose
```

**generation0** 1000

generation1 100  
generation1 100

**generation0\_threshold** 1000

```
import gc
gc.collect()
```

generation0 1000

generation1 100

generation0\_threshold 1000

generation0 1000

generation1 100

generation0\_threshold 1000

```
>>> f = open("test.txt")
>>> del f
```

f.close() with

```
>>> with open("test.txt") as f:
...     pass
...     # do something with f
>>> #now the f object still exists, but it is closed
```

with while

<https://riptutorial.com/zh-TW/python/topic/2532/>

# 101:

## Examples

### inputraw\_input

#### Python 2.x 2.3

raw\_input◦

```
foo = raw_input("Put a message here that asks the user for input")
```

foo◦

#### Python 3.x 3.0

input◦

```
foo = input("Put a message here that asks the user for input")
```

foo◦

#### Python 3.x 3.0

#### Python 3

```
print("This string will be displayed in the output")  
# This string will be displayed in the output  
  
print("You can print \n escape characters too.")  
# You can print escape characters too.
```

#### Python 2.x 2.3

#### Python 2print◦

```
print "This string will be displayed in the output"  
# This string will be displayed in the output  
  
print "You can print \n escape characters too."  
# You can print escape characters too.
```

#### Python 2from \_\_future\_\_ import print\_functionPython 3print()◦ Python 2.6◦

```
def input_number(msg, err_msg=None):  
    while True:  
        try:  
            return float(raw_input(msg))
```

```

    except ValueError:
        if err_msg is not None:
            print(err_msg)

def input_number(msg, err_msg=None):
    while True:
        try:
            return float(input(msg))
        except ValueError:
            if err_msg is not None:
                print(err_msg)

```

```
user_number = input_number("input a number: ", "that's not a number!")
```

“”

```
user_number = input_number("input a number: ")
```

## Python 2.x 2.3

### Python 2.x printprint° °

```

print "Hello,",
print "World!"
# Hello, World!

```

## Python 3.x 3.0

### Python 3.x printend°

```

print("Hello, ", end="\n")
print("World!")
# Hello,
# World!

```

```

print("Hello, ", end="")
print("World!")
# Hello, World!

```

```

print("Hello, ", end="<br>")
print("World!")
# Hello, <br>World!

```

```

print("Hello, ", end="BREAK")
print("World!")
# Hello, BREAKWorld!

```

sys.stdout.write

```

import sys

sys.stdout.write("Hello, ")

```

```
sys.stdout.write("World!")
# Hello, World!
```

## stdin

Python`unix`◦ `stdin`

```
import sys

for line in sys.stdin:
    print(line)
```

`sys.stdin`◦ `for`◦

## python

```
$ cat myfile | python myprogram.py
```

`cat myfile``stdout``unix`◦

## fileinput

```
import fileinput
for line in fileinput.input():
    process(line)
```

◦ `open` ◦ `with <command> as <name>`“”`open`

```
with open('somefile.txt', 'r') as fileobj:
    # write code here using fileobj
```

◦

◦ ◦ `r` ◦ `rb` ◦ `a` ◦ `w` ◦ `r+` ◦ `open()` ◦ `r` ◦

```
# let's create an example file:
with open('shoppinglist.txt', 'w') as fileobj:
    fileobj.write('tomato\npasta\ngarlic')

with open('shoppinglist.txt', 'r') as fileobj:
    # this method makes a list where each line
    # of the file is an element in the list
    lines = fileobj.readlines()

print(lines)
# ['tomato\n', 'pasta\n', 'garlic']

with open('shoppinglist.txt', 'r') as fileobj:
    # here we read the whole content into one string:
    content = fileobj.read()
    # get a list of lines, just like int the previous example:
    lines = content.split('\n')
```

```
print(lines)
# ['tomato', 'pasta', 'garlic']
```

◦ ◦

```
with open('shoppinglist.txt', 'r') as fileobj:
    # this method reads line by line:
    lines = []
    for line in fileobj:
        lines.append(line.strip())
```

◦ for line in fileobj.strip()◦

fileobj.tell()◦ 0◦

```
fileobj = open('shoppinglist.txt', 'r')
pos = fileobj.tell()
print('We are at %u.' % pos) # We are at 0.
```

```
content = fileobj.read()
end = fileobj.tell()
print('This file was %u characters long.' % end)
# This file was 22 characters long.
fileobj.close()
```

```
fileobj = open('shoppinglist.txt', 'r')
fileobj.seek(7)
pos = fileobj.tell()
print('We are at character #%u.' % pos)
```

◦ read()◦ read()◦ rbr

```
# reads the next 4 characters
# starting at the current position
next4 = fileobj.read(4)
# what we got?
print(next4) # 'cucu'
# where we are now?
pos = fileobj.tell()
print('We are at %u.' % pos) # We are at 11, as we was at 7, and read 4 chars.

fileobj.close()
```

```
with open('shoppinglist.txt', 'r') as fileobj:
    print(type(fileobj.read())) # <class 'str'>

with open('shoppinglist.txt', 'rb') as fileobj:
    print(type(fileobj.read())) # <class 'bytes'>
```

<https://riptutorial.com/zh-TW/python/topic/266/>

# 102:

LIFO ◦ ◦ - ◦ ◦

- stack = []
- stack.append(object)
- stack.pop - > object
- list [-1] - > object

*push pop* ◦

LIFO ◦

Python append() push pop() pop ◦ O1 ◦

Python deque ◦ deque S ◦

## Examples

### ListStack

list ◦ python listStack ◦

```
#define a stack class
class Stack:
    def __init__(self):
        self.items = []

    #method to check the stack is empty or not
    def isEmpty(self):
        return self.items == []

    #method for pushing an item
    def push(self, item):
        self.items.append(item)

    #method for popping an item
    def pop(self):
        return self.items.pop()

    #check what item is on top of the stack without removing it
    def peek(self):
        return self.items[-1]

    #method to get the size
    def size(self):
        return len(self.items)

    #to view the entire stack
    def fullStack(self):
        return self.items
```

```

stack = Stack()
print('Current stack:', stack.fullStack())
print('Stack empty?:', stack.isEmpty())
print('Pushing integer 1')
stack.push(1)
print('Pushing string "Told you, I am generic stack!"')
stack.push('Told you, I am generic stack!')
print('Pushing integer 3')
stack.push(3)
print('Current stack:', stack.fullStack())
print('Popped item:', stack.pop())
print('Current stack:', stack.fullStack())
print('Stack empty?:', stack.isEmpty())

```

```

Current stack: []
Stack empty?: True
Pushing integer 1
Pushing string "Told you, I am generic stack!"
Pushing integer 3
Current stack: [1, 'Told you, I am generic stack!', 3]
Popped item: 3
Current stack: [1, 'Told you, I am generic stack!']
Stack empty?: False

```

◦ ◦

([])<sup>◦</sup> ( )<>)<sup>◦</sup> ([])<sup>◦</sup>

```

def checkParenth(str):
    stack = Stack()
    pushChars, popChars = "<{([, ">)}]"
    for c in str:
        if c in pushChars:
            stack.push(c)
        elif c in popChars:
            if stack.isEmpty():
                return False
            else:
                stackTop = stack.pop()
                # Checks to see whether the opening bracket matches the closing one
                balancingBracket = pushChars[popChars.index(c)]
                if stackTop != balancingBracket:
                    return False
        else:
            return False
    return not stack.isEmpty()

```

<https://riptutorial.com/zh-TW/python/topic/3807/>

# 103:

## Examples

worker◦

- `countUp()` 1◦
- `countDown()` 1◦

```
import multiprocessing
import time
from random import randint

def countUp():
    i = 0
    while i <= 3:
        print('Up:\t{}'.format(i))
        time.sleep(randint(1, 3)) # sleep 1, 2 or 3 seconds
        i += 1

def countDown():
    i = 3
    while i >= 0:
        print('Down:\t{}'.format(i))
        time.sleep(randint(1, 3)) # sleep 1, 2 or 3 seconds
        i -= 1

if __name__ == '__main__':
    # Initiate the workers.
    workerUp = multiprocessing.Process(target=countUp)
    workerDown = multiprocessing.Process(target=countDown)

    # Start the workers.
    workerUp.start()
    workerDown.start()

    # Join the workers. This will block in the main (parent) process
    # until the workers are complete.
    workerUp.join()
    workerDown.join()
```

```
Up:    0
Down:  3
Up:    1
Up:    2
Down:  2
Up:    3
Down:  1
Down:  0
```

```
from multiprocessing import Pool

def cube(x):
    return x ** 3
```



```
if __name__ == "__main__":  
    pool = Pool(5)  
    result = pool.map(cube, [0, 1, 2, 3])
```

PoolWorkers ◦

Pool(5) **5Pool** pool.map **map** ◦

map\_async apply apply\_async ◦

<https://riptutorial.com/zh-TW/python/topic/3601/>

# 104:

## Examples

◦ ◦ ◦ XYYX◦

```
class Shape:
    """
    This is a parent class that is intended to be inherited by other classes
    """

    def calculate_area(self):
        """
        This method is intended to be overridden in subclasses.
        If a subclass doesn't implement it but it is called, NotImplemented will be raised.

        """
        raise NotImplemented

class Square(Shape):
    """
    This is a subclass of the Shape class, and represents a square
    """
    side_length = 2    # in this example, the sides are 2 units long

    def calculate_area(self):
        """
        This method overrides Shape.calculate_area(). When an object of type
        Square has its calculate_area() method called, this is the method that
        will be called, rather than the parent class' version.

        It performs the calculation necessary for this shape, a square, and
        returns the result.
        """
        return self.side_length * 2

class Triangle(Shape):
    """
    This is also a subclass of the Shape class, and it represents a triangle
    """
    base_length = 4
    height = 3

    def calculate_area(self):
        """
        This method also overrides Shape.calculate_area() and performs the area
        calculation for a triangle, returning the result.
        """

        return 0.5 * self.base_length * self.height

def get_area(input_obj):
    """
    This function accepts an input object, and will call that object's
    calculate_area() method. Note that the object type is not specified. It
    could be a Square, Triangle, or Shape object.
    """
```

```

    print(input_obj.calculate_area())

# Create one object of each class
shape_obj = Shape()
square_obj = Square()
triangle_obj = Triangle()

# Now pass each object, one at a time, to the get_area() function and see the
# result.
get_area(shape_obj)
get_area(square_obj)
get_area(triangle_obj)

```

4

6

◦ get\_area()◦

```

class Square:

    side_length = 2

    def calculate_square_area(self):
        return self.side_length ** 2

class Triangle:

    base_length = 4
    height = 3

    def calculate_triangle_area(self):
        return (0.5 * self.base_length) * self.height

def get_area(input_obj):

    # Notice the type checks that are now necessary here. These type checks
    # could get very complicated for a more complex example, resulting in
    # duplicate and difficult to maintain code.

    if type(input_obj).__name__ == "Square":
        area = input_obj.calculate_square_area()

    elif type(input_obj).__name__ == "Triangle":
        area = input_obj.calculate_triangle_area()

    print(area)

# Create one object of each class
square_obj = Square()
triangle_obj = Triangle()

# Now pass each object, one at a time, to the get_area() function and see the
# result.
get_area(square_obj)
get_area(triangle_obj)

```

4

6

“Python 3。 Python 2.x3.xPython 2.xtypeinput\_obj。 “object。

Python。 Python。

```
class Duck:
    def quack(self):
        print("Quaaaaaack!")
    def feathers(self):
        print("The duck has white and gray feathers.")

class Person:
    def quack(self):
        print("The person imitates a duck.")
    def feathers(self):
        print("The person takes a feather from the ground and shows it.")
    def name(self):
        print("John Smith")

def in_the_forest(obj):
    obj.quack()
    obj.feathers()

donald = Duck()
john = Person()
in_the_forest(donald)
in_the_forest(john)
```

Quaaaaaack

- 
- 
- 

<https://riptutorial.com/zh-TW/python/topic/5100/>

# 105:

## Examples

◦

```
lst=[[1,2,3],[4,5,6],[7,8,9]]
```

lst◦ [1,2,3] [4,5,6] [7,8,9] ◦

```
print (lst[0])
#output: [1, 2, 3]

print (lst[1])
#output: [4, 5, 6]

print (lst[2])
#output: [7, 8, 9]
```

```
print (lst[0][0])
#output: 1

print (lst[0][1])
#output: 2
```

[ ]◦ 00[1,2,3] ◦ [ ]◦ 0101 12

```
lst[0]=[10,11,12]
```

[[10,11,12],[4,5,6],[7,8,9]] ◦ ◦

```
lst[1][2]=15
```

[[10,11,12],[4,5,15],[7,8,9]] ◦ ◦ 12615◦

...

◦

```
[[[111,112,113],[121,122,123],[131,132,133]],[[211,212,213],[221,222,223],[231,232,233]],[[311,312,313],[321,322,323],[331,332,333]]]
```

◦

```
[[[111,112,113],[121,122,123],[131,132,133]],\
 [[211,212,213],[221,222,223],[231,232,233]],\
 [[311,312,313],[321,322,323],[331,332,333]]]
```

◦

## 2D

```
print(myarray)
print(myarray[1])
print(myarray[2][1])
print(myarray[1][0][2])
etc.
```

```
myarray[1]=new_n-1_d_list
myarray[2][1]=new_n-2_d_list
myarray[1][0][2]=new_n-3_d_list #or a single number if you're dealing with 3D arrays
etc.
```

<https://riptutorial.com/zh-TW/python/topic/8186/>

# 106:

Python ◦

## Examples

threading.threading.Thread ◦ threading.Thread

```
import threading

def foo():
    print "Hello threading!"

my_thread = threading.Thread(target=foo)
```

target ◦ Threadstart ◦

```
my_thread.start() # prints 'Hello threading!'
```

my\_thread.start RuntimeError ◦ start() daemon=True kwarg my\_thread.daemon True Thread ◦

Thread.join() ◦

◦

```
import requests
from threading import Thread
from queue import Queue

q = Queue(maxsize=20)
def put_page_to_q(page_num):
    q.put(requests.get('http://some-website.com/page_%s.html' % page_num))

def compile(q):
    # magic function that needs all pages before being able to be executed
    if not q.full():
        raise ValueError
    else:
        print("Done compiling!")

threads = []
for page_num in range(20):
    t = Thread(target=requests.get, args=(page_num,))
    t.start()
    threads.append(t)

# Next, join all threads to make sure all threads are done running before
# we continue. join() is a blocking call (unless specified otherwise using
# the kwarg blocking=False when calling join)
for t in threads:
    t.join()

# Call compile() now, since all threads have completed
```

```
compile(q)
```

join()◦

threading.Thread **Thread**◦ run◦

```
from threading import Thread
import time

class Sleepy(Thread):

    def run(self):
        time.sleep(5)
        print("Hello form Thread")

if __name__ == "__main__":
    t = Sleepy()
    t.start()      # start method automatic call Thread class run method.
    # print 'The main program continues to run in foreground.'
    t.join()
    print("The main program continues to run in the foreground.")
```

◦

Queuequeue◦

```
from queue import Queue
from threading import Thread

# create a data producer
def producer(output_queue):
    while True:
        data = data_computation()

        output_queue.put(data)

# create a consumer
def consumer(input_queue):
    while True:
        # retrieve data (blocking)
        data = input_queue.get()

        # do something with the data

        # indicate data has been consumed
        input_queue.task_done()
```

```
q = Queue()
t1 = Thread(target=consumer, args=(q,))
t2 = Thread(target=producer, args=(q,))
t1.start()
t2.start()
```

threadingqueue



```

from socket import socket, AF_INET, SOCK_STREAM
from threading import Thread
from queue import Queue

def echo_server(addr, nworkers):
    print('Echo server running at', addr)
    # Launch the client workers
    q = Queue()
    for n in range(nworkers):
        t = Thread(target=echo_client, args=(q,))
        t.daemon = True
        t.start()

    # Run the server
    sock = socket(AF_INET, SOCK_STREAM)
    sock.bind(addr)
    sock.listen(5)
    while True:
        client_sock, client_addr = sock.accept()
        q.put((client_sock, client_addr))

echo_server('0.0.0.0', 15000), 128)

```

concurrent.futures.ThreadPoolExecutor

```

from socket import AF_INET, SOCK_STREAM, socket
from concurrent.futures import ThreadPoolExecutor

def echo_server(addr):
    print('Echo server running at', addr)
    pool = ThreadPoolExecutor(128)
    sock = socket(AF_INET, SOCK_STREAM)
    sock.bind(addr)
    sock.listen(5)
    while True:
        client_sock, client_addr = sock.accept()
        pool.submit(echo_client, client_sock, client_addr)

echo_server('0.0.0.0', 15000)

```

*Python Cookbook* 3 David Beazley Brian K. Jones O'Reilly. 2013 David Beazley Brian Jones 978-1-449-34037-7.

o

o “”。

```

#!/usr/bin/env python2

import threading
import Queue
import time
import sys
import subprocess
from backports.shutil_get_terminal_size import get_terminal_size

print q = Queue.Queue()

```

```

interrupt = False
lines = []

def main():

    ptt = threading.Thread(target=printer) # Turn the printer on
    ptt.daemon = True
    ptt.start()

    # Stupid example of stuff to print
    for i in xrange(1,100):
        printq.put(' '.join([str(x) for x in range(1,i)])) # The actual way to send
stuff to the printer
        time.sleep(.5)

def split_line(line, cols):
    if len(line) > cols:
        new_line = ''
        ww = line.split()
        i = 0
        while len(new_line) <= (cols - len(ww[i]) - 1):
            new_line += ww[i] + ' '
            i += 1
            print len(new_line)
        if new_line == '':
            return (line, '')

        return (new_line, ' '.join(ww[i:]))
    else:
        return (line, '')

def printer():

    while True:
        cols, rows = get_terminal_size() # Get the terminal dimensions
        msg = '#' + '-' * (cols - 2) + '#\n' # Create the
        try:
            new_line = str(printq.get_nowait())
            if new_line != '!@#EXIT#@!': # A nice way to turn the printer
                # thread out gracefully

                lines.append(new_line)
                printq.task_done()
            else:
                printq.task_done()
                sys.exit()
        except Queue.Empty:
            pass

        # Build the new message to show and split too long lines
        for line in lines:
            res = line # The following is to split lines which are
                # longer than cols.

            while len(res) !=0:
                toprint, res = split_line(res, cols)
                msg += '\n' + toprint

        # Clear the shell and print the new output
        subprocess.check_call('clear') # Keep the shell clean
        sys.stdout.write(msg)
        sys.stdout.flush()

```

```
time.sleep(.5)
```

## while

```
import threading
import time

class StoppableThread(threading.Thread):
    """Thread class with a stop() method. The thread itself has to check
    regularly for the stopped() condition."""

    def __init__(self):
        super(StoppableThread, self).__init__()
        self._stop_event = threading.Event()

    def stop(self):
        self._stop_event.set()

    def join(self, *args, **kwargs):
        self.stop()
        super(StoppableThread, self).join(*args, **kwargs)

    def run():
        while not self._stop_event.is_set():
            print("Still running!")
            time.sleep(2)
        print("stopped!")
```

o

<https://riptutorial.com/zh-TW/python/topic/544/>

# 107:

- Python◦

|                    |      |
|--------------------|------|
| socket.AF_UNIX     | UNIX |
| socket.AF_INET     | IPv4 |
| socket.AF_INET6    | IPv6 |
| socket.SOCK_STREAM | TCP  |
| socket.SOCK_DGRAM  | UDP  |

## Examples

### UDP

UDP◦ ◦ ◦ UDP◦

UDPlocalhost6667

“UDP◦

```
from socket import socket, AF_INET, SOCK_DGRAM
s = socket(AF_INET, SOCK_DGRAM)
msg = ("Hello you there!").encode('utf-8') # socket.sendto() takes bytes as input, hence we
must encode the string first.
s.sendto(msg, ('localhost', 6667))
```

### UDP

UDP◦ ◦ socket.recvfrom msg [] addr []

socketUDP

```
from socket import socket, AF_INET, SOCK_DGRAM
sock = socket(AF_INET, SOCK_DGRAM)
sock.bind(('localhost', 6667))

while True:
    msg, addr = sock.recvfrom(8192) # This is the amount of bytes to read at maximum
    print("Got message from %s: %s" % (addr, msg))
```

socketserver.UDPServer

```
from socketserver import BaseRequestHandler, UDPServer
```

```

class MyHandler(BaseRequestHandler):
    def handle(self):
        print("Got connection from: %s" % self.client_address)
        msg, sock = self.request
        print("It said: %s" % msg)
        sock.sendto("Got your message!".encode(), self.client_address) # Send reply

serv = UDPServer(('localhost', 6667), MyHandler)
serv.serve_forever()

```

sockets◦ ◦

## TCP

◦ ◦

b'Hello'localhost6667TCP

```

from socket import socket, AF_INET, SOCK_STREAM
s = socket(AF_INET, SOCK_STREAM)
s.connect(('localhost', 6667)) # The address of the TCP server listening
s.send(b'Hello')
s.close()

```

“”◦ ◦ send◦

◦

## TCP

TCP5000127.0.0.1◦ ◦

-c◦ JSON◦ -n◦ ◦

### client\_list.py

```

import argparse
import json
import socket
import threading

def handle_client(client_list, conn, address):
    name = conn.recv(1024)
    entry = dict(zip(['name', 'address', 'port'], [name, address[0], address[1]]))
    client_list[name] = entry
    conn.sendall(json.dumps(client_list))
    conn.shutdown(socket.SHUT_RDWR)
    conn.close()

def server(client_list):
    print "Starting server..."
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

```

```

s.bind(('127.0.0.1', 5000))
s.listen(5)
while True:
    (conn, address) = s.accept()
    t = threading.Thread(target=handle_client, args=(client_list, conn, address))
    t.daemon = True
    t.start()

def client(name):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('127.0.0.1', 5000))
    s.send(name)
    data = s.recv(1024)
    result = json.loads(data)
    print json.dumps(result, indent=4)

def parse_arguments():
    parser = argparse.ArgumentParser()
    parser.add_argument('-c', dest='client', action='store_true')
    parser.add_argument('-n', dest='name', type=str, default='name')
    result = parser.parse_args()
    return result

def main():
    client_list = dict()
    args = parse_arguments()
    if args.client:
        client(args.name)
    else:
        try:
            server(client_list)
        except KeyboardInterrupt:
            print "Keyboard interrupt"

if __name__ == '__main__':
    main()

```

```

$ python client_list.py
Starting server...

```

```

$ python client_list.py -c -n name1
{
  "name1": {
    "address": "127.0.0.1",
    "port": 62210,
    "name": "name1"
  }
}

```

## 1024。JSON。

```

ValueError: Unterminated string starting at: line 1 column 1023 (char 1022)

```

## Linux

```

sudo ethtool -K eth1 tx off

```

## SOCK\_RAW

```
#!/usr/bin/env python
from socket import socket, AF_PACKET, SOCK_RAW
s = socket(AF_PACKET, SOCK_RAW)
s.bind(("eth1", 0))

# We're putting together an ethernet frame here,
# but you could have anything you want instead
# Have a look at the 'struct' module for more
# flexible packing/unpacking of binary data
# and 'binascii' for 32 bit CRC
src_addr = "\x01\x02\x03\x04\x05\x06"
dst_addr = "\x01\x02\x03\x04\x05\x06"
payload = ("["*30)+"PAYLOAD"+("]"*30)
checksum = "\x1a\x2b\x3c\x4d"
ethertype = "\x08\x01"

s.send(dst_addr+src_addr+ethertype+payload+checksum)
```

<https://riptutorial.com/zh-TW/python/topic/1530/>

## 108:

- `subprocess.callargs*stdin = None``stdout = None``stderr = None``shell = False``timeout = None`
- `subprocess.Popenargsbufsize = -1``executable = None``stdin = None``stdout = None``stderr = None``preexec_fn = None``close_fds = True``shell = False``cwd = None``env = None``universal_newlines = False``startupinfo = None``creationflags = 0``restore_signals = True``start_new_session = False``pass_fds =`

|       |                         |
|-------|-------------------------|
| args  | - 'ls' ['ls', '-la']    |
| shell | shell POSIX/bin/shshell |
| cwd   | .                       |

## Examples

`subprocess.call` ◦ ◦ ◦ ◦

```
subprocess.call([r'C:\path\to\app.exe', 'arg1', '--flag', 'arg'])
```

`shell` `shell=True` ◦

```
subprocess.call('echo "Hello, world"', shell=True)
```

`exit status` ◦ `shell=True` ◦

`subprocess.call` `subprocess.check_output` ◦ ◦

## Popen

`subprocess.Popen` `subprocess.call` ◦

```
process = subprocess.Popen([r'C:\path\to\app.exe', 'arg1', '--flag', 'arg'])
```

`Popen.call`; `Popen.call` ◦

```
process = subprocess.Popen([r'C:\path\to\app.exe', 'arg1', '--flag', 'arg'])  
process.wait()
```



```

process = subprocess.Popen([r'C:\path\to\app.exe'], stdout=subprocess.PIPE,
stderr=subprocess.PIPE)

# This will block until process completes
stdout, stderr = process.communicate()
print stdout
print stderr

```

stdinstdout ◦ ◦

```

process = subprocess.Popen([r'C:\path\to\app.exe'], stdout = subprocess.PIPE, stdin =
subprocess.PIPE)

process.stdin.write('line of input\n') # Write input

line = process.stdout.readline() # Read a line from stdout

# Do logic on line read.

```

communicate() stdinstdout ◦

```

process = subprocess.Popen(<your_command>, stdout=subprocess.PIPE)
while process.poll() is None:
    output_line = process.stdout.readline()

```

EOL ◦

```

process = subprocess.Popen(<your_command>, stdout=subprocess.PIPE)
while process.poll() is None:
    output_line = process.stdout.read(1)

```

readlread1 ◦ ◦ 0read EOF ◦

process.poll()None ◦ ◦

stderr ◦

shell\_mode=True ◦

◦ shlex ◦

```

import shlex
cmd_to_subprocess = shlex.split(command_used_in_the_shell)

```

```

import shlex
shlex.split('ls --color -l -t -r')

out: ['ls', '--color', '-l', '-t', '-r']

```

<https://riptutorial.com/zh-TW/python/topic/1393/>

# 109:

- `mydict = {}`
- `mydict [k] =`
- `value = mydict [k]`
- `value = mydict.getk`
- `value = mydict.getk“default_value”`



- 
- `hash;TypeError`
- ◦

## Examples

```
dictionary = {"Hello": 1234, "World": 5678}
print(dictionary["Hello"])
```

1234 ◦

"Hello" ◦ dict ◦

dict1234 ◦ "Hello" dict ◦

KeyError ◦ KeyErrordictionary.get ◦ None ◦ None ◦

```
w = dictionary.get("whatever")
x = dictionary.get("whatever", "nuh-uh")
```

wNone x"nuh-uh" ◦

## dict

dict() ◦

```
dict(a=1, b=2, c=3) # {'a': 1, 'b': 2, 'c': 3}
dict([('d', 4), ('e', 5), ('f', 6)]) # {'d': 4, 'e': 5, 'f': 6}
dict(['a', 1], b=2, c=3) # {'a': 1, 'b': 2, 'c': 3}
dict({'a' : 1, 'b' : 2}, c=3) # {'a': 1, 'b': 2, 'c': 3}
```

## KeyError

◦ `KeyError`

```
mydict = {}
mydict['not there']
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'not there'
```

dict.get◦

```
value = mydict.get(key, default_value)
```

mydict[key] default\_value ◦ mydictkey ◦ mydict.setdefault(key, default\_value) ◦

```
mydict = {}
print(mydict)
# {}
print(mydict.get("foo", "bar"))
# bar
print(mydict)
# {}
print(mydict.setdefault("foo", "bar"))
# bar
print(mydict)
# {'foo': 'bar'}
```

```
try:
    value = mydict[key]
except KeyError:
    value = default_value
```

in◦

```
if key in mydict:
    value = mydict[key]
else:
    value = default_value
```

◦

dictcollections.defaultdictdefault\_factorynew\_keydict◦

for◦

```
mydict = {
    'a': '1',
    'b': '2'
}
```

keys()

```
print(mydict.keys())
# Python2: ['a', 'b']
```

```
# Python3: dict_keys(['b', 'a'])
```

values()

```
print(mydict.values())
# Python2: ['1', '2']
# Python3: dict_values(['2', '1'])
```

items()

```
print(mydict.items())
# Python2: [('a', '1'), ('b', '2')]
# Python3: dict_items([('b', '2'), ('a', '1')])
```

dict keys() values() items() ◦ sort() sorted() OrderedDict ◦

**Python 2/3** Python 3 Python 2 iterkeys() itervalues() iteritems() ◦ ◦ [PEP 3106](#) ◦

Python ◦ ◦ ◦ ◦

---

```
d = {} # empty dict
d = {'key': 'value'} # dict with initial values
```

## Python 3.x 3.5

```
# Also unpacking one or multiple dictionaries with the literal syntax is possible

# makes a shallow copy of otherdict
d = {**otherdict}
# also updates the shallow copy with the contents of the yetanotherdict.
d = {**otherdict, **yetanotherdict}
```

## dict

```
d = {k:v for k,v in [('key', 'value',)]}
```

**dict()**

```
d = dict() # empty dict
d = dict(key='value') # explicit keyword arguments
d = dict([('key', 'value')]) # passing in a list of key/value pairs
# make a shallow copy of another dict (only possible if keys are only strings!)
d = dict(**otherdict)
```

---

```
d['newkey'] = 42
```

listdictionary

```
d['new_list'] = [1, 2, 3]
d['new_dict'] = {'nested_dict': 1}
```

```
del d['newkey']
```

## defaultdict

```
from collections import defaultdict

d = defaultdict(int)
d['key']           # 0
d['key'] = 5
d['key']           # 5

d = defaultdict(lambda: 'empty')
d['key']           # 'empty'
d['key'] = 'full'
d['key']           # 'full'
```

[\*]dict using dict.setdefault()

```
>>> d = {}
{}
>>> d.setdefault('Another_key', []).append("This worked!")
>>> d
{'Another_key': ['This worked!']}
```

dict.setdefault() [] - °

[\*] *Python Cookbook* 3 David Beazley Brian K. Jones O'Reilly. 2013 David Beazley Brian Jones 978-1-449-34037-7.

°

collections.OrderedDict ° °

```
from collections import OrderedDict

d = OrderedDict()
d['first'] = 1
d['second'] = 2
d['third'] = 3
d['last'] = 4

# Outputs "first 1", "second 2", "third 3", "last 4"
for key in d:
    print(key, d[key])
```

\*\*

\*\*°

```

>>>
>>> def parrot(voltage, state, action):
...     print("This parrot wouldn't", action, end=' ')
...     print("if you put", voltage, "volts through it.", end=' ')
...     print("E's", state, "!")
...
>>> d = {"voltage": "four million", "state": "bleedin' demised", "action": "VOOM"}
>>> parrot(**d)

```

This parrot wouldn't VOOM if you put four million volts through it. E's bleedin' demised !

## Python 3.5 dict

```

>>> fish = {'name': "Nemo", 'hands': "fins", 'special': "gills"}
>>> dog = {'name': "Clifford", 'hands': "paws", 'color': "red"}
>>> fishdog = **fish, **dog
>>> fishdog

{'hands': 'paws', 'color': 'red', 'name': 'Clifford', 'special': 'gills'}

```

lattermost "Clifford" "Nemo".

```

>>> fish = {'name': "Nemo", 'hands': "fins", 'special': "gills"}
>>> dog = {'name': "Clifford", 'hands': "paws", 'color': "red"}

```

## Python 3.5+

```

>>> fishdog = **fish, **dog
>>> fishdog

{'hands': 'paws', 'color': 'red', 'name': 'Clifford', 'special': 'gills'}

```

lattermost "Clifford" "Nemo".

## Python 3.3+

```

>>> from collections import ChainMap
>>> dict(ChainMap(fish, dog))

{'hands': 'fins', 'color': 'red', 'special': 'gills', 'name': 'Nemo'}

```

"Clifford" "Nemo".

## Python 2.x/3.x

```

>>> from itertools import chain
>>> dict(chain(fish.items(), dog.items()))

```

```
{'hands': 'paws', 'color': 'red', 'name': 'Clifford', 'special': 'gills'}
```

lattermost\*\*“Clifford”“Nemo”。

```
>>> fish.update(dog)
>>> fish
{'color': 'red', 'hands': 'paws', 'name': 'Clifford', 'special': 'gills'}
```

dict.updatedict。

。

```
role = {"By day": "A typical programmer",
        "By night": "Still a typical programmer", }
```

PEP 8。

```
options = {
    "x": ["a", "b"],
    "y": [10, 20, 30]
}
```

。 "x"="a" "y"=10 "x"="a" "y"=10 。

。

```
import itertools

options = {
    "x": ["a", "b"],
    "y": [10, 20, 30]}

keys = options.keys()
values = (options[key] for key in keys)
combinations = [dict(zip(keys, combination)) for combination in itertools.product(*values)]
print combinations
```

combinations

```
[{'x': 'a', 'y': 10},
 {'x': 'b', 'y': 10},
 {'x': 'a', 'y': 20},
 {'x': 'b', 'y': 20},
 {'x': 'a', 'y': 30},
 {'x': 'b', 'y': 30}]
```

for 。

```
d = {'a': 1, 'b': 2, 'c':3}
for key in d:
    print(key, d[key])
# c 3
```



```
# b 2
# a 1
```

```
print([key for key in d])
# ['c', 'b', 'a']
```

## Python 3.x 3.0

items()

```
for key, value in d.items():
    print(key, value)
# c 3
# b 2
# a 1
```

values()

```
for key, value in d.values():
    print(key, value)
# 3
# 2
# 1
```

## Python 2.x 2.2

keys() values() items() iterkeys() itervalues() iteritems()◦

- 
- hash, TypeError
- ◦

```
# Creating and populating it with values
stock = {'eggs': 5, 'milk': 2}

# Or creating an empty dictionary
dictionary = {}

# And populating it after
dictionary['eggs'] = 5
dictionary['milk'] = 2

# Values can also be lists
mydict = {'a': [1, 2, 3], 'b': ['one', 'two', 'three']}

# Use list.append() method to add new elements to the values list
mydict['a'].append(4) # => {'a': [1, 2, 3, 4], 'b': ['one', 'two', 'three']}
mydict['b'].append('four') # => {'a': [1, 2, 3, 4], 'b': ['one', 'two', 'three', 'four']}

# We can also create a dictionary using a list of two-items tuples
iterable = [('eggs', 5), ('milk', 2)]
dictionary = dict(iterables)

# Or using keyword argument:
dictionary = dict(eggs=5, milk=2)
```

```
# Another way will be to use the dict.fromkeys:
dictionary = dict.fromkeys((milk, eggs)) # => {'milk': None, 'eggs': None}
dictionary = dict.fromkeys((milk, eggs), (2, 5)) # => {'milk': 2, 'eggs': 5}
```

o

```
car = {}
car["wheels"] = 4
car["color"] = "Red"
car["model"] = "Corvette"
```

o

```
print "Little " + car["color"] + " " + car["model"] + "!"
# This would print out "Little Red Corvette!"
```

## JSON

```
car = {"wheels": 4, "color": "Red", "model": "Corvette"}
```

```
for key in car:
    print key + ": " + car[key]

# wheels: 4
# color: Red
# model: Corvette
```

<https://riptutorial.com/zh-TW/python/topic/396/>

## 110:

- `str.capitalize` - > str
- `str.casefold` - > str [Python> 3.3]
- `str.centerwidth` [fillchar] - > str
- `str.countsub` [start [end]] - > int
- `str.decodeencoding` =“utf-8”[errors] - > unicode [Python 2.x]
- `str.encodeencoding` =“utf-8”errors =“strict” - > bytes
- `str.endswithsuffix` [start [end]] - > bool
- `str.expandtabstabsz` = 8 - > str
- `str.findsub` [start [end]] - > int
- `str.format* args** kwargs` - > str
- `str.format_mapmapping` - > str
- `str.indexsub` [start [end]] - > int
- `str.isalnum` - > bool
- `str.isalpha` - > bool
- `str.isdecimal` - > bool
- `str.isdigit` - > bool
- `str.isidentifier` - > bool
- `str.islower` - > bool
- `str.isnumeric` - > bool
- `str.isprintable` - > bool
- `str.isspace` - > bool
- `str.istitle` - > bool
- `str.isupper` - > bool
- `str.joiniterable` - > str
- `str.ljustwidth` [fillchar] - > str
- `str.lower` - > str
- `str.lstrip`[chars] - > str
- static `str.maketransx` [y [z]]
- `str.partitionsep` - >headseptail
- `str.replaceoldnew` [count] - > str
- `str.rfindsub` [start [end]] - > int
- `str.rindexsub` [start [end]] - > int
- `str.rjustwidth` [fillchar] - > str
- `str.rpartitionsep` - >headseptail
- `str.rsplitsep` = Nonemaxsplit = -1 - >
- `str.rstrip`[chars] - > str
- `str.splitsep` = Nonemaxsplit = -1 - >
- `str.splitlines`[keepends] - >
- `str.startswith`[start [end]] - > bool
- `str.strip`[chars] - > str
- `str.swapcase` - > str
- `str.title` - > str
- `str.translatetable` - > str

- `str.upper` - > `str`
- `str.zfillwidth` - > `str`

◦ `str str`◦

## Examples

Python◦

- `str.casefold`
- `str.upper`
- `str.lower`
- `str.capitalize`
- `str.title`
- `str.swapcase`

unicodePython 3 11◦ ◦

### Python 3.x 3.3

`str.casefold()`

`str.casefold`◦ `str.lower`◦

```
"XβΣ".casefold()
# 'xssσ'

"XβΣ".lower()
# 'xβς'
```

CaseFolding.txtUnicode Consortium◦

`str.upper()`

`str.upper`

```
"This is a 'string'.".upper()
# "THIS IS A 'STRING'."
```

`str.lower()`

`str.lower`;

```
"This IS a 'string'.".lower()
# "this is a 'string'."
```

`str.capitalize()`

`str.capitalize`

```
"this Is A 'String'".capitalize() # Capitalizes the first character and lowercases all others
# "This is a 'string'."
```

---

`str.title()`

`str.title`

```
"this Is a 'String'".title()
# "This Is A 'String'"
```

---

`str.swapcase()`

`str.swapcase`

```
"this iS A STRiNg".swapcase() #Swaps case of each character
# "THIS Is a strIng"
```

---

`str`

`strstr.upper`

```
str.upper("This is a 'string'")
# "THIS IS A 'STRING'"
```

`map`

```
map(str.upper, ["These", "are", "some", "'strings'"])
# ['THESE', 'ARE', 'SOME', "'STRINGS'"]
```

`str.split(sep=None, maxsplit=-1)`

`str.split` ◦ `sep`

`sep None` ◦

```
>>> "This is a sentence.".split()
['This', 'is', 'a', 'sentence.']

>>> " This is    a sentence. ".split()
['This', 'is', 'a', 'sentence.']

>>> "          ".split()
[]
```

`sep` ◦ ◦ ◦

```
>>> "This is a sentence.".split(' ')
```

```

['This', 'is', 'a', 'sentence.']

>>> "Earth,Stars,Sun,Moon".split(',')
['Earth', 'Stars', 'Sun', 'Moon']

>>> " This is    a sentence. ".split(' ')
['', 'This', 'is', '', '', '', 'a', 'sentence.', '', '']

>>> "This is a sentence.".split('e')
['This is a s', 'nt', 'nc', '.']

>>> "This is a sentence.".split('en')
['This is a s', 't', 'ce.']

```

maxsplit<sup>o</sup> -1

```

>>> "This is a sentence.".split('e', maxsplit=0)
['This is a sentence.']

>>> "This is a sentence.".split('e', maxsplit=1)
['This is a s', 'ntence.']

>>> "This is a sentence.".split('e', maxsplit=2)
['This is a s', 'nt', 'nce.']

>>> "This is a sentence.".split('e', maxsplit=-1)
['This is a s', 'nt', 'nc', '.']

```

**str.rsplit(sep=None, maxsplit=-1)**

maxsplit str.rsplit “right split” str.split “left split”<sup>o</sup>

```

>>> "This is a sentence.".rsplit('e', maxsplit=1)
['This is a sentenc', '.']

>>> "This is a sentence.".rsplit('e', maxsplit=2)
['This is a sent', 'nc', '.']

```

Python<sup>o</sup> <sup>o</sup>

Python<sup>o</sup> str<sup>o</sup> [re.sub](#)<sup>o</sup>

**str.replace(old, new[, count])**

str.replaceoldnewoldnew<sup>o</sup> count

'foo''spam' old = 'foo'new = 'spam' str.replace

```

>>> "Make sure to foo your sentence.".replace('foo', 'spam')
"Make sure to spam your sentence."

```

oldnew

```
>>> "It can foo multiple examples of foo if you want.".replace('foo', 'spam')
'It can spam multiple examples of spam if you want.'
```

count ◦ count

```
>>> """It can foo multiple examples of foo if you want, \
... or you can limit the foo with the third argument.""".replace('foo', 'spam', 1)
'It can spam multiple examples of foo if you want, or you can limit the foo with the third
argument.'
```

## str.format-strings

Python2.6 `str.format` 3.6 `f-strings` ◦

```
i = 10
f = 1.5
s = "foo"
l = ['a', 1, 2]
d = {'a': 1, 2: 'foo'}
```

```
"10 1.5 foo ['a', 1, 2] {'a': 1, 2: 'foo'}"
```

```
>>> "{} {} {} {} {}".format(i, f, s, l, d)
>>> str.format("{} {} {} {} {}", i, f, s, l, d)
>>> "{0} {1} {2} {3} {4}".format(i, f, s, l, d)
>>> "{0:d} {1:0.1f} {2} {3!r} {4!r}".format(i, f, s, l, d)
>>> "{i:d} {f:0.1f} {s} {l!r} {d!r}".format(i=i, f=f, s=s, l=l, d=d)
```

```
>>> f"{i} {f} {s} {l} {d}"
>>> f"{i:d} {f:0.1f} {s} {l!r} {d!r}"
```

PythonC ◦ `str.format`

```
"%d %0.1f %s %r %r" % (i, f, s, l, d)
"%(i)d %(f)0.1f %(s)s %(l)r %(d)r" % dict(i=i, f=f, s=s, l=l, d=d)
```

`str.format` ◦

```
"I am from Australia. I love cupcakes from Australia!"
```

```
>>> "I am from {}. I love cupcakes from {}".format("Australia", "Australia")
>>> "I am from {0}. I love cupcakes from {0}!".format("Australia")
```

python [pyformat.info](https://pyformat.info) ◦

```
{}
```

```
"{'a': 5, 'b': 6}"
```

```
>>> "{'{}': {}, '{}': {}}".format("a", 5, "b", 6)
```

```
>>> f"{'{a}': {5}, '{b}': {6}}"
```

◦ `str.format()` [PEP 3101](#) [PEP 498](#) f◦

`str.count` `str.count` ◦

---

**`str.count(sub[, start[, end]])`**

`str.count` `int` `sub` ◦ `start` `end` ◦ `start = 0` `end = len(str)`

```
>>> s = "She sells seashells by the seashore."
>>> s.count("sh")
2
>>> s.count("se")
3
>>> s.count("sea")
2
>>> s.count("seashells")
1
```

`start` `end` `start` 13

```
>>> s.count("sea", start)
1
```

```
>>> t = s[start:]
>>> t.count("sea")
1
```

**Python** `str.startswith()` `str.endswith()` ◦

---

**`str.startswith(prefix[, start[, end]])`**

`str.startswith` `prefix` ◦

```
>>> s = "This is a test string"
>>> s.startswith("T")
True
>>> s.startswith("Thi")
True
>>> s.startswith("thi")
False
```

`start` `end` ◦ 22



```
>>> s.startswith("is", 2)
True
```

s[2] == 'i's[3] == 's' True ◦

tuple

```
>>> s.startswith(('This', 'That'))
True
>>> s.startswith(('ab', 'bc'))
False
```

---

**str.endswith(prefix[, start[, end]])**

str.endswithstr.startswith ◦

```
>>> s = "this ends in a full stop."
>>> s.endswith('.')
True
>>> s.endswith('!!')
False
```

startswith

```
>>> s.endswith('stop.')
True
>>> s.endswith('Stop.')
False
```

tuple

```
>>> s.endswith(('.', 'something'))
True
>>> s.endswith(('ab', 'bc'))
False
```

**Python**str ◦ str.isalpha str.isdigit str.isalnum str.isspace ◦ str.isupper str.islower  
str.istitlestr.istitle ◦

---

**str.isalpha**

str.isalphaTrue

```
>>> "Hello World".isalpha() # contains a space
False
>>> "Hello2World".isalpha() # contains a number
False
>>> "HelloWorld!".isalpha() # contains punctuation
False
>>> "HelloWorld".isalpha()
True
```

`"".isalpha()`False ◦

---

`str.isupper str.islower str.istitle`

◦

`str.isupper`TrueFalse ◦

```
>>> "HeLLO WORLD".isupper()
False
>>> "HELLO WORLD".isupper()
True
>>> "".isupper()
False
```

`str.islower`TrueFalse ◦

```
>>> "Hello world".islower()
False
>>> "hello world".islower()
True
>>> "".islower()
False
```

`str.istitle`True ;◦

```
>>> "hello world".istitle()
False
>>> "Hello world".istitle()
False
>>> "Hello World".istitle()
True
>>> "".istitle()
False
```

---

`str.isdecimal str.isdigit str.isnumeric`

`str.isdecimal`◦

`str.isdigit`◦

`str.isnumeric`0-9◦

|                                | <code>isdecimal</code> | <code>isdigit</code> | <code>isnumeric</code> |
|--------------------------------|------------------------|----------------------|------------------------|
| 12345                          | True                   | True                 | True                   |
| □2□□5                          | True                   | True                 | True                   |
| ① <sup>23</sup> □ <sub>5</sub> | False                  | True                 | True                   |
| ⓪□                             | False                  | False                | True                   |
| Five                           | False                  | False                | False                  |

## Bytestrings Python 3 bytes Python 2 str isdigit ASCII

`str.isalpha` False

---

### `str.isalnum`

`str.isalpha` `str.isnumeric` True

```
>>> "Hello2World".isalnum()
True
>>> "HelloWorld".isalnum()
True
>>> "2016".isalnum()
True
>>> "Hello World".isalnum() # contains whitespace
False
```

---

### `str.isspace`

True

```
>>> "\t\r\n".isspace()
True
>>> " ".isspace()
True
```

“

```
>>> "".isspace()
False
```

```
>>> my_str = ''
>>> my_str.isspace()
False
>>> my_str.isspace() or not my_str
True
```

### `strip`

```
>>> not my_str.strip()
True
```

## `str.translate`

### Python `str.translate`

```
str.translate(table[, deletechars])
```

|             |   |
|-------------|---|
| table       | ◦ |
| deletechars | ◦ |

maketrans str.maketrans Python 3 string.maketrans Python 2 ◦

```
>>> translation_table = str.maketrans("aeiou", "12345")
>>> my_string = "This is a string!"
>>> translated = my_string.translate(translation_table)
'Th3s 3s 1 str3ng!'
```

translate ◦

table None ◦

```
>>> 'this syntax is very useful'.translate(None, 'aeiou')
'ths syntx s vry sfl'
```

/

str.strip str.rstrip str.lstrip ◦

**str.strip([chars])**

str.stripchars chars ; chars " None ◦

```
>>> "   a line with leading and trailing space   ".strip()
'a line with leading and trailing space'
```

chars ◦

```
>>> ">>> a Python prompt".strip('> ') # strips '>' character and space character
'a Python prompt'
```

**str.rstrip([chars]) str.lstrip([chars])**

str.strip() ◦ str.rstrip() str.lstrip() ◦

str.rstrip

```
>>> "   spacious string   ".rstrip()
'   spacious string'
```

str.lstrip

```
>>> "    spacious string    ".rstrip()
'spacious string    '
```

- unicodePython 3. Python 2Python 3 - unicode.

unicode ◦ `text.lower() != text.upper().lower()` "ß"

```
>>> "ß".lower()
'ß'

>>> "ß".upper().lower()
'ss'
```

"Buße""BUSSE""Buße" ◦ "BUSSE""BU E" - ◦ `casefold`

## Python 3.x 3.3

```
>>> help(str.casefold)
"""
Help on method_descriptor:

casefold(...)
    S.casefold() -> str

    Return a version of S suitable for caseless comparisons.
"""
```

`lower` ◦ `casefold.upper().lower()` ◦

- "ê" == "ê " -

```
>>> "ê" == "ê "
False
```

```
>>> import unicodedata

>>> [unicodedata.name(char) for char in "ê"]
['LATIN SMALL LETTER E WITH CIRCUMFLEX']

>>> [unicodedata.name(char) for char in "ê "]
['LATIN SMALL LETTER E', 'COMBINING CIRCUMFLEX ACCENT']
```

`unicodedata.normalize` ◦ **NFKD** ◦

```
>>> unicodedata.normalize("NFKD", "ê") == unicodedata.normalize("NFKD", "ê ")
True
```

```
import unicodedata

def normalize_caseless(text):
    return unicodedata.normalize("NFKD", text.casefold())
```

```
def caseless_equal(left, right):
    return normalize_caseless(left) == normalize_caseless(right)
```

join()° °

```
>>> " ".join(["once", "upon", "a", "time"])
"once upon a time"
```

°

```
>>> "---".join(["once", "upon", "a", "time"])
"once---upon---a---time"
```

**Python**string° string

```
>>> import string
```

**string.ascii\_letters**

ascii\_lowercaseascii\_uppercase

```
>>> string.ascii_letters
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

---

**string.ascii\_lowercase**

**ASCII**

```
>>> string.ascii_lowercase
'abcdefghijklmnopqrstuvwxyz'
```

---

**string.ascii\_uppercase**

**ASCII**

```
>>> string.ascii_uppercase
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

---

**string.digits**

```
>>> string.digits
'0123456789'
```

---

**string.hexdigits**

```
>>> string.hexdigits
'0123456789abcdefABCDEF'
```

---

### string.octaldigits

```
>>> string.octaldigits
'01234567'
```

---

### string.punctuation

C

```
>>> string.punctuation
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

---

### string.whitespace

## ASCII

```
>>> string.whitespace
'\t\n\r\x0b\x0c'
```

```
print(string.whitespace) str°
```

---

### string.printable

```
; string.digits string.ascii_letters string.punctuation string.whitespace °
```

```
>>> string.printable
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~ \t\n\r\x0b\x0c'
```

**reverse** reversed() reversed() °

```
>>> reversed('hello')
<reversed object at 0x0000000000000000>
>>> [char for char in reversed('hello')]
['o', 'l', 'l', 'e', 'h']
```

reversed() `''.join()` °

```
>>> ''.join(reversed('hello'))
'olleh'
```

reversed() Python-1°

```
>>> def reversed_string(main_string):
...     return main_string[::-1]
...
>>> reversed_string('hello')
'olleh'
```

## Python

`str.ljust``str.rjust`

```
interstates_lengths = {
    5: (1381, 2222),
    19: (63, 102),
    40: (2555, 4112),
    93: (189, 305),
}
for road, length in interstates_lengths.items():
    miles, kms = length
    print('{} -> {} mi. ({} km.)'.format(str(road).rjust(4), str(miles).ljust(4),
str(kms).ljust(4)))
```

```
40 -> 2555 mi. (4112 km.)
19 -> 63 mi. (102 km.)
5 -> 1381 mi. (2222 km.)
93 -> 189 mi. (305 km.)
```

`ljust``rjust`° `width``fillchar`° `width`° `width` **alread**° `fillchar` ' '°

`ljust``fillchar``width`° `rjust``rjust`° `lrfillchar`°

## strunicode

° **unicode**°

Python 2`strUnicode`° ' ' "ASCIIASCII° Unicode`u`' ' `u`"°

## Python 2.x 2.3

```
# You get "© abc" encoded in UTF-8 from a file, network, or other data source

s = '\xc2\xa9 abc' # s is a byte array, not a string of characters
                    # Doesn't know the original was UTF-8
                    # Default form of string literals in Python 2
s[0]               # '\xc2' - meaningless byte (without context such as an encoding)
type(s)           # str - even though it's not a useful one w/o having a known encoding

u = s.decode('utf-8') # u'\xa9 abc'
                    # Now we have a Unicode string, which can be read as UTF-8 and printed
                    # properly

                    # In Python 2, Unicode string literals need a leading u
                    # str.decode converts a string which may contain escaped bytes to a
Unicode string
u[0]              # u'\xa9' - Unicode Character 'COPYRIGHT SIGN' (U+00A9) '©'
type(u)          # unicode
```



```
u.encode('utf-8') # '\xc2\xa9 abc'
                  # unicode.encode produces a string with escaped bytes for non-ASCII
characters
```

Python 3 `Unicode`. `Unicode` `bytestring` `b''` `b'''`. `True` `isinstance(some_val, bytes)` `some_val`.

## Python 3.x 3.0

```
# You get from file or network "@ abc" encoded in UTF-8

s = b'\xc2\xa9 abc' # s is a byte array, not characters
                    # In Python 3, the default string literal is Unicode; byte array literals
                    # need a leading b
s[0]                # b'\xc2' - meaningless byte (without context such as an encoding)
type(s)             # bytes - now that byte arrays are explicit, Python can show that.

u = s.decode('utf-8') # '@ abc' on a Unicode terminal
                    # bytes.decode converts a byte array to a string (which will, in Python
3, be Unicode)
u[0]                # '\u00a9' - Unicode Character 'COPYRIGHT SIGN' (U+00A9) '@'
type(u)             # str
                    # The default string literal in Python 3 is UTF-8 Unicode

u.encode('utf-8')   # b'\xc2\xa9 abc'
                    # str.encode produces a byte array, showing ASCII-range bytes as unescaped
characters.
```

## Python `in`

```
>>> "foo" in "foo.baz.bar"
True
```

True

```
>>> "" in "test"
True
```

<https://riptutorial.com/zh-TW/python/topic/278/>

# 111:

## ◦ Python◦

- “{}”◦ format42==>“42”
- “{0}”◦ format42==>“42”
- “{0.2f}”◦ format42==>“42.00”
- “{0.0f}”◦ 42.1234==>“42”
- “{answer}”◦ formatno\_answer = 41answer = 42==>“42”
- “{answer.2f}”◦ no\_answer = 41answer = 42==>“42.00”
- “{[key]}”◦ format{key"value"}==>“value”
- “{[1]}”◦ format[zero"one"two]==>“one”
- “{answer} = {answer}”◦ formatanswer = 42==>“42 = 42”
- ”◦ join[stack"overflow]==>“”

- [PyFormat.info/](#)

## Examples

```
foo = 1
bar = 'bar'
baz = 3.14
```

str.format◦

```
print('{} , {} and {}'.format(foo, bar, baz))
# Out: "1, bar and 3.14"
```

◦ str.format0◦

```
print('{0}, {1}, {2}, and {1}'.format(foo, bar, baz))
# Out: "1, bar, 3.14, and bar"
print('{0}, {1}, {2}, and {3}'.format(foo, bar, baz))
# Out: index out of range error
```

```
print("X value is: {x_val}. Y value is: {y_val}.".format(x_val=2, y_val=3))
# Out: "X value is: 2. Y value is: 3."
```

str.format

```
class AssignValue(object):
    def __init__(self, value):
        self.value = value
my_value = AssignValue(6)
print('My value is: {0.value}'.format(my_value)) # "0" is optional
# Out: "My value is: 6"
```

```
my_dict = {'key': 6, 'other_key': 7}
print("My other key is: {0[other_key]}".format(my_dict)) # "0" is optional
# Out: "My other key is: 7"
```

## listtuple

```
my_list = ['zero', 'one', 'two']
print("2nd element is: {0[2]}".format(my_list)) # "0" is optional
# Out: "2nd element is: two"
```

str.format Python% [PEP 3101](#) - ◦ str.format%◦

◦ :◦ ◦ :~^20 ^20

```
'{:~^20}'.format('centered')
# Out: '~~~~~centered~~~~~'
```

format%

```
t = (12, 45, 22222, 103, 6)
print '{0} {2} {1} {2} {3} {2} {4} {2}'.format(*t)
# Out: 12 22222 45 22222 103 22222 6 22222
```

format

```
number_list = [12,45,78]
print map('the number is {}'.format, number_list)
# Out: ['the number is 12', 'the number is 45', 'the number is 78']
```

```
from datetime import datetime,timedelta
```

```
once_upon_a_time = datetime(2010, 7, 1, 12, 0, 0)
delta = timedelta(days=13, hours=8, minutes=20)
```

```
gen = (once_upon_a_time + x * delta for x in xrange(5))
```

```
print '\n'.join(map('{:%Y-%m-%d %H:%M:%S}'.format, gen))
#Out: 2010-07-01 12:00:00
#     2010-07-14 20:20:00
#     2010-07-28 04:40:00
#     2010-08-10 13:00:00
#     2010-08-23 21:20:00
```

## Python 2.x 2.6

format()◦ : [fill\_char] [align\_operator] [width] align\_operator

- <width◦
- >width◦
- ^width◦
- =◦

fill\_char ◦

```
{:~<9s}, World'.format('Hello')
# 'Hello~~~~, World'

{:~>9s}, World'.format('Hello')
# '~~~~Hello, World'

{:~^9s}'.format('Hello')
# '~~Hello~~'

{:0=6d}'.format(-123)
# '-00123'
```

ljust() rjust() center() zfill() ljust() **2.5** ◦

## f-string

**PEP 498 Python3.6** f.format ◦

```
>>> foo = 'bar'
>>> f'Foo is {foo}'
'Foo is bar'
```

◦

```
>>> f'{foo:^7s}'
' bar '
```

f'b'bytes u'unicode **python2** ◦ ◦

```
>>> price = 478.23
>>> f'{f'${price:0.2f}':*>20s}'
'*****$478.23'
```

f-string ◦

```
>>> def fn(l, incr):
...     result = l[0]
...     l[0] += incr
...     return result
...
>>> lst = [0]
>>> f'{fn(lst,2)} {fn(lst,3)}'
'0 2'
>>> f'{fn(lst,2)} {fn(lst,3)}'
'5 7'
>>> lst
[10]
```

## datetime

`__format__` ◦ Python `datetimestr.formatstr.format strftime`

```
>>> from datetime import datetime
>>> 'North America: {dt:%m/%d/%Y}. ISO: {dt:%Y-%m-%d}'.format(dt=datetime.now())
'North America: 07/21/2016. ISO: 2016-07-21.'
```

◦

## GetitemGetattr

`__getitem__`

```
person = {'first': 'Arthur', 'last': 'Dent'}
'{p[first]} {p[last]}'.format(p=person)
# 'Arthur Dent'
```

`getattr()`

```
class Person(object):
    first = 'Zaphod'
    last = 'Beeblebrox'

'{p.first} {p.last}'.format(p=Person())
# 'Zaphod Beeblebrox'
```

```
>>> '{0:.0f}'.format(42.12345)
'42'
```

```
>>> '{0:.1f}'.format(42.12345)
'42.1'
```

```
>>> '{0:.3f}'.format(42.12345)
'42.123'
```

```
>>> '{0:.5f}'.format(42.12345)
'42.12345'
```

```
>>> '{0:.7f}'.format(42.12345)
'42.1234500'
```

```
>>> '{:.3f}'.format(42.12345)
'42.123'
```

```
>>> '{answer:.3f}'.format(answer=42.12345)
'42.123'
```

```
>>> '{0:.3e}'.format(42.12345)
'4.212e+01'
```

```
>>> '{0:.0%}'.format(42.12345)
'4212%'
```

`{0}{name}` ◦ **1**

```
>>> s = 'Hello'
>>> a, b, c = 1.12345, 2.34567, 34.5678
>>> digits = 2

>>> '{0}! {1:.{n}f}, {2:.{n}f}, {3:.{n}f}'.format(s, a, b, c, n=digits)
'Hello! 1.12, 2.35, 34.57'
```

.format()

```
>>> '{:c}'.format(65) # Unicode character
'A'

>>> '{:d}'.format(0x0a) # base 10
'10'

>>> '{:n}'.format(0x0a) # base 10 using current locale for separators
'10'
```

```
>>> '{0:x}'.format(10) # base 16, lowercase - Hexadecimal
'a'

>>> '{0:X}'.format(10) # base 16, uppercase - Hexadecimal
'A'

>>> '{:o}'.format(10) # base 8 - Octal
'12'

>>> '{:b}'.format(10) # base 2 - Binary
'1010'

>>> '{0:#b}, {0:#o}, {0:#x}'.format(42) # With prefix
'0b101010, 0o52, 0x2a'

>>> '8 bit: {0:08b}; Three bytes: {0:06x}'.format(42) # Add zero padding
'8 bit: 00101010; Three bytes: 00002a'
```

## RGB

```
>>> r, g, b = (1.0, 0.4, 0.0)
>>> '#{0:02X}{0:02X}{0:02X}'.format(int(255 * r), int(255 * g), int(255 * b))
'FF6600'
```

```
>>> '{:x}'.format(42.0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: Unknown format code 'x' for object of type 'float'
```

str.formatformat◦ ◦

formatPython \_\_format\_\_format\_\_◦ \_\_format\_\_format\_\_◦

\_\_str\_\_format\_\_◦ 1

```
object.__format__(self, format_spec)
```

```

# Example in Python 2 - but can be easily applied to Python 3

class Example(object):
    def __init__(self,a,b,c):
        self.a, self.b, self.c = a,b,c

    def __format__(self, format_spec):
        """ Implement special semantics for the 's' format specifier """
        # Reject anything that isn't an s
        if format_spec[-1] != 's':
            raise ValueError('{} format specifier not understood for this object',
format_spec[:-1])

        # Output in this example will be (<a>,<b>,<c>)
        raw = "(" + ",".join([str(self.a), str(self.b), str(self.c)]) + ")"
        # Honor the format language by using the inbuilt string format
        # Since we know the original format_spec ends in an 's'
        # we can take advantage of the str.format method with a
        # string argument we constructed above
        return "{r:{f}}".format( r=raw, f=format_spec )

inst = Example(1,2,3)
print "{0:>20s}".format( inst )
# out :                (1,2,3)
# Note how the right align and field width of 20 has been honored.

```

`__format__format` Python2 `__str__repr__repr` s° Python3 `format__format__`°

```

>>> '{:.>10}'.format('foo')
'.....foo'

```

`format{}`

```

>>> '{:.>{}}'.format('foo', 10)
'.....foo'
'{:({}){}}'.format('foo', '*', '^', 15)
'*****foo*****'

```

`'{:({}){}}' * 15` "15" 'foo' °

```

>>> data = ["a", "bbbbbbb", "ccc"]
>>> m = max(map(len, data))
>>> for d in data:
...     print('{:>{}}'.format(d, m))
a
bbbbbbb
ccc

```

3°

`{}`°

```

s = ""
pad

```

```

{:3}          :{a:3}:

truncate
{:.3}        :{e:.3}:

combined
{:>3.3}      :{a:>3.3}:
{:3.3}       :{a:3.3}:
{:3.3}       :{c:3.3}:
{:3.3}       :{e:3.3}:
"""

print (s.format(a="1"*1, c="3"*3, e="5"*5))

```

```

pad
{:3}          :1  :

truncate
{:.3}        :555:

combined
{:>3.3}      : 1:
{:3.3}       :1  :
{:3.3}       :333:
{:3.3}       :555:

```

format◦

## Python 2.x

```

>>> data = {'first': 'Hodor', 'last': 'Hodor!'}
>>> '{first} {last}'.format(**data)
'Hodor Hodor!'

```

## Python 3.2+

```

>>> '{first} {last}'.format_map(data)
'Hodor Hodor!'

```

[str.format\\_map](#)◦ [str.format\\_map](#)◦ [datadict](#)◦

```

>>> '{first} {last}'.format(first='Hodor', last='Hodor!')
'Hodor Hodor!'

```

<https://riptutorial.com/zh-TW/python/topic/1019/>



# 112:

Python ◦ Python/ ◦

- hashlib.new
- hashlib.pbkdf2\_hmac ◦ passwordsaltroundsdklen = None

hashlib ◦ hashlib.new().update() ◦ hashlib.pbkdf2\_hmac ◦ b

```
"This is a string"
b"This is a buffer of bytes"
```

## Examples

hashlib.new ◦

```
import hashlib

h = hashlib.new('sha256')
h.update(b'Nobody expects the Spanish Inquisition.')
h.digest()
# ==>
b'.\xdf\xda\xdaVR[\x12\x90\xff\x16\xfb\x17D\xcf\xb4\x82\xdd)\x14\xff\xbc\xb6Iy\x0c\x0eX\x9eF-
='
```

digest ◦ update ◦ hexdigest

```
h.hexdigest()
# ==> '2edfdada56525b1290ff16fb1744cfb482dd2914ffbc649790c0e589e462d3d'
```

hashlib.new ◦ Python ◦ hashlib.algorithms\_available

```
import hashlib
hashlib.algorithms_available
# ==> {'sha256', 'DSA-SHA', 'SHA512', 'SHA224', 'dsaWithSHA', 'SHA', 'RIPEMD160', 'ecdsa-with-
SHA1', 'sha1', 'SHA384', 'md5', 'SHA1', 'MD5', 'MD4', 'SHA256', 'sha384', 'md4', 'ripemd160',
'sha224', 'sha512', 'DSA', 'dsaEncryption', 'sha', 'whirlpool'}
```

;

hashlib.algorithms\_guaranteed

```
hashlib.algorithms_guaranteed
# ==> {'sha256', 'sha384', 'sha1', 'sha224', 'md5', 'sha512'}
```

hashlib ◦ PBKDF2 ◦ ◦

```
import hashlib
import os

salt = os.urandom(16)
hash = hashlib.pbkdf2_hmac('sha256', b'password', salt, 100000)
```

## PBKDF2SHA256。 。 。 。

binascii

```
import binascii
hexhash = binascii.hexlify(hash)
```

## PBKDF2bcryptscript。 Python。

。 。 。

hashlib

```
import hashlib

hasher = hashlib.new('sha256')
with open('myfile', 'r') as f:
    contents = f.read()
    hasher.update(contents)

print hasher.hexdigest()
```

```
import hashlib
SIZE = 65536
hasher = hashlib.new('sha256')
with open('myfile', 'r') as f:
    buffer = f.read(SIZE)
    while len(buffer) > 0:
        hasher.update(buffer)
        buffer = f.read(SIZE)
print (hasher.hexdigest())
```

## pycrypto

### Python。 pycrypto 。 AES 。

```
import hashlib
import math
import os

from Crypto.Cipher import AES

IV_SIZE = 16 # 128 bit, fixed for the AES algorithm
KEY_SIZE = 32 # 256 bit meaning AES-256, can also be 128 or 192 bits
SALT_SIZE = 16 # This size is arbitrary

cleartext = b'Lorem ipsum'
password = b'highly secure encryption password'
```

```

salt = os.urandom(SALT_SIZE)
derived = hashlib.pbkdf2_hmac('sha256', password, salt, 100000,
                             dklen=IV_SIZE + KEY_SIZE)

iv = derived[0:IV_SIZE]
key = derived[IV_SIZE:]

encrypted = salt + AES.new(key, AES.MODE_CFB, iv).encrypt(cleartext)

```

AESIV。AES。◦ [PBKDF2128256](#)。

◦ ◦ ◦

```

salt = encrypted[0:SALT_SIZE]
derived = hashlib.pbkdf2_hmac('sha256', password, salt, 100000,
                             dklen=IV_SIZE + KEY_SIZE)

iv = derived[0:IV_SIZE]
key = derived[IV_SIZE:]
cleartext = AES.new(key, AES.MODE_CFB, iv).decrypt(encrypted[SALT_SIZE:])

```

## pycryptoRSA

RSA。RSA。- ◦ [pycrypto](#) ◦

```

import errno

from Crypto.Hash import SHA256
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5

message = b'This message is from me, I promise.'

try:
    with open('privkey.pem', 'r') as f:
        key = RSA.importKey(f.read())
except IOError as e:
    if e.errno != errno.ENOENT:
        raise
    # No private key, generate a new one. This can take a few seconds.
    key = RSA.generate(4096)
    with open('privkey.pem', 'wb') as f:
        f.write(key.exportKey('PEM'))
    with open('pubkey.pem', 'wb') as f:
        f.write(key.publickey().exportKey('PEM'))

hasher = SHA256.new(message)
signer = PKCS1_v1_5.new(key)
signature = signer.sign(hasher)

```

```

with open('pubkey.pem', 'rb') as f:
    key = RSA.importKey(f.read())
hasher = SHA256.new(message)
verifier = PKCS1_v1_5.new(key)
if verifier.verify(hasher, signature):
    print('Nice, the signature is valid!')
else:
    print('No, the message was signed with the wrong private key or modified')

```

PKCS1 v1.5。 pycryptoPKCS1 PSS PKCS1\_PSSPKCS1\_PSSPKCS1\_v1\_5。 ◦

## pycryptoRSA

◦ ◦ [pycrypto](#) ◦

```
from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA

message = b'This is a very secret message.'
```

```
with open('privkey.pem', 'rb') as f:
    key = RSA.importKey(f.read())
    cipher = PKCS1_OAEP.new(key)
    decrypted = cipher.decrypt(encrypted)
```

PKCS1 OAEP。 pycryptoPKCS1 v1.5。

<https://riptutorial.com/zh-TW/python/topic/2598/>

# 113: /

◦ IDEAMODE CTRPython/◦

RSA◦ HashingSHA-1◦ ◦ IDEA◦

Python 2.7 <https://www.python.org/downloads/>

\* **PyCrypto** <https://pypi.python.org/pypi/pycrypto>

\* **PyCryptoPlus** <https://github.com/doegox/python-cryptoplus>

**PyCrypto**◦ linuxalt + ctrl + tCMDshift ++Windows◦ python setup.py installWindowsSure Python

**PyCryptoPlus**◦

◦ ◦

- ◦ “import socket”IP◦

-----

```
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = raw_input("Server Address To Be Connected -> ")
port = int(input("Port of The Server -> "))
server.connect((host, port))
```

-----

```
try:
    #setting up socket
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((host, port))
    server.listen(5)
except BaseException: print "-----Check Server Address or Port-----"
```

**“socket.AF\_INETsocket.SOCK\_STREAM”accept. “socket.AF\_INET socket.SOCK\_DGRAM”setblockingvalue.**

- CLIENT◦ ◦ CryptoRandomCrypto.PublicKeyRSA◦

```
random_generator = Random.new().read
key = RSA.generate(1024, random_generator)
public = key.publickey().exportKey()
```

random\_generator“ ”◦ **“Crypto.PublicKeyRSA”1024. Public.**

- CLIENTSHA-1◦ SHA-1“import hashlib”◦

```
hash_object = hashlib.shal(public)
hex_digest = hash_object.hexdigest()
```

```
hash_object.hex_digest, hex_digest, publicServer, ..., rsa_server_public_key =
RSA.importKey(getpbk) getpbk
```

- “os” “key = os.urandom(16)” “AES.MODE\_CTR” “SHA-1”

```
#encrypt CTR MODE session key
en = AES.new(key_128, AES.MODE_CTR, counter = lambda: key_128) encrypto =
en.encrypt(key_128)
#hashing sha1
en_object = hashlib.shal(encrypto)
en_digest = en_object.hexdigest()
```

en\_digest

- SERVER

```
#encrypting session key and public key
E = server_public_key.encrypt(encrypto, 16)
```

.

- CLIENT eval

```
en = eval(msg)
decrypt = key.decrypt(en)
# hashing sha1
en_object = hashlib.shal(decrypt) en_digest = en_object.hexdigest()
```

SHA-1

IDEA.MODE\_CTR, IDEA.MODE\_CTR

- IDEA 16bit, MODE\_CTR, 40IDEA, python string [valuevalue], “key [16]” 016, [1:17] [16], IDEA.new(IDEA3, KEYIDEA, IDEA.MODE\_CTR, counter = counter = counter = lambda KEY, Counter.Util, lambda counter = Counter.Util, crypto)

```
ideaEncrypt = IDEA.new(key, IDEA.MODE_CTR, counter=lambda : key)
```

“ideaEncrypt” IDEA

```
eMsg = ideaEncrypt.encrypt(whole)
#converting the encrypted message to HEXADECIMAL to readable eMsg =
eMsg.encode("hex").upper()
```

wholeeMsg, HEXADECIMALupper

•

o o o

```
decoded = newmess.decode("hex")
ideaDecrypt = IDEA.new(key, IDEA.MODE_CTR, counter=lambda: key)
dMsg = ideaDecrypt.decrypt(decoded)
```

o

## Examples

```
import socket
import hashlib
import os
import time
import itertools
import threading
import sys
import Crypto.Cipher.AES as AES
from Crypto.PublicKey import RSA
from CryptoPlus.Cipher import IDEA

#server address and port number input from admin
host= raw_input("Server Address - > ")
port = int(input("Port - > "))
#boolean for checking server and port
check = False
done = False

def animate():
    for c in itertools.cycle(['....', '.....', '.....', '.....']):
        if done:
            break
        sys.stdout.write('\rCHECKING IP ADDRESS AND NOT USED PORT '+c)
        sys.stdout.flush()
        time.sleep(0.1)
    sys.stdout.write('\r -----SERVER STARTED. WAITING FOR CLIENT-----\n')
try:
    #setting up socket
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((host, port))
    server.listen(5)
    check = True
except BaseException:
    print "-----Check Server Address or Port-----"
    check = False

if check is True:
    # server Quit
    shutdown = False
# printing "Server Started Message"
thread_load = threading.Thread(target=animate)
thread_load.start()

time.sleep(4)
done = True
#binding client and address
client, address = server.accept()
print ("CLIENT IS CONNECTED. CLIENT'S ADDRESS ->", address)
```

```

print ("\n-----WAITING FOR PUBLIC KEY & PUBLIC KEY HASH-----\n")

#client's message(Public Key)
getpbk = client.recv(2048)

#conversion of string to KEY
server_public_key = RSA.importKey(getpbk)

#hashing the public key in server side for validating the hash from client
hash_object = hashlib.shal(getpbk)
hex_digest = hash_object.hexdigest()

if getpbk != "":
    print (getpbk)
    client.send("YES")
    gethash = client.recv(1024)
    print ("\n-----HASH OF PUBLIC KEY----- \n"+gethash)
if hex_digest == gethash:
    # creating session key
    key_128 = os.urandom(16)
    #encrypt CTR MODE session key
    en = AES.new(key_128,AES.MODE_CTR,counter = lambda:key_128)
    encrypto = en.encrypt(key_128)
    #hashing shal
    en_object = hashlib.shal(encrypto)
    en_digest = en_object.hexdigest()

    print ("\n-----SESSION KEY-----\n"+en_digest)

    #encrypting session key and public key
    E = server_public_key.encrypt(encrypto,16)
    print ("\n-----ENCRYPTED PUBLIC KEY AND SESSION KEY-----\n"+str(E))
    print ("\n-----HANDSHAKE COMPLETE-----")
    client.send(str(E))
    while True:
        #message from client
        newmess = client.recv(1024)
        #decoding the message from HEXADECIMAL to decrypt the ecrypted version of the message
only
        decoded = newmess.decode("hex")
        #making en_digest(session_key) as the key
        key = en_digest[:16]
        print ("\nENCRYPTED MESSAGE FROM CLIENT -> "+newmess)
        #decrypting message from the client
        ideaDecrypt = IDEA.new(key, IDEA.MODE_CTR, counter=lambda: key)
        dMsg = ideaDecrypt.decrypt(decoded)
        print ("\n**New Message** "+time.ctime(time.time()) +" > "+dMsg+"\n")
        mess = raw_input("\nMessage To Client -> ")
        if mess != "":
            ideaEncrypt = IDEA.new(key, IDEA.MODE_CTR, counter=lambda : key)
            eMsg = ideaEncrypt.encrypt(mess)
            eMsg = eMsg.encode("hex").upper()
            if eMsg != "":
                print ("ENCRYPTED MESSAGE TO CLIENT-> " + eMsg)
                client.send(eMsg)
        client.close()
    else:
        print ("\n-----PUBLIC KEY HASH DOESNOT MATCH-----\n")

```

```
import time
```



```

import socket
import threading
import hashlib
import itertools
import sys
from Crypto import Random
from Crypto.PublicKey import RSA
from CryptoPlus.Cipher import IDEA

#animating loading
done = False
def animate():
    for c in itertools.cycle(['....','.....','.....','.....']):
        if done:
            break
        sys.stdout.write('\rCONFIRMING CONNECTION TO SERVER '+c)
        sys.stdout.flush()
        time.sleep(0.1)

#public key and private key
random_generator = Random.new().read
key = RSA.generate(1024,random_generator)
public = key.publickey().exportKey()
private = key.exportKey()

#hashing the public key
hash_object = hashlib.shal(public)
hex_digest = hash_object.hexdigest()

#Setting up socket
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

#host and port input user
host = raw_input("Server Address To Be Connected -> ")
port = int(input("Port of The Server -> "))
#binding the address and port
server.connect((host, port))
# printing "Server Started Message"
thread_load = threading.Thread(target=animate)
thread_load.start()

time.sleep(4)
done = True

def send(t,name,key):
    mess = raw_input(name + " : ")
    key = key[:16]
    #merging the message and the name
    whole = name+" : "+mess
    ideaEncrypt = IDEA.new(key, IDEA.MODE_CTR, counter=lambda : key)
    eMsg = ideaEncrypt.encrypt(whole)
    #converting the encrypted message to HEXADECIMAL to readable
    eMsg = eMsg.encode("hex").upper()
    if eMsg != "":
        print ("ENCRYPTED MESSAGE TO SERVER-> "+eMsg)
        server.send(eMsg)
def recv(t,key):
    newmess = server.recv(1024)
    print ("\nENCRYPTED MESSAGE FROM SERVER-> " + newmess)
    key = key[:16]
    decoded = newmess.decode("hex")

```

```

ideaDecrypt = IDEA.new(key, IDEA.MODE_CTR, counter=lambda: key)
dMsg = ideaDecrypt.decrypt(decoded)
print ("\n**New Message From Server** " + time.ctime(time.time()) + " : " + dMsg + "\n")

while True:
    server.send(public)
    confirm = server.recv(1024)
    if confirm == "YES":
        server.send(hex_digest)

    #connected msg
    msg = server.recv(1024)
    en = eval(msg)
    decrypt = key.decrypt(en)
    # hashing sha1
    en_object = hashlib.sha1(decrypt)
    en_digest = en_object.hexdigest()

    print ("\n-----ENCRYPTED PUBLIC KEY AND SESSION KEY FROM SERVER-----")
    print (msg)
    print ("\n-----DECRYPTED SESSION KEY-----")
    print (en_digest)
    print ("\n-----HANDSHAKE COMPLETE-----\n")
    alais = raw_input("\nYour Name -> ")

    while True:
        thread_send = threading.Thread(target=send,args=("-----Sending Message-----",alais,en_digest))
        thread_rcv = threading.Thread(target=recv,args=("-----Recieving Message-----",en_digest))
        thread_send.start()
        thread_rcv.start()

        thread_send.join()
        thread_rcv.join()
        time.sleep(0.5)
    time.sleep(60)
    server.close()

```

<https://riptutorial.com/zh-TW/python/topic/8710/>

# 114:

## Examples

### CHello World

Chello.chellogreet()

```
#include <Python.h>
#include <stdio.h>

#if PY_MAJOR_VERSION >= 3
#define IS_PY3K
#endif

static PyObject *hello_greet(PyObject *self, PyObject *args)
{
    const char *input;
    if (!PyArg_ParseTuple(args, "s", &input)) {
        return NULL;
    }
    printf("%s", input);
    Py_RETURN_NONE;
}

static PyMethodDef HelloMethods[] = {
    { "greet", hello_greet, METH_VARARGS, "Greet the user" },
    { NULL, NULL, 0, NULL }
};

#ifdef IS_PY3K
static struct PyModuleDef hellomodule = {
    PyModuleDef_HEAD_INIT, "hello", NULL, -1, HelloMethods
};

PyMODINIT_FUNC PyInit_hello(void)
{
    return PyModule_Create(&hellomodule);
}
#else
PyMODINIT_FUNC inithello(void)
{
    (void) Py_InitModule("hello", HelloMethods);
}
#endif
```

gcc

gcc /path/to/your/file/hello.c -o /path/to/your/file/hello

greet()hello.py

```
import hello          # imports the compiled library
hello.greet("Hello!") # runs the greet() function with "Hello!" as an argument
```

## C Extensions

### PythonC。

PyObject\_AsFileDescriptor

```
PyObject *fobj;
int fd = PyObject_AsFileDescriptor(fobj);
if (fd < 0){
    return NULL;
}
```

pythonPyFile\_FromFd。

```
int fd; /* Existing file descriptor */
PyObject *fobj = PyFile_FromFd(fd, "filename", "r", -1, NULL, NULL, NULL, 1);
```

## Cc ++Boost

### C ++BoostC。

# C ++

hello.cppC ++

```
#include <boost/python/module.hpp>
#include <boost/python/list.hpp>
#include <boost/python/class.hpp>
#include <boost/python/def.hpp>

// Return a hello world string.
std::string get_hello_function()
{
    return "Hello world!";
}

// hello class that can return a list of count hello world strings.
class hello_class
{
public:

    // Taking the greeting message in the constructor.
    hello_class(std::string message) : _message(message) {}

    // Returns the message count times in a python list.
    boost::python::list as_list(int count)
    {
        boost::python::list res;
        for (int i = 0; i < count; ++i) {
            res.append(_message);
        }
        return res;
    }
}
```

```

private:
    std::string _message;
};

// Defining a python module naming it to "hello".
BOOST_PYTHON_MODULE(hello)
{
    // Here you declare what functions and classes that should be exposed on the module.

    // The get_hello_function exposed to python as a function.
    boost::python::def("get_hello", get_hello_function);

    // The hello_class exposed to python as a class.
    boost::python::class_<hello_class>("Hello", boost::python::init<std::string>())
        .def("as_list", &hello_class::as_list)
        ;
}

```

pythonpythonboost。 Ubuntu 12.04python 3.4gcc。 Boost。 Ubuntu

```
sudo apt-get install gcc libboost-dev libpython3.4-dev
```

.sopython

```
gcc -shared -o hello.so -fPIC -I/usr/include/python3.4 hello.cpp -lboost_python-py34 -
lboost_system -l:libpython3.4m.so
```

example.pypython

```

import hello

print(hello.get_hello())

h = hello.Hello("World hello!")
print(h.as_list(3))

```

python3 example.py

```

Hello world!
['World hello!', 'World hello!', 'World hello!']

```

<https://riptutorial.com/zh-TW/python/topic/557/>

# 115: PythonSQL Server

## Examples

```
$ pip install pymssql
```

```
import pymssql

SERVER = "servername"
USER = "username"
PASSWORD = "password"
DATABASE = "dbname"

connection = pymssql.connect(server=SERVER, user=USER,
                             password=PASSWORD, database=DATABASE)

cursor = connection.cursor() # to access field as dictionary use cursor(as_dict=True)
cursor.execute("SELECT TOP 1 * FROM TableName")
row = cursor.fetchone()

##### CREATE TABLE #####
cursor.execute("""
CREATE TABLE posts (
    post_id INT PRIMARY KEY NOT NULL,
    message TEXT,
    publish_date DATETIME
)
""")

##### INSERT DATA IN TABLE #####
cursor.execute("""
    INSERT INTO posts VALUES(1, "Hey There", "11.23.2016")
""")
# commit your work to database
connection.commit()

##### ITERATE THROUGH RESULTS #####
cursor.execute("SELECT TOP 10 * FROM posts ORDER BY publish_date DESC")
for row in cursor:
    print("Message: " + row[1] + " | " + "Date: " + row[2])
    # if you pass as_dict=True to cursor
    # print(row["message"])

connection.close()
```

SQLexecuteCRUD.

with [pymssql.org](https://pymssql.org)

PythonSQL Server <https://riptutorial.com/zh-TW/python/topic/7985/pythonsql-server>

# 116:

- `import module_name`
- `import module_name.submodule_name`
- `module_name import *`
- `from module_name import submodule_name [ class_name function_name ... etc]`
- `module_namesome_namenew_name`
- `module_name.submodule_name import class_name [ function_name ... etc]`

Python◦ `if __name__ == '__main__':◦`

## Examples

import

```
>>> import random
>>> print(random.randint(1, 10))
4
```

import modulemodule.name - ◦ randomrandint◦ randomrandom.randintrandint ◦

```
>>> import random as rn
>>> print(rn.randint(1, 10))
4
```

Pythonmain.pycustom.py ◦

```
import custom
```

```
>>> from math import sin
>>> sin(1)
0.8414709848078965
```

import

```
from urllib.request import urlopen
```

python◦ import from ◦ import ◦ hello.pyworld.pyfunction ◦ import◦

```
from hello import function
from world import function
```

```
function() #world's function will be invoked. Not hello's
```

import◦

```
import hello
```

```
import world

hello.function() # exclusively hello's function will be invoked
world.function() # exclusively world's function will be invoked
```

from

```
>>> # Multiple modules
>>> import time, sockets, random
>>> # Multiple functions
>>> from math import sin, cos, tan
>>> # Multiple constants
>>> from math import pi, e
```

```
>>> print(pi)
3.141592653589793
>>> print(cos(45))
0.5253219888177297
>>> print(time.time())
1482807222.7240417
```

```
>>> from urllib.request import urlopen as geturl, pathname2url as path2url, getproxies
>>> from math import factorial as fact, gamma, atan as arctan
>>> import random.randint, time, sys
```

```
>>> print(time.time())
1482807222.7240417
>>> print(arctan(60))
1.554131203080956
>>> filepath = "/dogs/jumping poodle (december).png"
>>> print(path2url(filepath))
/dogs/jumping%20poodle%20%28december%29.png
```

```
from random import randint # Syntax "from MODULENAME import NAME1[, NAME2[, ...]]"
print(randint(1, 10))      # Out: 5
```

from random **python** import randint◦

```
from math import pi
print(pi) # Out: 3.14159265359
```

---

```
random.randrange(1, 10) # works only if "import random" has been run before
```

```
NameError: name 'random' is not defined
```

**python** random ◦ import random

```
import random
random.randrange(1, 10)
```

```
from module_name import *
```



```
from math import *
sqrt(2)      # instead of math.sqrt(2)
ceil(2.7)    # instead of math.ceil(2.7)
```

math◦

◦ from math import sqrt, ceilfrom math import sqrt, ceil

```
def sqrt(num):
    print("I don't know what's the square root of {}".format(num))
```

```
sqrt(4)
# Output: I don't know what's the square root of 4.
```

```
from math import *
sqrt(4)
# Output: 2.0
```

◦ SyntaxError ◦

```
def f():
    from math import *
```

```
class A:
    from math import *
```

```
SyntaxError: import * only allowed at module level
```

## \_\_all\_\_

\_\_all\_\_from mymodule import \*from mymodule import \*

```
# mymodule.py
__all__ = ['imported_by_star']

imported_by_star = 42
not_imported_by_star = 21
```

```
from mymodule import *imported_by_star imported_by_star
```

```
>>> from mymodule import *
>>> imported_by_star
42
>>> not_imported_by_star
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'not_imported_by_star' is not defined
```

not\_imported\_by\_star

```
>>> from mymodule import not_imported_by_star
>>> not_imported_by_star
21
```

## Python 2.x 2.7

### importlib2.7Python

```
import importlib
random = importlib.import_module("random")
```

```
importlib.import_module()
```

```
collections_abc = importlib.import_module("collections.abc")
```

### Pythonimp°

## Python 2.x 2.7

```
imp.find_moduleimp.load_module°
```

```
import imp, sys
def import_module(name):
    fp, pathname, description = imp.find_module(name)
    try:
        return imp.load_module(name, fp, pathname, description)
    finally:
        if fp:
            fp.close()
```

```
__import__()sys.modules fromlistimportlib.import_module()°
```

### Pythonsys.path° python°

```
import sys
sys.path.append("/path/to/directory/containing/your/module")
import mymodule
```

```
mymodule°
```

## PEP8

### PEP8

1. 

```
from math import sqrt, ceil # Not recommended
from math import sqrt      # Recommended
from math import ceil
```

2.
  - 
  - 
  - /

3. ◦ `from module import * module ◦ from module import *`

4. ;◦

```
from module.submodule import function
```

`module.submodulefunction ◦`

## \_\_import\_\_

`__import__()`

```
if user_input == "os":
    os = __import__("os")

# equivalent to import os
```

```
mod = __import__(r"C:/path/to/file/anywhere/on/computer/module.py")
```

◦ ◦

`import`

```
import math
math.pi = 3
print(math.pi)    # 3
import math
print(math.pi)    # 3
```

◦ ◦ `import`

```
print(math.pi)    # 3
import sys
if 'math' in sys.modules: # Is the ``math`` module in the register?
    del sys.modules['math'] # If so, remove it.
import math
print(math.pi)    # 3.141592653589793
```

◦

## Python 2

`reload`

### Python 2.x 2.3

```
import math
math.pi = 3
print(math.pi)    # 3
reload(math)
```

```
print(math.pi)    # 3.141592653589793
```

## Python 3

```
reloadimportlib
```

### Python 3.x 3.0

```
import math
math.pi = 3
print(math.pi)    # 3
from importlib import reload
reload(math)
print(math.pi)    # 3.141592653589793
```

<https://riptutorial.com/zh-TW/python/topic/249/>

# 117:

Python 2object.

## Examples

### @property

@property ◦

foobar.py

```
class Foo(object):
    def __init__(self):
        self.__bar = None

    @property
    def bar(self):
        if self.__bar is None:
            self.__bar = some_expensive_lookup_operation()
        return self.__bar
```

```
>>> from foobar import Foo
>>> foo = Foo()
>>> print(foo.bar) # This will take some time since bar is None after initialization
42
>>> print(foo.bar) # This is much faster since bar has a value now
42
```

### @property

@property

```
class Cash(object):
    def __init__(self, value):
        self.value = value

    @property
    def formatted(self):
        return '${:.2f}'.format(self.value)

    @formatted.setter
    def formatted(self, new):
        self.value = float(new[1:])
```

```
>>> wallet = Cash(2.50)
>>> print(wallet.formatted)
$2.50
>>> print(wallet.value)
2.5
>>> wallet.formatted = '$123.45'
>>> print(wallet.formatted)
```

```
$123.45
>>> print(wallet.value)
123.45
```

## gettersetterdeleter

getter setterdeleter

```
class BaseClass(object):
    @property
    def foo(self):
        return some_calculated_value()

    @foo.setter
    def foo(self, value):
        do_something_with_value(value)

class DerivedClass(BaseClass):
    @BaseClass.foo.setter
    def foo(self, value):
        do_something_different_with_value(value)
```

setterdeleter。

@。 Python 3.x

```
class A:
    p = 1234
    def getX (self):
        return self._x

    def setX (self, value):
        self._x = value

    def getY (self):
        return self._y

    def setY (self, value):
        self._y = 1000 + value    # Weird but possible

    def getY2 (self):
        return self._y

    def setY2 (self, value):
        self._y = value

    def getT (self):
        return self._t

    def setT (self, value):
        self._t = value

    def getU (self):
        return self._u + 10000

    def setU (self, value):
```

```

        self._u = value - 5000

    x, y, y2 = property (getX, setX), property (getY, setY), property (getY2, setY2)
    t = property (getT, setT)
    u = property (getU, setU)

A.q = 5678

class B:
    def getZ (self):
        return self.z_

    def setZ (self, value):
        self.z_ = value

    z = property (getZ, setZ)

class C:
    def __init__ (self):
        self.offset = 1234

    def getW (self):
        return self.w_ + self.offset

    def setW (self, value):
        self.w_ = value - self.offset

    w = property (getW, setW)

a1 = A ()
a2 = A ()

a1.y2 = 1000
a2.y2 = 2000

a1.x = 5
a1.y = 6

a2.x = 7
a2.y = 8

a1.t = 77
a1.u = 88

print (a1.x, a1.y, a1.y2)
print (a2.x, a2.y, a2.y2)
print (a1.p, a2.p, a1.q, a2.q)

print (a1.t, a1.u)

b = B ()
c = C ()

b.z = 100100
c.z = 200200
c.w = 300300

print (a1.x, b.z, c.z, c.w)

c.w = 400400
c.z = 500500

```

```
b.z = 600600  
print (a1.x, b.z, c.z, c.w)
```

<https://riptutorial.com/zh-TW/python/topic/2050/>



# 118:

- `x.title` # Accesses the title attribute using the dot notation
- `x.title = "Hello World"` # Sets the property of the title attribute using the dot notation
- `@property` # Used as a decorator before the getter method for properties
- `@title.setter` # Used as a decorator before the setter method for properties

## Examples

◦

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

book1 = Book(title="Right Ho, Jeeves", author="P.G. Wodehouse")
```

Python ◦

```
>>> book1.title
'P.G. Wodehouse'
```

Python

```
>>> book1.series
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
AttributeError: 'Book' object has no attribute 'series'
```

◦ **gettersetter**◦

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
```

◦ **gettersetter**◦ `""`◦ ◦

*authorgettersetter*◦

```
class P:
    def __init__(self, title, author):
        self.title = title
        self.setAuthor(author)

    def get_author(self):
        return self.author

    def set_author(self, author):
```

```
if not author:
    self.author = "Unknown"
else:
    self.author = author
```

◦

public◦

```
>>> book = Book(title="Ancient Manuscript", author="Some Guy")
>>> book.author = "" #Cos Some Guy didn't write this one!
```

◦ Python◦ @property◦ setter@ attributeName.setter◦

◦

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

    @property
    def author(self):
        return self.__author

    @author.setter
    def author(self, author):
        if not author:
            self.author = "Unknown"
        else:
            self.author = author
```

Python◦ Python◦

```
>>> book = Book(title="Ancient Manuscript", author="Some Guy")
>>> book.author = "" #Cos Some Guy didn't write this one!
>>> book.author
Unknown
```

<https://riptutorial.com/zh-TW/python/topic/4392/>

# 119:

## Examples

### ◦ falsey◦

```
x = True
y = True
z = x and y # z = True

x = True
y = False
z = x and y # z = False

x = False
y = True
z = x and y # z = False

x = False
y = False
z = x and y # z = False

x = 1
y = 1
z = x and y # z = y, so z = 1, see `and` and `or` are not guaranteed to be a boolean

x = 0
y = 1
z = x and y # z = x, so z = 0 (see above)

x = 1
y = 0
z = x and y # z = y, so z = 0 (see above)

x = 0
y = 0
z = x and y # z = x, so z = 0 (see above)
```

### 1truthy 0falsey◦

### truthy◦ ◦

```
x = True
y = True
z = x or y # z = True

x = True
y = False
z = x or y # z = True

x = False
y = True
z = x or y # z = True

x = False
```

```

y = False
z = x or y # z = False

x = 1
y = 1
z = x or y # z = x, so z = 1, see `and` and `or` are not guaranteed to be a boolean

x = 1
y = 0
z = x or y # z = x, so z = 1 (see above)

x = 0
y = 1
z = x or y # z = y, so z = 1 (see above)

x = 0
y = 0
z = x or y # z = y, so z = 0 (see above)

```

## 1truthy 0falsey。

```

x = True
y = not x # y = False

x = False
y = not x # y = True

```

## Python 。

```

>>> def true_func():
...     print("true_func()")
...     return True
...
>>> def false_func():
...     print("false_func()")
...     return False
...
>>> true_func() or false_func()
true_func()
True
>>> false_func() or true_func()
false_func()
true_func()
True
>>> true_func() and false_func()
true_func()
false_func()
False
>>> false_func() and false_func()
false_func()
False

```

\*\*\*\*

or ◦ or

```
def or_(a, b):
    if a:
        return a
    else:
        return b
```

and

```
def and_(a, b):
    if not a:
        return a
    else:
        return b
```

## Python -

```
if 3.14 < x < 3.142:
    print("x is near pi")
```

$(3.14 < x) < 3.142$  Python  $3.14 < x$  and  $x < 3.142$  ◦

<https://riptutorial.com/zh-TW/python/topic/1731/>

# 120:

Python◦ Python◦

Python◦

## Examples

for

```
alist = [0, 1, 2]
for index, value in enumerate(alist):
    alist.pop(index)
print(alist)
# Out: [1]
```

list.pop()◦

◦

```
# Iteration #1
index = 0
alist = [0, 1, 2]
alist.pop(0) # removes '0'

# Iteration #2
index = 1
alist = [1, 2]
alist.pop(1) # removes '2'

# loop terminates, but alist is not empty:
alist = [1]
```

◦

```
alist = [1,2,3,4,5,6,7]
for index, item in reversed(list(enumerate(alist))):
    # delete all even items
    if item % 2 == 0:
        alist.pop(index)
print(alist)
# Out: [1, 3, 5, 7]
```

◦ alist ◦

```
alist = [0, 1, 2]
for index, value in enumerate(alist):
    # break to avoid infinite loop:
    if index == 20:
        break
```

```
alist.insert(index, 'a')
print(alist)
# Out (abbreviated): ['a', 'a', ..., 'a', 'a', 0, 1, 2]
```

break'a' ◦ ◦

---

for **list**

```
alist = [1,2,3,4]
for item in alist:
    if item % 2 == 0:
        item = 'even'
print(alist)
# Out: [1,2,3,4]
```

**item** ◦ alist[2] enumerate()

```
alist = [1,2,3,4]
for index, item in enumerate(alist):
    if item % 2 == 0:
        alist[index] = 'even'
print(alist)
# Out: [1, 'even', 3, 'even']
```

**while**

```
zlist = [0, 1, 2]
while zlist:
    print(zlist[0])
    zlist.pop(0)
print('After: zlist =', zlist)

# Out: 0
#      1
#      2
# After: zlist = []
```

zlist;

```
zlist = []
```

len()x

```
zlist = [0, 1, 2]
x = 1
while len(zlist) > x:
    print(zlist[0])
    zlist.pop(0)
print('After: zlist =', zlist)

# Out: 0
#      1
# After: zlist = [2]
```

```

zlist = [1,2,3,4,5]
i = 0
while i < len(zlist):
    if zlist[i] % 2 == 0:
        zlist.pop(i)
    else:
        i += 1
print(zlist)
# Out: [1, 3, 5]

```

i ◦ zlist[i] 1 izlist[i] ◦

---

◦ while

```

zlist = [1,2,3,4,5]

z_temp = []
for item in zlist:
    if item % 2 != 0:
        z_temp.append(item)
zlist = z_temp
print(zlist)
# Out: [1, 3, 5]

```

◦ ◦

## Python /◦

```

zlist = [1,2,3,4,5]
[item for item in zlist if item % 2 != 0]
# Out: [1, 3, 5]

```

```

def foo(li=[]):
    li.append(1)
    print(li)

```

```

foo([2])
# Out: [2, 1]
foo([3])
# Out: [3, 1]

```

```

foo()
# Out: [1] As expected...

```

```

foo()
# Out: [1, 1] Not as expected...

```

◦ li◦

```

def foo(li=None):
    if not li:
        li = []
    li.append(1)

```



```
print(li)

foo()
# Out: [1]

foo()
# Out: [1]
```

if not liFalse ◦

```
x = []
foo(li=x)
# Out: [1]

foo(li="")
# Out: [1]

foo(li=0)
# Out: [1]
```

None

```
def foo(li=None):
    if li is None:
        li = []
    li.append(1)
    print(li)

foo()
# Out: [1]
```

```
li = [[]] * 3
print(li)
# Out: [[], [], []]
```

3◦ 1

```
li[0].append(1)
print(li)
# Out: [[1], [], []]
```

1li◦

[[]] \* 3list3list◦ list list3◦ li[0] li◦

```
li = []
element = [[]]
li = element + element + element
print(li)
# Out: [[], [], []]
element.append(1)
print(li)
# Out: [[1], [], []]
```

idlist

```
li = [[]] * 3
print([id(inner_list) for inner_list in li])
# Out: [6830760, 6830760, 6830760]
```

```
li = [[] for _ in range(3)]
```

**3list3**◦ id

```
print([id(inner_list) for inner_list in li])
# Out: [6331048, 6331528, 6331488]
```

◦ append◦

```
>>> li = []
>>> li.append([])
>>> li.append([])
>>> li.append([])
>>> for k in li: print(id(k))
...
4315469256
4315564552
4315564808
```

◦

```
for i in range(len(tab)):
    print(tab[i])
```

```
for elem in tab:
    print(elem)
```

for◦

◦

```
for i, elem in enumerate(tab):
    print((i, elem))
```

**“==”TrueFalse**

```
if (var == True):
    # this will execute if var is True or 1, 1.0, 1L

if (var != True):
    # this will execute if var is neither True nor 1

if (var == False):
    # this will execute if var is False or 0 (or 0.0, 0L, 0j)

if (var == None):
```

```

# only execute if var is None

if var:
    # execute if var is a non-empty string/list/dictionary/tuple, non-0, etc

if not var:
    # execute if var is "", {}, [], (), 0, None, etc.

if var is True:
    # only execute if var is boolean True, not 1

if var is False:
    # only execute if var is boolean False, not 0

if var is None:
    # same as var == None

```

## Pythonistas<sup>™</sup>.

```

if os.path.isfile(file_path):
    file = open(file_path)
else:
    # do something

```

```

try:
    file = open(file_path)
except OSError as e:
    # do something

```

Python 2.6+

```

with open(file_path) as file:

```

◦ try/except ◦ ◦

Python ◦ [duck typing](#) ◦ `str()` ◦ `list()` ◦ `iterable` ◦

```

def foo(name):
    if isinstance(name, str):
        print(name.lower())

def bar(listing):
    if isinstance(listing, list):
        listing.extend((1, 2, 3))
        return ", ".join(listing)

```

```

def foo(name) :
    print(str(name).lower())

def bar(listing) :
    l = list(listing)
    l.extend((1, 2, 3))
    return ", ".join(l)

```

foo° bar° °

## object

° Python 2.x° ° ° “”° object°

```
class Father:
    pass

class Child(Father):
    pass
```

```
class Father(object):
    pass

class Child(Father):
    pass
```

Python 3.x°

## init

JavaPHP° ° Python° ° °

- class°
- °

```
class Car(object):
    color = "red"
    wheels = [Wheel(), Wheel(), Wheel(), Wheel()]
```

```
class Car(object):
    def __init__(self):
        self.color = "red"
        self.wheels = [Wheel(), Wheel(), Wheel(), Wheel()]
```

## Python°

```
>>> -8 is (-7 - 1)
False
>>> -3 is (-2 - 1)
True
```

```
>>> (255 + 1) is (255 + 1)
True
>>> (256 + 1) is (256 + 1)
False
```

isTrue -3 256 -8 257°

```
[-5, 256] is True; False
```

-

`==` **identity** is

---

**Python** is

```
>>> 'python' is 'py' + 'thon'
True
```

'python'**Python**'python'

.

```
>>> 'this is not a common string' is 'this is not' + ' a common string'
False
>>> 'this is not a common string' == 'this is not' + ' a common string'
True
```

`==` **identity** is

**int literals**

**Python** 7 bit\_length

```
x = 7
x.bit_length()
# Out: 3
```

7.bit\_length() SyntaxError 7.27.bit\_length()

**int literals**

```
# parenthesis
(7).bit_length()
# a space
7 .bit_length()
```

7..bit\_length() floatbit\_length()

float float.

```
7.2.as_integer_ratio()
# Out: (8106479329266893, 1125899906842624)
```

```
if a == 3 or b == 3 or c == 3:
```

```
if a or b or c == 3: # Wrong
```

```
; or == if (a) or (b) or (c == 3):
```

```
if a == 3 or b == 3 or c == 3: # Right Way
```

```
any() or
```

```
if any([a == 3, b == 3, c == 3]): # Right
```

```
if any(x == 3 for x in (a, b, c)): # Right
```

```
if 3 in (a, b, c): # Right
```

```
in
```

```
if a == 1 or 2 or 3:
```

```
if a in (1, 2, 3):
```

## sys.argv [0]

sys.argv[0]python

```
# script.py
import sys

print(sys.argv[0])
print(sys.argv)
```

```
$ python script.py
=> script.py
=> ['script.py']

$ python script.py fizz
=> script.py
=> ['script.py', 'fizz']

$ python script.py fizz buzz
=> script.py
=> ['script.py', 'fizz', 'buzz']
```

## PythonC ++ std::map

```
myDict = {'first': 1, 'second': 2, 'third': 3}
print(myDict)
# Out: {'first': 1, 'second': 2, 'third': 3}
```

```
print([k for k in myDict])
# Out: ['second', 'third', 'first']
```

## Python

/collections.OrderedDict

```
from collections import OrderedDict

oDict = OrderedDict([('first', 1), ('second', 2), ('third', 3)])

print([k for k in oDict])
# Out: ['first', 'second', 'third']
```

OrderedDict

## Python 3.6

### Python 3.x 3.6

```
def func(**kw): print(kw.keys())

func(a=1, b=2, c=3, d=4, e=5)
dict_keys(['a', 'b', 'c', 'd', 'e']) # expected order
```

“ ”

## GIL

### Python GIL

```
import math
from threading import Thread

def calc_fact(num):
    math.factorial(num)

num = 600000
t = Thread(target=calc_fact, daemon=True, args=[num])
print("About to calculate: {}".format(num))
t.start()
print("Calculating...")
t.join()
print("Calculated")
```

Calculating... **C** math.factorial **GIL**

Python **C**

```
def calc_fact(num):
    """ A slow version of factorial in native Python """
    res = 1
    while num >= 1:
```

```
    res = res * num
    num -= 1
return res
```

sleep() C.

```
def calc_fact(num):
    sleep(0.001)
    math.factorial(num)
```

## Python 2.x 2.7

```
i = 0
a = [i for i in range(3)]
print(i) # Outputs 2
```

Python 2"" 。 **Python 3**。

## Python 3.x 3.0

```
i = 0
a = [i for i in range(3)]
print(i) # Outputs 0
```

for

```
i = 0
for i in range(3):
    pass
print(i) # Outputs 2
```

Python 2Python 3。

。

xyzab

```
def xyz():
    return a, b
```

xyzxyz

```
t = xyz()
```

tABt。

TypeErrortype tuple...

```
a, b = xyz()
```



## Pythonic JSON

```
my_var = 'bla';  
api_key = 'key';  
...lots of code here...  
params = {"language": "en", my_var: api_key}
```

JavaScriptPython ◦ JavaScriptparams

```
{  
  "language": "en",  
  "my_var": "key"  
}
```

Python

```
{  
  "language": "en",  
  "bla": "key"  
}
```

my\_var ◦

<https://riptutorial.com/zh-TW/python/topic/3553/>

# 121: CPython

CPython。

NuGetMongoDB.Bson。

JSON。

- JSON

```
BsonDocument argsBson = BsonDocument.Parse("{ 'x' : '1', 'y' : '2' }");
string argsFile = string.Format("{0}\\{1}.txt", Path.GetDirectoryName(pyScriptPath),
Guid.NewGuid());
```

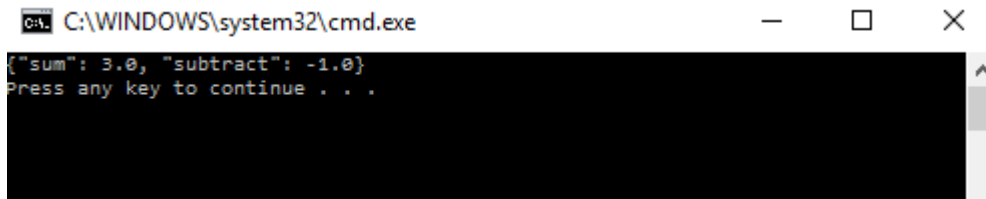
- Pythonpython.exepythonJSON

```
filename = sys.argv[ 1 ]
with open( filename ) as data_file:
    input_args = json.loads( data_file.read() )

x, y = [ float(input_args.get( key )) for key in [ 'x', 'y' ] ]
```

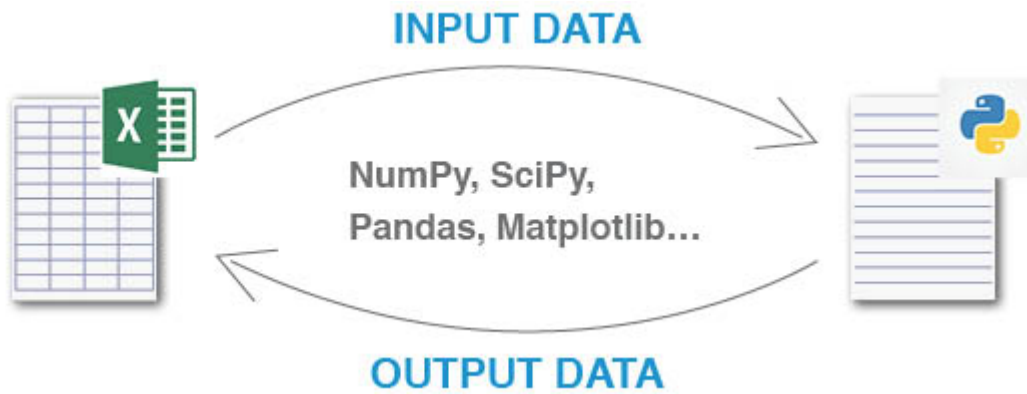
- PythonJSON

```
print json.dumps( { 'sum' : x + y , 'subtract' : x - y } )
```



- CJSON

```
using (StreamReader myStreamReader = process.StandardOutput)
{
    outputString = myStreamReader.ReadLine();
    process.WaitForExit();
}
```



CPythonExcelPython。

ExcelDNAC - Excel。

GitHub 。

4 。

- 
- 
- 
- 
- 

。

## Examples

### CPython

```
import sys
import json

# load input arguments from the text file
filename = sys.argv[ 1 ]
with open( filename ) as data_file:
    input_args = json.loads( data_file.read() )

# cast strings to floats
x, y = [ float(input_args.get( key )) for key in [ 'x', 'y' ] ]

print json.dumps( { 'sum' : x + y , 'subtract' : x - y } )
```

### CPython

```
using MongoDB.Bson;
using System;
using System.Diagnostics;
using System.IO;
```

```

namespace python_csharp
{
    class Program
    {
        static void Main(string[] args)
        {
            // full path to .py file
            string pyScriptPath = "...../sum.py";
            // convert input arguments to JSON string
            BsonDocument argsBson = BsonDocument.Parse("{ 'x' : '1', 'y' : '2' }");

            bool saveInputFile = false;

            string argsFile = string.Format("{0}\\{1}.txt",
Path.GetDirectoryName(pyScriptPath), Guid.NewGuid());

            string outputString = null;
            // create new process start info
            ProcessStartInfo prcStartInfo = new ProcessStartInfo
            {
                // full path of the Python interpreter 'python.exe'
                FileName = "python.exe", // string.Format(@"{0}", "python.exe"),
                UseShellExecute = false,
                RedirectStandardOutput = true,
                CreateNoWindow = false
            };

            try
            {
                // write input arguments to .txt file
                using (StreamWriter sw = new StreamWriter(argsFile))
                {
                    sw.WriteLine(argsBson);
                    prcStartInfo.Arguments = string.Format("{0} {1}",
string.Format(@"{0}", pyScriptPath), string.Format(@"{0}", argsFile));
                }
                // start process
                using (Process process = Process.Start(prcStartInfo))
                {
                    // read standard output JSON string
                    using (StreamReader myStreamReader = process.StandardOutput)
                    {
                        outputString = myStreamReader.ReadLine();
                        process.WaitForExit();
                    }
                }
            }
            finally
            {
                // delete/save temporary .txt file
                if (!saveInputFile)
                {
                    File.Delete(argsFile);
                }
            }
            Console.WriteLine(outputString);
        }
    }
}

```

# 122:

python switch。 PEP [PEP-3103](#)。

pythonswitch。

- <http://stackoverflow.com/questions/60208/replacements-for-switch-statement-in-python>
- <http://code.activestate.com/recipes/269708-some-python-style-switches/>
- <http://code.activestate.com/recipes/410692-readable-switch-construction-without-lambdas-or-di/>
- ...

## Examples

if / else。

switch / caseif / else

```
def switch(value):
    if value == 1:
        return "one"
    if value == 2:
        return "two"
    if value == 42:
        return "the answer to the question about life, the universe and everything"
    raise Exception("No case found!")
```

```
>>> switch(1)
one
>>> switch(2)
two
>>> switch(3)
...
Exception: No case found!
>>> switch(42)
the answer to the question about life the universe and everything
```

```
switch = {
    1: lambda: 'one',
    2: lambda: 'two',
    42: lambda: 'the answer of life the universe and everything',
}
```

```
def default_case():
    raise Exception('No case found!')
```

get。 default\_case。

```
>>> switch.get(1, default_case)()
```

```

one
>>> switch.get(2, default_case)()
two
>>> switch.get(3, default_case)()
...
Exception: No case found!
>>> switch.get(42, default_case)()
the answer of life the universe and everything

```

```

def run_switch(value):
    return switch.get(value, default_case)()

>>> run_switch(1)
one

```

/◦ getattr()“case”◦

\_\_call\_\_()◦

```

class SwitchBase:
    def switch(self, case):
        m = getattr(self, 'case_{}'.format(case), None)
        if not m:
            return self.default
        return m

    __call__ = switch

```

SwitchBasecase

```

class CustomSwitcher:
    def case_1(self):
        return 'one'

    def case_2(self):
        return 'two'

    def case_42(self):
        return 'the answer of life, the universe and everything!'

    def default(self):
        raise Exception('Not a case!')

```

```

>>> switch = CustomSwitcher()
>>> print(switch(1))
one
>>> print(switch(2))
two
>>> print(switch(3))
...
Exception: Not a case!
>>> print(switch(42))
the answer of life, the universe and everything!

```

if / elsetrue

```
class Switch:
    def __init__(self, value):
        self._val = value
    def __enter__(self):
        return self
    def __exit__(self, type, value, traceback):
        return False # Allows traceback to occur
    def __call__(self, cond, *mconds):
        return self._val in (cond,)+mconds
```

switch / case

```
def run_switch(value):
    with Switch(value) as case:
        if case(1):
            return 'one'
        if case(2):
            return 'two'
        if case(3):
            return 'the answer to the question about life, the universe and everything'
        # default
        raise Exception('Not a case!')
```

```
>>> run_switch(1)
one
>>> run_switch(2)
two
>>> run_switch(3)
...
Exception: Not a case!
>>> run_switch(42)
the answer to the question about life, the universe and everything
```

### *Nota Bene*

- [pypi](#) ◦

<https://riptutorial.com/zh-TW/python/topic/4268/>

# 123: CSV

.CSV ◦ ◦

|                                     |     |
|-------------------------------------|-----|
| “/ path /” “mode”                   | CSV |
| “”                                  |     |
| csv.writer                          | CSV |
| csv.writerfile <b>delimiter =</b> ” |     |

```
open( path, "wb")
```

"wb" - ◦

"wb"bWindows◦

```
csv.writer ( csv_file, delimiter=',' )
```

,◦ ,◦

## Examples

```
import csv

#----- We will write to CSV in this function -----

def csv_writer(data, path):

    #Open CSV file whose path we passed.
    with open(path, "wb") as csv_file:

        writer = csv.writer(csv_file, delimiter=',')
        for line in data:
            writer.writerow(line)

#---- Define our list here, and call function -----

if __name__ == "__main__":

    """
    data = our list that we want to write.
    Split it so we get a list of lists.
    """
    data = ["first_name,last_name,age".split(","),
            "John,Doe,22".split(","),
            "Jane,Doe,31".split(","),
            "Jack,Reacher,27".split(",")]
```



```
]

# Path to CSV file we want to write to.
path = "output.csv"
csv_writer(data, path)
```

## CSV

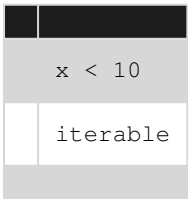
```
def append_to_csv(input_string):
    with open("fileName.csv", "a") as csv_file:
        csv_file.write(input_row + "\n")
```

CSV <https://riptutorial.com/zh-TW/python/topic/10862/csv>

# 124:

◦ ◦ Python◦

- <boolean expression>
- <iterable><variable>
- <variable><number>
- <variable><start\_number><end\_number>
- <variable><start\_number><end\_number><step\_size>
- for i<variable> in enumerate<iterable>with index i
- zip<variable1><variable2><iterable1><iterable2>



## Examples

for

```
for x in ['one', 'two', 'three', 'four']:
    print(x)
```

```
one
two
three
four
```

rangefor◦

```
for x in range(1, 6):
    print(x)
```

python> = 3python <= 2◦ for◦

```
1
2
3
4
5
```

Pythonenumerate

```
for index, item in enumerate(['one', 'two', 'three', 'four']):
    print(index, '::', item)
```

enumerate index item ◦

```
(0, '::', 'one')
(1, '::', 'two')
(2, '::', 'three')
(3, '::', 'four')
```

map lambda lambda

```
x = map(lambda e : e.upper(), ['one', 'two', 'three', 'four'])
print(x)
```

```
['ONE', 'TWO', 'THREE', 'FOUR'] # Python 2.x
```

Python 3.x `mapprint(list(x))` <http://www.riptutorial.com/python//8186/map--http://www.riptutorial.com/python/topic/809/incompatibilities-moving-from-python-2-to-python-3> ◦

for list dict ◦

```
for i in [0, 1, 2, 3, 4]:
    print(i)
```

for ◦

i◦ 0 1 2 ◦

```
0
1
2
3
4
```

range for

```
for i in range(5):
    print(i)
```

for ◦ 50 ◦

`for __getitem__ __getitem__ iter__ ◦ __iter__ next iterable ◦`

**break**

break ""

```
i = 0
while i < 7:
    print(i)
```

```
if i == 4:
    print("Breaking from loop")
    break
i += 1
```

break◦ break ◦ break◦

break4

```
0
1
2
3
4
Breaking from loop
```

breakforPython

```
for i in (0, 1, 2, 3, 4):
    print(i)
    if i == 2:
        break
```

```
0
1
2
```

34◦

else break◦

---

**continue**

continue◦ break continue

```
for i in (0, 1, 2, 3, 4, 5):
    if i == 2 or i == 4:
        continue
    print(i)
```

```
0
1
3
5
```

24 continueprint(i)i == 2i == 4◦

---

breakcontinue◦ forwhile

```
while True:
```

```

for i in range(1,5):
    if i == 2:
        break    # Will only break out of the inner loop!

```

## Python-breakreturn

### returnbreak

return

returnbreak

```

def break_loop():
    for i in range(1, 5):
        if (i == 2):
            return(i)
        print(i)
    return(5)

```

return

```

def break_all():
    for j in range(1, 5):
        for i in range(1,4):
            if i*j == 6:
                return(i)
            print(i*j)

```

```

1 # 1*1
2 # 1*2
3 # 1*3
4 # 1*4
2 # 2*1
4 # 2*2
# return because 2*3 = 6, the remaining iterations of both loops are not executed

```

## “else”

forwhileelse

elseforwhilefalse while

```

for i in range(3):
    print(i)
else:
    print('done')

i = 0
while i < 3:
    print(i)
    i += 1
else:
    print('done')

```

```
0
1
2
done
```

```
else break
```

```
for i in range(2):
    print(i)
    if i == 1:
        break
else:
    print('done')
```

```
0
1
```

```
else◦ else◦
```

**Donald Knuth** if goto else◦

```
while loop_condition():
    ...
    if break_condition():
        break
    ...
```

```
# pseudocode
```

```
<<start>>:
if loop_condition():
    ...
    if break_condition():
        goto <<end>>
    ...
    goto <<start>>

<<end>>:
```

```
else◦
```

```
while loop_condition():
    ...
    if break_condition():
        break
    ...
else:
    print('done')
```

```
# pseudocode
```

```
<<start>>:
if loop_condition():
```

```
...
if break_condition():
    goto <<end>>
...
goto <<start>>
else:
    print('done')

<<end>>:
```

elsefor° **True**°

for...else

```
a = [1, 2, 3, 4]
for i in a:
    if type(i) is not int:
        print(i)
        break
else:
    print("no exception")
```

else “*if not break*” “*if not found*”。

[\[Python-ideas\] for ... else thread for Summary pythonforwhile'else'](#)

```
d = {"a": 1, "b": 2, "c": 3}
```

```
for key in d:
    print(key)
```

```
"a"
"b"
"c"
```

```
for key in d.keys():
    print(key)
```

## Python 2

```
for key in d.iterkeys():
    print(key)
```

```
for value in d.values():
    print(value)
```

```
1
2
3
```

```
for key, value in d.items():
    print(key, ":", value)
```

```
a :: 1
b :: 2
c :: 3
```

**Python 2** `.keys()` `.values()` `.items()` `list*` `.iterkeys()` `.itervalues()` `.iteritems()` `*`

`.keys().iterkeys()` `.values().itervalues()` `.items().iteritems()` `iter*` `*` `list*`

**Python 3** `*`

## Loop

`while` `false` `*` `4` `*`

```
i = 0
while i < 4:
    #loop statements
    i = i + 1
```

`for` `while` `*` `myObject` `*`

```
myObject = anObject()
while myObject.isNotReady():
    myObject.tryToGetReady()
```

`while` `True`

```
import cmath

complex_num = cmath.sqrt(-1)
while complex_num:      # You can also replace complex_num with any number, True or a value of
    any type
    print(complex_num)  # Prints 1j forever
```

`while` `break` `return` `*`

```
while True:
    print "Infinite loop"
# Infinite loop
# Infinite loop
# Infinite loop
# ...
```

`pass` `null` `Python` `for` `while` `*` `*`

```
for x in range(10):
    pass #we don't want to do anything, or are not ready to do anything here, so we'll pass
```



◦ for ◦ pass ◦

passwhile ◦

```
while x == y:
    pass
```

◦ ◦ Python ◦ ◦

```
lst = ['alpha', 'bravo', 'charlie', 'delta', 'echo']
```

---

for

```
for s in lst:
    print s[:1] # print the first letter
```

forlstS ◦

```
a
b
c
d
e
```

◦ enumerate ◦

```
for idx, s in enumerate(lst):
    print("%s has an index of %d" % (s, idx))
```

idxs ◦

```
alpha has an index of 0
bravo has an index of 1
charlie has an index of 2
delta has an index of 3
echo has an index of 4
```

---

Pythonrange ◦

```
for i in range(2,4):
    print("lst at %d contains %s" % (i, lst[i]))
```

```
lst at 2 contains charlie
lst at 3 contains delta
```

◦ 12 ◦ for ◦

```
for s in lst[1::2]:
    print(s)

for i in range(1, len(lst), 2):
    print(lst[i])
```

```
bravo
delta
```

◦

“”

Python `do-untild` `while True` `break`

```
a = 10
while True:
    a = a-1
    print(a)
    if a<7:
        break
print('Done.')
```

```
9
8
7
6
Done.
```

```
collection = [('a', 'b', 'c'), ('x', 'y', 'z'), ('1', '2', '3')]
```

```
for item in collection:
    i1 = item[0]
    i2 = item[1]
    i3 = item[2]
    # logic
```

```
for item in collection:
    i1, i2, i3 = item
    # logic
```

```
for i1, i2, i3 in collection:
    # logic
```

◦

<https://riptutorial.com/zh-TW/python/topic/237/>

# 125:

Python ◦ ◦ // ◦ Donald Knuth

“97. 3. “

## Examples

◦ // ◦ Donald Knuth

“97. 3”

cProfile profile line\_profiler timeit ◦ ◦

cProfile cProfile cProfile ◦ [ <https://docs.python.org/2/library/profile.html>] [1 ]

```
import cProfile
def f(x):
    return "42!"
cProfile.run('f(12)')
```

```
import cProfile, pstats, StringIO
pr = cProfile.Profile()
pr.enable()
# ... do something ...
# ... long ...
pr.disable()
sortby = 'cumulative'
ps = pstats.Stats(pr, stream=StringIO()).sort_stats(sortby)
ps.print_stats()
print ps.getvalue()
```

◦

3 function calls in 0.000 seconds

Ordered by: standard name

| ncalls | totttime | percall | cumtime | percall | filename:lineno(function)                        |
|--------|----------|---------|---------|---------|--------------------------------------------------|
| 1      | 0.000    | 0.000   | 0.000   | 0.000   | <stdin>:1(f)                                     |
| 1      | 0.000    | 0.000   | 0.000   | 0.000   | <string>:1(<module>)                             |
| 1      | 0.000    | 0.000   | 0.000   | 0.000   | {method 'disable' of '_lsprof.Profiler' objects} |

line\_profiler [ [https://github.com/rkern/line\\_profiler](https://github.com/rkern/line_profiler)] [1 ] ◦ ◦ kernprof.

```
$ kernprof -l script_to_profile.py
```

kernprof LineProfiler profile\_\_builtins\_\_ ◦ @profile @profile ◦

```
@profile
```

```
def slow_function(a, b, c):  
    ...
```

**kernprof** script\_to\_profile.py.lprof ◦ **kernprof**[-v / - view]◦

```
$ python -m line_profiler script_to_profile.py.lprof
```

**timeit** python shell◦ list()list()◦ setup-s◦

```
>>> import timeit  
>>> timeit.timeit('"-".join(str(n) for n in range(100))', number=10000)  
0.8187260627746582
```

```
$ python -m timeit '"-".join(str(n) for n in range(100))'  
10000 loops, best of 3: 40.3 usec per loop
```

<https://riptutorial.com/zh-TW/python/topic/5889/>

# 126:

## Examples

`dict.get` ◦ ◦ `dict[key]` `KeyError` ◦

```
def add_student():
    try:
        students['count'] += 1
    except KeyError:
        students['count'] = 1
```

```
def add_student():
    students['count'] = students.get('count', 0) + 1
```

◦

```
x = True
y = False
x, y = y, x
x
# False
y
# True
```

## Python

```
# Good examples, using implicit truth testing
if attr:
    # do something

if not attr:
    # do something

# Bad examples, using specific types
if attr == 1:
    # do something

if attr == True:
    # do something

if attr != '':
    # do something

# If you are looking to specifically check for None, use 'is' or 'is not'
if attr is None:
    # do something
```

◦

`False` ◦

“`__main__`”

`__name__`°

```
import sys

def main():
    # Your code starts here

    # Don't forget to provide a return code
    return 0

if __name__ == "__main__":
    sys.exit(main())
```

;

```
python my_program.py
```

`import`° `import __main__`

```
# A new program file
import my_program          # main() is not run

# But you can run main() explicitly if you really want it to run:
my_program.main()
```

<https://riptutorial.com/zh-TW/python/topic/3070/>

# 127:

## Examples

### Python 3 print◦

```
print('hello world!')
# out: hello world!

foo = 1
bar = 'bar'
baz = 3.14

print(foo)
# out: 1
print(bar)
# out: bar
print(baz)
# out: 3.14
```

### print

```
print(foo, bar, baz)
# out: 1 bar 3.14
```

### print+

```
print(str(foo) + " " + bar + " " + str(baz))
# out: 1 bar 3.14
```

### +◦ string1"bar"3.14◦

```
# Wrong:
# type:int str float
print(foo + bar + baz)
# will result in an error
```

### print

```
print(4 + 5)
# out: 9
print("4" + "5")
# out: 45
print([4] + [5])
# out: [4, 5]
```

### +◦

```
import random
#telling python to include a function to create random numbers
randnum = random.randint(0, 12)
```

```
#make a random number between 0 and 12 and assign it to a variable
print("The randomly generated number was - " + str(randnum))
```

endprint

```
print("this has no newline at the end of it... ", end="")
print("see?")
# out: this has no newline at the end of it... see?
```

file

```
with open('my_file.txt', 'w+') as my_file:
    print("this goes to the file!", file=my_file)
```

◦ print◦

sep ◦

```
>>> print('apples', 'bannas', 'cherries', sep=', ')
apple, bannas, cherries
>>> print('apple', 'banna', 'cherries', sep=', ')
apple, banna, cherries
>>>
```

end end

endprint()◦ ""◦

```
>>> print("<a", end=''); print(" class='jldn'" if 1 else "", end=''); print(">")
<a class='jldn'/>
>>> print("paragraph1", end="\n\n"); print("paragraph2")
paragraph1

paragraph2
>>>
```

file **sys.stdout**◦

**stdoutStringIO**◦ ◦

```
>>> def sendit(out, *values, sep=' ', end='\n'):
...     print(*values, sep=sep, end=end, file=out)
...
>>> sendit(sys.stdout, 'apples', 'bannas', 'cherries', sep='\t')
apples    bannas    cherries
>>> with open("delete-me.txt", "w+") as f:
...     sendit(f, 'apples', 'bannas', 'cherries', sep=' ', end='\n')
...
>>> with open("delete-me.txt", "rt") as f:
...     print(f.read())
...
apples bannas cherries

>>>
```



flush◦

<https://riptutorial.com/zh-TW/python/topic/1360/>

# 128: abc

## Examples

### ABCMeta

- 
- 

### NotImplementedError ◦ ◦

```
class Fruit:

    def check_ripeness(self):
        raise NotImplementedError("check_ripeness method not implemented!")

class Apple(Fruit):
    pass

a = Apple()
a.check_ripeness() # raises NotImplementedError
```

- abc

```
from abc import ABCMeta

class AbstractClass(object):
    # the metaclass attribute must always be set as a class variable
    __metaclass__ = ABCMeta

    # the abstractmethod decorator registers this method as undefined
    @abstractmethod
    def virtual_method_subclasses_must_define(self):
        # Can be left completely blank, or a base implementation can be provided
        # Note that ordinarily a blank interpretation implicitly returns `None`,
        # but by registering, this behaviour is no longer enforced.
```

```
class Subclass(AbstractClass):
    def virtual_method_subclasses_must_define(self):
        return
```

### /ABCMeta@abstractmethod

### ABCs ◦

### Van Rossum ◦ Base“MontyPython” ◦

```

class MontyPython:
    def joke(self):
        raise NotImplementedError()

    def punchline(self):
        raise NotImplementedError()

class ArgumentClinic(MontyPython):
    def joke(self):
        return "Hahahahahah"

```

punchline()◦

```

>>> sketch = ArgumentClinic()
>>> sketch.punchline()
NotImplementedError

```

ArgumentClinic◦ punchline◦

ABC◦

```

from abc import ABCMeta, abstractmethod

class MontyPython(metaclass=ABCMeta):
    @abstractmethod
    def joke(self):
        pass

    @abstractmethod
    def punchline(self):
        pass

class ArgumentClinic(MontyPython):
    def joke(self):
        return "Hahahahahah"

```

TypeError

```

>>> c = ArgumentClinic()
TypeError:
"Can't instantiate abstract class ArgumentClinic with abstract methods punchline"

```

TypeErrors

```

class ArgumentClinic(MontyPython):
    def joke(self):
        return "Hahahahahah"

    def punchline(self):
        return "Send in the constable!"

```

[abc https://riptutorial.com/zh-TW/python/topic/5442/-abc-](https://riptutorial.com/zh-TW/python/topic/5442/-abc-)

# 129:

## Examples

### python

#### pythondocstring

```
#!/usr/local/bin/python3

import ast
import sys

""" The data we collect. Each key is a function name; each value is a dict
with keys: firstline, sigend, docend, and lastline and values of line numbers
where that happens. """
functions = {}

def process(functions):
    """ Handle the function data stored in functions. """
    for funcname,data in functions.items():
        print("function:",funcname)
        print("\tstarts at line:",data['firstline'])
        print("\tsignature ends at line:",data['sigend'])
        if ( data['sigend'] < data['docend'] ):
            print("\tdocstring ends at line:",data['docend'])
        else:
            print("\tno docstring")
        print("\tfunction ends at line:",data['lastline'])
        print()

class FuncLister(ast.NodeVisitor):
    def visit_FunctionDef(self, node):
        """ Recursively visit all functions, determining where each function
starts, where its signature ends, where the docstring ends, and where
the function ends. """
        functions[node.name] = {'firstline':node.lineno}
        sigend = max(node.lineno,lastline(node.args))
        functions[node.name]['sigend'] = sigend
        docstring = ast.get_docstring(node)
        docstringlength = len(docstring.split('\n')) if docstring else -1
        functions[node.name]['docend'] = sigend+docstringlength
        functions[node.name]['lastline'] = lastline(node)
        self.generic_visit(node)

def lastline(node):
    """ Recursively find the last line of a node """
    return max( [ node.lineno if hasattr(node,'lineno') else -1 , ]
               +[lastline(child) for child in ast.iter_child_nodes(node)] )

def readin(pythonfilename):
    """ Read the file name and store the function data into functions. """
    with open(pythonfilename) as f:
        code = f.read()
    FuncLister().visit(ast.parse(code))
```

```
def analyze(file,process):
    """ Read the file and process the function data. """
    readin(file)
    process(functions)

if __name__ == '__main__':
    if len(sys.argv)>1:
        for file in sys.argv[1:]:
            analyze(file,process)
    else:
        analyze(sys.argv[0],process)
```

<https://riptutorial.com/zh-TW/python/topic/5370/>

# 130:

◦ ◦ ◦ Python◦

- $x \ll y$
- $x \gg y$
- $xy$ AND
- $x | y$ OR
- $\sim x$ NOT
- $x \wedge y$  #Bitwise XOR

## Examples

### AND

&AND ◦

```
# 0 & 0 = 0
# 0 & 1 = 0
# 1 & 0 = 0
# 1 & 1 = 1

# 60 = 0b111100
# 30 = 0b011110
60 & 30
# Out: 28
# 28 = 0b11100

bin(60 & 30)
# Out: 0b11100
```

### OR

|“”◦

```
# 0 | 0 = 0
# 0 | 1 = 1
# 1 | 0 = 1
# 1 | 1 = 1

# 60 = 0b111100
# 30 = 0b011110
60 | 30
# Out: 62
# 62 = 0b111110
```

```
bin(60 | 30)
# Out: 0b111110
```

^**XOR** 11 ◦ 1 ◦

```
# 0 ^ 0 = 0
# 0 ^ 1 = 1
# 1 ^ 0 = 1
# 1 ^ 1 = 0

# 60 = 0b111100
# 30 = 0b011110
60 ^ 30
# Out: 34
# 34 = 0b100010

bin(60 ^ 30)
# Out: 0b100010
```

<<“”◦

```
# 2 = 0b10
2 << 2
# Out: 8
# 8 = 0b1000

bin(2 << 2)
# Out: 0b1000
```

12

```
7 << 1
# Out: 14
```

n2\*\*n

```
3 << 4
# Out: 48
```

>>“”◦

```
# 8 = 0b1000
8 >> 2
# Out: 2
# 2 = 0b10

bin(8 >> 2)
# Out: 0b10
```

12

```
36 >> 1
```

```
# Out: 18
```

```
15 >> 1
```

```
# Out: 7
```

$n2^{**n}$

```
48 >> 4
```

```
# Out: 3
```

```
59 >> 3
```

```
# Out: 7
```

## NOT

$\sim 0 = -10$

80000 00000111 111101271xxx xxxx

|           |     |      |
|-----------|-----|------|
| 0000 0000 | 0   | 0    |
| 0000 0001 | 1   | 1    |
| 0000 0010 | 2   | 2    |
| 0111 1110 | 126 | 126  |
| 0111 1111 | 127 | 127  |
| 1000 0000 | 128 | -128 |
| 1000 0001 | 129 | -127 |
| 1000 0010 | 130 | -126 |
| 1111 1110 | 254 | -2   |
| 1111 1111 | 255 | -1   |

1010 0110166  $(128 * 1) + (64 * 0) + (32 * 1) + (16 * 0) + (8 * 0) + (4 * 1) + (2 * 1) + (1 * 0) -$   
90  $(128 * 1) - (64 * 0) - (32 * 1) - (16 * 0) - (8 * 0) - (4 * 1) - (2 * 1) - (1 * 0)$

-128 1000 0000 0 0000 0000 -11111 1111

$\sim n = -n - 1$

```
# 0 = 0b0000 0000
```

```
~0
```

```
# Out: -1
```



```

# -1 = 0b1111 1111

# 1 = 0b0000 0001
~1
# Out: -2
# -2 = 1111 1110

# 2 = 0b0000 0010
~2
# Out: -3
# -3 = 0b1111 1101

# 123 = 0b0111 1011
~123
# Out: -124
# -124 = 0b1000 0100

```

~n -> -|n+1|

~-n -> |n-1|

.....

```

# -0 = 0b0000 0000
~-0
# Out: -1
# -1 = 0b1111 1111
# 0 is the obvious exception to this rule, as -0 == 0 always

# -1 = 0b1000 0001
~-1
# Out: 0
# 0 = 0b0000 0000

# -2 = 0b1111 1110
~-2
# Out: 1
# 1 = 0b0000 0001

# -123 = 0b1111 1011
~-123
# Out: 122
# 122 = 0b0111 1010

```

## Bitwise~

```

a = 0b001
a &= 0b010
# a = 0b000

a = 0b001
a |= 0b010
# a = 0b011

a = 0b001
a <<= 2
# a = 0b100

a = 0b100

```

```
a >>= 2
# a = 0b001

a = 0b101
a ^= 0b011
# a = 0b110
```

<https://riptutorial.com/zh-TW/python/topic/730/>

# 131:

## Examples

```
min(7,2,1,5)
# Output: 1

max(7,2,1,5)
# Output: 7
```

/

```
list_of_tuples = [(0, 10), (1, 15), (2, 8)]
min(list_of_tuples)
# Output: (0, 10)
```

### key -argument

```
min(list_of_tuples, key=lambda x: x[0])          # Sorting by first element
# Output: (0, 10)

min(list_of_tuples, key=lambda x: x[1])          # Sorting by second element
# Output: (2, 8)

sorted(list_of_tuples, key=lambda x: x[0])        # Sorting by first element (increasing)
# Output: [(0, 10), (1, 15), (2, 8)]

sorted(list_of_tuples, key=lambda x: x[1])        # Sorting by first element
# Output: [(2, 8), (0, 10), (1, 15)]

import operator
# The operator module contains efficient alternatives to the lambda function
max(list_of_tuples, key=operator.itemgetter(0)) # Sorting by first element
# Output: (2, 8)

max(list_of_tuples, key=operator.itemgetter(1)) # Sorting by second element
# Output: (1, 15)

sorted(list_of_tuples, key=operator.itemgetter(0), reverse=True) # Reversed (decreasing)
# Output: [(2, 8), (1, 15), (0, 10)]

sorted(list_of_tuples, key=operator.itemgetter(1), reverse=True) # Reversed(decreasing)
# Output: [(1, 15), (0, 10), (2, 8)]
```

### maxmin

maxmin

```
min([])
```

ValueError: min() arg

## Python 3 default

```
max([], default=42)
# Output: 42
max([], default=0)
# Output: 0
```

## sorted dict

```
adict = {'a': 3, 'b': 5, 'c': 1}
min(adict)
# Output: 'a'
max(adict)
# Output: 'c'
sorted(adict)
# Output: ['a', 'b', 'c']
```

## .items()

```
min(adict.items())
# Output: ('a', 3)
max(adict.items())
# Output: ('c', 1)
sorted(adict.items())
# Output: [('a', 3), ('b', 5), ('c', 1)]
```

## sorted OrderedDict dict

```
from collections import OrderedDict
OrderedDict(sorted(adict.items()))
# Output: OrderedDict([('a', 3), ('b', 5), ('c', 1)])
res = OrderedDict(sorted(adict.items()))
res['a']
# Output: 3
```

---

## key

```
min(adict.items(), key=lambda x: x[1])
# Output: ('c', 1)
max(adict.items(), key=operator.itemgetter(1))
# Output: ('b', 5)
sorted(adict.items(), key=operator.itemgetter(1), reverse=True)
# Output: [('b', 5), ('a', 3), ('c', 1)]
```

```
sorted((7, 2, 1, 5))           # tuple
# Output: [1, 2, 5, 7]

sorted(['c', 'A', 'b'])       # list
# Output: ['A', 'b', 'c']

sorted({11, 8, 1})           # set
# Output: [1, 8, 11]
```

```
sorted({'11': 5, '3': 2, '10': 15}) # dict
# Output: ['10', '11', '3']         # only iterates over the keys

sorted('bdca')                     # string
# Output: ['a', 'b', 'c', 'd']
```

list ;°

sorted

```
min([2, 7, 5])
# Output: 2
sorted([2, 7, 5])[0]
# Output: 2
```

sorted **order** max°

```
max([2, 7, 5])
# Output: 7
sorted([2, 7, 5])[-1]
# Output: 7
```

```
class MyClass(object):
    def __init__(self, value, name):
        self.value = value
        self.name = name

    def __lt__(self, other):
        return self.value < other.value

    def __repr__(self):
        return str(self.name)

sorted([MyClass(4, 'first'), MyClass(1, 'second'), MyClass(4, 'third')])
# Output: [second, first, third]
max([MyClass(4, 'first'), MyClass(1, 'second'), MyClass(4, 'third')])
# Output: first
```

<>°

min maxsorted° **6** `__lt__` `__gt__` `__ge__` `__le__` `__ne__` `__ge__` `__le__` `__ne__` `__eq__`

```
class IntegerContainer(object):
    def __init__(self, value):
        self.value = value

    def __repr__(self):
        return "{}({})".format(self.__class__.__name__, self.value)

    def __lt__(self, other):
        print('{!r} - Test less than {!r}'.format(self, other))
        return self.value < other.value

    def __le__(self, other):
        print('{!r} - Test less than or equal to {!r}'.format(self, other))
```

```

        return self.value <= other.value

    def __gt__(self, other):
        print('{!r} - Test greater than {!r}'.format(self, other))
        return self.value > other.value

    def __ge__(self, other):
        print('{!r} - Test greater than or equal to {!r}'.format(self, other))
        return self.value >= other.value

    def __eq__(self, other):
        print('{!r} - Test equal to {!r}'.format(self, other))
        return self.value == other.value

    def __ne__(self, other):
        print('{!r} - Test not equal to {!r}'.format(self, other))
        return self.value != other.value

```

o

```

alist = [IntegerContainer(5), IntegerContainer(3),
        IntegerContainer(10), IntegerContainer(7)
        ]

res = max(alist)
# Out: IntegerContainer(3) - Test greater than IntegerContainer(5)
#      IntegerContainer(10) - Test greater than IntegerContainer(5)
#      IntegerContainer(7) - Test greater than IntegerContainer(10)
print(res)
# Out: IntegerContainer(10)

res = min(alist)
# Out: IntegerContainer(3) - Test less than IntegerContainer(5)
#      IntegerContainer(10) - Test less than IntegerContainer(3)
#      IntegerContainer(7) - Test less than IntegerContainer(3)
print(res)
# Out: IntegerContainer(3)

res = sorted(alist)
# Out: IntegerContainer(3) - Test less than IntegerContainer(5)
#      IntegerContainer(10) - Test less than IntegerContainer(3)
#      IntegerContainer(10) - Test less than IntegerContainer(5)
#      IntegerContainer(7) - Test less than IntegerContainer(5)
#      IntegerContainer(7) - Test less than IntegerContainer(10)
print(res)
# Out: [IntegerContainer(3), IntegerContainer(5), IntegerContainer(7), IntegerContainer(10)]

```

reverse=True sorted\_\_lt\_\_

```

res = sorted(alist, reverse=True)
# Out: IntegerContainer(10) - Test less than IntegerContainer(7)
#      IntegerContainer(3) - Test less than IntegerContainer(10)
#      IntegerContainer(3) - Test less than IntegerContainer(10)
#      IntegerContainer(3) - Test less than IntegerContainer(7)
#      IntegerContainer(5) - Test less than IntegerContainer(7)
#      IntegerContainer(5) - Test less than IntegerContainer(3)
print(res)
# Out: [IntegerContainer(10), IntegerContainer(7), IntegerContainer(5), IntegerContainer(3)]

```

sorted\_gt\_\_

```
del IntegerContainer.__lt__ # The IntegerContainer no longer implements "less than"

res = min(alist)
# Out: IntegerContainer(5) - Test greater than IntegerContainer(3)
#      IntegerContainer(3) - Test greater than IntegerContainer(10)
#      IntegerContainer(3) - Test greater than IntegerContainer(7)
print(res)
# Out: IntegerContainer(3)
```

\_\_lt\_\_\_\_gt\_\_\_\_gt\_\_TypeError

```
del IntegerContainer.__gt__ # The IntegerContainer no longer implements "greater than"

res = min(alist)
```

## TypeErrorunorderableIntegerContainer<IntegerContainer

---

[functools.total\\_ordering](#)° total\_ordering \_\_eq\_\_ \_\_ne\_\_ \_\_lt\_\_ \_\_le\_\_ \_\_ge\_\_ \_\_gt\_\_

```
import functools

@functools.total_ordering
class IntegerContainer(object):
    def __init__(self, value):
        self.value = value

    def __repr__(self):
        return "{}({})".format(self.__class__.__name__, self.value)

    def __lt__(self, other):
        print('{!r} - Test less than {!r}'.format(self, other))
        return self.value < other.value

    def __eq__(self, other):
        print('{!r} - Test equal to {!r}'.format(self, other))
        return self.value == other.value

    def __ne__(self, other):
        print('{!r} - Test not equal to {!r}'.format(self, other))
        return self.value != other.value

IntegerContainer(5) > IntegerContainer(6)
# Output: IntegerContainer(5) - Test less than IntegerContainer(6)
# Returns: False

IntegerContainer(6) > IntegerContainer(5)
# Output: IntegerContainer(6) - Test less than IntegerContainer(5)
# Output: IntegerContainer(6) - Test equal to IntegerContainer(5)
# Returns True
```

> \_\_eq\_\_° °

## NN

### [heapqnlargestnsmallest](#)

```
import heapq

# get 5 largest items from the range

heapq.nlargest(5, range(10))
# Output: [9, 8, 7, 6, 5]

heapq.nsmallest(5, range(10))
# Output: [0, 1, 2, 3, 4]
```

iterable。 ◦

min maxsortedkey◦

## 1000

```
import heapq
with open(filename) as f:
    longest_lines = heapq.nlargest(1000, f, key=len)
```

fnlargest ◦ ;nlargestlen◦ len ◦

## 1000

```
longest_lines = sorted(f, key=len)[1000:]
```

◦

[https://riptutorial.com/zh-TW/python/topic/252/-](https://riptutorial.com/zh-TW/python/topic/252/)



# 132:

## Examples

◦ `__get__()` `__set__()` `__delete__()` ◦ ◦ ◦ ◦

**AttributeError** ◦ **set** ◦

```
descr.__get__(self, obj, type=None) --> value
descr.__set__(self, obj, value) --> None
descr.__delete__(self, obj) --> None
```

```
class DescPrinter(object):
    """A data descriptor that logs activity."""
    _val = 7

    def __get__(self, obj, objtype=None):
        print('Getting ...')
        return self._val

    def __set__(self, obj, val):
        print('Setting', val)
        self._val = val

    def __delete__(self, obj):
        print('Deleting ...')
        del self._val

class Foo():
    x = DescPrinter()

i = Foo()
i.x
# Getting ...
# 7

i.x = 100
# Setting 100
i.x
# Getting ...
# 100

del i.x
# Deleting ...
i.x
# Getting ...
# 7
```

◦

◦

```
>>> oscillator = Oscillator(freq=100.0) # Set frequency to 100.0 Hz
>>> oscillator.period # Period is 1 / frequency, i.e. 0.01 seconds
0.01
>>> oscillator.period = 0.02 # Set period to 0.02 seconds
>>> oscillator.freq # The frequency is automatically adjusted
50.0
>>> oscillator.freq = 200.0 # Set the frequency to 200.0 Hz
>>> oscillator.period # The period is automatically adjusted
0.005
```

“”

```
class Hertz(object):
    def __get__(self, instance, owner):
        return self.value

    def __set__(self, instance, value):
        self.value = float(value)
```

“”。

```
class Second(object):
    def __get__(self, instance, owner):
        # When reading period, convert from frequency
        return 1 / instance.freq

    def __set__(self, instance, value):
        # When setting period, update the frequency
        instance.freq = 1 / float(value)
```

```
class Oscillator(object):
    period = Second() # Set the other value as a class attribute

    def __init__(self, freq):
        self.freq = Hertz() # Set the anchor value as an instance attribute
        self.freq = freq # Assign the passed value - self.period will be adjusted
```

<https://riptutorial.com/zh-TW/python/topic/3405/>

# 133: /

Python。

。

Python。 Exception。 。

## Examples

CustomErrorException。 raise。

```
class CustomError(Exception):
    pass

x = 1

if x == 1:
    raise CustomError('This is custom error')
```

```
Traceback (most recent call last):
  File "error_custom.py", line 8, in <module>
    raise CustomError('This is custom error')
__main__.CustomError: This is custom error
```

## Exception

```
class CustomError(Exception):
    pass

try:
    raise CustomError('Can you catch me ?')
except CustomError as e:
    print ('Caught CustomError :{}'.format(e))
except Exception as e:
    print ('Generic exception: {}'.format(e))
```

```
Caught CustomError :Can you catch me ?
```

[/ https://riptutorial.com/zh-TW/python/topic/10882/-](https://riptutorial.com/zh-TW/python/topic/10882/)

# 134:

## Examples

mixin ◦ mixin ◦

◦ Mixins ◦ mixin ◦

mixins ◦ mixin ◦

```
class Mixin1(object):
    def test(self):
        print "Mixin1"

class Mixin2(object):
    def test(self):
        print "Mixin2"

class MyClass(Mixin1, Mixin2):
    pass
```

MyClasstest

```
>>> obj = MyClass()
>>> obj.test()
Mixin1
```

Mixin1 ◦ ◦

```
class MyClass(Mixin2, Mixin1):
    pass
```

```
>>> obj = MyClass()
>>> obj.test()
Mixin2
```

Mixins ◦

## Python 3.x 3.0

```
class Base(object):
    def test(self):
        print("Base.")

class PluginA(object):
    def test(self):
        super().test()
        print("Plugin A.")

class PluginB(object):
    def test(self):
```

```

        super().test()
        print("Plugin B.")

plugins = PluginA, PluginB

class PluginSystemA(PluginA, Base):
    pass

class PluginSystemB(PluginB, Base):
    pass

PluginSystemA().test()
# Base.
# Plugin A.

PluginSystemB().test()
# Base.
# Plugin B.

```

Python 3.6 [PEP 487](#) `__init_subclass__` ° °

## Python 3.x 3.6

```

class Base:
    plugins = []

    def __init_subclass__(cls, **kwargs):
        super().__init_subclass__(**kwargs)
        cls.plugins.append(cls)

    def test(self):
        print("Base.")

class PluginA(Base):
    def test(self):
        super().test()
        print("Plugin A.")

class PluginB(Base):
    def test(self):
        super().test()
        print("Plugin B.")

```

```

PluginA().test()
# Base.
# Plugin A.

PluginB().test()
# Base.
# Plugin B.

Base.plugins
# [__main__.PluginA, __main__.PluginB]

```

<https://riptutorial.com/zh-TW/python/topic/4724/>

# 135:

`nO(n) ◦ bisect.bisect_left()O(log(n)) ◦`

## Examples

### `str.indexstr.rindexstr.findstr.rfind`

`Stringindexstr.find ◦ ◦`

```
astring = 'Hello on StackOverflow'
astring.index('o') # 4
astring.rindex('o') # 20

astring.find('o') # 4
astring.rfind('o') # 20
```

`index / rindexfind / rfind`

```
astring.index('q') # ValueError: substring not found
astring.find('q') # -1
```

```
astring.index('o', 5) # 6
astring.index('o', 6) # 6 - start is inclusive
astring.index('o', 5, 7) # 6
astring.index('o', 5, 6) # - end is not inclusive
```

### ValueError

```
astring.rindex('o', 20) # 20
astring.rindex('o', 19) # 20 - still from left to right

astring.rindex('o', 4, 7) # 6
```

### Python<sub>in</sub> ◦

```
alist = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
5 in alist # True
10 in alist # False
```

```
atuple = ('0', '1', '2', '3', '4')
4 in atuple # False
'4' in atuple # True
```

```
astring = 'i am a string'
'a' in astring # True
'am' in astring # True
'I' in astring # False
```

```
aset = {(10, 10), (20, 20), (30, 30)}
(10, 10) in aset # True
10 in aset      # False
```

dictin ◦ ◦ ◦

```
adict = {0: 'a', 1: 'b', 2: 'c', 3: 'd'}
1 in adict          # True   - implicitly searches in keys
'a' in adict        # False
2 in adict.keys()   # True   - explicitly searches in keys
'a' in adict.values() # True  - explicitly searches in values
(0, 'a') in adict.items() # True - explicitly searches key/value pairs
```

## list.indextuple.index

listtupleindex -method

```
alist = [10, 16, 26, 5, 2, 19, 105, 26]
# search for 16 in the list
alist.index(16) # 1
alist[1]        # 16

alist.index(15)
```

### ValueError15

```
atuple = (10, 16, 26, 5, 2, 19, 105, 26)
atuple.index(26) # 2
atuple[2]        # 26
atuple[7]        # 26 - is also 26!
```

## dict

dict◦

```
def getKeysForValue(dictionary, value):
    foundkeys = []
    for keys in dictionary:
        if dictionary[key] == value:
            foundkeys.append(key)
    return foundkeys
```

```
def getKeysForValueComp(dictionary, value):
    return [key for key in dictionary if dictionary[key] == value]
```

```
def getOneKeyForValue(dictionary, value):
    return next(key for key in dictionary if dictionary[key] == value)
```

keyslist

```
adict = {'a': 10, 'b': 20, 'c': 10}
getKeysForValue(adict, 10)      # ['c', 'a'] - order is random could as well be ['a', 'c']
getKeysForValueComp(adict, 10) # ['c', 'a'] - dito
getKeysForValueComp(adict, 20) # ['b']
getKeysForValueComp(adict, 25) # []
```

```
getOneKeyForValue(adict, 10)    # 'c' - depending on the circumstances this could also be 'a'
getOneKeyForValue(adict, 20)    # 'b'
```

dict StopIteration - Exception

```
getOneKeyForValue(adict, 25)
```

## StopIteration

## bisect.bisect\_left

bisect.bisect\_left() <sup>1</sup>

```
import bisect

def index_sorted(sorted_seq, value):
    """Locate the leftmost value exactly equal to x or raise a ValueError"""
    i = bisect.bisect_left(sorted_seq, value)
    if i != len(sorted_seq) and sorted_seq[i] == value:
        return i
    raise ValueError

alist = [i for i in range(1, 100000, 3)] # Sorted list from 1 to 100000 with step 3
index_sorted(alist, 97285) # 32428
index_sorted(alist, 4)    # 1
index_sorted(alist, 97286)
```

## ValueError

### ◦ 500

```
%timeit index_sorted(alist, 97285)
# 100000 loops, best of 3: 3 µs per loop
%timeit alist.index(97285)
# 1000 loops, best of 3: 1.58 ms per loop
```

```
%timeit index_sorted(alist, 4)
# 100000 loops, best of 3: 2.98 µs per loop
%timeit alist.index(4)
# 1000000 loops, best of 3: 580 ns per loop
```

tuple list dict◦

```
def outer_index(nested_sequence, value):
    return next(index for index, inner in enumerate(nested_sequence)
                for item in inner)
```



```

        if item == value)

alist_of_tuples = [(4, 5, 6), (3, 1, 'a'), (7, 0, 4.3)]
outer_index(alist_of_tuples, 'a') # 1
outer_index(alist_of_tuples, 4.3) # 2

```

```

def outer_inner_index(nested_sequence, value):
    return next((oindex, iindex) for oindex, inner in enumerate(nested_sequence)
                for iindex, item in enumerate(inner)
                if item == value)

outer_inner_index(alist_of_tuples, 'a') # (1, 2)
alist_of_tuples[1][2] # 'a'

outer_inner_index(alist_of_tuples, 7) # (2, 0)
alist_of_tuples[2][0] # 7

```

next◦

**\_\_content\_\_ iter\_\_**

in **for\_\_contains\_\_ \_\_iter\_\_ -method◦**

list list

```

class ListList:
    def __init__(self, value):
        self.value = value
        # Create a set of all values for fast access
        self.setofvalues = set(item for sublist in self.value for item in sublist)

    def __iter__(self):
        print('Using __iter__.')
        # A generator over all sublist elements
        return (item for sublist in self.value for item in sublist)

    def __contains__(self, value):
        print('Using __contains__.')
        # Just lookup if the value is in the set
        return value in self.setofvalues

        # Even without the set you could use the iter method for the contains-check:
        # return any(item == value for item in iter(self))

```

in

```

a = ListList([[1,1,1],[0,1,1],[1,5,1]])
10 in a # False
# Prints: Using __contains__.
5 in a # True
# Prints: Using __contains__.

```

**\_\_contains\_\_**

```
del ListList.__contains__
```

```
5 in a      # True
# Prints: Using __iter__.
```

in for i in a \_\_iter\_\_\_\_contains\_\_°

<https://riptutorial.com/zh-TW/python/topic/350/>

# 136:

## Examples

+operator -function operator.add for +

```
1 + 1
# Output: 2
from operator import add
add(1, 1)
# Output: 2
```

```
from operator import mul
mul('a', 10)
# Output: 'aaaaaaaaaa'
mul([3], 3)
# Output: [3, 3, 3]
```

Python ◦

## Methodcaller

lambda

```
alist = ['wolf', 'sheep', 'duck']
list(filter(lambda x: x.startswith('d'), alist)) # Keep only elements that start with 'd'
# Output: ['duck']
```

```
from operator import methodcaller
list(filter(methodcaller('startswith', 'd'), alist)) # Does the same but is faster.
# Output: ['duck']
```

## Itemgetter

itemgetter

```
from itertools import groupby
from operator import itemgetter
adict = {'a': 1, 'b': 5, 'c': 1}

dict((i, dict(v)) for i, v in groupby(adict.items(), itemgetter(1)))
# Output: {1: {'a': 1, 'c': 1}, 5: {'b': 5}}
```

lambda

```
dict((i, dict(v)) for i, v in groupby(adict.items(), lambda x: x[1]))
```

```
alist_of_tuples = [(5,2), (1,3), (2,2)]
sorted(alist_of_tuples, key=itemgetter(1,0))
```

```
# Output: [(2, 2), (5, 2), (1, 3)]
```

<https://riptutorial.com/zh-TW/python/topic/257/>

---

# 137: XML

XML。 XML。 APIXML;XML。

## Examples

### ElementTree

#### ElementTree.xml

```
import xml.etree.ElementTree as ET
tree = ET.parse("yourXMLfile.xml")
root = tree.getroot()
```

。

```
for child in root:
    print(child.tag, child.attrib)
```

```
print(root[0][1].text)
```

.find.findall

```
print(root.findall("myTag"))
print(root[0].find("myOtherTag"))
```

## XML

### xmlxml

```
import xml.etree.ElementTree as ET
tree = ET.parse('sample.xml')
root=tree.getroot()
element = root[0] #get first child of root element
```

```
element.set('attribute_name', 'attribute_value') #set the attribute to xml element
element.text="string_text"
```

### Element.remove

```
root.remove(element)
```

### ElementTree.writexmlxml。

```
tree.write('output.xml')
```

## XML

```
import xml.etree.ElementTree as ET
```

### ElementXML

```
p=ET.Element('parent')
```

### SubElementgive

```
c = ET.SubElement(p, 'child1')
```

### dumpxml

```
ET.dump(p)
# Output will be like this
#<parent><child1 /></parent>
```

### ElementTreexmlwrite

```
tree = ET.ElementTree(p)
tree.write("output.xml")
```

### Commentxml

```
comment = ET.Comment('user comment')
p.append(comment) #this comment will be appended to parent element
```

## iterparseXML

XML。 ◦ iterparseXML。

### ElementTree

```
import xml.etree.ElementTree as ET
```

### .xml

```
for event, elem in ET.iterparse("yourXMLfile.xml"):
    ... do something ...
```

/◦ “end”

```
events=("start", "end", "start-ns", "end-ns")
for event, elem in ET.iterparse("yourXMLfile.xml", events=events):
    ... do something ...
```

```

for event, elem in ET.iterparse("yourXMLfile.xml", events=("start","end")):
    if elem.tag == "record_tag" and event == "end":
        print elem.text
        elem.clear()
        ... do something else ...

```

## XPathXML

### 2.7 ElementTreeXPath。 XPathxmlSQL。 findfindallXPath。 xml

```

<Catalog>
  <Books>
    <Book id="1" price="7.95">
      <Title>Do Androids Dream of Electric Sheep?</Title>
      <Author>Philip K. Dick</Author>
    </Book>
    <Book id="5" price="5.95">
      <Title>The Colour of Magic</Title>
      <Author>Terry Pratchett</Author>
    </Book>
    <Book id="7" price="6.95">
      <Title>The Eye of The World</Title>
      <Author>Robert Jordan</Author>
    </Book>
  </Books>
</Catalog>

```

```

import xml.etree.cElementTree as ET
tree = ET.parse('sample.xml')
tree.findall('Books/Book')

```

'''

```

tree.find("Books/Book[Title='The Colour of Magic']")
# always use '=' in the right side of the comparison

```

#### id = 5

```

tree.find("Books/Book[@id='5']")
# searches with xml attributes must have '@' before the name

```

```

tree.find("Books/Book[2]")
# indexes starts at 1, not 0

```

```

tree.find("Books/Book[last()]")
# 'last' is the only xpath function allowed in ElementTree

```

```

tree.findall("./Author")
#searches with // must use a relative path

```

**XML** <https://riptutorial.com/zh-TW/python/topic/479/xml>

# 138:

Shelvepython。 shelveAPIPythonpickle。 Python。 。 pickleanydbm。

```
;close() shelve.open()
```

```
with shelve.open('spam') as db:
    db['eggs'] = 'eggs'
```

shelvepickle。 pickle。

1。 dbm.ndbmdbm.gnu。 dbm。 dbm - 。

2./。 。 。 UnixUnix。

## Examples

```
import shelve
database = shelve.open(filename.suffix)
object = Object()
database['key'] = object
```

```
import shelve

d = shelve.open(filename) # open -- file may get suffix added by low-level
                           # library

d[key] = data              # store data at key (overwrites old data if
                           # using an existing key)
data = d[key]             # retrieve a COPY of data at key (raise KeyError
                           # if no such key)
del d[key]                # delete data stored at key (raises KeyError
                           # if no such key)

flag = key in d           # true if the key exists
klist = list(d.keys())    # a list of all existing keys (slow!)

# as d was opened WITHOUT writeback=True, beware:
d['xx'] = [0, 1, 2]       # this works as expected, but...
d['xx'].append(3)        # *this doesn't!* -- d['xx'] is STILL [0, 1, 2]!

# having opened d without writeback=True, you need to code carefully:
temp = d['xx']            # extracts the copy
temp.append(5)           # mutates the copy
d['xx'] = temp            # stores the copy right back, to persist it

# or, d=shelve.open(filename,writeback=True) would let you just code
# d['xx'].append(5) and have it work as expected, BUT it would also
# consume more memory and make the d.close() operation slower.

d.close()                # close it
```



## DbfilenameShelf。 anydbm。 shelve.open

```
import shelve

s = shelve.open('test_shelf.db')
try:
    s['key1'] = { 'int': 10, 'float':9.5, 'string':'Sample data' }
finally:
    s.close()
```

```
import shelve

s = shelve.open('test_shelf.db')
try:
    existing = s['key1']
finally:
    s.close()

print existing
```

```
$ python shelve_create.py
$ python shelve_existing.py

{'int': 10, 'float': 9.5, 'string': 'Sample data'}
```

## dbm。 shelve。

```
import shelve

s = shelve.open('test_shelf.db', flag='r')
try:
    existing = s['key1']
finally:
    s.close()

print existing
```

### ◦ anydbm。

◦ ◦

```
import shelve

s = shelve.open('test_shelf.db')
try:
    print s['key1']
    s['key1']['new_value'] = 'this was not here before'
finally:
    s.close()

s = shelve.open('test_shelf.db', writeback=True)
try:
    print s['key1']
finally:
    s.close()
```

'key1'.

```
$ python shelve_create.py
$ python shelve_withoutwriteback.py

{'int': 10, 'float': 9.5, 'string': 'Sample data'}
{'int': 10, 'float': 9.5, 'string': 'Sample data'}
```

◦ ◦ ◦

```
import shelve

s = shelve.open('test_shelf.db', writeback=True)
try:
    print s['key1']
    s['key1']['new_value'] = 'this was not here before'
    print s['key1']
finally:
    s.close()

s = shelve.open('test_shelf.db', writeback=True)
try:
    print s['key1']
finally:
    s.close()
```

◦ ◦ ◦ ◦

```
$ python shelve_create.py
$ python shelve_writeback.py

{'int': 10, 'float': 9.5, 'string': 'Sample data'}
{'int': 10, 'new_value': 'this was not here before', 'float': 9.5, 'string': 'Sample data'}
{'int': 10, 'new_value': 'this was not here before', 'float': 9.5, 'string': 'Sample data'}
```

<https://riptutorial.com/zh-TW/python/topic/10629/>

# 139:

collectionsdict list tuple set MutableSetItemsView

1. UserDict
2. UserList
3. UserString

UserDict dict data ;

## Examples

### collections.Counter

Counter dict

```
import collections
counts = collections.Counter([1,2,3])
```

count Counter({1: 1, 2: 1, 3: 1})

```
>>> collections.Counter('Happy Birthday')
Counter({'a': 2, 'p': 2, 'y': 2, 'i': 1, 'r': 1, 'B': 1, ' ': 1, 'H': 1, 'd': 1, 'h': 1, 't': 1})
```

```
>>> collections.Counter('I am Sam Sam I am That Sam-I-am That Sam-I-am! I do not like that
Sam-I-am'.split())
Counter({'I': 3, 'Sam': 2, 'Sam-I-am': 2, 'That': 2, 'am': 2, 'do': 1, 'Sam-I-am!': 1, 'that': 1, 'not': 1, 'like': 1})
```

```
>>> c = collections.Counter({'a': 4, 'b': 2, 'c': -2, 'd': 0})
```

```
>>> c['a']
4
```

```
>>> c['c'] = -3
>>> c
Counter({'a': 4, 'b': 2, 'd': 0, 'c': -3})
```

4 + 2 + 0 - 3

```
>>> sum(c.itervalues()) # negative numbers are counted!
3
```

```
>>> list(c.elements())
['a', 'a', 'a', 'a', 'b', 'b']
```

## 0

```
>>> c = collections.Counter()
Counter({'a': 4, 'b': 2})
```

```
>>> c.clear()
>>> c
Counter()
```

```
>>> c.update({'a': 3, 'b':3})
>>> c.update({'a': 2, 'c':2}) # adds to existing, sets if they don't exist
>>> c
Counter({'a': 5, 'b': 3, 'c': 2})
>>> c.subtract({'a': 3, 'b': 3, 'c': 3}) # subtracts (negative values are allowed)
>>> c
Counter({'a': 2, 'b': 0, 'c': -1})
```

## collections.defaultdict

[collections.defaultdict](#) default\_factory dict ◦ ◦ None ◦

```
>>> state_capitals = collections.defaultdict(str)
>>> state_capitals
defaultdict(<class 'str'>, {})
```

defaultdict default dict default\_factory ◦

default dict str int list dict default\_factory

```
>>> str()
''
>>> int()
0
>>> list
[]
```

default dict ◦

```
>>> state_capitals['Alaska']
''
>>> state_capitals
defaultdict(<class 'str'>, {'Alaska': ''})
```

int

```
>>> fruit_counts = defaultdict(int)
>>> fruit_counts['apple'] += 2 # No errors should occur
>>> fruit_counts
defaultdict(int, {'apple': 2})
>>> fruit_counts['banana'] # No errors should occur
0
>>> fruit_counts # A new key is created
```

```
default_dict(int, {'apple': 2, 'banana': 0})
```

```
>>> state_capitals['Alabama'] = 'Montgomery'  
>>> state_capitals  
defaultdict(<class 'str'>, {'Alabama': 'Montgomery', 'Alaska': ''})
```

list **default\_factory**.

```
>>> s = [('NC', 'Raleigh'), ('VA', 'Richmond'), ('WA', 'Seattle'), ('NC', 'Asheville')]  
>>> dd = collections.defaultdict(list)  
>>> for k, v in s:  
...     dd[k].append(v)  
>>> dd  
defaultdict(<class 'list'>,  
           {'VA': ['Richmond'],  
            'NC': ['Raleigh', 'Asheville'],  
            'WA': ['Seattle']})
```

## collections.OrderedDict

Python.

```
>>> d = {'foo': 5, 'bar': 6}  
>>> print(d)  
{'foo': 5, 'bar': 6}  
>>> d['baz'] = 7  
>>> print(a)  
{'baz': 7, 'foo': 5, 'bar': 6}  
>>> d['foobar'] = 8  
>>> print(a)  
{'baz': 7, 'foo': 5, 'bar': 6, 'foobar': 8}  
...
```

.

for.

collections.OrderedDict. OrderedDict

```
>>> from collections import OrderedDict  
>>> d = OrderedDict([('foo', 5), ('bar', 6)])  
>>> print(d)  
OrderedDict([('foo', 5), ('bar', 6)])  
>>> d['baz'] = 7  
>>> print(d)  
OrderedDict([('foo', 5), ('bar', 6), ('baz', 7)])  
>>> d['foobar'] = 8  
>>> print(d)  
OrderedDict([('foo', 5), ('bar', 6), ('baz', 7), ('foobar', 8)])
```

OrderedDict

```
>>> o = OrderedDict()
```

```
>>> o['key1'] = "value1"
>>> o['key2'] = "value2"
>>> print(o)
OrderedDict([('key1', 'value1'), ('key2', 'value2')])
```

OrderedDict◦

```
>>> d['foo'] = 4
>>> print(d)
OrderedDict([('foo', 4), ('bar', 6), ('baz', 7), ('foobar', 8)])
```

OrderedDict◦

## collections.namedtuple

`namedtuple`Person

```
Person = namedtuple('Person', ['age', 'height', 'name'])
```

◦

```
Person = namedtuple('Person', 'age height name')
```

```
Person = namedtuple('Person', 'age height name')
```

```
dave = Person(30, 178, 'Dave')
```

```
jack = Person(age=30, height=178, name='Jack S.')
```

## namedtuple

```
print(jack.age) # 30
print(jack.name) # 'Jack S.'
```

`namedtuple`'Person' typename ◦ typename

```
Human = namedtuple('Person', 'age height name')
dave = Human(30, 178, 'Dave')
print(dave) # yields: Person(age=30, height=178, name='Dave')
```

## collections.deque

`deque`appenditerable◦ iterable◦ `deque`◦

Deque“deck”◦ Deque◦ `deque`◦ O1◦

pop◦ insert◦ v◦ On◦



## collections.ChainMap

ChainMap **3.3**

maps ChainMap ◦ dicts ◦

ChainMap ◦ python Django Context ◦ ◦ update() ◦

ChainMap ◦ ◦ web POST GET Django Flask ◦ ChainMap ◦

maps ◦ ◦ ◦

```
import collections

# define two dictionaries with at least some keys overlapping.
dict1 = {'apple': 1, 'banana': 2}
dict2 = {'coconut': 1, 'date': 1, 'apple': 3}

# create two ChainMaps with different ordering of those dicts.
combined_dict = collections.ChainMap(dict1, dict2)
reverse_ordered_dict = collections.ChainMap(dict2, dict1)
```

```
for k, v in combined_dict.items():
    print(k, v)

date 1
apple 1
banana 2
coconut 1

for k, v in reverse_ordered_dict.items():
    print(k, v)

date 1
apple 3
banana 2
coconut 1
```

<https://riptutorial.com/zh-TW/python/topic/498/>



# 140:

## Examples

### ceiltrunc

round mathfloor ceiltrunc ◦

```
x = 1.55
y = -1.55

# round to the nearest integer
round(x)      # 2
round(y)      # -2

# the second argument gives how many decimal places to round to (defaults to 0)
round(x, 1)   # 1.6
round(y, 1)   # -1.6

# math is a module so import it first, then use it.
import math

# get the largest integer less than x
math.floor(x) # 1
math.floor(y) # -2

# get the smallest integer greater than x
math.ceil(x)  # 2
math.ceil(y)  # -1

# drop fractional part of x
math.trunc(x) # 1, equivalent to math.floor for positive numbers
math.trunc(y) # -1, equivalent to math.ceil for negative numbers
```

### Python 2.x 2.7

floor ceil truncroundfloat ◦

```
round(1.3) # 1.0
```

round ◦

```
round(0.5) # 1.0
round(1.5) # 2.0
```

### Python 3.x 3.0

floor ceiltruncIntegralroundIntegral ◦

```
round(1.3)      # 1
round(1.33, 1)  # 1.3
```

round<sup>o</sup> ◦

```
round(0.5) # 0
round(1.5) # 2
```

◦ ◦

```
round(2.675, 2) # 2.67, not 2.68!
```

## PythonC ++Java<sup>o</sup>

```
>>> math.floor(-1.7)
-2.0
>>> -5 // 2
-3
```

math.log(x) ex ◦

```
math.log(math.e) # 1.0
math.log(1) # 0.0
math.log(100) # 4.605170185988092
```

math.log<sub>10</sub> ◦ **1** math.log<sub>10</sub> **1**

```
math.log(1 + 1e-20) # 0.0
math.log1p(1e-20) # 1e-20
```

math.log<sub>10</sub> **10**

```
math.log10(10) # 1.0
```

## Python 2.x 2.3.0

math.log(x, base) math.log(x, base) basexlog(x) / log(base) ◦

```
math.log(100, 10) # 2.0
math.log(27, 3) # 3.0
math.log(1, 10) # 0.0
```

Python 2.6 math.copysign(x, y) x y ◦ float ◦

## Python 2.x 2.6

```
math.copysign(-2, 3) # 2.0
math.copysign(3, -3) # -3.0
math.copysign(4, 14.2) # 4.0
math.copysign(1, -0.0) # -1.0, on a platform which supports signed zero
```

```
math.hypot(2, 4) # Just a shorthand for SquareRoot(2**2 + 4**2)
```

```
# Out: 4.47213595499958
```

/

math

```
math.radians(45)          # Convert 45 degrees to radians
# Out: 0.7853981633974483
```

```
math.degrees(math.asin(1)) # Convert the result of asin to degrees
# Out: 90.0
```

```
# Sine and arc sine
math.sin(math.pi / 2)
# Out: 1.0
math.sin(math.radians(90)) # Sine of 90 degrees
# Out: 1.0
```

```
math.asin(1)
# Out: 1.5707963267948966 # "= pi / 2"
math.asin(1) / math.pi
# Out: 0.5
```

```
# Cosine and arc cosine:
math.cos(math.pi / 2)
# Out: 6.123233995736766e-17
# Almost zero but not exactly because "pi" is a float with limited precision!
```

```
math.acos(1)
# Out: 0.0
```

```
# Tangent and arc tangent:
math.tan(math.pi/2)
# Out: 1.633123935319537e+16
# Very large but not exactly "Inf" because "pi" is a float with limited precision
```

## Python 3.x 3.5

```
math.atan(math.inf)
# Out: 1.5707963267948966 # This is just "pi / 2"
```

```
math.atan(float('inf'))
# Out: 1.5707963267948966 # This is just "pi / 2"
```

math.atanmath.atan2

```
math.atan2(1, 2) # Equivalent to "math.atan(1/2)"
# Out: 0.4636476090008061 # ≈ 26.57 degrees, 1st quadrant
```

```
math.atan2(-1, -2) # Not equal to "math.atan(-1/-2)" == "math.atan(1/2)"
# Out: -2.677945044588987 # ≈ -153.43 degrees (or 206.57 degrees), 3rd quadrant
```

```
math.atan2(1, 0) # math.atan(1/0) would raise ZeroDivisionError
# Out: 1.5707963267948966 # This is just "pi / 2"
```

```

# Hyperbolic sine function
math.sinh(math.pi) # = 11.548739357257746
math.asinh(1)      # = 0.8813735870195429

# Hyperbolic cosine function
math.cosh(math.pi) # = 11.591953275521519
math.acosh(1)      # = 0.0

# Hyperbolic tangent function
math.tanh(math.pi) # = 0.99627207622075
math.atanh(0.5)    # = 0.5493061443340549

```

math°

- math.pi - pi
- math.e - e

```

>>> from math import pi, e
>>> pi
3.141592653589793
>>> e
2.718281828459045
>>>

```

Python 3.5NaN<sup>°</sup>. float()<sup>°</sup>

## Python 3.x 3.5

```

math.inf == float('inf')
# Out: True

-math.inf == float('-inf')
# Out: True

# NaN never compares equal to anything, even itself
math.nan == float('nan')
# Out: False

```

Python<sup>°</sup>“j”“J”<sup>°</sup>.

```

1j          # Equivalent to the square root of -1.
1j * 1j     # = (-1+0j)

```

NaN<sup>°</sup>“”

PythonNaN<sup>°</sup>“”

```

pos_inf = float('inf')    # positive infinity
neg_inf = float('-inf')   # negative infinity
not_a_num = float('nan')  # NaN ("not a number")

```

Python 3.5<sup>°</sup>math.infmath.nan

## Python 3.x 3.5

```
pos_inf = math.inf
neg_inf = -math.inf
not_a_num = math.nan
```

inf-infnan

```
pos_inf, neg_inf, not_a_num
# Out: (inf, -inf, nan)
```

isinf

```
math.isinf(pos_inf)
# Out: True
```

```
math.isinf(neg_inf)
# Out: True
```

```
pos_inf == float('inf')    # or == math.inf in Python 3.5+
# Out: True
```

```
neg_inf == float('-inf')   # or == -math.inf in Python 3.5+
# Out: True
```

```
neg_inf == pos_inf
# Out: False
```

## Python 3.2

### Python 3.x 3.2

```
math.isfinite(pos_inf)
# Out: False
```

```
math.isfinite(0.0)
# Out: True
```

```
import sys
```

```
sys.float_info.max
# Out: 1.7976931348623157e+308 (this is system-dependent)
```

```
pos_inf > sys.float_info.max
# Out: True
```

```
neg_inf < -sys.float_info.max
# Out: True
```

float

```
pos_inf == sys.float_info.max * 1.0000001
# Out: True
```

```
neg_inf == -sys.float_info.max * 1.0000001
```

```
# Out: True
```

ZeroDivisionError◦

```
try:
    x = 1.0 / 0.0
    print(x)
except ZeroDivisionError:
    print("Division by zero")

# Out: Division by zero
```

## NaN

```
-5.0 * pos_inf == neg_inf
# Out: True

-5.0 * neg_inf == pos_inf
# Out: True

pos_inf * neg_inf == neg_inf
# Out: True

0.0 * pos_inf
# Out: nan

0.0 * neg_inf
# Out: nan

pos_inf / pos_inf
# Out: nan
```

## NaN◦ isnan

```
not_a_num == not_a_num
# Out: False

math.isnan(not_a_num)
Out: True
```

## NaN“”

```
not_a_num != 5.0    # or any random value
# Out: True

not_a_num > 5.0    or    not_a_num < 5.0    or    not_a_num == 5.0
# Out: False
```

## NaN◦ -1“NaN”◦

```
5.0 * not_a_num
# Out: nan

float('-nan')
```

```
# Out: nan
```

## Python 3.x 3.5

```
-math.nan  
# Out: nan
```

## NaNInfinityfloatPython 3.5+ math

## Python 3.x 3.5

```
math.inf is math.inf, math.nan is math.nan  
# Out: (True, True)  
  
float('inf') is float('inf'), float('nan') is float('nan')  
# Out: (False, False)
```

## timeit

```
> python -m timeit 'for x in xrange(50000): b = x**3'  
10 loops, best of 3: 51.2 msec per loop  
> python -m timeit 'from math import pow' 'for x in xrange(50000): b = pow(x,3)'  
100 loops, best of 3: 9.15 msec per loop
```

## \*\*math.pow◦ pow

```
> from math import pow  
> pow(5,5)  
3125.0
```

## cmath

cmathmath◦

Python◦ import cmath◦

j J i◦

```
z = 1 + 3j
```

1jj◦

```
1j * 1j  
Out: (-1+0j)  
  
1j ** 1j  
# Out: (0.20787957635076193+0j) # "i to the i" == math.e ** -(math.pi/2)
```

realimag conjugate

```
# real part and imaginary part are both float type
z.real, z.imag
# Out: (1.0, 3.0)

z.conjugate()
# Out: (1-3j)      # z.conjugate() == z.real - z.imag * 1j
```

abscomplex

```
abs(1 + 1j)
# Out: 1.4142135623730951      # square root of 2

complex(1)
# Out: (1+0j)

complex(imag=1)
# Out: (1j)

complex(1, 1)
# Out: (1+1j)
```

complex

```
complex('1+1j')
# Out: (1+1j)

complex('1 + 1j')
# Exception: ValueError: complex() arg is a malformed string
```

sqrt

```
import cmath

cmath.sqrt(-1)
# Out: 1j
```

sqrt◦ math

```
import math

math.sqrt(-1)
# Exception: ValueError: math domain error
```

```
cmath.polar(1 + 1j)
# Out: (1.4142135623730951, 0.7853981633974483)      # == (sqrt(1 + 1), atan2(1, 1))

abs(1 + 1j), cmath.phase(1 + 1j)
# Out: (1.4142135623730951, 0.7853981633974483)      # same as previous calculation

cmath.rect(math.sqrt(2), math.atan(1))
# Out: (1.0000000000000002+1.0000000000000002j)
```

“”。 IEEE 754“”。 Python



```
cmath.phase(complex(-1.0, 0.0))
# Out: 3.141592653589793

cmath.phase(complex(-1.0, -0.0))
# Out: -3.141592653589793
```

cmathmath°

sqrt exp log log10 sin cos tan asin acos atan sinh cosh tanh asinh acosh atanh ° math.atan2  
arctangent°

```
cmath.log(1+1j)
# Out: (0.34657359027997264+0.7853981633974483j)

cmath.exp(1j * cmath.pi)
# Out: (-1+1.2246467991473532e-16j) # e to the i pi == -1, within rounding error
```

pie ° floatcomplex °

```
type(cmath.pi)
# Out: <class 'float'>
```

cmathisinfPython 3.2+isfinite ° “InfinityNaN” ° °

```
cmath.isinf(complex(float('inf'), 0.0))
# Out: True
```

cmathisnan° “InfinityNaN” ° “nan” °

```
cmath.isnan(0.0, float('nan'))
# Out: True
```

math.infmath.nancmathPython 3.5

## Python 3.x 3.5

```
cmath.isinf(complex(0.0, math.inf))
# Out: True

cmath.isnan(complex(math.nan, 0.0))
# Out: True

cmath.inf
# Exception: AttributeError: module 'cmath' has no attribute 'inf'
```

Python 3.5 cmathmathisclose°

## Python 3.x 3.5

```
z = cmath.rect(*cmath.polar(1+1j))

z
```

```
# Out: (1.0000000000000002+1.0000000000000002j)
cmath.isclose(z, 1+1j)
# True
```

<https://riptutorial.com/zh-TW/python/topic/230/>

# 141:

- unpickled\_string = pickle.loadsstring
- unpickled\_string = pickle.loadfile\_object
- pickled\_string = pickle.dumps["'cmplx'{'object'None}]pickle.HIGHEST\_PROTOCOL
- pickle.dump["'cmplx'{'object'None}]file\_objectpickle.HIGHEST\_PROTOCOL
- unjsoned\_string = json.loadsstring
- unjsoned\_string = json.loadfile\_object
- jsoned\_string = json.dumps'a"b"c'[1,2,3]
- json.dump'a"b"c'[1,2,3]file\_object

```
protocol picklecPickle /° pickle.HIGHEST_PROTOCOL °
```

## JSON

- 
- 
- pickle

## JSON

- Pythonic
- °
- Pythonic
- °

## Pickle

- 
- 

## Examples

### JSON

### JSON

*int float boolean string listdict* ° - > [JSON Wiki](#)

### JSON -

```
import json

families = (['John'], ['Mark', 'David', {'name': 'Avraham'}])
```

```

# Dumping it into string
json_families = json.dumps(families)
# [{"John"}, {"Mark", "David", {"name": "Avraham"}}]

# Dumping it to file
with open('families.json', 'w') as json_file:
    json.dump(families, json_file)

# Loading it from string
json_families = json.loads(json_families)

# Loading it from file
with open('families.json', 'r') as json_file:
    json_families = json.load(json_file)

```

## JSONJSON-Module ◦

## Pickle

-

```

# Importing pickle
try:
    import cPickle as pickle # Python 2
except ImportError:
    import pickle # Python 3

# Creating Pythonic object:
class Family(object):
    def __init__(self, names):
        self.sons = names

    def __str__(self):
        return ' '.join(self.sons)

my_family = Family(['John', 'David'])

# Dumping to string
pickle_data = pickle.dumps(my_family, pickle.HIGHEST_PROTOCOL)

# Dumping to file
with open('family.p', 'w') as pickle_file:
    pickle.dump(families, pickle_file, pickle.HIGHEST_PROTOCOL)

# Loading from string
my_family = pickle.loads(pickle_data)

# Loading from file
with open('family.p', 'r') as pickle_file:
    my_family = pickle.load(pickle_file)

```

## PicklePickle◦

pickle◦ ◦

<https://riptutorial.com/zh-TW/python/topic/3347/>

# 142:

Python。 API。 APIPEP 249。

APIPython。 Python。 [PEP-249](#)

## Examples

### MySQLdbMySQL

connect。 ◦

executecommit。 ◦

◦

Dbconnect。 ◦

```
import MySQLdb

class Dbconnect(object):

    def __init__(self):

        self.dbconnection = MySQLdb.connect(host='host_example',
   port=int('port_example'),
   user='user_example',
   passwd='pass_example',
   db='schema_example')

        self.dbcursor = self.dbconnection.cursor()

    def commit_db(self):
        self.dbconnection.commit()

    def close_db(self):
        self.dbcursor.close()
        self.dbconnection.close()
```

◦ **execute**。 ◦

```
db = Dbconnect()
db.dbcursor.execute('SELECT * FROM %s' % 'table_example')
```

◦ ◦

```
db = Dbconnect()
db.callproc('stored_procedure_name', [parameters])
```

◦ ◦

```
results = db.dbcursor.fetchall()
for individual_row in results:
    first_field = individual_row[0]
```

```
for individual_row in db.dbcursor:
    first_field = individual_row[0]
```

```
db.commit_db()
```

```
db.close_db()
```

## SQLite

### SQLite ◦ ◦

```
import sqlite3

conn = sqlite3.connect("users.db")
c = conn.cursor()

c.execute("CREATE TABLE user (name text, age integer)")

c.execute("INSERT INTO user VALUES ('User A', 42)")
c.execute("INSERT INTO user VALUES ('User B', 43)")

conn.commit()

c.execute("SELECT * FROM user")
print(c.fetchall())

conn.close()
```

users.db ◦ **SQL** ◦

```
[(u'User A', 42), (u'User B', 43)]
```

---

# SQLite

## 1. sqlite

```
>>> import sqlite3
```

## 2. Connection ◦ example.db

```
>>> conn = sqlite3.connect('users.db')
```

:memory:RAM

```
>>> conn = sqlite3.connect(':memory:')
```

### 3. Connection Cursor.execute() SQL

```
c = conn.cursor()

# Create table
c.execute('''CREATE TABLE stocks
            (date text, trans text, symbol text, qty real, price real)''')

# Insert a row of data
c.execute("INSERT INTO stocks VALUES ('2006-01-05','BUY','RHAT',100,35.14)")

# Save (commit) the changes
conn.commit()

# We can also close the connection if we are done with it.
# Just be sure any changes have been committed or they will be lost.
conn.close()
```

#### Connection

##### 1. isolation\_level

- DEFERRED IMMEDIATEEXCLUSIVE ◦

##### 2. cursor

SQL◦

##### 3. commit()

- 

##### 4. rollback()

commit()

##### 5. close()

- commit() ◦ commit()close() ◦

##### 6. total\_changes

- 

##### 7. execute executemanyexecutescript

- 

##### 8. row\_factory

- 

```
def dict_factory(cursor, row):
```

```

d = {}
for i, col in enumerate(cursor.description):
    d[col[0]] = row[i]
return d

conn = sqlite3.connect(":memory:")
conn.row_factory = dict_factory

```

## Cursor

### 1. execute(sql[, parameters])

SQL。 SQLSQL。 sqlite3? “qmark style”:name “named style”。

```

import sqlite3
conn = sqlite3.connect(":memory:")
cur = conn.cursor()
cur.execute("create table people (name, age)")

who = "Sophia"
age = 37
# This is the qmark style:
cur.execute("insert into people values (?, ?)",
            (who, age))

# And this is the named style:
cur.execute("select * from people where name=:who and age=:age",
            {"who": who, "age": age}) # the keys correspond to the placeholders in SQL

print(cur.fetchone())

```

%sSQLSQLSQL。

### 2. executemany(sql, seq\_of\_parameters)

sqlSQL。 sqlite3。

```

L = [(1, 'abcd', 'dfj', 300), # A list of tuples to be inserted into the database
      (2, 'cfgd', 'dyfj', 400),
      (3, 'sdd', 'dfjh', 300.50)]

conn = sqlite3.connect("test1.db")
conn.execute("create table if not exists book (id int, name text, author text, price
real)")
conn.executemany("insert into book values (?, ?, ?, ?)", L)

for row in conn.execute("select * from book"):
    print(row)

```

executemany。。

```

import sqlite3

class IterChars:
    def __init__(self):
        self.count = ord('a')

```



```

def __iter__(self):
    return self

def __next__(self):          # (use next(self) for Python 2)
    if self.count > ord('z'):
        raise StopIteration
    self.count += 1
    return (chr(self.count - 1),)

conn = sqlite3.connect("abc.db")
cur = conn.cursor()
cur.execute("create table characters(c)")

theIter = IterChars()
cur.executemany("insert into characters(c) values (?)", theIter)

rows = cur.execute("select c from characters")
for row in rows:
    print(row[0]),

```

### 3. executescrpt(sql\_script)

**SQL**◦ COMMIT**SQL**◦

sql\_scriptstrbytes◦

```

import sqlite3
conn = sqlite3.connect(":memory:")
cur = conn.cursor()
cur.executescript("""
    create table person(
        firstname,
        lastname,
        age
    );

    create table book(
        title,
        author,
        published
    );

    insert into book(title, author, published)
    values (
        'Dirk Gently''s Holistic Detective Agency',
        'Douglas Adams',
        1987
    );
""")

```

**SQL**SELECT◦ SELECTfetchone() fetchall()◦

```

import sqlite3
stocks = [('2006-01-05', 'BUY', 'RHAT', 100, 35.14),
          ('2006-03-28', 'BUY', 'IBM', 1000, 45.0),
          ('2006-04-06', 'SELL', 'IBM', 500, 53.0),
          ('2006-04-05', 'BUY', 'MSFT', 1000, 72.0)]

```

```

conn = sqlite3.connect(":memory:")
conn.execute("create table stocks (date text, buysell text, symb text, amount int, price
real)")
conn.executemany("insert into stocks values (?, ?, ?, ?, ?)", stocks)
cur = conn.cursor()

for row in cur.execute('SELECT * FROM stocks ORDER BY price'):
    print(row)

# Output:
# ('2006-01-05', 'BUY', 'RHAT', 100, 35.14)
# ('2006-03-28', 'BUY', 'IBM', 1000, 45.0)
# ('2006-04-06', 'SELL', 'IBM', 500, 53.0)
# ('2006-04-05', 'BUY', 'MSFT', 1000, 72.0)

```

#### 4. fetchone()

None.

```

cur.execute('SELECT * FROM stocks ORDER BY price')
i = cur.fetchone()
while(i):
    print(i)
    i = cur.fetchone()

# Output:
# ('2006-01-05', 'BUY', 'RHAT', 100, 35.14)
# ('2006-03-28', 'BUY', 'IBM', 1000, 45.0)
# ('2006-04-06', 'SELL', 'IBM', 500, 53.0)
# ('2006-04-05', 'BUY', 'MSFT', 1000, 72.0)

```

#### 5. fetchmany(size=cursor.arraysize)

size. sizefetchmany. .

```

cur.execute('SELECT * FROM stocks ORDER BY price')
print(cur.fetchmany(2))

# Output:
# [('2006-01-05', 'BUY', 'RHAT', 100, 35.14), ('2006-03-28', 'BUY', 'IBM', 1000, 45.0)]

```

#### 6. fetchall()

.

```

cur.execute('SELECT * FROM stocks ORDER BY price')
print(cur.fetchall())

# Output:
# [('2006-01-05', 'BUY', 'RHAT', 100, 35.14), ('2006-03-28', 'BUY', 'IBM', 1000, 45.0),
('2006-04-06', 'SELL', 'IBM', 500, 53.0), ('2006-04-05', 'BUY', 'MSFT', 1000, 72.0)]

```

## SQLitePython

SQLiteNULLINTEGERREALTEXTBLOB.

## SQLPython。

|       |     |                 |
|-------|-----|-----------------|
| None  | <-> | NULL            |
| int   | <-> | INTEGER/INT     |
| float | <-> | REAL/FLOAT      |
| str   | <-> | TEXT/VARCHAR(n) |
| bytes | <-> | BLOB            |

## psycopg2PostgreSQL

### psycopg2PostgreSQL。 PostgreSQL。

#### Python DB API 2.0

```
import psycopg2

# Establish a connection to the database.
# Replace parameter values with database credentials.
conn = psycopg2.connect(database="testpython",
                        user="postgres",
                        host="localhost",
                        password="abc123",
                        port="5432")

# Create a cursor. The cursor allows you to execute database queries.
cur = conn.cursor()

# Create a table. Initialise the table name, the column names and data type.
cur.execute("""CREATE TABLE FRUITS (
                id            INT ,
                fruit_name    TEXT,
                color          TEXT,
                price          REAL
            ) """)
conn.commit()
conn.close()
```

```
# After creating the table as shown above, insert values into it.
cur.execute("""INSERT INTO FRUITS (id, fruit_name, color, price)
            VALUES (1, 'Apples', 'green', 1.00) """)

cur.execute("""INSERT INTO FRUITS (id, fruit_name, color, price)
            VALUES (1, 'Bananas', 'yellow', 0.80) """)
```

```
# Set up a query and execute it
cur.execute("""SELECT id, fruit_name, color, price
            FROM fruits """)

# Fetch the data
rows = cur.fetchall()

# Do stuff with the data
for row in rows:
```

```
print "ID = {}".format(row[0])
print "FRUIT NAME = {}".format(row[1])
print ("COLOR = {}".format(row[2]))
print ("PRICE = {}".format(row[3]))
```

```
ID = 1
NAME = Apples
COLOR = green
PRICE = 1.0
```

```
ID = 2
NAME = Bananas
COLOR = yellow
PRICE = 0.8
```

## **psycopg2** :)

## **Oracle**

- **cx\_Oracle** -
- **Oracle** - [Windows x64](#) [Linux x64](#)

- **cx\_Oracle**

```
sudo rpm -i <YOUR_PACKAGE_FILENAME>
```

- **Oracle**

```
ORACLE_HOME=<PATH_TO_INSTANTCLIENT>
PATH=$ORACLE_HOME:$PATH
LD_LIBRARY_PATH=<PATH_TO_INSTANTCLIENT>:$LD_LIBRARY_PATH
```

```
import cx_Oracle

class OraExec(object):
    _db_connection = None
    _db_cur = None

    def __init__(self):
        self._db_connection =
            cx_Oracle.connect('<USERNAME>/<PASSWORD>@<HOSTNAME>:<PORT>/<SERVICE_NAME>')
        self._db_cur = self._db_connection.cursor()
```

```
ver = con.version.split(".")
print ver
```

[**'12' '1' '0' '2' '0'**]

## **SELECT**

```
_db_cur.execute("select * from employees order by emp_id")
for result in _db_cur:
    print result
```

## Python

```
10'SYSADMIN'IT-INFRA'7
```

```
23'''6
```

### INSERT

```
_db_cur.execute("insert into employees(emp_id, title, dept, grade)
                 values (31, 'MTS', 'ENGINEERING', 7)
_db_connection.commit()
```

Oracle//commit ° °

### INSERT

° SQL °

```
rows = [ (1, "First" ),
          (2, "Second" ),
          (3, "Third" ) ]
_db_cur.bindarraysize = 3
_db_cur.setinputsizes(int, 10)
_db_cur.executemany("insert into mytab(id, data) values (:1, :2)", rows)
_db_connection.commit()
```

```
_db_connection.close()
```

close ° °

PEP 249 connect() connect() Connection ° ° °

```
import MyDBAPI

con = MyDBAPI.connect(*database_dependent_args)
```

### 1

```
con.close()
```

° Connection.\_\_del\_\_ Connection.\_\_del\_\_ ° °

### 2

```
con.commit()
```

°

### 3

```
con.rollback()
```

◦ ◦

## 4

```
cur = con.cursor()
```

Cursor◦ ◦

## sqlalchemy

### sqlalchemy

```
from sqlalchemy import create_engine
from sqlalchemy.engine.url import URL

url = URL(drivertype='mysql',
          username='user',
          password='passwd',
          host='host',
          database='db')

engine = create_engine(url) # sqlalchemy engine
```

### pandasmysql

```
import pandas as pd

con = engine.connect()
dataframe = pd.read_sql(sql=query, con=con)
```

<https://riptutorial.com/zh-TW/python/topic/4240/>

# 143:

## Examples

### iterablecollections.Counter

```
from collections import Counter

c = Counter(["a", "b", "c", "d", "a", "b", "a", "c", "d"])
c
# Out: Counter({'a': 3, 'b': 2, 'c': 2, 'd': 2})
c["a"]
# Out: 3

c[7]      # not in the list (7 occurred 0 times!)
# Out: 0
```

collections.Counter **iterable**

dictcollections.Mapping **likeCounter**

```
Counter({"e": 2})
# Out: Counter({"e": 2})

Counter({"e": "e"})      # warning Counter does not verify the values are int
# Out: Counter({"e": "e"})
```

### -collections.Counter.most\_common

collections.CounterMapping

```
from collections import Counter
adict = {'a': 5, 'b': 3, 'c': 5, 'd': 2, 'e': 2, 'q': 5}
Counter(adict.values())
# Out: Counter({2: 2, 3: 1, 5: 3})
```

most\_common **-method**

```
# Sorting them from most-common to least-common value:
Counter(adict.values()).most_common()
# Out: [(5, 3), (2, 2), (3, 1)]

# Getting the most common value
Counter(adict.values()).most_common(1)
# Out: [(5, 3)]

# Getting the two most common values
Counter(adict.values()).most_common(2)
# Out: [(5, 3), (2, 2)]
```

## list.counttuple.count

```
alist = [1, 2, 3, 4, 1, 2, 1, 3, 4]
alist.count(1)
# Out: 3

atuple = ('bear', 'weasel', 'bear', 'frog')
atuple.count('bear')
# Out: 2
atuple.count('fox')
# Out: 0
```

## str.count

```
astring = 'thisisashorttext'
astring.count('t')
# Out: 4
```

```
astring.count('th')
# Out: 1
astring.count('is')
# Out: 2
astring.count('text')
# Out: 1
```

collections.Counter

```
from collections import Counter
Counter(astring)
# Out: Counter({'a': 1, 'e': 1, 'h': 2, 'i': 2, 'o': 1, 'r': 1, 's': 3, 't': 4, 'x': 1})
```

## numpy

### numpy.

```
>>> import numpy as np
>>> a=np.array([0,3,4,3,5,4,7])
>>> print np.sum(a==3)
2
```

### 1. . dtype.

### numpy. bincount. bincount1d.

```
>>> unique,counts=np.unique(a,return_counts=True)
>>> print unique,counts # counts[i] is equal to occurrences of unique[i] in a
[0 3 4 5 7] [1 2 2 1 1]
>>> bin_count=np.bincount(a)
>>> print bin_count # bin_count[i] is equal to occurrences of i in a
[1 0 0 2 2 1 0 1]
```



numpynumpy。

<https://riptutorial.com/zh-TW/python/topic/476/>

# 144:

Python“CJava。。

|   |          |
|---|----------|
| b | 1        |
| B | 1        |
| c | 1        |
| u | 2unicode |
| h | 2        |
| H | 2        |
| i | 2        |
| I | 2        |
| w | 4unicode |
| l | 4        |
| L | 4        |
| f | 4        |
| d | 8        |

## Examples

。 Python。

pythonpython。 Python。

PythonPython。

pythonarray。 。 pythonarray

```
from array import *
```

array。

```
arrayIdentifierName = array(typecode, [Initializers])
```

arrayIdentifierName typecode **python** Initializers

Typecodes

**python**

```
my_array = array('i', [1,2,3,4])
```

typecode **i**

**5**

```
from array import *
my_array = array('i', [1,2,3,4,5])
for i in my_array:
    print(i)
# 1
# 2
# 3
# 4
# 5
```

**Python**

```
my_array = array('i', [1,2,3,4,5])
print(my_array[1])
# 2
print(my_array[2])
# 3
print(my_array[0])
# 1
```

**append**

```
my_array = array('i', [1,2,3,4,5])
my_array.append(6)
# array('i', [1, 2, 3, 4, 5, 6])
```

**6**

**insert**

insert()

```
my_array = array('i', [1,2,3,4,5])
my_array.insert(0,0)
#array('i', [0, 1, 2, 3, 4, 5])
```

**00**

## extendpython

extend()python

```
my_array = array('i', [1,2,3,4,5])
my_extnd_array = array('i', [7,8,9,10])
my_array.extend(my_extnd_array)
# array('i', [1, 2, 3, 4, 5, 7, 8, 9, 10])
```

my\_arraymy\_extnd\_array

## fromlist

```
my_array = array('i', [1,2,3,4,5])
c=[11,12,13]
my_array.fromlist(c)
# array('i', [1, 2, 3, 4, 5, 11, 12, 13])
```

11,1213cmy\_array

## remove

```
my_array = array('i', [1,2,3,4,5])
my_array.remove(4)
# array('i', [1, 2, 3, 5])
```

4

## pop

pop

```
my_array = array('i', [1,2,3,4,5])
my_array.pop()
# array('i', [1, 2, 3, 4])
```

5

## index

index()

```
my_array = array('i', [1,2,3,4,5])
print(my_array.index(5))
# 5
my_array = array('i', [1,2,3,3,5])
print(my_array.index(3))
# 3
```

## reversepython

reverse() ◦

```
my_array = array('i', [1,2,3,4,5])
my_array.reverse()
# array('i', [5, 4, 3, 2, 1])
```

## buffer\_info

◦

```
my_array = array('i', [1,2,3,4,5])
my_array.buffer_info()
(33881712, 5)
```

## count

count() ◦ 3◦

```
my_array = array('i', [1,2,3,3,5])
my_array.count(3)
# 2
```

## tostring

tostring() ◦

```
my_char_array = array('c', ['g','e','e','k'])
# array('c', 'geek')
print(my_char_array.tostring())
# geek
```

## tolistpython

Python listtolist() ◦

```
my_array = array('i', [1,2,3,4,5])
c = my_array.tolist()
# [1, 2, 3, 4, 5]
```

## fromstringchar

fromstring()

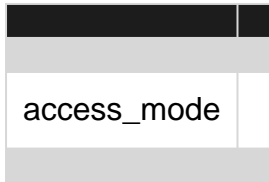
```
my_char_array = array('c', ['g','e','e','k'])
my_char_array.fromstring("stuff")
print(my_char_array)
#array('c', 'geekstuff')
```



# 145: I / O.

Python◦ Python/◦ Python◦

- file\_object = open(filename [access\_mode] [buffering])



Python◦ open() encoding◦ ◦

linuxutf-8MACWindows◦

```
import sys
sys.getdefaultencoding()
```

python◦

◦

```
with open('somefile.txt', 'r', encoding='UTF-8') as f:
    for line in f:
        print(line)
```

## Examples

mode◦

- 'r' - ◦ ◦ ◦ ◦ ◦
- 'w' - ◦ ◦ ◦
- 'a' - ◦ ◦ ◦ ◦
- 'rb' - ◦ ◦ r◦ ◦
- 'r+' - ◦ ◦ rw◦
- 'rb+' - ◦ ◦ r+
- 'wb' - ◦ ◦ w◦
- 'w+' - ◦ ◦ r+◦ ◦

- 'wb+' - ◦ w+◦
- 'ab' - ◦ a◦
- 'a+' - ◦ w+◦ ◦
- 'ab+' - ◦ a+◦

```
with open(filename, 'r') as f:
    f.read()
with open(filename, 'w') as f:
    f.write(filedata)
with open(filename, 'a') as f:
    f.write('\n' + newdata)
```

|  | [R | R + | w ^ | W + |   | A + |
|--|----|-----|-----|-----|---|-----|
|  | ✓  | ✓   | x   | ✓   | x | ✓   |
|  | x  | ✓   | ✓   | ✓   | ✓ | ✓   |
|  | x  | x   | ✓   | ✓   | ✓ | ✓   |
|  | x  | x   | ✓   | ✓   | x | x   |

### Python 3exclusive creation◦

- 'x' - FileNotFoundError
- 'xb' - ◦ x◦
- 'x+' - ◦ w+◦ FileNotFoundError ◦
- 'xb+' - ◦ x+

|  | X | X + |
|--|---|-----|
|  | x | ✓   |
|  | ✓ | ✓   |
|  | ✓ | ✓   |
|  | x | x   |

pythonic

Python 3.x 3.3



```

try:
    with open("fname", "r") as fout:
        # Work with your open file
except FileExistsError:
    # Your error handling goes here

```

## Python 2

### Python 2.x 2.0

```

import os.path
if os.path.isfile(fname):
    with open("fname", "w") as fout:
        # Work with your open file
else:
    # Your error handling goes here

```

```

with open('myfile.txt', 'r') as fp:
    for line in fp:
        print(line)

```

readline()◦

```

with open('myfile.txt', 'r') as fp:
    while True:
        cur_line = fp.readline()
        # If the result is an empty string
        if cur_line == '':
            # We have reached the end of the file
            break
        print(cur_line)

```

forreadline◦

readlines()

```

with open("myfile.txt", "r") as fp:
    lines = fp.readlines()
    for i in range(len(lines)):
        print("Line " + str(i) + ": " + line)

```

0

1

i/owith◦ ◦

```

with open('myfile.txt') as in_file:
    content = in_file.read()

print(content)

```

withclose

```
in_file = open('myfile.txt', 'r')
content = in_file.read()
print(content)
in_file.close()
```

with

```
in_file = open('myfile.txt', 'r')
raise Exception("oops")
in_file.close() # This will never be called
```

```
with open('myfile.txt', 'w') as f:
    f.write("Line 1")
    f.write("Line 2")
    f.write("Line 3")
    f.write("Line 4")
```

myfile.txt

12Line 3Line 4

## Python

```
with open('myfile.txt', 'w') as f:
    f.write("Line 1\n")
    f.write("Line 2\n")
    f.write("Line 3\n")
    f.write("Line 4\n")
```

1  
2  
3  
4

os.linesep;\n°

encodingopen

```
with open('my_file.txt', 'w', encoding='utf-8') as f:
    f.write('utf-8 text')
```

print° Python 2Python 3°

## Python 3.x 3.0

```
with open('fred.txt', 'w') as outfile:
    s = "I'm Not Dead Yet!"
    print(s) # writes to stdout
    print(s, file = outfile) # writes to outfile

#Note: it is possible to specify the file parameter AND write to the screen
#by making sure file ends up with a None value either directly or via a variable
myfile = None
```

```
print(s, file = myfile) # writes to stdout
print(s, file = None)   # writes to stdout
```

## Python 2

### Python 2.x 2.0

```
outfile = open('fred.txt', 'w')
s = "I'm Not Dead Yet!"
print s    # writes to stdout
print >> outfile, s    # writes to outfile
```

### writeprint.

```
with open(input_file, 'r') as in_file, open(output_file, 'w') as out_file:
    for line in in_file:
        out_file.write(line)
```

- shutil

```
import shutil
shutil.copyfile(src, dst)
```

### EAFP<sup>try</sup>.

```
import errno

try:
    with open(path) as f:
        # File exists
except IOError as e:
    # Raise the exception if it is not ENOENT (No such file or directory)
    if e.errno != errno.ENOENT:
        raise
    # No such file or directory
```

o

- os

```
import os
os.path.isfile('/path/to/some/file.txt')
```

## Python 3.x 3.4

- pathlib

```
import pathlib
path = pathlib.Path('/path/to/some/file.txt')
if path.is_file():
    ...
```

## EAFP

```
import os
path = "/home/myFiles/directory1"

if os.path.exists(path):
    ## Do stuff
```

```
import shutil
source='//192.168.1.2/Daily Reports'
destination='D:\\Reports\\Today'
shutil.copytree(source, destination)
```

◦

## os.walk

```
import os
for root, folders, files in os.walk(root_dir):
    for filename in files:
        print root, filename
```

root\_dir“。 ”。

## Python 3.x 3.5

### os.scandir

```
for entry in os.scandir(path):
    if not entry.name.startswith('.') and entry.is_file():
        print(entry.name)
```

### itertools

```
import itertools

with open('myfile.txt', 'r') as f:
    for line in itertools.islice(f, 12, 30):
        # do something here
```

## 1320python0。 10

next()◦

readline()

## mmap

mmap◦ ◦

```
import mmap
```

```

with open('filename.ext', 'r') as fd:
    # 0: map the whole file
    mm = mmap.mmap(fd.fileno(), 0)

    # print characters at indices 5 through 10
    print mm[5:10]

    # print the line starting from mm's current position
    print mm.readline()

    # write a character to the 5th index
    mm[5] = 'a'

    # return mm's position to the beginning of the file
    mm.seek(0)

    # close the mmap object
    mm.close()

```

```

import fileinput

replacements = {'Search1': 'Replace1',
                'Search2': 'Replace2'}

for line in fileinput.input('filename.txt', inplace=True):
    for search_for in replacements:
        replace_with = replacements[search_for]
        line = line.replace(search_for, replace_with)
    print(line, end='')

```

```

>>> import os
>>> os.stat(path_to_file).st_size == 0

```

```

>>> import os
>>> os.path.getsize(path_to_file) > 0

```

o

```

import os
def is_empty_file(fpath):
    return os.path.isfile(fpath) and os.path.getsize(fpath) > 0

```

boolo

I / O. <https://riptutorial.com/zh-TW/python/topic/267/i---o->

# 146:

## Examples

PythonPython 23print with ()

```
class Parent(object):
    def introduce(self):
        print("Hello!")

    def print_name(self):
        print("Parent")

class Child(Parent):
    def print_name(self):
        print("Child")

p = Parent()
c = Child()

p.introduce()
p.print_name()

c.introduce()
c.print_name()

$ python basic_override.py
Hello!
Parent
Hello!
Child
```

ChildParent◦ ◦ introduceChildParentChild◦

Childprint\_name◦ c.print\_name() "Parent" ◦ ChildParentprint\_namec.print\_name() "Child"◦

<https://riptutorial.com/zh-TW/python/topic/3131/>

# 147:

Python◦

## Examples

datetimePython 3.2+%z◦

UTC+HHMM-HHMM -HHMM -HHMM◦

Python 3.x 3.2

```
import datetime
dt = datetime.datetime.strptime("2016-04-15T08:27:18-0500", "%Y-%m-%dT%H:%M:%S%z")
```

Pythondateutildatetime◦

```
import dateutil.parser
dt = dateutil.parser.parse("2016-04-15T08:27:18-0500")
```

dtdatetime

```
datetime.datetime(2016, 4, 15, 8, 27, 18, tzinfo=tzoffset(None, -18000))
```

◦ ◦ [timedelta](#)

```
import datetime

today = datetime.date.today()
print('Today:', today)

yesterday = today - datetime.timedelta(days=1)
print('Yesterday:', yesterday)

tomorrow = today + datetime.timedelta(days=1)
print('Tomorrow:', tomorrow)

print('Time between tomorrow and yesterday:', tomorrow - yesterday)
```

```
Today: 2016-04-15
Yesterday: 2016-04-14
Tomorrow: 2016-04-16
Difference between tomorrow and yesterday: 2 days, 0:00:00
```

datetime - ◦

```
import datetime
```

```

# Date object
today = datetime.date.today()
new_year = datetime.date(2017, 01, 01) #datetime.date(2017, 1, 1)

# Time object
noon = datetime.time(12, 0, 0) #datetime.time(12, 0)

# Current datetime
now = datetime.datetime.now()

# Datetime object
millenium_turn = datetime.datetime(2000, 1, 1, 0, 0, 0) #datetime.datetime(2000, 1, 1, 0, 0)

```

## TypeError.

```

# subtraction of noon from today
noon-today
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for -: 'datetime.time' and 'datetime.date'
However, it is straightforward to convert between types.

# Do this instead
print('Time since the millenium at midnight: ',
      datetime.datetime(today.year, today.month, today.day) - millenium_turn)

# Or this
print('Time since the millenium at noon: ',
      datetime.datetime.combine(today, noon) - millenium_turn)

```

### ◦ datetimetimedelta

```

import datetime

# The size of each step in days
day_delta = datetime.timedelta(days=1)

start_date = datetime.date.today()
end_date = start_date + 7*day_delta

for i in range((end_date - start_date).days):
    print(start_date + i*day_delta)

```

```

2016-07-21
2016-07-22
2016-07-23
2016-07-24
2016-07-25
2016-07-26
2016-07-27

```

### dateutil ""

### CST - tzinfo



```

from dateutil import tz
from dateutil.parser import parse

ET = tz.gettz('US/Eastern')
CT = tz.gettz('US/Central')
MT = tz.gettz('US/Mountain')
PT = tz.gettz('US/Pacific')

us_tzinfos = {'CST': CT, 'CDT': CT,
              'EST': ET, 'EDT': ET,
              'MST': MT, 'MDT': MT,
              'PST': PT, 'PDT': PT}

dt_est = parse('2014-01-02 04:00:00 EST', tzinfos=us_tzinfos)
dt_pst = parse('2016-03-11 16:00:00 PST', tzinfos=us_tzinfos)

```

```

dt_est
# datetime.datetime(2014, 1, 2, 4, 0, tzinfo=tzfile('/usr/share/zoneinfo/US/Eastern'))
dt_pst
# datetime.datetime(2016, 3, 11, 16, 0, tzinfo=tzfile('/usr/share/zoneinfo/US/Pacific'))

```

pytz

```

from dateutil.parser import parse
import pytz

EST = pytz.timezone('America/New_York')
dt = parse('2014-02-03 09:17:00 EST', tzinfos={'EST': EST})

```

pytzpytz

```

dt.tzinfo # Will be in Local Mean Time!
# <DstTzInfo 'America/New_York' LMT-1 day, 19:04:00 STD>

```

localize

```

dt_fixed = dt.tzinfo.localize(dt.replace(tzinfo=None))
dt_fixed.tzinfo # Now it's EST.
# <DstTzInfo 'America/New_York' EST-1 day, 19:00:00 STD>

```

datetime.tzinfo.tzinfo **UTC**

**UTC** Python 3.2+ datetime.timezone tzinfo.time\_delta name

**Python 3.x 3.2**

```

from datetime import datetime, timedelta, timezone
JST = timezone(timedelta(hours=+9))

dt = datetime(2015, 1, 1, 12, 0, 0, tzinfo=JST)
print(dt)
# 2015-01-01 12:00:00+09:00

print(dt.tzname())
# UTC+09:00

```

```
dt = datetime(2015, 1, 1, 12, 0, 0, tzinfo=timezone(timedelta(hours=9), 'JST'))
print(dt.tzname)
# 'JST'
```

**3.2 Python `dateutil`** ◦ `dateutil.tz.tzoffset` **2.5.3** `dateutil.tz.tzoffset(tzname, offset)` `offset`

**Python 3.x 3.2**

**Python 2.x 2.7**

```
from datetime import datetime, timedelta
from dateutil import tz

JST = tz.tzoffset('JST', 9 * 3600) # 3600 seconds per hour
dt = datetime(2015, 1, 1, 12, 0, tzinfo=JST)
print(dt)
# 2015-01-01 12:00:00+09:00
print(dt.tzname)
# 'JST'
```

`python` ◦ `pytz` `dateutil` ◦

`dateutil` `tz` ◦ `tz.gettz()` `datetime`

```
from datetime import datetime
from dateutil import tz
local = tz.gettz() # Local time
PT = tz.gettz('US/Pacific') # Pacific time

dt_l = datetime(2015, 1, 1, 12, tzinfo=local) # I am in EST
dt_pst = datetime(2015, 1, 1, 12, tzinfo=PT)
dt_pdt = datetime(2015, 7, 1, 12, tzinfo=PT) # DST is handled automatically
print(dt_l)
# 2015-01-01 12:00:00-05:00
print(dt_pst)
# 2015-01-01 12:00:00-08:00
print(dt_pdt)
# 2015-07-01 12:00:00-07:00
```

**2.5.3** `dateutil` ◦ `dateutil` `2015-11-01 1:30 EDT-4` ◦

`pytz` `pytz` ◦ `localize` `pytz`

```
from datetime import datetime, timedelta
import pytz

PT = pytz.timezone('US/Pacific')
dt_pst = PT.localize(datetime(2015, 1, 1, 12))
dt_pdt = PT.localize(datetime(2015, 11, 1, 0, 30))
print(dt_pst)
# 2015-01-01 12:00:00-08:00
print(dt_pdt)
# 2015-11-01 00:30:00-07:00
```

`pytz` **UTC** `pytz.normalize()`

```
dt_new = dt_pdt + timedelta(hours=3) # This should be 2:30 AM PST
print(dt_new)
# 2015-11-01 03:30:00-07:00
dt_corrected = PT.normalize(dt_new)
print(dt_corrected)
# 2015-11-01 02:30:00-08:00
```

`dateutil`。

```
from dateutil.parser import parse

dt = parse("Today is January 1, 2047 at 8:21:00AM", fuzzy=True)
print(dt)
```

`dt.datetime.datetime(2047, 1, 1, 8, 21)`。

。

```
from datetime import datetime
from dateutil import tz

utc = tz.tzutc()
local = tz.tzlocal()

utc_now = datetime.utcnow()
utc_now # Not timezone-aware.

utc_now = utc_now.replace(tzinfo=utc)
utc_now # Timezone-aware.

local_now = utc_now.astimezone(local)
local_now # Converted to local time.
```

## ISO 8601

PythonISO 8601。 `strptime`。 `datetime`ISO 86016

```
str(datetime.datetime(2016, 7, 22, 9, 25, 59, 555555))
# '2016-07-22 09:25:59.555555'
```

0

```
str(datetime.datetime(2016, 7, 22, 9, 25, 59, 0))
# '2016-07-22 09:25:59'
```

`strptime`。 `strptime` does not support at all parsing minute timezones that have a in it, thus `2016-07-22 092559 + 0300` can be parsed, but the standard format `2016-07-22 09:25:59 +0300``。

`iso8601`ISO 8601。

T

```

import iso8601
iso8601.parse_date('2016-07-22 09:25:59')
# datetime.datetime(2016, 7, 22, 9, 25, 59, tzinfo=<iso8601.Utc>)
iso8601.parse_date('2016-07-22 09:25:59+03:00')
# datetime.datetime(2016, 7, 22, 9, 25, 59, tzinfo=<FixedOffset '+03:00' ...>)
iso8601.parse_date('2016-07-22 09:25:59Z')
# datetime.datetime(2016, 7, 22, 9, 25, 59, tzinfo=<iso8601.Utc>)
iso8601.parse_date('2016-07-22T09:25:59.000111+03:00')
# datetime.datetime(2016, 7, 22, 9, 25, 59, 111, tzinfo=<FixedOffset '+03:00' ...>)

```

iso8601.parse\_date**UTC**◦ default\_zone◦ None

```

iso8601.parse_date('2016-07-22T09:25:59', default_timezone=None)
# datetime.datetime(2016, 7, 22, 9, 25, 59)
iso8601.parse_date('2016-07-22T09:25:59Z', default_timezone=None)
# datetime.datetime(2016, 7, 22, 9, 25, 59, tzinfo=<iso8601.Utc>)

```

datetime**POSIX** timestamp**ITC** datetime◦

**197011**◦

```

import time
from datetime import datetime
seconds_since_epoch=time.time() #1469182681.709

utc_date=datetime.utcnowfromtimestamp(seconds_since_epoch) #datetime.datetime(2016, 7, 22, 10,
18, 1, 709000)

```

calendar

```

import calendar
from datetime import date

def monthdelta(date, delta):
    m, y = (date.month+delta) % 12, date.year + ((date.month)+delta-1) // 12
    if not m: m = 12
    d = min(date.day, calendar.monthrange(y, m)[1])
    return date.replace(day=d,month=m, year=y)

next_month = monthdelta(date.today(), 1) #datetime.date(2016, 10, 23)

```

dateutils

```

import datetime
import dateutil.relativedelta

d = datetime.datetime.strptime("2013-03-31", "%Y-%m-%d")
d2 = d - dateutil.relativedelta.relativedelta(months=1) #datetime.datetime(2013, 2, 28, 0, 0)

```

timedelta

```

from datetime import datetime, timedelta
now = datetime.now()
then = datetime(2016, 5, 23) # datetime.datetime(2016, 05, 23, 0, 0, 0)

```

datetime

```
delta = now-then
```

deltatimedelta

```
print(delta.days)
# 60
print(delta.seconds)
# 40826
```

n

n

```
def get_n_days_after_date(date_format="%d %B %Y", add_days=120):

    date_n_days_after = datetime.datetime.now() + timedelta(days=add_days)
    return date_n_days_after.strftime(date_format)
```

```
def get_n_days_before_date(self, date_format="%d %B %Y", days_before=120):

    date_n_days_ago = datetime.datetime.now() - timedelta(days=days_before)
    return date_n_days_ago.strftime(date_format)
```

## ISO 8601

---

```
from datetime import datetime

datetime.now().isoformat()
# Out: '2016-07-31T23:08:20.886783'
```

```
from datetime import datetime
from dateutil.tz import tzlocal

datetime.now(tzlocal()).isoformat()
# Out: '2016-07-31T23:09:43.535074-07:00'
```

```
from datetime import datetime
from dateutil.tz import tzlocal

datetime.now(tzlocal()).replace(microsecond=0).isoformat()
# Out: '2016-07-31T23:10:30-07:00'
```

[ISO 8601](#) | [ISO 8601](#) ◦

<https://riptutorial.com/zh-TW/python/topic/484/>

# 148:

## Examples

```
from datetime import datetime

a = datetime(2016,10,06,0,0,0)
b = datetime(2016,10,01,23,59,59)

a-b
# datetime.timedelta(4, 1)

(a-b).days
# 4
(a-b).total_seconds()
# 518399.0
```

### datetime

C。

```
from datetime import datetime
datetime_string = 'Oct 1 2016, 00:00:00'
datetime_string_format = '%b %d %Y, %H:%M:%S'
datetime.strptime(datetime_string, datetime_string_format)
# datetime.datetime(2016, 10, 1, 0, 0)
```

### datetime

C。

```
from datetime import datetime
datetime_for_string = datetime(2016,10,1,0,0)
datetime_string_format = '%b %d %Y, %H:%M:%S'
datetime.strftime(datetime_for_string,datetime_string_format)
# Oct 01 2016, 00:00:00
```

<https://riptutorial.com/zh-TW/python/topic/7284/>

---

# 149:

## Examples

```
from PIL import Image

im = Image.open("Image.bmp")
```

## JPEG

```
from __future__ import print_function
import os, sys
from PIL import Image

for infile in sys.argv[1:]:
    f, e = os.path.splitext(infile)
    outfile = f + ".jpg"
    if infile != outfile:
        try:
            Image.open(infile).save(outfile)
        except IOError:
            print("cannot convert", infile)
```

<https://riptutorial.com/zh-TW/python/topic/6841/>

# 150:

[PEP 435](#) 3.4Python。

## Examples

### Python 2.43.3

Python 3.3Python 3.4Python 2.4。 [PyPlenum34](#) backport。

```
pip install enum34
```

### Python 3.4+

```
from enum import Enum

class Color(Enum):
    red = 1
    green = 2
    blue = 3

print(Color.red) # Color.red
print(Color(1)) # Color.red
print(Color['red']) # Color.red
```

```
class Color(Enum):
    red = 1
    green = 2
    blue = 3

[c for c in Color] # [<Color.red: 1>, <Color.green: 2>, <Color.blue: 3>]
```

<https://riptutorial.com/zh-TW/python/topic/947/>



# 151:

ifelifelsePythonTrueFalse。 Python。

- <expression> if <conditional> else <expression> #Ternary Operator

## Examples

### ifelif

Pythonif elif else。

```
number = 5

if number > 2:
    print("Number is bigger than 2.")
elif number < 2: # Optional clause (you can have multiple elifs)
    print("Number is smaller than 2.")
else: # Optional clause (you can only have one else)
    print("Number is 2.")
```

Number is bigger than 2

else ifelif。

“”

。。

- CRubyJavaPython“”。
- “”。

```
n = 5

"Greater than 2" if n > 2 else "Smaller than or equal to 2"
# Out: 'Greater than 2'
```

- True。

### Tenary

```
n = 5
"Hello" if n > 10 else "Goodbye" if n > 5 else "Good day"
```

lambda。

```
if condition:
    body
```

if True if False

```
if True:
    print "It is true!"
>> It is true!

if False:
    print "This won't get printed.."
```

```
if 2 + 2 == 4:
    print "I know math!"
>> I know math!
```

```
if condition:
    body
else:
    body
```

## Falseelse

```
if True:
    print "It is true!"
else:
    print "This won't get printed.."

# Output: It is true!

if False:
    print "This won't get printed.."
else:
    print "It is false!"

# Output: It is false!
```

TrueFalse TrueFalse Pythonicif-else

---

True and False

```
>>> 1 and 2
2

>>> 1 and 0
0

>>> 1 and "Hello World"
"Hello World"

>>> "" and "Pancakes"
""
```

---

orTrueTrue ◦

```
>>> 1 or 2
1

>>> None or 1
1

>>> 0 or []
[]
```

◦ ◦

```
>>> def print_me():
...     print('I am here!')
>>> 0 and print_me()
0
```

print\_mePython0 False False ◦ print\_me◦

◦

**2. - if (a) and (b > 2) ◦ a bool(a)True ◦**

```
>>> a = 1
>>> b = 6
>>> if a and b > 2:
...     print('yes')
... else:
...     print('no')

yes
```

◦

```
>>> if a > 2 and b > 2:
...     print('yes')
... else:
...     print('no')

no
```

◦ - if (a == 3) or (4) or (6) ◦ bool(4)bool(6)True

```
>>> a = 1
>>> if a == 3 or 4 or 6:
...     print('yes')
... else:
```

```
...     print('no')
```

```
yes
```

```
>>> if a == 3 or a == 4 or a == 6:
```

```
...     print('yes')
```

```
... else:
```

```
...     print('no')
```

```
no
```

**in**

```
>>> if a in (3, 4, 6):
```

```
...     print('yes')
```

```
... else:
```

```
...     print('no')
```

```
no
```

False

- 
- 
- 0 0L 0.0 0j
- '' "" () []
- {}
- `__bool__` `__len__` 0 False

Python True

Falsey if `foo()` is None if `foo()`

**cmp**

Python 2 `cmp`

'greater than' `x > y` 'less than' `x < y` 'equal' `x == y`

```
['equal', 'greater than', 'less than', ][cmp(x,y)]
```

```
# x,y = 1,1 output: 'equal'
```

```
# x,y = 1,2 output: 'less than'
```

```
# x,y = 2,1 output: 'greater than'
```

`cmp(x,y)`

```
x < y    -1
```

|                       |                |
|-----------------------|----------------|
| <code>x == y</code>   | <code>0</code> |
| <code>x &gt; y</code> | <code>1</code> |

Python 3. Python 3 `functools.cmp_to_key(func)` ◦

Python ◦

```
[value_false, value_true][<conditional-test>]
```

```
>> n = 16
>> print [10, 20][n <= 15]
10
```

`n <= 15` False Python 0. Python

```
[10, 20][n <= 15]
=> [10, 20][False]
=> [10, 20][0] #False==0, True==1 (Check Boolean Equivalencies in Python)
=> 10
```

Python 2.x 2.7

`__cmp__` 30,1-1 `cmp(x,y)` objects `x > y` -1 `x < y`

01-1 ◦

```
[value_equals, value_greater, value_less][<conditional-test>]
```

Python ◦

```
[lambda: value_false, lambda: value_true][<test>]()
```

`() lambda` ◦ ◦

```
count = [lambda:0, lambda:N+1][count==N]()
```

None

None ◦ aDate ◦

is None ◦

```
if aDate is None:
    aDate=datetime.date.today()
```

Pythonic is None == None ◦

not NoneTrue◦

```
if not aDate:  
    aDate=datetime.date.today()
```

## Pythonic◦

```
aDate=aDate or datetime.date.today()
```

◦ aDate not None ◦ is None datetime.date.today() aDate ◦

<https://riptutorial.com/zh-TW/python/topic/1111/>

# 152:

## PythonZIP

```
import sys
sys.path.append("package.zip")
```

## Examples

### Python◦

module.py

```
def hi():
    print("Hello world!")
```

my\_script.py

```
import module
module.hi()
```

```
>>> from module import hi
>>> hi()
# Hello world!
```

### PythonCC ++◦

package

- \_\_init\_\_.py
- dog.py
- hi.py

\_\_init\_\_.py

```
from package.dog import woof
from package.hi import hi
```

dog.py

```
def woof():
    print("WOOF!!!")
```

hi.py

```
def hi():
    print("Hello world!")
```

Python\_\_init\_\_.py◦ import package import package \_\_init\_\_.py◦ package.hipackage.woof◦

<https://riptutorial.com/zh-TW/python/topic/3142/>



# 153:

|         |             |
|---------|-------------|
| os.F_OK | accessmode◦ |
| os.R_OK | accessmode◦ |
| os.W_OK | accessmode◦ |
| os.X_OK | accessmode◦ |

## Examples

### os.access

os.access◦

```
import os
path = "/home/myFiles/directory1"

## Check if path exists
os.access(path, os.F_OK)

## Check if path is Readable
os.access(path, os.R_OK)

## Check if path is Writable
os.access(path, os.W_OK)

## Check if path is Executable
os.access(path, os.X_OK)
```

```
os.access(path, os.F_OK & os.R_OK & os.W_OK & os.X_OK)
```

True/False ◦ **unix/windows**◦

<https://riptutorial.com/zh-TW/python/topic/1262/>

# 154:

## Python<sub>re</sub>

◦ 'amount\D+\d+'amount amount=100 amount is 3 amount is equal to: 33

- 
- re.matchpatternstringflag = 0#OutstringpatternNone
- re.searchpatternstringflag = 0#OutstringpatternNone
- re.findallpatternstringflag = 0#Out[]
- re.finditerpatternstringflag = 0#Outre.findall
- re.subpatternreplacementstringflag = 0#out
- 
- precompiled\_pattern = re.compilepatternflag = 0
- precompiled\_pattern.matchstring#Out
- precompiled\_pattern.searchstring#Out
- precompiled\_pattern.findallstring#out
- precompiled\_pattern.substring / pattern / functionstring#Out

## Examples

re.match()

```
import re

pattern = r"123"
string = "123zzb"

re.match(pattern, string)
# Out: <_sre.SRE_Match object; span=(0, 3), match='123'>

match = re.match(pattern, string)

match.group()
# Out: '123'
```

r ◦

\“”\n \t \ \t \r ◦ ◦

r"\n"

2 \n ◦ \d ◦ r"\d" "\d" r"\d" ◦

```
string = "\\t123zzb" # here the backslash is escaped, so there's no tab, just '\' and 't'
pattern = "\\t123" # this will match \t (escaping the backslash) followed by 123
re.match(pattern, string).group() # no match
re.match(pattern, "\t123zzb").group() # matches '\t123'

pattern = r"\\t123"
re.match(pattern, string).group() # matches '\\t123'
```

◦ [re.search](#)

```
match = re.match(r"(123)", "a123zzb")

match is None
# Out: True

match = re.search(r"(123)", "a123zzb")

match.group()
# Out: '123'
```

```
pattern = r"(your base)"
sentence = "All your base are belong to us."

match = re.search(pattern, sentence)
match.group(1)
# Out: 'your base'

match = re.search(r"(belong.*)", sentence)
match.group(1)
# Out: 'belong to us.'
```

re.match ◦ re.findall ◦

^

```
match = re.search(r"^123", "123zzb")
match.group(0)
# Out: '123'

match = re.search(r"^123", "a123zzb")
match is None
# Out: True
```

\$

```
match = re.search(r"123$", "zzb123")
match.group(0)
# Out: '123'

match = re.search(r"123$", "123zzb")
match is None
# Out: True
```

^\$

```
match = re.search(r"^123$", "123")
match.group(0)
# Out: '123'
```

◦ `group()` ◦

```
match.group() # Group without argument returns the entire match found
# Out: '123'
match.group(0) # Specifying 0 gives the same result as specifying no argument
# Out: '123'
```

`group()` ◦

;

`groups()` ◦

```
sentence = "This is a phone number 672-123-456-9910"
pattern = r".*(phone).*?([\d-]+)"

match = re.match(pattern, sentence)

match.groups() # The entire match as a list of tuples of the parenthesized subgroups
# Out: ('phone', '672-123-456-9910')

m.group() # The entire match as a string
# Out: 'This is a phone number 672-123-456-9910'

m.group(0) # The entire match as a string
# Out: 'This is a phone number 672-123-456-9910'

m.group(1) # The first parenthesized subgroup.
# Out: 'phone'

m.group(2) # The second parenthesized subgroup.
# Out: '672-123-456-9910'

m.group(1, 2) # Multiple arguments give us a tuple.
# Out: ('phone', '672-123-456-9910')
```

---

```
match = re.search(r'My name is (?P<name>[A-Za-z ]+)', 'My name is John Smith')
match.group('name')
# Out: 'John Smith'

match.group(1)
# Out: 'John Smith'
```

◦

---

(?:)

。 “”。

```
re.match(r'(\d+) (\+(\d+))?', '11+22').groups()
# Out: ('11', '+22', '22')

re.match(r'(\d+) (?:\+(\d+))?', '11+22').groups()
# Out: ('11', '22')
```

11+2211 11+ ◦ +◦ +◦

[]

```
match = re.search(r'[b]', 'a[b]c')
match.group()
# Out: 'b'
```

```
match = re.search(r'\[b\]', 'a[b]c')
match.group()
# Out: '[b]'
```

re.escape()

```
re.escape('a[b]c')
# Out: 'a\[b\]c'
match = re.search(re.escape('a[b]c'), 'a[b]c')
match.group()
# Out: 'a[b]c'
```

re.escape()

```
username = 'A.C.' # suppose this came from the user
re.findall(r'Hi { }!'.format(username), 'Hi A.C.! Hi ABCD!')
# Out: ['Hi A.C.!', 'Hi ABCD!']
re.findall(r'Hi { }!'.format(re.escape(username)), 'Hi A.C.! Hi ABCD!')
# Out: ['Hi A.C.!', 'Hi ABCD!']
```

re.sub◦

```
re.sub(r"t[0-9][0-9]", "foo", "my name t13 is t44 what t99 ever t44")
# Out: 'my name foo is foo what foo ever foo'
```

```
re.sub(r"t([0-9])([0-9])", r"t\2\1", "t13 t19 t81 t25")
# Out: 't31 t91 t18 t52'
```

ID“10” \10“ID10”◦ \g<i>

```
re.sub(r"t([0-9])([0-9])", r"t\g<2>\g<1>", "t13 t19 t81 t25")
# Out: 't31 t91 t18 t52'
```

```
items = ["zero", "one", "two"]
re.sub(r"a\[([0-3])\]", lambda match: items[int(match.group(1))], "Items: a[0], a[1], something, a[2]")
# Out: 'Items: zero, one, something, two'
```

```
re.findall(r"[0-9]{2,3}", "some 1 text 12 is 945 here 4445588899")
# Out: ['12', '945', '444', '558', '889']
```

`"[0-9]{2,3}"rpython;`

`re.finditer()``re.findall()``SRE_Match`

```
results = re.finditer(r"([0-9]{2,3})", "some 1 text 12 is 945 here 4445588899")
print(results)
# Out: <callable-iterator object at 0x105245890>
for result in results:
    print(result.group(0))
''' Out:
12
945
444
558
889
'''
```

```
import re

precompiled_pattern = re.compile(r"(\d+)")
matches = precompiled_pattern.search("The answer is 41!")
matches.group(1)
# Out: 41

matches = precompiled_pattern.search("Or was it 42?")
matches.group(1)
# Out: 42
```

◦ [Python docs](#) [SO](#)

```
import re

precompiled_pattern = re.compile(r"(.*\d+)")
matches = precompiled_pattern.match("The answer is 41!")
print(matches.group(1))
# Out: The answer is 41

matches = precompiled_pattern.match("Or was it 42?")
print(matches.group(1))
# Out: Or was it 42
```

`re.match`

`azAZ0-9`

```
import re
```

```
def is_allowed(string):
    characterRegex = re.compile(r'^a-zA-Z0-9.>')
    string = characterRegex.search(string)
    return not bool(string)

print (is_allowed("abyzABYZ0099"))
# Out: 'True'

print (is_allowed("#*#@#$$%^"))
# Out: 'False'
```

[^a-zA-Z0-9.][^a-z0-9.] ◦

<http://stackoverflow.com/a/1325265/2697955>

◦

```
import re
data = re.split(r'\s+', 'James 94 Samantha 417 Scarlett 74')
print( data )
# Output: ['James', '94', 'Samantha', '417', 'Scarlett', '74']
```

◦ flags◦

re.searchre◦

```
m = re.search("b", "ABC")
m is None
# Out: True

m = re.search("b", "ABC", flags=re.IGNORECASE)
m.group()
# Out: 'B'

m = re.search("a.b", "A\nBC", flags=re.IGNORECASE)
m is None
# Out: True

m = re.search("a.b", "A\nBC", flags=re.IGNORECASE|re.DOTALL)
m.group()
# Out: 'A\nB'
```

|                    |      |
|--------------------|------|
| re.IGNORECASE re.I |      |
| re.DOTALL re.S     | .    |
| re.MULTILINE re.M  | ^ \$ |
| re.DEBUG           |      |

(?iLmsux) 'i"L"m"s"u"x'◦

;re.Ire.Lre.Mre.Sre.UUnicodere.X。 re.compile。

X。 。 。

## `re.finditer`

re.finditer。 re.findall

```
import re
text = 'You can try to find an ant in this string'
pattern = 'an?\w' # find 'an' either with or without a following word character

for match in re.finditer(pattern, text):
    # Start index of match (integer)
    sStart = match.start()

    # Final index of match (integer)
    sEnd = match.end()

    # Complete match (string)
    sGroup = match.group()

    # Print match
    print('Match "{}" found at: [{} , {}]'.format(sGroup, sStart, sEnd))
```

```
Match "an" found at: [5,7]
Match "an" found at: [20,22]
Match "ant" found at: [23,26]
```

。

An apple a day keeps the doctor away (I eat an apple everyday).

“**regex**”。

forget\_this | or this | and this as well | (but keep this)

```
import regex as re
string = "An apple a day keeps the doctor away (I eat an apple everyday)."
```

|                      |                                                                    |
|----------------------|--------------------------------------------------------------------|
| rx = re.compile(r''' |                                                                    |
| \([^\)]*\)           | (*SKIP)(*FAIL) # match anything in parentheses and "throw it away" |
|                      | # or                                                               |
| apple                | # match an apple                                                   |
| ''', re.VERBOSE)     |                                                                    |

```
apples = rx.findall(string)
print(apples)
# only one
```

“apple”。

- (\*SKIP)“””。 (\*FAIL)。



- (\*SKIP)◦ (\*SKIP)◦

<https://riptutorial.com/zh-TW/python/topic/632/-->

# 155:

- = -
- == -
- > -
- < -
- >= -
- <= -
- is -
- is not = test



## Examples

```
x > y
x < y
```

◦

```
12 > 4
# True
12 < 4
# False
1 < 4
# True
```

◦

```
"alpha" < "beta"
# True
"gamma" > "beta"
# True
"gamma" < "OMEGA"
# False
```

“” "gamma" < "OMEGA" ◦

```
"GAMMA" < "OMEGA"  
# True
```

<>◦

```
x != y
```

True xyFalse ◦

```
12 != 1  
# True  
12 != '12'  
# True  
'12' != '12'  
# False
```

```
x == y
```

xy◦ **int** 12'12' ◦

```
12 == 12  
# True  
12 == 1  
# False  
'12' == '12'  
# True  
'spam' == 'spam'  
# True  
'spam' == 'spam '  
# False  
'12' == 12  
# False
```

◦ ◦ TrueFalse ◦

```
x > y > z
```

```
x > y and y > z
```

TrueTrue ◦

```
a OP b OP c OP d ...
```

OP◦

```
0 != 1 != 0True 0 != 0False ◦ x != y != zx yz◦ ==◦
```

```
1 > -1 < 2 > 0.5 < 100 != 24
```

True True ◦ ◦ “”

```
1 > x > -4 > y != 8
```

False False ◦

a > exp > bexp◦

```
a > exp and exp > b
```

expa > exp◦

## `is` vs `==`

is== ◦

a == bab◦

a is bab ◦

```
a = 'Python is fun!'
b = 'Python is fun!'
a == b # returns True
a is b # returns False

a = [1, 2, 3, 4, 5]
b = a      # b references a
a == b    # True
a is b    # True
b = a[:]   # b now references a copy of a
a == b    # True
a is b    # False [!!]
```

isid(a) == id(b) ◦

◦ PythonisTrue ◦

```
a = 'short'
b = 'short'
c = 5
d = 5
a is b # True
c is d # True
```

◦

```
a = 'not so short'
b = 'not so short'
c = 1000
```

```
d = 1000
a is b # False
c is d # False
```

isNone

```
if myvar is not None:
    # not None
    pass
if myvar is None:
    # None
    pass
```

isis“”。

```
sentinel = object()
def myfunc(var=sentinel):
    if var is sentinel:
        # value wasn't provided
        pass
    else:
        # value was provided
        pass
```

`__eq__` `__ne__` `==` `!=` `__lt__` `<` `__le__` `<=` `__gt__` `>` `__ge__` `>` `Python` `==` `not` `<` `not` `>`

```
class Foo(object):
    def __init__(self, item):
        self.my_item = item
    def __eq__(self, other):
        return self.my_item == other.my_item

a = Foo(5)
b = Foo(5)
a == b      # True
a != b      # False
a is b      # False
```

other 。

```
class Bar(object):
    def __init__(self, item):
        self.other_item = item
    def __eq__(self, other):
        return self.other_item == other.other_item
    def __ne__(self, other):
        return self.other_item != other.other_item

c = Bar(5)
a == c      # throws AttributeError: 'Foo' object has no attribute 'other_item'
```

isinstance()。

## Python

## Java

```
if("asgdsrf" == 0) {  
    //do stuff  
}
```

.....◦ ◦ **Python - False** ◦

```
myVariable = "1"  
if 1 == myVariable:  
    #do stuff
```

False ◦

<https://riptutorial.com/zh-TW/python/topic/248/>

# 156:

- `class ClassName MainClass Mixin1 Mixin2 ...ClassName mainfirstclass MainClass mixins Mixin1 Mixin2.`
- `class ClassName Mixin1 MainClass Mixin2 ...'main';mixin`

`mixin Foo Foo.isinstance(instance, Foo) true`

## Examples

**Mixin** ;

`Car BoatPlane . travel . ; Vehicle`

```
class Vehicle(object):
    """A generic vehicle class."""

    def __init__(self, position):
        self.position = position

    def travel(self, destination):
        route = calculate_route(from=self.position, to=destination)
        self.move_along(route)

class Car(Vehicle):
    ...

class Boat(Vehicle):
    ...

class Plane(Vehicle):
    ...
```

`car.travel("Montana") boat.travel("Hawaii") plane.travel("France") travel`

`Carplay_song_on_station Clock . CarClock Machine . ; BoatPlane . mixin . Pythonmixin`

```
class Foo(main_super, mixin): ...
```

`Foo main_super mixin`

`CarClockCar`

```
class RadioUserMixin(object):
    def __init__(self):
        self.radio = Radio()

    def play_song_on_station(self, station):
        self.radio.set_station(station)
        self.radio.play_song()
```

```
class Car(Vehicle, RadioUserMixin):
    ...

class Clock(Vehicle, RadioUserMixin):
    ...
```

```
car.play_song_on_station(98.7) clock.play_song_on_station(101.3)
boat.play_song_on_station(100.5)
```

mixins“”。 mixins/。

Mixins“”。 mixin。 mixinPython。

```
class Mixin1(object):
    def test(self):
        print "Mixin1"

class Mixin2(object):
    def test(self):
        print "Mixin2"

class BaseClass(object):
    def test(self):
        print "Base"

class MyClass(BaseClass, Mixin1, Mixin2):
    pass
```

Mixin2Mixin1BaseClass。

```
>>> x = MyClass()
>>> x.test()
Base
```

Base。

<https://riptutorial.com/zh-TW/python/topic/4359/>



# 157:

FlaskPythonWebPinterestTwilioLinkedIn FlaskWeb

- @ app.route("/ urlpath"methods = ["GET""POST""DELETE""PUTS""HEAD""OPTIONS"]
- @ app.route("/ urlpath / <param>"methods = ["GET""POST""DELETE""PUTS""HEAD""OPTIONS"]

## Examples

```
# Imports the Flask class
from flask import Flask
# Creates an app and checks if its the main or imported
app = Flask(__name__)

# Specifies what URL triggers hello_world()
@app.route('/')
# The function run on the index route
def hello_world():
    # Returns the text to be displayed
    return "Hello World!"

# If this script isn't an import
if __name__ == "__main__":
    # Run the app until stopped
    app.run()
```

◦ 127.0.0.1localhost ◦ 5000◦ WebWebURL localhost:5000127.0.0.1:5000 ◦ ◦

app.run() ◦ 127.0.0.1 0.0.0.0WebURLIP◦ 500080 80◦ TrueFlask◦

```
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=80, debug=True)
```

## URL

FlaskURL◦ URL◦ Flaskwww.example.com◦

```
@app.route("/")
def index():
    return "You went to www.example.com"

@app.route("/about")
def about():
    return "You went to www.example.com/about"

@app.route("/users/guido-van-rossum")
def user():
    return "You went to www.example.com/guido-van-rossum"
```

/ users /URL◦ @app.route() FlaskURL

```

@app.route("/users/<username>")
def profile(username):
    return "Welcome to the profile of " + username

cities = ["OMAHA", "MELBOURNE", "NEPAL", "STUTTGART", "LIMA", "CAIRO", "SHANGHAI"]

@app.route("/stores/locations/<city>")
def storefronts(city):
    if city in cities:
        return "Yes! We are located in " + city
    else:
        return "No. We are not located in " + city

```

## HTTP

HTTPGETPOST。 FlaskHTTPURL。 WebURL。 GETURLPOST。 DELETEPUT HTTP。

```

@app.route("/login", methods=["GET"])
def login_form():
    return "This is the login form"
@app.route("/login", methods=["POST"])
def login_auth():
    return "Processing your data"
@app.route("/login", methods=["DELETE", "PUT"])
def deny():
    return "This method is not allowed"

```

flaskrequest

```

from flask import request

@app.route("/login", methods=["GET", "POST", "DELETE", "PUT"])
def login():
    if request.method == "DELETE" or request.method == "PUT":
        return "This method is not allowed"
    elif request.method == "GET":
        return "This is the login forum"
    elif request.method == "POST":
        return "Processing your data"

```

POSTrequest

```

from flask import request
@app.route("/login", methods=["GET", "POST", "DELETE", "PUT"])
def login():
    if request.method == "DELETE" or request.method == "PUT":
        return "This method is not allowed"
    elif request.method == "GET":
        return "This is the login forum"
    elif request.method == "POST":
        return "Username was " + request.form["username"] + " and password was " +
request.form["password"]

```

render\_template() HTML

```

from flask import Flask
from flask import render_template
app = Flask(__name__)

@app.route("/about")
def about():
    return render_template("about-us.html")

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=80, debug=True)

```

about-us.html ◦

```

- application.py
/templates
  - about-us.html
  - login-form.html
/static
  /styles
    - about-style.css
    - login-style.css
  /scripts
    - about-script.js
    - login-script.js

```

## HTML

```
<link rel="stylesheet" type="text/css", href="{{url_for('static', filename='styles/about-style.css')}}">
```

**styles**about-style.css ◦ ◦

Meteor.jsFlask◦ FlaskJinja Templating◦ HTML◦

HTML◦ HTML◦

```

@app.route("/users/<username>")
def profile(username):
    joinedDate = get_joined_date(username) # This function's code is irrelevant
    awards = get_awards(username) # This function's code is irrelevant
    # The joinDate is a string and awards is an array of strings
    return render_template("profile.html", username=username, joinDate=joinDate,
        awards=awards)

```

render\_template()◦ profile.html

```

<!DOCTYPE html>
<html>
  <head>
    # if username
    <title>Profile of {{ username }}</title>
    # else
    <title>No User Found</title>
    # endif
  </head>
  <body>

```

```

    {% if username %}
        <h1>{{ username }} joined on the date {{ date }}</h1>
        {% if len(awards) > 0 %}
            <h3>{{ username }} has the following awards:</h3>
            <ul>
                {% for award in awards %}
                    <li>{{award}}</li>
                {% endfor %}
            </ul>
        {% else %}
            <h3>{{ username }} has no awards</h3>
        {% endif %}
    {% else %}
        <h1>No user was found under that username</h1>
    {% endif %}
    {# This is a comment and doesn't affect the output #}
</body>
</html>

```

- {% ... %}
- {{ ... }}
- {# ... #}
- {# ... ##

request◦

```
from flask import request
```

request.methodrequest.form request.argsURL◦

```

@app.route("/api/users/<username>")
def user_api(username):
    try:
        token = request.args.get("key")
        if key == "pA55w0Rd":
            if isUser(username): # The code of this method is irrelevant
                joined = joinDate(username) # The code of this method is irrelevant
                return "User " + username + " joined on " + joined
            else:
                return "User not found"
        else:
            return "Incorrect key"
    # If there is no key parameter
    except KeyError:
        return "No key provided"

```

## URL

www.example.com/api/users/guido-van-rossum?key=pa55w0Rd

POST<sub>request</sub>

```
@app.route("/upload", methods=["POST"])
```

```
def upload_file():
    f = request.files["wordlist-upload"]
    f.save("/var/www/uploads/" + f.filename) # Store with the original filename
```

---

## URLcookie

```
@app.route("/home")
def home():
    try:
        username = request.cookies.get("username")
        return "Your stored username is " + username
    except KeyError:
        return "No username cookies was found")
```

<https://riptutorial.com/zh-TW/python/topic/8682/>

# 158: `exec` `eval`

- `eval[globals = None [locals = None]]`
- EXEC
- `exec`
- `exec`

expression	code
object	code
globals	◦ locals◦ globals()◦
locals	◦ globals◦ globals()locals()globalslocals globals◦

`exec globalslocals◦ globalslocals globals◦`

`globals__builtins__ keyPython◦ printisinstanceglobals__builtins__None◦◦`

Python 2; Python 3Python 2◦ <s>

- `exec` object
- `exec` object in `globals`
- `exec` object in `globals`, `locals`

## Examples

### exec

```
>>> code = """for i in range(5):\n    print('Hello world!')"""
>>> exec(code)
Hello world!
Hello world!
Hello world!
Hello world!
Hello world!
```

### eval

```
>>> expression = '5 + 3 * a'
>>> a = 5
>>> result = eval(expression)
>>> result
20
```

`compile;eval◦◦ compile3'eval'◦`

```
>>> code = compile('a * b + c', '<string>', 'eval')
>>> code
<code object <module> at 0x7f0e51a58830, file "<string>", line 1>
>>> a, b, c = 1, 2, 3
>>> eval(code)
5
```

## eval

```
>>> variables = {'a': 6, 'b': 7}
>>> eval('a * b', globals=variables)
42
```

```
>>> eval('variables')
{'a': 6, 'b': 7}
>>> eval('variables', globals=variables)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<string>", line 1, in <module>
NameError: name 'variables' is not defined
```

defaultdict

```
>>> from collections import defaultdict
>>> variables = defaultdict(int, {'a': 42})
>>> eval('a * c', globals=variables) # note that 'c' is not explicitly defined
0
```

## ast.literal\_evalPython

Python `ast.literal_eval` `eval` `◦ ◦`

```
>>> import ast
>>> code = """(1, 2, {'foo': 'bar'})"""
>>> object = ast.literal_eval(code)
>>> object
(1, 2, {'foo': 'bar'})
>>> type(object)
<class 'tuple'>
```

```
>>> import ast
>>> ast.literal_eval('(') * 1000000
[5] 21358 segmentation fault (core dumped) python3
```

(`CPython` `CPython` `◦` `CPython` `◦`)

## execast.literal\_eval

`eval` `exec` `◦` `ast.literal_eval` `◦` `◦`

Python `◦`

``exec`eval`` <https://riptutorial.com/zh-TW/python/topic/2251/-exec--eval->



# 159: Matplotlib

Matplotlib <https://matplotlib.org/NumPy2D>。 <https://matplotlib.org/2.0.2/gallery.html>  
<https://matplotlib.org/2.0.2/examples/index.html>[http //www.riptutorial.com/topic/881](http://www.riptutorial.com/topic/881)

## Examples

### Matplotlib

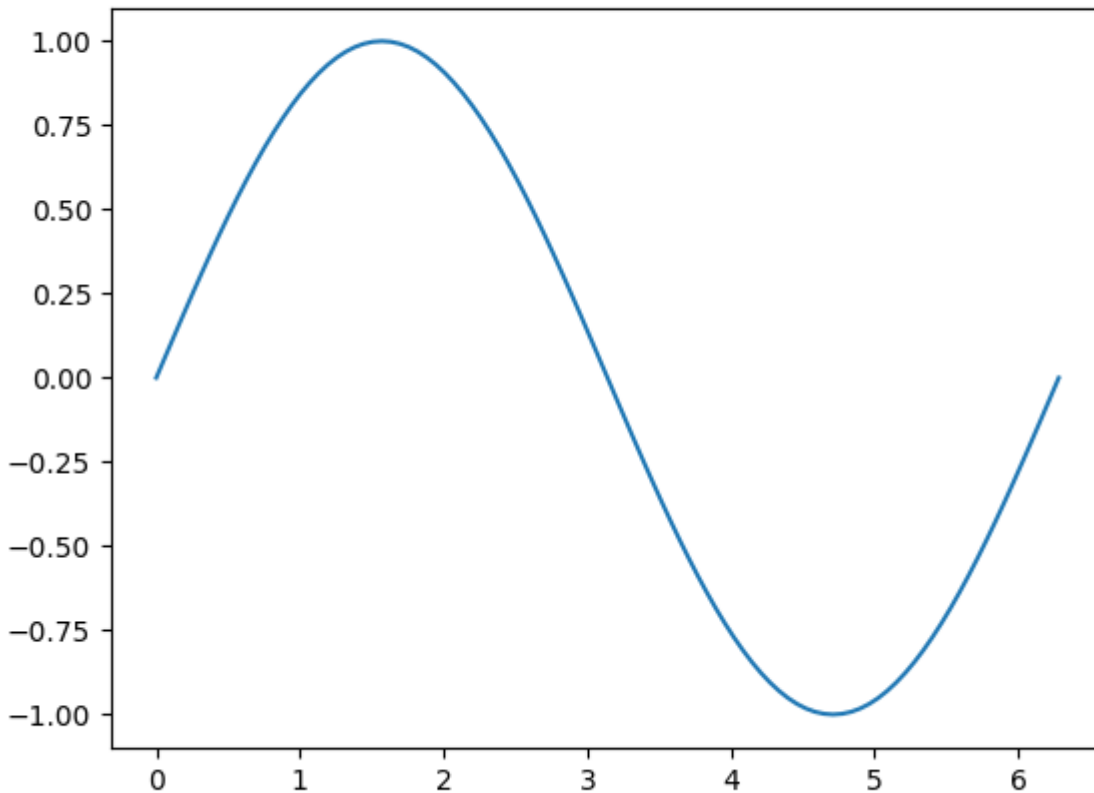
### Matplotlib

```
# Plotting tutorials in Python
# Launching a simple plot

import numpy as np
import matplotlib.pyplot as plt

# angle varying between 0 and 2*pi
x = np.linspace(0, 2.0*np.pi, 101)
y = np.sin(x)                # sine function

plt.plot(x, y)
plt.show()
```



;

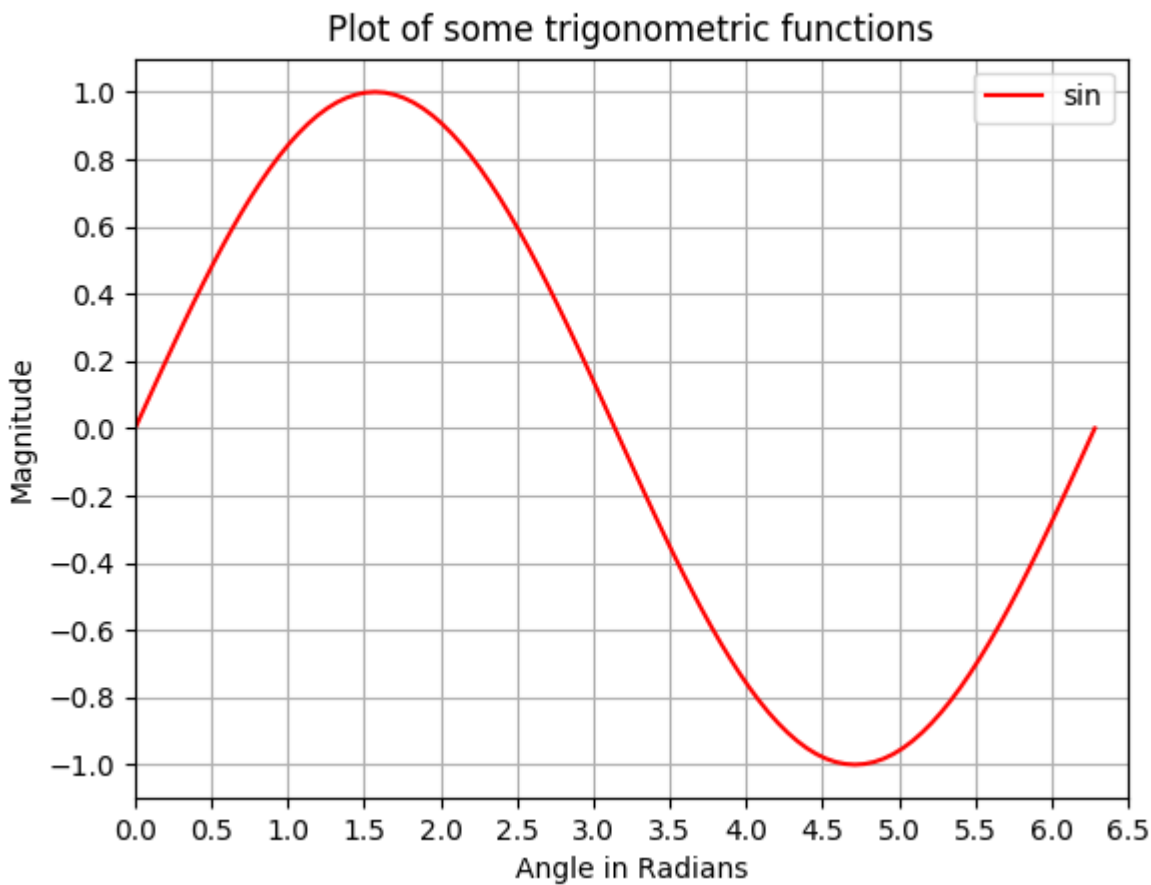
```
# Plotting tutorials in Python
# Enhancing a plot

import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2.0*np.pi, 101)
y = np.sin(x)

# values for making ticks in x and y axis
xnumbers = np.linspace(0, 7, 15)
ynumbers = np.linspace(-1, 1, 11)

plt.plot(x, y, color='r', label='sin') # r - red colour
plt.xlabel("Angle in Radians")
plt.ylabel("Magnitude")
plt.title("Plot of some trigonometric functions")
plt.xticks(xnumbers)
plt.yticks(ynumbers)
plt.legend()
plt.grid()
plt.axis([0, 6.5, -1.1, 1.1]) # [xstart, xend, ystart, yend]
plt.show()
```



## MATLAB

```

# Plotting tutorials in Python
# Adding Multiple plots by superimposition
# Good for plots sharing similar x, y limits
# Using single plot command and legend

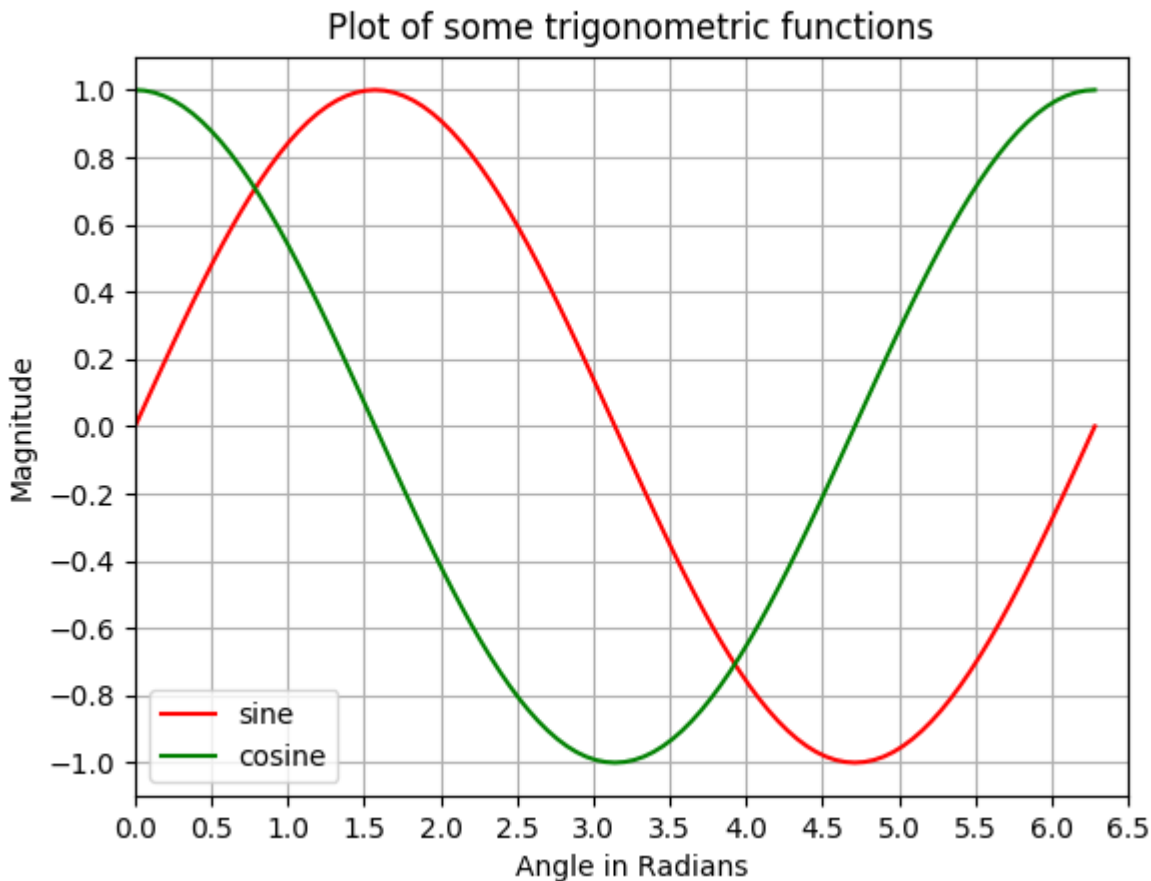
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2.0*np.pi, 101)
y = np.sin(x)
z = np.cos(x)

# values for making ticks in x and y axis
xnumbers = np.linspace(0, 7, 15)
ynumbers = np.linspace(-1, 1, 11)

plt.plot(x, y, 'r', x, z, 'g') # r, g - red, green colour
plt.xlabel("Angle in Radians")
plt.ylabel("Magnitude")
plt.title("Plot of some trigonometric functions")
plt.xticks(xnumbers)
plt.yticks(ynumbers)
plt.legend(['sine', 'cosine'])
plt.grid()
plt.axis([0, 6.5, -1.1, 1.1]) # [xstart, xend, ystart, yend]
plt.show()

```



## ◦ Pythonic◦

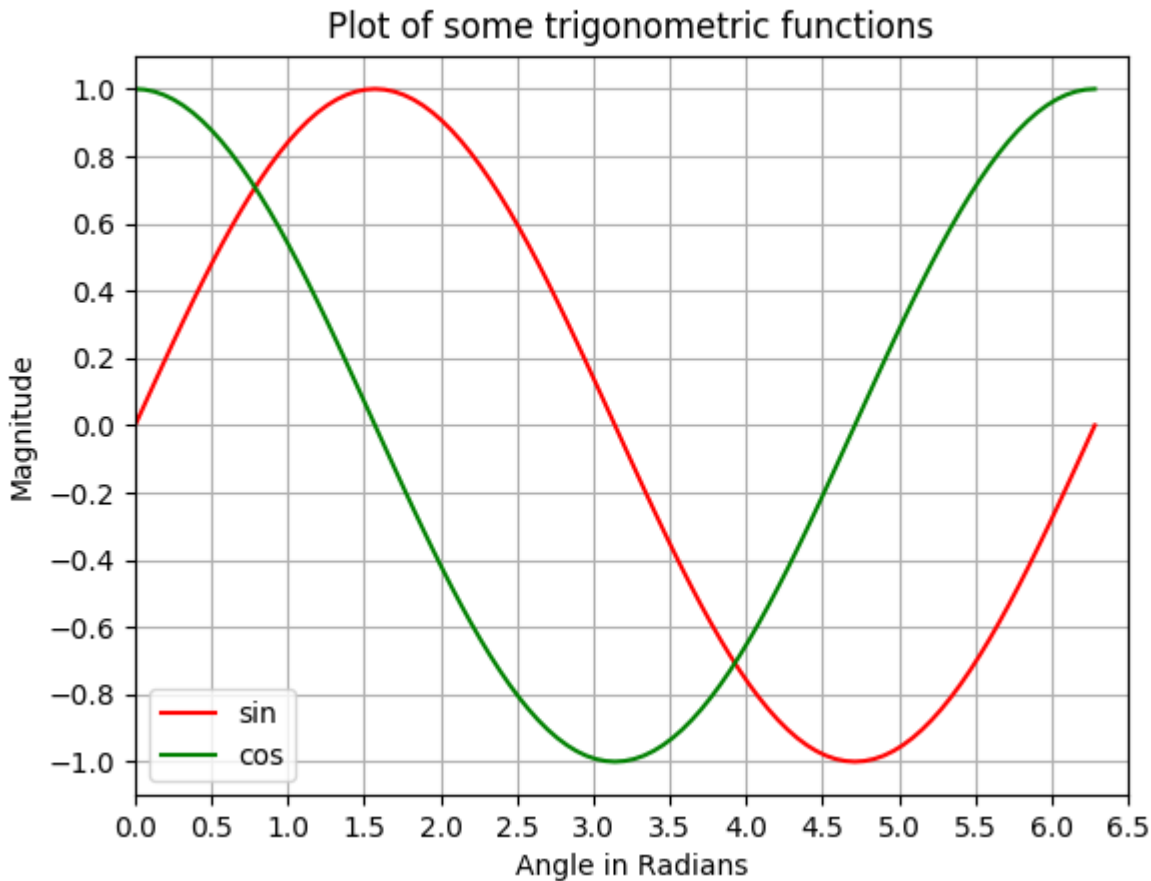
```
# Plotting tutorials in Python
# Adding Multiple plots by superimposition
# Good for plots sharing similar x, y limits
# Using multiple plot commands
# Much better and preferred than previous

import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2.0*np.pi, 101)
y = np.sin(x)
z = np.cos(x)

# values for making ticks in x and y axis
xnumbers = np.linspace(0, 7, 15)
ynumbers = np.linspace(-1, 1, 11)

plt.plot(x, y, color='r', label='sin') # r - red colour
plt.plot(x, z, color='g', label='cos') # g - green colour
plt.xlabel("Angle in Radians")
plt.ylabel("Magnitude")
plt.title("Plot of some trigonometric functions")
plt.xticks(xnumbers)
plt.yticks(ynumbers)
plt.legend()
plt.grid()
plt.axis([0, 6.5, -1.1, 1.1]) # [xstart, xend, ystart, yend]
plt.show()
```



## XYtwinx

xy. **twinx**.

```
# Plotting tutorials in Python
# Adding Multiple plots by twin x axis
# Good for plots having different y axis range
# Separate axes and figure objects
# replicate axes object and plot curves
# use axes to set attributes

# Note:
# Grid for second curve unsuccessful : let me know if you find it! :(

import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 2.0*np.pi, 101)
y = np.sin(x)
z = np.sinh(x)

# separate the figure object and axes object
# from the plotting object
fig, ax1 = plt.subplots()

# Duplicate the axes with a different y axis
# and the same x axis
ax2 = ax1.twinx() # ax2 and ax1 will have common x axis and different y axis
```

```

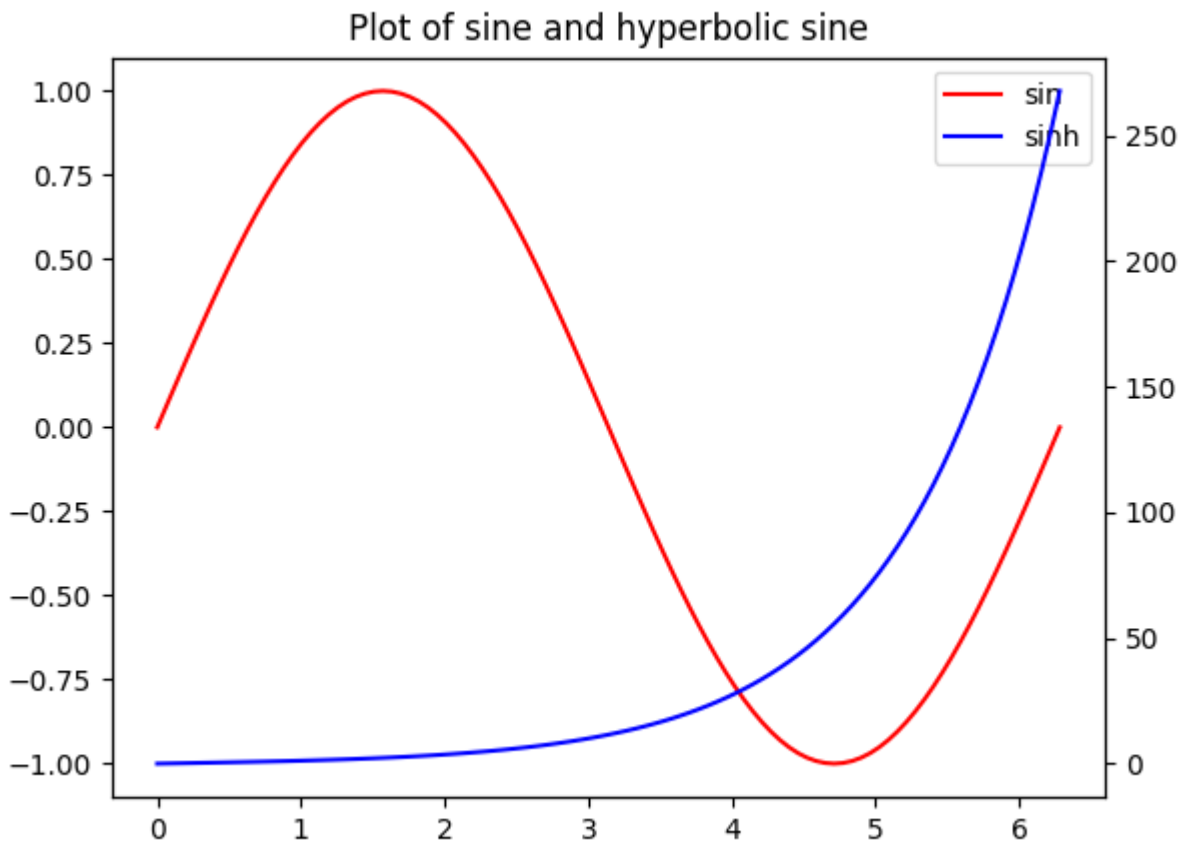
# plot the curves on axes 1, and 2, and get the curve handles
curve1, = ax1.plot(x, y, label="sin", color='r')
curve2, = ax2.plot(x, z, label="sinh", color='b')

# Make a curves list to access the parameters in the curves
curves = [curve1, curve2]

# add legend via axes 1 or axes 2 object.
# one command is usually sufficient
# ax1.legend() # will not display the legend of ax2
# ax2.legend() # will not display the legend of ax1
ax1.legend(curves, [curve.get_label() for curve in curves])
# ax2.legend(curves, [curve.get_label() for curve in curves]) # also valid

# Global figure properties
plt.title("Plot of sine and hyperbolic sine")
plt.show()

```



## twinyYX

### twinyyx ◦ ◦

```

# Plotting tutorials in Python
# Adding Multiple plots by twin y axis
# Good for plots having different x axis range
# Separate axes and figure objects
# replicate axes object and plot curves
# use axes to set attributes

```

```

import numpy as np
import matplotlib.pyplot as plt

y = np.linspace(0, 2.0*np.pi, 101)
x1 = np.sin(y)
x2 = np.sinh(y)

# values for making ticks in x and y axis
ynumbers = np.linspace(0, 7, 15)
xnumbers1 = np.linspace(-1, 1, 11)
xnumbers2 = np.linspace(0, 300, 7)

# separate the figure object and axes object
# from the plotting object
fig, ax1 = plt.subplots()

# Duplicate the axes with a different x axis
# and the same y axis
ax2 = ax1.twinx() # ax2 and ax1 will have common y axis and different x axis

# plot the curves on axes 1, and 2, and get the axes handles
curve1, = ax1.plot(x1, y, label="sin", color='r')
curve2, = ax2.plot(x2, y, label="sinh", color='b')

# Make a curves list to access the parameters in the curves
curves = [curve1, curve2]

# add legend via axes 1 or axes 2 object.
# one command is usually sufficient
# ax1.legend() # will not display the legend of ax2
# ax2.legend() # will not display the legend of ax1
# ax1.legend(curves, [curve.get_label() for curve in curves])
ax2.legend(curves, [curve.get_label() for curve in curves]) # also valid

# x axis labels via the axes
ax1.set_xlabel("Magnitude", color=curve1.get_color())
ax2.set_xlabel("Magnitude", color=curve2.get_color())

# y axis label via the axes
ax1.set_ylabel("Angle/Value", color=curve1.get_color())
# ax2.set_ylabel("Magnitude", color=curve2.get_color()) # does not work
# ax2 has no property control over y axis

# y ticks - make them coloured as well
ax1.tick_params(axis='y', colors=curve1.get_color())
# ax2.tick_params(axis='y', colors=curve2.get_color()) # does not work
# ax2 has no property control over y axis

# x axis ticks via the axes
ax1.tick_params(axis='x', colors=curve1.get_color())
ax2.tick_params(axis='x', colors=curve2.get_color())

# set x ticks
ax1.set_xticks(xnumbers1)
ax2.set_xticks(xnumbers2)

# set y ticks
ax1.set_yticks(ynumbers)
# ax2.set_yticks(ynumbers) # also works

```

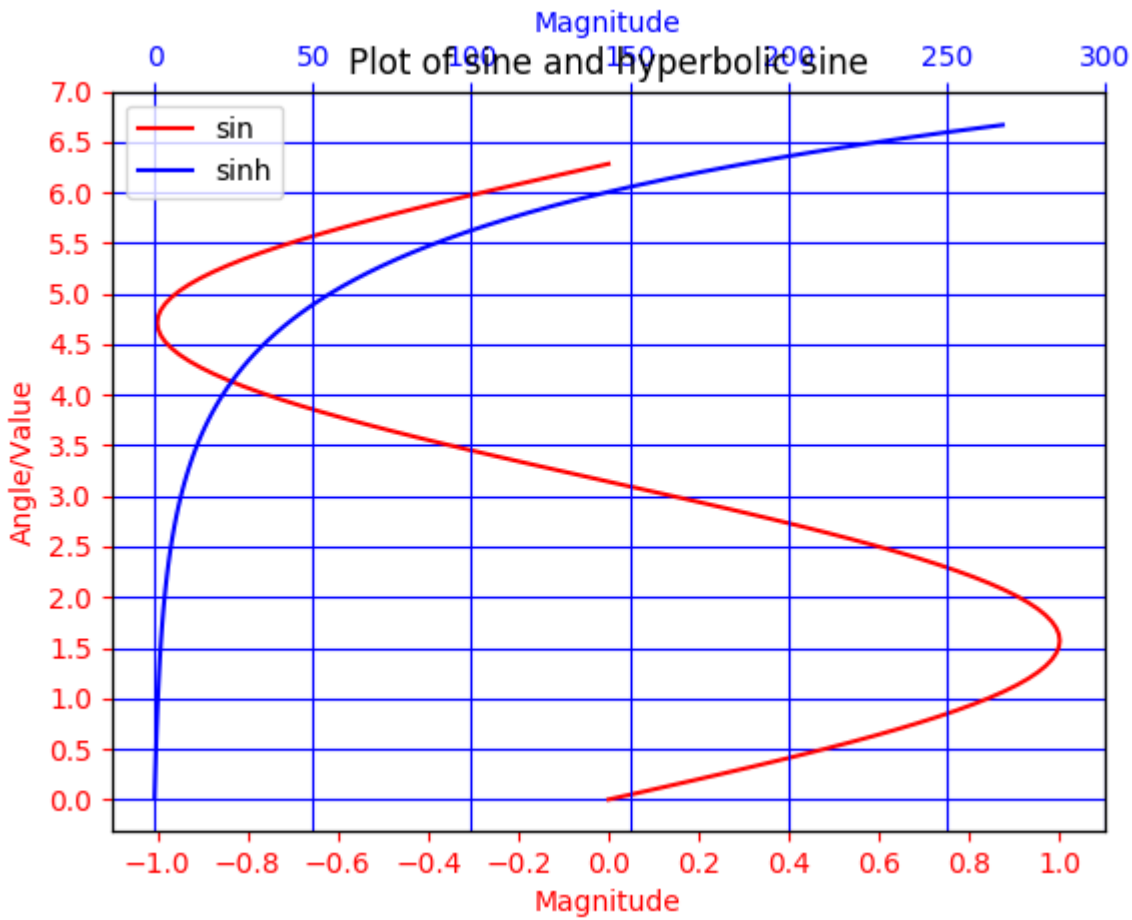
```

# Grids via axes 1 # use this if axes 1 is used to
# define the properties of common x axis
# ax1.grid(color=curve1.get_color())

# To make grids using axes 2
ax1.grid(color=curve2.get_color())
ax2.grid(color=curve2.get_color())
ax1.xaxis.grid(False)

# Global figure properties
plt.title("Plot of sine and hyperbolic sine")
plt.show()

```



Matplotlib <https://riptutorial.com/zh-TW/python/topic/10264/matplotlib>



# 160:

## Examples

o

```
class A(object):
    # func: A user-defined function object
    #
    # Note that func is a function object when it's defined,
    # and an unbound method object when it's retrieved.
    def func(self):
        pass

    # classMethod: A class method
    @classmethod
    def classMethod(self):
        pass

class B(object):
    # unboundMeth: A unbound user-defined method object
    #
    # Parent.func is an unbound user-defined method object here,
    # because it's retrieved.
    unboundMeth = A.func

a = A()
b = B()

print A.func
# output: <unbound method A.func>
print a.func
# output: <bound method A.func of <__main__.A object at 0x10e9ab910>>
print B.unboundMeth
# output: <unbound method A.func>
print b.unboundMeth
# output: <unbound method A.func>
print A.classMethod
# output: <bound method type.classMethod of <class '__main__.A'>>
print a.classMethod
# output: <bound method type.classMethod of <class '__main__.A'>>
```

;o

```
# Parent: The class stored in the original method object
class Parent(object):
    # func: The underlying function of original method object
    def func(self):
        pass
    func2 = func

# Child: A derived class of Parent
class Child(Parent):
    func = Parent.func

# AnotherClass: A different class, neither subclasses nor subclassed
```

```

class AnotherClass(object):
    func = Parent.func

print Parent.func is Parent.func           # False, new object created
print Parent.func2 is Parent.func2         # False, new object created
print Child.func is Child.func             # False, new object created
print AnotherClass.func is AnotherClass.func # True, original object used

```

∞.

```

import turtle, time, random #tell python we need 3 different modules
turtle.speed(0) #set draw speed to the fastest
turtle.colormode(255) #special colormode
turtle.pensize(4) #size of the lines that will be drawn
def triangle(size): #This is our own function, in the parenthesis is a variable we have
defined that will be used in THIS FUNCTION ONLY. This fucntion creates a right triangle
    turtle.forward(size) #to begin this function we go forward, the amount to go forward by is
the variable size
    turtle.right(90) #turn right by 90 degree
    turtle.forward(size) #go forward, again with variable
    turtle.right(135) #turn right again
    turtle.forward(size * 1.5) #close the triangle. thanks to the Pythagorean theorem we know
that this line must be 1.5 times longer than the other two(if they are equal)
while(1): #INFINITE LOOP
    turtle.setpos(random.randint(-200, 200), random.randint(-200, 200)) #set the draw point to
a random (x,y) position
    turtle.pencolor(random.randint(1, 255), random.randint(1, 255), random.randint(1, 255))
#randomize the RGB color
    triangle(random.randint(5, 55)) #use our function, because it has only one variable we can
simply put a value in the parenthesis. The value that will be sent will be random between 5 -
55, end the end it really just changes ow big the triangle is.
    turtle.pencolor(random.randint(1, 255), random.randint(1, 255), random.randint(1, 255))
#randomize color again

```

<https://riptutorial.com/zh-TW/python/topic/3965/>

# 161:

`yield (an_expression for x in an_iterator)` ◦

- `yield <expr>`
- `<expr>`
- `<var> = yield <expr>`
- `<iter>`

## Examples

◦ `next()` Python 3.x `__next__()` `__iter__` ◦

```
# naive partial implementation of the Python 2.x xrange()
def xrange(n):
    i = 0
    while i < n:
        yield i
        i += 1

# looping
for i in xrange(10):
    print(i) # prints the values 0, 1, ..., 9

# unpacking
a, b, c = xrange(3) # 0, 1, 2

# building a list
l = list(xrange(10)) # [0, 1, ..., 9]
```

## next

`next()` ◦

```
def nums():
    yield 1
    yield 2
    yield 3
generator = nums()

next(generator, None) # 1
next(generator, None) # 2
next(generator, None) # 3
next(generator, None) # None
next(generator, None) # None
# ...
```

`next(iterator[, default])` ◦ ◦ `StopIteration` ◦

`send()` ◦

```

def accumulator():
    total = 0
    value = None
    while True:
        # receive sent value
        value = yield total
        if value is None: break
        # aggregate values
        total += value

generator = accumulator()

# advance until the first "yield"
next(generator)      # 0

# from this point on, the generator aggregates values
generator.send(1)    # 1
generator.send(10)   # 11
generator.send(100)  # 111
# ...

# Calling next(generator) is equivalent to calling generator.send(None)
next(generator)     # StopIteration

```

- `next(generator)` yields `total`.
- `generator.send(x)` yields `value`.
- `next(generator)` with `None` yields `None`.

```

generator = (i * 2 for i in range(3))

next(generator) # 0
next(generator) # 2
next(generator) # 4
next(generator) # raises StopIteration

```

```

sum(i ** 2 for i in range(4)) # 0^2 + 1^2 + 2^2 + 3^2 = 0 + 1 + 4 + 9 = 14

```

Python `[0, 1, 2, 3]`.

```

expression = (x**2 for x in range(10))

```

`100x = 0`.

`yield` returns

```

def function():
    for x in range(10):

```

```
yield x**2
```

◦

◦

---

```
sum(i for i in range(10) if i % 2 == 0) #Output: 20
any(x = 0 for x in foo) #Output: True or False depending on foo
type(a > b for a in foo if a % 2 == 1) #Output: <class 'generator'>
```

```
sum((i for i in range(10) if i % 2 == 0))
any((x = 0 for x in foo))
type((a > b for a in foo if a % 2 == 1))
```

```
fooFunction(i for i in range(10) if i % 2 == 0,foo,bar)
return x = 0 for x in foo
barFunction(baz, a > b for a in foo if a % 2 == 1)
```

---

◦ ◦

```
g1 = function()
print(g1) # Out: <generator object function at 0x1012e1888>
```

function() **print** ◦ ◦

◦ ◦

list ◦ “” ◦

```
for x in g1:
    print("Received", x)

# Output:
# Received 0
# Received 1
# Received 4
# Received 9
# Received 16
# Received 25
# Received 36
# Received 49
# Received 64
# Received 81

arr1 = list(g1)
# arr1 = [], because the loop above already consumed all the values.
g2 = function()
arr2 = list(g2) # arr2 = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

`next()` ◦ ◦

`next()` Python `yield` `yield` `next()` `yield`

Python `yield` `StopIteration`

```
g3 = function()
a = next(g3) # a becomes 0
b = next(g3) # b becomes 1
c = next(g3) # c becomes 2
...
j = next(g3) # Raises StopIteration, j remains undefined
```

Python 2. `next()` Python 3. `__next__()`

- None ◦

- ◦ ◦

- **Fibonacci** ◦

```
def fib(a=0, b=1):
    """Generator that yields Fibonacci numbers. `a` and `b` are the seed values"""
    while True:
        yield a
        a, b = b, a + b

f = fib()
print(', '.join(str(next(f)) for _ in range(10)))
```

**0,1,1,2,3,5,8,13,21,34**

```
def integers_starting_from(n):
    while True:
        yield n
        n += 1

natural_numbers = integers_starting_from(1)
```

`itertools.count`

```
natural_numbers = itertools.count(1)
```

```
multiples_of_two = (x * 2 for x in natural_numbers)
multiples_of_three = (x for x in natural_numbers if x % 3 == 0)
```

```
list(multiples_of_two) # will never terminate, or raise an OS-specific error
```

list / set comprehensions with `range` `xrange` for python <3.0

```
first_five_multiples_of_three = [next(multiples_of_three) for _ in range(5)]
# [3, 6, 9, 12, 15]
```

`itertools.islice()`

```
from itertools import islice
multiples_of_four = (x * 4 for x in integers_starting_from(1))
first_five_multiples_of_four = list(islice(multiples_of_four, 5))
# [4, 8, 12, 16, 20]
```

“”

```
next(natural_numbers)    # yields 16
next(multiples_of_two)   # yields 34
next(multiples_of_four)  # yields 24
```

`for`° `break`

```
for idx, number in enumerate(multiplies_of_two):
    print(number)
    if idx == 9:
        break # stop after taking the first 10 multiplies of two
```

---

■

```
import itertools

def fibonacci():
    a, b = 1, 1
    while True:
        yield a
        a, b = b, a + b

first_ten_fibs = list(itertools.islice(fibonacci(), 10))
# [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

def nth_fib(n):
    return next(itertools.islice(fibonacci(), n - 1, n))

ninety_nineth_fib = nth_fib(99) # 354224848179261915075
```

## Python 3.x 3.3

`iterable`° `yield from`

```
def foob(x):
    yield from range(x * 2)
    yield from range(2)

list(foob(5)) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1]
```

°

```
def fibto(n):
    a, b = 1, 1
```

```

while True:
    if a >= n: break
    yield a
    a, b = b, a + b

def usefib():
    yield from fibto(10)
    yield from fibto(20)

list(usefib()) # [1, 1, 2, 3, 5, 8, 1, 1, 2, 3, 5, 8, 13]

```

```

# create and advance generator to the first yield
def coroutine(func):
    def start(*args,**kwargs):
        cr = func(*args,**kwargs)
        next(cr)
        return cr
    return start

# example coroutine
@coroutine
def adder(sum = 0):
    while True:
        x = yield sum
        sum += x

# example use
s = adder()
s.send(1) # 1
s.send(2) # 3

```

◦ ◦

```

from os import listdir
from os.path import isfile, join, exists

```

```

def get_files(path):
    for file in listdir(path):
        full_path = join(path, file)
        if isfile(full_path):
            if exists(full_path):
                yield full_path

```

```

def get_directories(path):
    for directory in listdir(path):
        full_path = join(path, directory)
        if not isfile(full_path):
            if exists(full_path):
                yield full_path

```

```

def get_files_recursive(directory):
    for file in get_files(directory):
        yield file
    for subdirectory in get_directories(directory):
        for file in get_files_recursive(subdirectory): # here the recursive call
            yield file

```



yield from

```
def get_files_recursive(directory):
    yield from get_files(directory)
    for subdirectory in get_directories(directory):
        yield from get_files_recursive(subdirectory)
```

zip

```
for x, y in zip(a,b):
    print(x,y)
```

```
1 x
2 y
3 z
```

python 2 [itertools.izip](#) ◦ [zip](#) ◦

iterables [zip](#) ◦ [itertools.zip\\_longest\(\)](#) ◦

```
def create():
    result = []
    # logic here...
    result.append(value) # possibly in several places
    # more logic...
    return result # possibly in several places
```

```
values = create()
```

```
def create_gen():
    # logic...
    yield value
    # more logic
    return # not needed if at the end of the function, of course
```

```
values = list(create_gen())
```

yield from `"""`

```
def preorder_traversal(node):
    yield node.value
    for child in node.children:
        yield from preorder_traversal(child)
```

next ◦ next ◦

```
def find_and_transform(sequence, predicate, func):
    for element in sequence:
        if predicate(element):
            return func(element)
    raise ValueError

item = find_and_transform(my_sequence, my_predicate, my_func)
```

```
item = next(my_func(x) for x in my_sequence if my_predicate(x))
# StopIteration will be raised if there are no matches; this exception can
# be caught and transformed, if desired.
```

```
first = next
```

```
def first(generator):
    try:
        return next(generator)
    except StopIteration:
        raise ValueError
```

<https://riptutorial.com/zh-TW/python/topic/292/>

# 162: urllib

## Examples

### HTTP GET

Python 2.x 2.7

### Python 2

```
import urllib
response = urllib.urlopen('http://stackoverflow.com/documentation/')
```

`urllib.urlopen()` ◦

```
print response.code
# Prints: 200
```

`response.code` `http` ◦ `200404NotFound` ◦

```
print response.read()
'<!DOCTYPE html>\r\n<html>\r\n<head>\r\n\r\n<title>Documentation - Stack. etc'
```

`response.read()` `response.readlines()` `html` ◦ `file.read*`

Python 3.x 3.0

### Python 3

```
import urllib.request

print(urllib.request.urlopen("http://stackoverflow.com/documentation/"))
# Prints: <http.client.HTTPResponse at 0x7f37a97e3b00>

response = urllib.request.urlopen("http://stackoverflow.com/documentation/")

print(response.code)
# Prints: 200
print(response.read())
# Prints: b'<!DOCTYPE html>\r\n<html>\r\n<head>\r\n\r\n<title>Documentation - Stack
Overflow</title>
```

Python 3.x ◦ `urllib.request.urlopen` ◦

### HTTP POST

POSTurlopen

## Python 2.x 2.7

# Python 2

```
import urllib
query_parms = {'username':'stackoverflow', 'password':'me.me'}
encoded_parms = urllib.urlencode(query_parms)
response = urllib.urlopen("https://stackoverflow.com/users/login", encoded_parms)
response.code
# Output: 200
response.read()
# Output: '<!DOCTYPE html>\r\n<html>\r\n<head>\r\n\r\n<title>Log In - Stack Overflow'
```

## Python 3.x 3.0

# Python 3

```
import urllib
query_parms = {'username':'stackoverflow', 'password':'me.me'}
encoded_parms = urllib.parse.urlencode(query_parms).encode('utf-8')
response = urllib.request.urlopen("https://stackoverflow.com/users/login", encoded_parms)
response.code
# Output: 200
response.read()
# Output: b'<!DOCTYPE html>\r\n<html>....etc'
```

## Python 3.x 3.0

```
import urllib.request

response = urllib.request.urlopen("http://stackoverflow.com/")
data = response.read()

encoding = response.info().get_content_charset()
html = data.decode(encoding)
```

## Python 2.x 2.7

```
import urllib2
response = urllib2.urlopen("http://stackoverflow.com/")
data = response.read()

encoding = response.info().getencoding()
html = data.decode(encoding)
```

[urllib](https://riptutorial.com/zh-TW/python/topic/2645/urllib) <https://riptutorial.com/zh-TW/python/topic/2645/urllib>

# 163: WebSockets

## Examples

### aiohhttp

[aiohhttp](#) [websockets](#).

### Python 3.x 3.5

```
import asyncio
from aiohttp import ClientSession

with ClientSession() as session:
    async def hello_world():

        websocket = await session.ws_connect("wss://echo.websocket.org")

        websocket.send_str("Hello, world!")

        print("Received:", (await websocket.receive()).data)

        await websocket.close()

loop = asyncio.get_event_loop()
loop.run_until_complete(hello_world())
```

### aiohhttp

[aiohttp](#).[ClientSession](#) [WebSocket](#).

### Python 3.x 3.5

```
import asyncio
from aiohttp import ClientSession

class EchoWebSocket(ClientSession):

    URL = "wss://echo.websocket.org"

    def __init__(self):
        super().__init__()
        self.websocket = None

    async def connect(self):
        """Connect to the WebSocket."""
        self.websocket = await self.ws_connect(self.URL)

    async def send(self, message):
        """Send a message to the WebSocket."""
        assert self.websocket is not None, "You must connect first!"
        self.websocket.send_str(message)
        print("Sent:", message)
```

```

async def receive(self):
    """Receive one message from the WebSocket."""
    assert self.websocket is not None, "You must connect first!"
    return (await self.websocket.receive()).data

async def read(self):
    """Read messages from the WebSocket."""
    assert self.websocket is not None, "You must connect first!"

    while self.websocket.receive():
        message = await self.receive()
        print("Received:", message)
        if message == "Echo 9!":
            break

async def send(websocket):
    for n in range(10):
        await websocket.send("Echo {}".format(n))
        await asyncio.sleep(1)

loop = asyncio.get_event_loop()

with EchoWebSocket() as websocket:

    loop.run_until_complete(websocket.connect())

    tasks = (
        send(websocket),
        websocket.read()
    )

    loop.run_until_complete(asyncio.wait(tasks))

    loop.close()

```

## WebSocket

AutobahnPython Web。

[Python Autobahn](#)

terminal

Linux

```
sudo pip install autobahn
```

Windows

```
python -m pip install autobahn
```

Pythonecho

```

from autobahn.asyncio.websocket import WebSocketServerProtocol
class MyServerProtocol(WebSocketServerProtocol):

```

```

'''When creating server protocol, the
user defined class inheriting the
WebSocketServerProtocol needs to override
the onMessage, onConnect, et-c events for
user specified functionality, these events
define your server's protocol, in essence'''
def onMessage(self,payload,isBinary):
    '''The onMessage routine is called
when the server receives a message.
It has the required arguments payload
and the bool isBinary. The payload is the
actual contents of the "message" and isBinary
is simply a flag to let the user know that
the payload contains binary data. I typically
elsewise assume that the payload is a string.
In this example, the payload is returned to sender verbatim.'''
    self.sendMessage(payload,isBinary)
if __name__=='__main__':
    try:
        import asyncio
    except ImportError:
        '''Trollius = 0.3 was renamed'''
        import trollius as asyncio
    from autobahn.asyncio.websocket import WebSocketServerFactory
    factory=WebSocketServerFactory()
    '''Initialize the websocket factory, and set the protocol to the
above defined protocol(the class that inherits from
autobahn.asyncio.websocket.WebSocketServerProtocol)'''
    factory.protocol=MyServerProtocol
    '''This above line can be thought of as "binding" the methods
onConnect, onMessage, et-c that were described in the MyServerProtocol class
to the server, setting the servers functionality, ie, protocol'''
    loop=asyncio.get_event_loop()
    coro=loop.create_server(factory,'127.0.0.1',9000)
    server=loop.run_until_complete(coro)
    '''Run the server in an infinite loop'''
    try:
        loop.run_forever()
    except KeyboardInterrupt:
        pass
    finally:
        server.close()
        loop.close()

```

9000localhost127.0.0.1。 IP。 LAN。 WAN IP9000WebSocketWAN IP。

9000。

**WebSockets** <https://riptutorial.com/zh-TW/python/topic/4751/websockets>

# 164:

Python。 Python。

numbers

	numbers.Number	numbers.Integral	numbers.Rational	numbers.Real	numbers.Complex
	✓	✓	✓	✓	✓
INT	✓	✓	✓	✓	✓
fractions.Fraction	✓	-	✓	✓	✓
	✓	-	-	✓	✓
	✓	-	-	-	✓
decimal.Decimal	✓	-	-	-	-

## Examples

```
a, b = 1, 2

# Using the "+" operator:
a + b          # = 3

# Using the "in-place" "+=" operator to add and assign:
a += b         # a = 3 (equivalent to a = a + b)

import operator          # contains 2 argument arithmetic functions for the examples

operator.add(a, b)      # = 5 since a is set to 3 right before this line

# The "+=" operator is equivalent to:
a = operator.iadd(a, b) # a = 5 since a is set to 3 right before this line
```

- intint int
- intfloat float
- intcomplex complex
- floatfloat float
- floatcomplex complex
- complexcomplex complex

+



```
"first string " + "second string"    # = 'first string second string'
[1, 2, 3] + [4, 5, 6]                # = [1, 2, 3, 4, 5, 6]
```

```
a, b = 1, 2
# Using the "-" operator:
b - a                                # = 1
import operator                      # contains 2 argument arithmetic functions
operator.sub(b, a)                   # = 1
```

- intint int
- intfloat float
- intcomplex complex
- floatfloat float
- floatcomplex complex
- complexcomplex complex

```
a, b = 2, 3
a * b                                # = 6
import operator
operator.mul(a, b)                   # = 6
```

- intint int
- intfloat float
- intcomplex complex
- floatfloat float
- floatcomplex complex
- complexcomplex complex

\*

```
3 * 'ab' # = 'ababab'
3 * ('a', 'b') # = ('a', 'b', 'a', 'b', 'a', 'b')
```

Python。 PythonPython 2.x3.x。

```
a, b, c, d, e = 3, 2, 2.0, -3, 10
```

Python 2.x 2.7

Python 2'/'。

```
a / b                                # = 1
```

```
a / c          # = 1.5
d / b          # = -2
b / a          # = 0
d / e          # = -1
```

abint int ◦

◦

ca / cfloat ◦

```
import operator      # the operator module provides 2-argument arithmetic functions
operator.div(a, b)   # = 1
operator.__div__(a, b) # = 1
```

## Python 2.x 2.2

```
from __future__ import division # applies Python 3 style division to the entire module
a / b          # = 1.5
a // b         # = 1
```

```
a / (b * 1.0)      # = 1.5
1.0 * a / b        # = 1.5
a / b * 1.0        # = 1.0      (careful with order of operations)

from operator import truediv
truediv(a, b)      # = 1.5
```

## TypeError

```
float(a) / b       # = 1.5
a / float(b)       # = 1.5
```

## Python 2.x 2.2

### Python 2'/'◦

```
a // b            # = 1
a // c            # = 1.0
```

## Python 3.x 3.0

### Python 3 /'true'◦ //◦

```
a / b            # = 1.5
e / b            # = 5.0
a // b           # = 1
a // c           # = 1.0
```

```
import operator          # the operator module provides 2-argument arithmetic functions
operator.truediv(a, b)  # = 1.5
operator.floordiv(a, b) # = 1
operator.floordiv(a, c) # = 1.0
```

- `int``int` Python 2 `int` Python 3 `float`
- `int``float` `float`
- `int``complex` `complex`
- `float``float` `float`
- `float``complex` `complex`
- `complex``complex` `complex`

## PEP 238 ◦

```
a, b = 2, 3

(a ** b)          # = 8
pow(a, b)         # = 8

import math
math.pow(a, b)    # = 8.0 (always float; does not allow complex results)

import operator
operator.pow(a, b) # = 8
```

`pow``math.pow``pow`

```
a, b, c = 2, 3, 2

pow(2, 3, 2)      # 0, calculates (2 ** 3) % 2, but as per Python docs,
                  # does so more efficiently
```

`math.sqrt(x)``x` ◦

```
import math
import cmath
c = 4
math.sqrt(c)      # = 2.0 (always float; does not allow complex results)
cmath.sqrt(c)     # = (2+0j) (always complex)
```

◦ ◦

```
import math
x = 8
math.pow(x, 1/3) # evaluates to 2.0
x**(1/3) # evaluates to 2.0
```

`math.exp(x)``e ** x` ◦

```
math.exp(0) # 1.0
```

```
math.exp(1) # 2.718281828459045 (e)
```

`math.expm1(x)`  $e^{**x} - 1$   $\circ$  `cmath.exp(x) - 1`

```
math.expm1(0) # 0.0

math.exp(1e-6) - 1 # 1.0000004999621837e-06
math.expm1(1e-6) # 1.0000005000001665e-06
# exact result # 1.000000500000166666708333341666...
```

`math.log`  $e^{\circ}$  `base`  $\circ$

```
import math
import cmath

math.log(5) # = 1.6094379124341003
# optional base argument. Default is math.e
math.log(5, math.e) # = 1.6094379124341003
cmath.log(5) # = (1.6094379124341003+0j)
math.log(1000, 10) # 3.0 (always returns float)
cmath.log(1000, 10) # (3+0j)
```

`math.log`  $\circ$

```
# Logarithm base e - 1 (higher precision for low values)
math.log1p(5) # = 1.791759469228055

# Logarithm base 2
math.log2(8) # = 3.0

# Logarithm base 10
math.log10(100) # = 2.0
cmath.log10(100) # = (2+0j)
```

```
a = a + 1
```

```
a = a * 2
```

```
a += 1
# and
a *= 2
```

'='

- -=
- +=
- \*=
- /=
- // = floorPython 3
- %=
- \*\* =

^ | etc

```
a, b = 1, 2

import math

math.sin(a) # returns the sine of 'a' in radians
# Out: 0.8414709848078965

math.cosh(b) # returns the inverse hyperbolic cosine of 'b' in radians
# Out: 3.7621956910836314

math.atan(math.pi) # returns the arc tangent of 'pi' in radians
# Out: 1.2626272556789115

math.hypot(a, b) # returns the Euclidean norm, same as math.sqrt(a*a + b*b)
# Out: 2.23606797749979
```

```
math.hypot(x, y) (0, 0) (x, y)
(x1, y1) (x2, y2) (x2, y2) math.hypot
math.hypot(x2-x1, y2-y1)
```

-> -> math.degrees math.radians

```
math.degrees(a)
# Out: 57.29577951308232

math.radians(57.29577951308232)
# Out: 1.0
```

## Python %

```
3 % 4 # 3
10 % 2 # 0
6 % 4 # 2
```

operator

```
import operator

operator.mod(3, 4) # 3
operator.mod(10, 2) # 0
operator.mod(6, 4) # 2
```

o

```
-9 % 7 # 5
9 % -7 # -5
-9 % -7 # -2
```

divmod

```
quotient, remainder = divmod(9, 4)
# quotient = 2, remainder = 1 as 4 * 2 + 1 == 9
```

<https://riptutorial.com/zh-TW/python/topic/298/>

# 165:

- OBJ []
- 
- []

Parameter	
obj	"""
start	objPythonobj0 ° 0 °
stop	obj° len(obj) °
step	step° 1 °

unicode1°

[start, end)° len(x[:n]) = nlen(x) >= n x[n:n+1]x[n:n+1] = [x[n]]xlen(x) >= n°

## Examples

Python°

```
iterable_name[start:stop:step]
```

- start° 0
- stop° len
- step

```
a = "abcdef"
a          # "abcdef"
           # Same as a[:] or a[::] since it uses the defaults for all three indices
a[-1]     # "f"
a[:]      # "abcdef"
a[::]     # "abcdef"
a[3:]     # "def" (from index 3, to end(defaults to size of iterable))
a[:4]     # "abcd" (from beginning(default 0) to position 4 (excluded))
a[2:4]    # "cd" (from position 2, to position 4 (excluded))
```

```
a[::2]    # "ace" (every 2nd element)
a[1:4:2]  # "bd" (from index 1, to index 4 (excluded), every 2nd element)
```

```
a[:-1]   # "abcde" (from index 0 (default), to the second last element (last element - 1))
a[:-2]   # "abcd" (from index 0 (default), to the third last element (last element -2))
a[-1:]   # "f" (from the last element to the end (default len()))
```

```
a[3:1:-1] # "dc" (from index 2 to None (default), in reverse order)
```

```
a[::-1] # "fedcba" (from last element (default len()-1), to first, in reverse order(-1))
```

end\_indexNone <http://stackoverflow.com/a/12521981>

```
a[5:None:-1] # "fedcba" (this is equivalent to a[::-1])
a[5:0:-1] # "fedcb" (from the last element (index 5) to second element (index 1))
```

```
arr[:]
```

◦ [:]start endslice ◦ 0 len(arr)1 arrarr◦

```
arr = ['a', 'b', 'c']
copy = arr[:]
arr.append('d')
print(arr) # ['a', 'b', 'c', 'd']
print(copy) # ['a', 'b', 'c']
```

```
arr.append('d')darr copy
```

```
arr.copy()◦
```

str listtuple **step**◦

```
s = 'reverse me!'
s[::-1] # '!em esrever'
```

◦ [::-1]startend -1◦

**\_\_getitem\_\_** **\_\_setitem\_\_** **\_\_delitem\_\_**

```
class MultiIndexingList:
    def __init__(self, value):
        self.value = value

    def __repr__(self):
        return repr(self.value)

    def __getitem__(self, item):
        if isinstance(item, (int, slice)):
            return self.__class__(self.value[item])
        return [self.value[i] for i in item]

    def __setitem__(self, item, value):
        if isinstance(item, int):
            self.value[item] = value
        elif isinstance(item, slice):
            raise ValueError('Cannot interpret slice with multiindexing')
        else:
            for i in item:
                if isinstance(i, slice):
```



```

        raise ValueError('Cannot interpret slice with multiindexing')
        self.value[i] = value

def __delitem__(self, item):
    if isinstance(item, int):
        del self.value[item]
    elif isinstance(item, slice):
        del self.value[item]
    else:
        if any(isinstance(elem, slice) for elem in item):
            raise ValueError('Cannot interpret slice with multiindexing')
        item = sorted(item, reverse=True)
        for elem in item:
            del self.value[elem]

```

```

a = MultiIndexingList([1,2,3,4,5,6,7,8])
a
# Out: [1, 2, 3, 4, 5, 6, 7, 8]
a[1,5,2,6,1]
# Out: [2, 6, 3, 7, 2]
a[4, 1, 5:, 2, ::2]
# Out: [5, 2, [6, 7, 8], 3, [1, 3, 5, 7]]
#      4|1-|----50:---|2-|-----::2----- <-- indicated which element came from which index

```

```

a[4] = 1000
a
# Out: [1, 2, 3, 4, 1000, 6, 7, 8]
a[2,6,1] = 100
a
# Out: [1, 100, 100, 4, 1000, 6, 100, 8]
del a[5]
a
# Out: [1, 100, 100, 4, 1000, 100, 8]
del a[4,2,5]
a
# Out: [1, 100, 4, 8]

```

## ◦ Python◦

```

lst = [1, 2, 3]
lst[1:3] = [4, 5]
print(lst) # Out: [1, 4, 5]

```

```

lst = [1, 2, 3, 4, 5]
lst[1:4] = [6]
print(lst) # Out: [1, 6, 5]

```

```

lst = [1, 2, 3]
lst[:] = [4, 5, 6]
print(lst) # Out: [4, 5, 6]

```

```

lst = [1, 2, 3]
lst[-2:] = [4, 5, 6]
print(lst) # Out: [1, 4, 5, 6]

```

slice()◦ ◦

```
>>> programmer_1 = [ 1956, 'Guido', 'van Rossum', 'Python', 'Netherlands']
>>> programmer_2 = [ 1815, 'Ada', 'Lovelace', 'Analytical Engine', 'England']
>>> name_columns = slice(1, 3)
>>> programmer_1[name_columns]
['Guido', 'van Rossum']
>>> programmer_2[name_columns]
['Ada', 'Lovelace']
```

## Python0 0

```
arr = ['a', 'b', 'c', 'd']
print(arr[0])
>> 'a'
```

12

```
print(arr[1])
>> 'b'
print(arr[2])
>> 'c'
```

◦ ◦ **index -1** **index -2**

```
print(arr[-1])
>> 'd'
print(arr[-2])
>> 'c'
```

IndexError

```
print arr[6]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

<https://riptutorial.com/zh-TW/python/topic/289/>

# 166:

- `empty_set = set`
- `literal_set = {'foo' 'bar' 'baz'}3`
- `set_from_list = set(['foo' 'bar' 'baz'])set`
- `set_from_iter = setxx30`
- `set_from_iter = {x for x in [random.randint(0,10)for i in range(10)]}`alternative notation

O1. . . .

- `frozenset` ◦
- `__hash__` ◦ `eq` ◦ `listset` ◦ `dict` ◦ `immutable` ◦

## Examples

- . .

```
restaurants = ["McDonald's", "Burger King", "McDonald's", "Chicken Chicken"]
unique_restaurants = set(restaurants)
print(unique_restaurants)
# prints {'Chicken Chicken', 'McDonald's', 'Burger King'}
```

; dict ◦

### Python list List

```
list(unique_restaurants)
# ['Chicken Chicken', 'McDonald's', 'Burger King']
```

```
# Removes all duplicates and returns another list
list(set(restaurants))
```

◦

```
# Intersection
{1, 2, 3, 4, 5}.intersection({3, 4, 5, 6}) # {3, 4, 5}
{1, 2, 3, 4, 5} & {3, 4, 5, 6} # {3, 4, 5}

# Union
{1, 2, 3, 4, 5}.union({3, 4, 5, 6}) # {1, 2, 3, 4, 5, 6}
{1, 2, 3, 4, 5} | {3, 4, 5, 6} # {1, 2, 3, 4, 5, 6}

# Difference
{1, 2, 3, 4}.difference({2, 3, 5}) # {1, 4}
{1, 2, 3, 4} - {2, 3, 5} # {1, 4}

# Symmetric difference with
{1, 2, 3, 4}.symmetric_difference({2, 3, 5}) # {1, 4, 5}
```

```

{1, 2, 3, 4} ^ {2, 3, 5}          # {1, 4, 5}

# Superset check
{1, 2}.issuperset({1, 2, 3}) # False
{1, 2} >= {1, 2, 3}          # False

# Subset check
{1, 2}.issubset({1, 2, 3}) # True
{1, 2} <= {1, 2, 3}        # True

# Disjoint check
{1, 2}.isdisjoint({3, 4}) # True
{1, 2}.isdisjoint({1, 4}) # False

```

```

# Existence check
2 in {1,2,3}      # True
4 in {1,2,3}      # False
4 not in {1,2,3}  # True

# Add and Remove
s = {1,2,3}
s.add(4)          # s == {1,2,3,4}

s.discard(3)     # s == {1,2,4}
s.discard(5)     # s == {1,2,4}

s.remove(2)      # s == {1,4}
s.remove(2)      # KeyError!

```

	$s   = t$	
	$s = t$	intersection_update
	$s - = t$ difference_update	
symmetric_difference	$s ^ = t$	symmetric_difference_update

```

s = {1, 2}
s.update({3, 4}) # s == {1, 2, 3, 4}

```

◦ ◦

```

>>> setA = {'a','b','b','c'}
>>> setA
set(['a', 'c', 'b'])

```

'a' 'b' 'b' 'c''b'◦

```

>>> listA = ['a','b','b','c']
>>> listA
['a', 'b', 'b', 'c']

```

◦

## PythoncollectionsCounter2.7

### Python 2.x 2.7

```
>>> from collections import Counter
>>> counterA = Counter(['a','b','b','c'])
>>> counterA
Counter({'b': 2, 'a': 1, 'c': 1})
```

Counter◦ ◦

## MethodsBuiltins

ab

```
>>> a = {1, 2, 2, 3, 4}
>>> b = {3, 3, 4, 4, 5}
```

{1}{ }dict ◦ set() ◦

---

a.intersection(b) ab

```
>>> a.intersection(b)
{3, 4}
```

---

a.union(b) ab

```
>>> a.union(b)
{1, 2, 3, 4, 5}
```

---

a.difference(b) ab

```
>>> a.difference(b)
{1, 2}
>>> b.difference(a)
{5}
```

---

a.symmetric\_difference(b) ab

```
>>> a.symmetric_difference(b)
{1, 2, 5}
>>> b.symmetric_difference(a)
{1, 2, 5}
```

```
a.symmetric_difference(b) == b.symmetric_difference(a)
```

```
c.issubset(a) ca °
```

```
a.issuperset(c) ca °
```

```
>>> c = {1, 2}
>>> c.issubset(a)
True
>>> a.issuperset(c)
True
```

<code>a.intersection(b)</code>	<code>a &amp; b</code>
<code>a.union(b)</code>	<code>a   b</code>
<code>a.difference(b)</code>	<code>a - b</code>
<code>a.symmetric_difference(b)</code>	<code>a ^ b</code>
<code>a.issubset(b)</code>	<code>a &lt;= b</code>
<code>a.issuperset(b)</code>	<code>a &gt;= b</code>

```
ad ad °
```

```
>>> d = {5, 6}
>>> a.isdisjoint(b) # {2, 3, 4} are in both sets
False
>>> a.isdisjoint(d)
True

# This is an equivalent check, but less efficient
>>> len(a & d) == 0
True

# This is even less efficient
>>> a & d == set()
True
```

```
in occurrences
```

```
>>> 1 in a
True
>>> 6 in a
False
```

len()

```
>>> len(a)
4
>>> len(b)
3
```

```
{{1,2}, {3,4}}
```

```
TypeError: unhashable type: 'set'
```

frozenset

```
{frozenset({1, 2}), frozenset({3, 4})}
```

<https://riptutorial.com/zh-TW/python/topic/497/>

# 167:

## Examples

unicode ◦ ◦

```
>>> u'𐀀'.encode('utf-8')  
'\xf0\x9f\x90\x8d'
```

unicode ◦ ◦

```
>>> b'\xf0\x9f\x90\x8d'.decode('utf-8')  
u'\U0001f40d'
```

<https://riptutorial.com/zh-TW/python/topic/5618/>



# 168:

## Examples

- IndentationError◦ IndentationError

```
a = 7
if a > 5:
    print "foo"
else:
    print "bar"
print "done"
```

IndentationError

```
if True:
print "true"
```

IndentationError

```
if True:
    a = 6
        b = 5
```

- NoneFalse

```
def isEven(a):
    if a%2 ==0:
        return True
        #this next line should be even with the if
        return False
print isEven(7)
```

Python Guido van Rossum◦ “Python”◦ :

```
class ExampleClass:
    #Every function belonging to a class must be indented equally
    def __init__(self):
        name = "example"

    def someFunction(self, a):
        #Notice everything belonging to a function must be indented
        if a > 5:
            return True
        else:
            return False

#If a function is not indented to the same level it will not be considers as part of the
parent class
def separateFunction(b):
    for i in b:
```

```

#Loops are also indented and nested conditions start a new indentation
    if i == 1:
        return True
return False

separateFunction([2,3,5,6,1])

```

#### 4. Python Python 3 Python 2.

o

o 0. "INDENT". "INDENT" IndentationError o

o "DEDENT". "DEDENT".

"INDENT" "DEDENT".

"DEDENT"0.

```

if foo:
    if bar:
        x = 42
else:
    print foo

```

```

<if> <foo> <:> [0]
<INDENT> <if> <bar> <:> [0, 4]
<INDENT> <x> <=> <42> [0, 4, 8]
<DEDENT> <DEDENT> <else> <:> [0]
<INDENT> <print> <foo> [0, 2]
<DEDENT>

```

"INDENT" "DEDENT".

<https://riptutorial.com/zh-TW/python/topic/2597/>

---

# 169:

Stack Overflow "ImportError: No module named '?????'" SyntaxError: invalid syntaxNameError: name '???' is not defined. ◦ ◦

## Examples

### IndentationErrors

PythonFORTRANMakefilesWhitespace.◦

---

## IndentationError / SyntaxError

◦

Python 2.x 2.0 2.7

```
print "This line is ok"
    print "This line isn't ok"
```

Python 3.x 3.0

```
print("This line is ok")
    print("This line isn't ok")
```

◦

Python 2.x 2.0 2.7

```
print "This line is ok"
print "This line isn't ok"
```

Python 3.x 3.0

```
print("This line is ok")
print("This line isn't ok")
```

---

## IndentationError / SyntaxErrorunindent

◦

Python 2.x 2.0 2.7

```
def foo():
```

```
print "This should be part of foo()"
print "ERROR!"
print "This is not a part of foo()"
```

## Python 3.x 3.0

```
print("This line is ok")
print("This line isn't ok")
```

---

# IndentationError

◦ ◦

```
if ok:
doStuff()
```

pass if else except class methoddefinition/except

```
def foo():
    pass
```

---

# IndentationError

```
def foo():
    if ok:
        return "Two != Four != Tab"
        return "i dont care i do whatever i want"
```

◦ PEP8 Python ◦

- 1. 4◦
- 2. 4◦
- 3. 8◦

◦

## TypeErrors

---

# TypeError[/]

◦

```
def foo(a): return a
foo(a,b,c,d) #And a,b,c,d are defined
```

```
def foo(a,b,c,d): return a += b + c + d
foo(a) #And a is defined
```

\*args\*\*kwargs ◦ \* args\*\* kwargs

---

## TypeError[]'???'???'

◦

+◦ + ing 'set1''tuple1'setset◦

```
set1, tuple1 = {1,2}, (3,4)
a = set1 + tuple1
```

intstring +

```
b = 400 + 'foo'
```

```
c = ["a","b"] - [1,2]
```

intfloat

```
d = 1 + 1.0
```

---

## TypeError'???' object/

IndexError \_\_iter\_\_ \_\_iterator\_\_ \_\_getitem\_\_◦

bar1

```
foo = 1
bar = foo[0]
```

forxamount [0] amountint

```
amount = 10
for x in amount: print(x)
```

---

# TypeError'???'

```
foo = "notAFunction"  
foo()
```

## NameErrorname'???'

```
◦ ◦ import ◦ ◦
```

```
foo # This variable is not defined  
bar() # This function is not defined
```

```
baz()  
  
def baz():  
    pass
```

### import

```
#needs import math  
  
def sqrt():  
    x = float(input("Value: "))  
    return math.sqrt(x)
```

## PythonLEGB

### LEGBPython◦

```
Local → Enclosed → Global → Built-in.
```

- **L**ocal◦
- **E**nclosing◦
- **G**lobal◦
- **B**uilt-in◦

```
for i in range(4):  
    d = i * 2  
print(d)
```

### dfor

```
def noaccess():  
    for i in range(4):  
        d = i * 2
```

```
noaccess()
print(d)
```

PythonNameError: name 'd' is not defined

---

## AssertError

assert◦

```
assert condition
```

```
assert condition, message
```

```
if __debug__:
    if not condition: raise AssertionError(message)
```

◦

**debug** TrueFalse-O◦ ◦ ◦

---

## KeyboardInterrupt

Ctrl + Cdel◦

---

## ZeroDivisionError

1/0◦

Python 2.x 2.0 2.7

```
div = float(raw_input("Divisors of: "))
for x in xrange(div+1): #includes the number itself and zero
    if div/x == div//x:
        print x, "is a divisor of", div
```

Python 3.x 3.0

```
div = int(input("Divisors of: "))
for x in range(div+1): #includes the number itself and zero
    if div/x == div//x:
        print(x, "is a divisor of", div)
```

ZeroDivisionErrorforx◦

Python 2.x 2.0 2.7

```
div = float(raw_input("Divisors of: "))
for x in xrange(1,div+1): #includes the number itself but not zero
    if div/x == div//x:
        print x, "is a divisor of", div
```

## Python 3.x 3.0

```
div = int(input("Divisors of: "))
for x in range(1,div+1): #includes the number itself but not zero
    if div/x == div//x:
        print(x, "is a divisor of", div)
```

## SyntaxError

```
def my_print():
    x = (1 + 1
    print(x)
```

```
File "<input>", line 3
    print(x)
      ^
SyntaxError: invalid syntax
```

/o

## Python 3

### Python 3.x 3.0

```
>>> print "hello world"
File "<stdin>", line 1
    print "hello world"
      ^
SyntaxError: invalid syntax
```

`printprint()`

```
print("hello world") # Note this is valid for both Py2 & Py3
```

<https://riptutorial.com/zh-TW/python/topic/9300/>



---

# 170:

Python。 “Project X1.xProject Y4.x”。

- 
- 

1. root
- 2.
3. python

## Examples

virtualenvPython。 Python。

---

# virtualenv

- virtualenv Debianpython-virtualenvpython3-virtualenv ◦

pipvirtualenv

```
$ pip install virtualenv
```

- 

```
$ virtualenv foo
```

pythonfoo◦ ◦ virtualenv◦ ◦

---

shellPythonfoo◦ activateshell

```
$ source foo/bin/activate
```

## Windows

```
$ foo\Scripts\activate.bat
```

pythonpipfoo◦ pip◦ ◦

```
# Installs 'requests' to foo only, not globally
(foo)$ pip install requests
```

pipfreeze◦ installPython◦ requirements.txt

```
(foo)$ pip freeze > requirements.txt
(foo)$ pip install -r requirements.txt
```

freeze◦ requirements.txt ◦

---

## shell

```
(foo)$ deactivate
```

---

\$ source bin/activate virtualenvmod\_wsgiAmazon API GatewayGoogle AppEngine◦ virtualenv  
sys.path ◦

virtualenvsys.pathsys.prefix

```
import os

mydir = os.path.dirname(os.path.realpath(__file__))
activate_this = mydir + '/bin/activate_this.py'
execfile(activate_this, dict(__file__=activate_this))
```

◦

virtualenvbin/activate\_this.py lib/python2.7/site-packagessys.path

activate\_this.pybinlib/python2.7/site-packages◦

---

## Python 3.x 3.3

Python 3.3 venv◦ pyvenv

```
$ pyvenv foo
$ source foo/bin/activate
```

```
$ python3 -m venv foo
$ source foo/bin/activate
```

virtualenv◦ root◦

virtualenv

```
(<Virtualenv Name> $ which python
/<Virtualenv Directory>/bin/python
```

```
(Virtualenv Name) $ which pip
/<Virtualenv Directory>/bin/pip
```

**pip**virtualenv

```
/<Virtualenv Directory>/lib/python2.7/site-packages/
```

◦

## requirements.txt

```
requests==2.10.0
```

```
# Install packages from requirements.txt
pip install -r requirements.txt
```

**2.10.0**requests ◦

```
# Get a list of installed packages
pip freeze
```

```
# Output list of packages and versions into a requirements.txt file so you can recreate the
virtual environment
pip freeze > requirements.txt
```

◦ **pip**◦

```
$ /<Virtualenv Directory>/bin/pip install requests
```

**pip**PIP ◦

- ◦

## python

pythonpython3 python3PythonPython 3

```
virtualenv -p python3 foo
```

```
virtualenv --python=python3 foo
```

```
python3 -m venv foo
```

```
pyvenv foo
```

**python**◦ /usr/bin//usr/local/bin/ **Linux**/Library/Frameworks/Python.framework/Versions/XX/bin/  
**python**--python-p◦

# virtualenvwrapper

[virtualenvwrapper/](#)◦

virtualenvwrapper~/.virtualenvs ◦

virtualenvwrapper ◦

## Debian / Ubuntu

```
apt-get install virtualenvwrapper
```

## Fedora/ CentOS/ RHEL

```
yum install python-virtualenvwrapper
```

## Arch Linux

```
pacman -S python-virtualenvwrapper
```

## pipPyPI

```
pip install virtualenvwrapper
```

## Windows[virtualenvwrapper-winvirtualenvwrapper-powershell](#)◦

mkvirtualenv◦ virtualenv◦

```
mkvirtualenv my-project
```

```
mkvirtualenv --system-site-packages my-project
```

◦ **shell**workon

```
workon my-project
```

workon. path/to/my-env/bin/activate workon;◦

-asetvirtualenvproject◦

```
mkvirtualenv -a /path/to/my-project my-project
```

```
workon my-project
cd /path/to/my-project
setvirtualenvproject
```

workoncdproject◦

virtualenvwrappervirtualenvlsvirtualenv◦

virtualenvrmvirtualenv

```
rmvirtualenv my-project
```

virtualenvwrappervirtualenv4bash preactivate postactivate predeactivatepostdeactivate◦

virtualenvbash; postactivatevirtualenv◦ ◦ 4.virtualenvs/<virtualenv\_name>/bin/◦

virtualenvwrapper◦

Linuxbash◦

```
(my-project-env) user@hostname:~$ which python
/home/user/my-project-env/bin/python
```

## Unix / Linuxpython

Linux shellpythonPythonshbang#!#!

```
#!/usr/bin/python
```

python myscript.pyPython./myscript.py#!#!◦ Python

```
#!/usr/bin/env python
```

shbang

```
chmod +x myscript.py
```

./myscript.py python myscript.pypython3 myscript.py◦

## virtualenv

virtualenvvirtualenvwrapper◦ virtualfish◦ virtualenvFish shell◦

- virtualfish

```
sudo pip install virtualfish
```

- fish shellpythonvirtualfish

```
$ echo "eval (python -m virtualfish)" > ~/.config/fish/config.fish
```

- fish\_prompt\$ funced fish\_prompt --editor vimvim

```
if set -q VIRTUAL_ENV
    echo -n -s (set_color -b blue white) "(" (basename "$VIRTUAL_ENV") ")" (set_color
normal) " "
end
```

```
vim$ funced fish_prompt --editor nano$ funced fish_prompt --editor gedit
```

- funcsave

```
funcsave fish_prompt
```

- vf new

```
vf new my_new_env # Make sure $HOME/.virtualenv exists
```

- python3-p

```
vf new -p python3 my_new_env
```

- vf deactivate vf activate another\_env
- <https://github.com/adambrenecki/virtualfish>
- <http://virtualfish.readthedocs.io/en/latest/>

## Anaconda

virtualenv [Anaconda](#) pip◦ Anaconda

```
conda create --name <envname> python=<version>
```

<envname> <version> Python◦

```
# Linux, Mac
source activate <envname>
source deactivate
```

```
# Windows
activate <envname>
deactivate
```

```
conda env list
```

```
conda env remove -n <envname>
```

[conda](#)◦

shell◦

## python

```
import sys
sys.prefix
sys.real_prefix
```

- `sys.prefixpython``sys.real_prefix` ◦
- `sys.prefixpython` `sys.real_prefixpython` ◦

`venv``sys.real_prefix` ◦ `sys.base_prefix``sys.prefix` ◦

<https://riptutorial.com/zh-TW/python/topic/868/>

# 171: virtualenvwrapper

ABC.ABpython 3。 Cpython 2.7。

。

VirtualenvVirtualenvwrapperConda

virtualenvwrapper。

## Examples

### virtualenvwrapper

ABC.ABpython 3。 Cpython 2.7。

。

VirtualenvVirtualenvwrapperConda

virtualenvwrapper。

### virtualenvwrapper。

```
$ pip install virtualenvwrapper

$ export WORKON_HOME=~/.Envs
$ mkdir -p $WORKON_HOME
$ source /usr/local/bin/virtualenvwrapper.sh
$ printf '\n%s\n%s\n%s' '# virtualenv' 'export WORKON_HOME=~/.virtualenvs' 'source
/home/salayhin/bin/virtualenvwrapper.sh' >> ~/.bashrc
$ source ~/.bashrc

$ mkvirtualenv python_3.5
Installing
setuptools.....
.....
.....done.
virtualenvwrapper.user_scripts Creating /Users/salayhin/Envs/python_3.5/bin/predeactivate
virtualenvwrapper.user_scripts Creating /Users/salayhin/Envs/python_3.5/bin/postdeactivate
virtualenvwrapper.user_scripts Creating /Users/salayhin/Envs/python_3.5/bin/preactivate
virtualenvwrapper.user_scripts Creating /Users/salayhin/Envs/python_3.5/bin/postactivate New
python executable in python_3.5/bin/python

(python_3.5)$ ls $WORKON_HOME
python_3.5 hook.log
```

。

```
(python_3.5)$ pip install django
```



```
Downloading/unpacking django
Downloading Django-1.1.1.tar.gz (5.6Mb): 5.6Mb downloaded
Running setup.py egg_info for package django
Installing collected packages: django
Running setup.py install for django
changing mode of build/scripts-2.6/django-admin.py from 644 to 755
changing mode of /Users/salayhin/Envs/env1/bin/django-admin.py to 755
Successfully installed django
```

## lssitepackages

```
(python_3.5)$ lssitepackages
Django-1.1.1-py2.6.egg-info easy-install.pth
setuptools-0.6.10-py2.6.egg pip-0.6.3-py2.6.egg
django setuptools.pth
```

o

## workon

```
(python_3.6)$ workon python_3.5
(python_3.5)$ echo $VIRTUAL_ENV
/Users/salayhin/Envs/env1
(python_3.5)$
```

## virtualenv

```
$ deactivate
```

**virtualenvwrapper** <https://riptutorial.com/zh-TW/python/topic/9983/virtualenvwrapper>

# 172:

◦ ◦ ◦ Python◦

- `def decorator_function(f):`  
`def decorator_function`
- `@decorator_function`  
`def ..._function_decorator_function`
- `decorated_function = decorator_function(decorated_function)`  
`@decorator_function`



## Examples

◦ ◦

```
# This simplest decorator does nothing to the function being decorated. Such
# minimal decorators can occasionally be used as a kind of code markers.
def super_secret_function(f):
    return f

@super_secret_function
def my_function():
    print("This is my secret function.")
```

@ -notation

```
my_function = super_secret_function(my_function)
```

◦ “unsugared”◦

```
def disabled(f):
    """
    This function returns nothing, and hence removes the decorated function
    from the local scope.
    """
    pass

@disabled
def my_function():
    print("This function can no longer be called...")

my_function()
# TypeError: 'NoneType' object is not callable
```

◦ ◦ ◦

```

#This is the decorator
def print_args(func):
    def inner_func(*args, **kwargs):
        print(args)
        print(kwargs)
        return func(*args, **kwargs) #Call the original function with its arguments.
    return inner_func

@print_args
def multiply(num_a, num_b):
    return num_a * num_b

print(multiply(3, 5))
#Output:
# (3,5) - This is actually the 'args' that the function receives.
# {} - This is the 'kwargs', empty because we didn't specify keyword arguments.
# 15 - The result of the function.

```

◦ my\_func = decorator(my\_func) ◦ decorator◦ my\_func◦ decorator◦ \_\_call\_\_()my\_func

```

class Decorator(object):
    """Simple decorator class."""

    def __init__(self, func):
        self.func = func

    def __call__(self, *args, **kwargs):
        print('Before the function call.')
        res = self.func(*args, **kwargs)
        print('After the function call.')
        return res

@Decorator
def testfunc():
    print('Inside the function.')

testfunc()
# Before the function call.
# Inside the function.
# After the function call.

```

“”

```

import types
isinstance(testfunc, types.FunctionType)
# False
type(testfunc)
# <class '__main__.Decorator'>

```

## \_\_get\_\_ -method

```

from types import MethodType

class Decorator(object):
    def __init__(self, func):

```

```

        self.func = func

    def __call__(self, *args, **kwargs):
        print('Inside the decorator.')
        return self.func(*args, **kwargs)

    def __get__(self, instance, cls):
        # Return a Method if it is called on an instance
        return self if instance is None else MethodType(self, instance)

class Test(object):
    @Decorator
    def __init__(self):
        pass

a = Test()

```

◦

---

```

from types import MethodType

class CountCallsDecorator(object):
    def __init__(self, func):
        self.func = func
        self.ncalls = 0    # Number of calls of this method

    def __call__(self, *args, **kwargs):
        self.ncalls += 1  # Increment the calls counter
        return self.func(*args, **kwargs)

    def __get__(self, instance, cls):
        return self if instance is None else MethodType(self, instance)

class Test(object):
    def __init__(self):
        pass

    @CountCallsDecorator
    def do_something(self):
        return 'something was done'

a = Test()
a.do_something()
a.do_something.ncalls    # 1
b = Test()
b.do_something()
b.do_something.ncalls    # 2

```

◦ ◦ ◦ [functools.wraps](#)◦

---

```

from functools import wraps

```

◦ ◦

```

def decorator(func):
    # Copies the docstring, name, annotations and module to the decorator
    @wraps(func)
    def wrapped_func(*args, **kwargs):
        return func(*args, **kwargs)
    return wrapped_func

@decorator
def test():
    pass

test.__name__

```

"

```

class Decorator(object):
    def __init__(self, func):
        # Copies name, module, annotations and docstring to the instance.
        self._wrapped = wraps(func)(self)

    def __call__(self, *args, **kwargs):
        return self._wrapped(*args, **kwargs)

@Decorator
def test():
    """Docstring of test."""
    pass

test.__doc__

```

'.' '

o o

o o

```

def decoratorfactory(message):
    def decorator(func):
        def wrapped_func(*args, **kwargs):
            print('The decorator wants to tell you: {}'.format(message))
            return func(*args, **kwargs)
        return wrapped_func
    return decorator

@decoratorfactory('Hello World')
def test():
    pass

test()

```

Hello World

```
@decoratorfactory # Without parentheses
def test():
    pass

test()
```

## TypeErrordecorator1'func'

```
def decoratorfactory(*decorator_args, **decorator_kwargs):

    class Decorator(object):
        def __init__(self, func):
            self.func = func

        def __call__(self, *args, **kwargs):
            print('Inside the decorator with arguments {}'.format(decorator_args))
            return self.func(*args, **kwargs)

    return Decorator

@decoratorfactory(10)
def test():
    pass

test()
```

10

/o o

```
def singleton(cls):
    instance = [None]
    def wrapper(*args, **kwargs):
        if instance[0] is None:
            instance[0] = cls(*args, **kwargs)
        return instance[0]

    return wrapper
```

o o

```
@singleton
class SomeSingletonClass:
    x = 2
    def __init__(self):
        print("Created!")

instance = SomeSingletonClass() # prints: Created!
instance = SomeSingletonClass() # doesn't print anything
print(instance.x) # 2

instance.x = 3
print(SomeSingletonClass().x) # 3
```

“”。

```
import time
def timer(func):
    def inner(*args, **kwargs):
        t1 = time.time()
        f = func(*args, **kwargs)
        t2 = time.time()
        print 'Runtime took {0} seconds'.format(t2-t1)
        return f
    return inner

@timer
def example_function():
    #do stuff

example_function()
```

<https://riptutorial.com/zh-TW/python/topic/229/>

# 173:

## Examples

◦

```
>>> import copy
>>> c = [[1,2]]
>>> d = copy.copy(c)
>>> c is d
False
>>> c[0] is d[0]
True
```

◦ ◦

```
>>> import copy
>>> c = [[1,2]]
>>> d = copy.deepcopy(c)
>>> c is d
False
>>> c[0] is d[0]
False
```

◦

```
>>> l1 = [1,2,3]
>>> l2 = l1[:]      # Perform the shallow copy.
>>> l2
[1,2,3]
>>> l1 is l2
False
```

copy ◦ ◦

```
>>> d1 = {1:[]}
>>> d2 = d1.copy()
>>> d1 is d2
False
>>> d1[1] is d2[1]
True
```

copy ◦ ◦

```
>>> s1 = {}
>>> s2 = s1.copy()
>>> s1 is s2
False
>>> s2.add(3)
>>> s1
{[]}
```



```
>>> s2  
{3, []}
```

<https://riptutorial.com/zh-TW/python/topic/920/>

# 174:

- `cmath.rectAbsoluteValuePhase`

## Examples

cmath

```
import cmath
```

```
z = 2+3j # A complex number
cmath.phase(z) # 0.982793723247329
```

```
cmath.polar(z) # (3.605551275463989, 0.982793723247329)
cmath.rect(2, cmath.pi/2) # (0+2j)
```

- `log log10`

```
cmath.exp(z) # (-7.315110094901103+1.0427436562359045j)
cmath.log(z) # (1.2824746787307684+0.982793723247329j)
cmath.log10(-100) # (2+1.3643763538418412j)
```

- `cmath.sqrt(z)` # (1.6741492280355401+0.8959774761298381j)

- ```
cmath.sin(z) # (9.15449914691143-4.168906959966565j)
cmath.cos(z) # (-4.189625690968807-9.109227893755337j)
cmath.tan(z) # (-0.003764025641504249+1.00323862735361j)
cmath.asin(z) # (0.5706527843210994+1.9833870299165355j)
cmath.acos(z) # (1.0001435424737972-1.9833870299165355j)
cmath.atan(z) # (1.4099210495965755+0.22907268296853878j)
cmath.sin(z)**2 + cmath.cos(z)**2 # (1+0j)
```

- ```
cmath.sinh(z) # (-3.59056458998578+0.5309210862485197j)
cmath.cosh(z) # (-3.7245455049153224+0.5118225699873846j)
cmath.tanh(z) # (0.965385879022133-0.009884375038322495j)
cmath.asinh(z) # (0.5706527843210994+1.9833870299165355j)
cmath.acosh(z) # (1.9833870299165355+1.0001435424737972j)
cmath.atanh(z) # (0.14694666622552977+1.3389725222944935j)
cmath.cosh(z)**2 - cmath.sinh(z)**2 # (1+0j)
cmath.cosh((0+1j)*z) - cmath.cos(z) # 0j
```

Python [j](#)

```
z = 2+3j # A complex number
w = 1-7j # Another complex number
```

```
z + w # (3-4j)
z - w # (1+10j)
```

```
z * w # (23-11j)
z / w # (-0.38+0.34j)
z**3 # (-46+9j)
```

## Python

```
z.real # 2.0
z.imag # 3.0
abs(z) # 3.605551275463989
z.conjugate() # (2-3j)
```

<https://riptutorial.com/zh-TW/python/topic/1142/>

# 175:

tarball ZIP Gzip Python tarfile zipfile gzip Python tarfile TarFile.extractall(path=".", members=None)  
tarball Python zipfile ZipFile.extractall([path[, members[, pwd]]) ZIP Python gzip GzipFile

## Examples

### Python ZipFile.extractall ZIP

```
file_unzip = 'filename.zip'  
unzip = zipfile.ZipFile(file_unzip, 'r')  
unzip.extractall()  
unzip.close()
```

### Python TarFile.extractall tarball

```
file_untar = 'filename.tar.gz'  
untar = tarfile.TarFile(file_untar)  
untar.extractall()  
untar.close()
```

<https://riptutorial.com/zh-TW/python/topic/9505/>

# 176:

◦ ◦ Python◦ Python◦

## Examples

### argparseHello

◦ ◦

```
import argparse

parser = argparse.ArgumentParser()

parser.add_argument('name',
                    help='name of user'
                    )

parser.add_argument('-g', '--greeting',
                    default='Hello',
                    help='optional alternate greeting'
                    )

args = parser.parse_args()

print("{greeting}, {name}!".format(
    greeting=args.greeting,
    name=args.name)
)
```

```
$ python hello.py --help
usage: hello.py [-h] [-g GREETING] name

positional arguments:
  name                name of user

optional arguments:
  -h, --help          show this help message and exit
  -g GREETING, --greeting GREETING
                      optional alternate greeting
```

```
$ python hello.py world
Hello, world!
$ python hello.py John -g Howdy
Howdy, John!
```

[argparse](#) ◦

## docopt

[docopt](#) ◦ [docopt](#) ◦

```

"""
Usage:
    script_name.py [-a] [-b] <path>

Options:
    -a            Print all the things.
    -b            Get more bees into the path.
"""
from docopt import docopt

if __name__ == "__main__":
    args = docopt(__doc__)
    import pprint; pprint.pprint(args)

```

```

$ python script_name.py
Usage:
    script_name.py [-a] [-b] <path>
$ python script_name.py something
{'-a': False,
 '-b': False,
 '<path>': 'something'}
$ python script_name.py something -a
{'-a': True,
 '-b': False,
 '<path>': 'something'}
$ python script_name.py -b something -a
{'-a': True,
 '-b': True,
 '<path>': 'something'}

```

## argparse

◦ `argparse.ArgumentParser.add_mutually_exclusive_group()` ◦ **foobar**◦

```

import argparse

parser = argparse.ArgumentParser()
group = parser.add_mutually_exclusive_group()
group.add_argument("-f", "--foo")
group.add_argument("-b", "--bar")
args = parser.parse_args()
print "foo = ", args.foo
print "bar = ", args.bar

```

`--foo--bar`◦

error: argument -b/--bar: not allowed with argument -f/--foo

## argv

Python ◦ `sys.argv`“argv”“argument vector”◦

`sys.argv`Python◦

```
# cli.py
import sys
print(sys.argv)

$ python cli.py
=> ['cli.py']

$ python cli.py fizz
=> ['cli.py', 'fizz']

$ python cli.py fizz buzz
=> ['cli.py', 'fizz', 'buzz']
```

argv° **sys.argv**° °

```
import getpass
import sys

words = sys.argv[1:]
sentence = " ".join(words)
print("[%s] %s" % (getpass.getuser(), sentence))
```

“”sys.argv°

```
# reverse and copy sys.argv
argv = reversed(sys.argv)
# extract the first element
arg = argv.pop()
# stop iterating when there's no more args to pop()
while len(argv) > 0:
    if arg in ('-f', '--foo'):
        print('seen foo!')
    elif arg in ('-b', '--bar'):
        print('seen bar!')
    elif arg in ('-a', '--with-arg'):
        arg = argv.pop()
        print('seen value: {}'.format(arg))
    # get the next value
    arg = argv.pop()
```

## argparse

° `argparse.ArgumentParser.error° --foo--barstderr°`

```
import argparse

parser = argparse.ArgumentParser()
parser.add_argument("-f", "--foo")
parser.add_argument("-b", "--bar")
args = parser.parse_args()
if args.foo and args.bar is None:
    parser.error("--foo requires --bar. You did not specify bar.")

print "foo =", args.foo
print "bar =", args.bar
```

```
sample.py python sample.py --foo ds_in_fridge
```

```
usage: sample.py [-h] [-f FOO] [-b BAR]
sample.py: error: --foo requires --bar. You did not specify bar.
```

## argparse.add\_argument\_group

```
argparse.ArgumentParser'-h'◦ example.pypython example.py -h◦
```

```
import argparse

parser = argparse.ArgumentParser(description='Simple example')
parser.add_argument('name', help='Who to greet', default='World')
parser.add_argument('--bar_this')
parser.add_argument('--bar_that')
parser.add_argument('--foo_this')
parser.add_argument('--foo_that')
args = parser.parse_args()
```

```
usage: example.py [-h] [--bar_this BAR_THIS] [--bar_that BAR_THAT]
                  [--foo_this FOO_THIS] [--foo_that FOO_THAT]
                  name
```

Simple example

positional arguments:

name Who to greet

optional arguments:

-h, --help show this help message and exit  
--bar\_this BAR\_THIS  
--bar\_that BAR\_THAT  
--foo\_this FOO\_THIS  
--foo\_that FOO\_THAT

◦ ◦ --foo\_\* args--bar\_\* args◦

```
import argparse

parser = argparse.ArgumentParser(description='Simple example')
parser.add_argument('name', help='Who to greet', default='World')
# Create two argument groups
foo_group = parser.add_argument_group(title='Foo options')
bar_group = parser.add_argument_group(title='Bar options')
# Add arguments to those groups
foo_group.add_argument('--bar_this')
foo_group.add_argument('--bar_that')
bar_group.add_argument('--foo_this')
bar_group.add_argument('--foo_that')
args = parser.parse_args()
```

```
python example.py -h
```

```
usage: example.py [-h] [--bar_this BAR_THIS] [--bar_that BAR_THAT]
                  [--foo_this FOO_THIS] [--foo_that FOO_THAT]
```



name

Simple example

positional arguments:

name Who to greet

optional arguments:

-h, --help show this help message and exit

Foo options:

--bar\_this BAR\_THIS

--bar\_that BAR\_THAT

Bar options:

--foo\_this FOO\_THIS

--foo\_that FOO\_THAT

## docoptdocopt\_dispatch

docopt[docopt\_dispatch]\_\_doc\_\_--help ◦ docdispatch ◦

if / else ◦

dispatch.on ◦

```
"""Run something in development or production mode.
```

```
Usage: run.py --development <host> <port>
       run.py --production <host> <port>
       run.py items add <item>
       run.py items delete <item>
```

```
"""
```

```
from docopt_dispatch import dispatch
```

```
@dispatch.on('--development')
def development(host, port, **kwargs):
    print('in *development* mode')
```

```
@dispatch.on('--production')
def development(host, port, **kwargs):
    print('in *production* mode')
```

```
@dispatch.on('items', 'add')
def items_add(item, **kwargs):
    print('adding item...')
```

```
@dispatch.on('items', 'delete')
def items_delete(item, **kwargs):
    print('deleting item...')
```

```
if __name__ == '__main__':
    dispatch(__doc__)
```

<https://riptutorial.com/zh-TW/python/topic/1382/>

# 177:

## Examples

### Python

◦

#### API Python

```
import logging

logger = logging.getLogger()
handler = logging.StreamHandler()
formatter = logging.Formatter(
    '%(asctime)s %(name)-12s %(levelname)-8s %(message)s')
handler.setFormatter(formatter)
logger.addHandler(handler)
logger.setLevel(logging.DEBUG)

logger.debug('this is a %s test', 'debug')
```

```
2016-07-26 18:53:55,332 root          DEBUG    this is a debug test
```

### INI

#### logging\_config.ini

```
[loggers]
keys=root

[handlers]
keys=stream_handler

[formatters]
keys=formatter

[logger_root]
level=DEBUG
handlers=stream_handler

[handler_stream_handler]
class=StreamHandler
level=DEBUG
formatter=formatter
args=(sys.stderr,)

[formatter_formatter]
format=%(asctime)s %(name)-12s %(levelname)-8s %(message)s
```

```
logging.config.fileConfig()
```

```
import logging
from logging.config import fileConfig

fileConfig('logging_config.ini')
logger = logging.getLogger()
logger.debug('often makes a very good meal of %s', 'visiting tourists')
```

## Python 2.7. [PEP 391](#).

```
import logging
from logging.config import dictConfig

logging_config = dict(
    version = 1,
    formatters = {
        'f': {'format':
              '%(asctime)s %(name)-12s %(levelname)-8s %(message)s'}
    },
    handlers = {
        'h': {'class': 'logging.StreamHandler',
              'formatter': 'f',
              'level': logging.DEBUG}
    },
    root = {
        'handlers': ['h'],
        'level': logging.DEBUG,
    },
)

dictConfig(logging_config)

logger = logging.getLogger()
logger.debug('often makes a very good meal of %s', 'visiting tourists')
```

`logging.exception(msg)`

```
>>> import logging
>>> logging.basicConfig()
>>> try:
...     raise Exception('foo')
... except:
...     logging.exception('bar')
...
ERROR:root:bar
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
Exception: foo
```

`logging.exception(msg)msg arg`

```
>>> try:
...     raise Exception('foo')
... except Exception as e:
...     logging.exception(e)
...
ERROR:root:foo
Traceback (most recent call last):
```

```
File "<stdin>", line 2, in <module>
Exception: foo
```

```
>>> try:
...     raise Exception(u'föö')
... except Exception as e:
...     logging.exception(e)
...
Traceback (most recent call last):
  File ".../python2.7/logging/__init__.py", line 861, in emit
    msg = self.format(record)
  File ".../python2.7/logging/__init__.py", line 734, in format
    return fmt.format(record)
  File ".../python2.7/logging/__init__.py", line 469, in format
    s = self._fmt % record.__dict__
UnicodeEncodeError: 'ascii' codec can't encode characters in position 1-2: ordinal not in
range(128)
Logged from file <stdin>, line 4
```

**unicode** ◦ `logging.exception(e)` ◦

◦ ◦

## python

```
>>> try:
...     raise Exception(u'föö')
... except Exception as e:
...     logging.exception('bar')
...
ERROR:root:bar
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
Exception: f\xf6\xf6
```

`e logging.exception(...)` ◦

## ERROR

**ERROR**`_exc_info`

```
logging.debug('exception occurred', exc_info=1)
logging.info('exception occurred', exc_info=1)
logging.warning('exception occurred', exc_info=1)
```

**unicodeutf-8** ◦ `repr(e)%r`

```
>>> try:
...     raise Exception(u'föö')
... except Exception as e:
...     logging.exception('received this exception: %r' % e)
...
ERROR:root:received this exception: Exception(u'f\xf6\xf6',)
Traceback (most recent call last):
```

```
File "<stdin>", line 2, in <module>  
Exception: f\x66\x66
```

<https://riptutorial.com/zh-TW/python/topic/4081/>

# 178: Python

## Examples

PythonPythonpython

Python

```
import dis

def fib(n):
    if n <= 2: return 1
    return fib(n-1) + fib(n-2)

# Display the disassembled bytecode of the function.
dis.dis(fib)
```

disdis.dis◦

CPython◦

\_\_code\_\_ co\_code ◦

```
def fib(n):
    if n <= 2: return 1
    return fib(n-1) + fib(n-2)
dir(fib.__code__)

def fib(n):
    if n <= 2: return 1
    return fib(n-1) + fib(n-2)
dir(fib.__code__)
```

Pythoninspect ◦ ◦ ◦

randomrandint

```
import random
import inspect

print(inspect.getsource(random.randint))
# Output:
#     def randint(self, a, b):
#         """Return random integer in range [a, b], including both end points.
#         """
#         return self.randrange(a, b+1)
```

```
print(inspect.getdoc(random.randint))
# Output:
# Return random integer in range [a, b], including both end points.
```

random.randint

```
print(inspect.getfile(random.randint))
# c:\Python35\lib\random.py
print(random.randint.__code__.co_filename) # equivalent to the above
# c:\Python35\lib\random.py
```

inspectdill.source.getsource

```
# define a new function in the interactive shell
def add(a, b):
    return a + b
print(add.__code__.co_filename) # Output: <stdin>

import dill
print(dill.source.getsource(add))
# def add(a, b):
#     return a + b
```

PythoncPythonMercurial<https://www.python.org/downloads/source/> ◦

```
print(inspect.getsource(sorted)) # raises a TypeError
type(sorted) # <class 'builtin_function_or_method'>
```

Python <https://riptutorial.com/zh-TW/python/topic/4351/python>

# 179:

- Python◦

## Examples

◦ ◦ ◦

“”◦

```
from types import MethodType

class Animal(object):

    def __init__(self, *args, **kwargs):
        self.name = kwargs.pop('name', None) or 'Animal'
        if kwargs.get('walk', None):
            self.walk = MethodType(kwargs.pop('walk'), self)

    def walk(self):
        """
        Cause animal instance to walk

        Walking functionality is a strategy, and is intended to
        be implemented separately by different types of animals.
        """
        message = '{} should implement a walk method'.format(
            self.__class__.__name__)
        raise NotImplementedError(message)

# Here are some different walking algorithms that can be used with Animal
def snake_walk(self):
    print('I am slithering side to side because I am a {}'.format(self.name))

def four_legged_animal_walk(self):
    print('I am using all four of my legs to walk because I am a(n) {}'.format(
        self.name))

def two_legged_animal_walk(self):
    print('I am standing up on my two legs to walk because I am a {}'.format(
        self.name))
```

```
generic_animal = Animal()
king_cobra = Animal(name='King Cobra', walk=snake_walk)
elephant = Animal(name='Elephant', walk=four_legged_animal_walk)
kangaroo = Animal(name='Kangaroo', walk=two_legged_animal_walk)

kangaroo.walk()
elephant.walk()
king_cobra.walk()
# This one will Raise a NotImplementedError to let the programmer
# know that the walk method is intended to be used as a strategy.
generic_animal.walk()
```



```

# OUTPUT:
#
# I am standing up on my two legs to walk because I am a Kangaroo.
# I am using all four of my legs to walk because I am a(n) Elephant.
# I am slithering side to side because I am a King Cobra.
# Traceback (most recent call last):
#   File "./strategy.py", line 56, in <module>
#     generic_animal.walk()
#   File "./strategy.py", line 30, in walk
#     raise NotImplementedError(message)
# NotImplementedError: Animal should implement a walk method

```

C ++Java Python types.MethodType

## Singleton

commonly occurring problems GoF(Gang of Four)

1. The pattern name
2. The problem
3. The solution
4. The consequences

- 1.
- 2.
- 3.
- 4.

- 1.
- 2.
- 3.

Creational Pattern

Structural Pattern

Behavioral Pattern

creational pattern

Singleton

Singleton Python Singleton Pattern

```

class Singleton(object):
    def __new__(cls):
        # hasattr method checks if the class object an instance property or not.
        if not hasattr(cls, 'instance'):
            cls.instance = super(Singleton, cls).__new__(cls)
        return cls.instance

s = Singleton()

```

```
print ("Object created", s)

s1 = Singleton()
print ("Object2 created", s1)
```

```
('Object created', <__main__.Singleton object at 0x10a7cc310>)
('Object2 created', <__main__.Singleton object at 0x10a7cc310>)
```

## C ++Java。 。 Python。

Creational pattern。 “ factory。 。 。

```
from abc import ABCMeta, abstractmethod

class Music():
    __metaclass__ = ABCMeta
    @abstractmethod
    def do_play(self):
        pass

class Mp3(Music):
    def do_play(self):
        print ("Playing .mp3 music!")

class Ogg(Music):
    def do_play(self):
        print ("Playing .ogg music!")

class MusicFactory(object):
    def play_sound(self, object_type):
        return eval(object_type)().do_play()

if __name__ == "__main__":
    mf = MusicFactory()
    music = input("Which music you want to play Mp3 or Ogg")
    mf.play_sound(music)
```

```
Which music you want to play Mp3 or Ogg"Ogg"
Playing .ogg music!
```

MusicFactoryMp3Ogg。

。

。

:(consumer\_service

```
from datetime import date
from operator import attrgetter

class Proxy:
    def __init__(self, current_user, reservation_service):
        self.current_user = current_user
        self.reservation_service = reservation_service
```

```

def highest_total_price_reservations(self, date_from, date_to, reservations_count):
    if self.current_user.can_see_reservations:
        return self.reservation_service.highest_total_price_reservations(
            date_from,
            date_to,
            reservations_count
        )
    else:
        return []

#Models and ReservationService:

class Reservation:
    def __init__(self, date, total_price):
        self.date = date
        self.total_price = total_price

class ReservationService:
    def highest_total_price_reservations(self, date_from, date_to, reservations_count):
        # normally it would be read from database/external service
        reservations = [
            Reservation(date(2014, 5, 15), 100),
            Reservation(date(2017, 5, 15), 10),
            Reservation(date(2017, 1, 15), 50)
        ]

        filtered_reservations = [r for r in reservations if (date_from <= r.date <= date_to)]

        sorted_reservations = sorted(filtered_reservations, key=attrgetter('total_price'),
reverse=True)

        return sorted_reservations[0:reservations_count]

class User:
    def __init__(self, can_see_reservations, name):
        self.can_see_reservations = can_see_reservations
        self.name = name

#Consumer service:

class StatsService:
    def __init__(self, reservation_service):
        self.reservation_service = reservation_service

    def year_top_100_reservations_average_total_price(self, year):
        reservations = self.reservation_service.highest_total_price_reservations(
            date(year, 1, 1),
            date(year, 12, 31),
            1
        )

        if len(reservations) > 0:
            total = sum(r.total_price for r in reservations)

            return total / len(reservations)
        else:
            return 0

#Test:

```

```
def test(user, year):
    reservations_service = Proxy(user, ReservationService())
    stats_service = StatsService(reservations_service)
    average_price = stats_service.year_top_100_reservations_average_total_price(year)
    print("{0} will see: {1}".format(user.name, average_price))

test(User(True, "John the Admin"), 2017)
test(User(False, "Guest"), 2017)
```

- 
- **ReservationService**◦
  - **date\_from date\_to reservations\_count** ◦
  - **StatsService**

---

## CAVEATS

- ◦

<https://riptutorial.com/zh-TW/python/topic/8056/>

# 180:

- 
- `print`
- `"""`  
`"""`

[PEP257 - Docstring Conventions](#) ◦ [Google Sphinx](#) ◦

## Examples

◦

Python ◦

# ◦

- 

```
# This is a single line comment in Python
```

- 

```
print("Hello World") # This line prints "Hello World"
```

- `"""` ◦

```
"""  
This type of comment spans multiple lines.  
These are mostly used for documentation of functions, classes and modules.  
"""
```

- - ◦

```
def func():  
    """This is a function that does nothing at all"""  
    return
```

`__doc__` **docstring**

```
print(func.__doc__)
```

```
help(func)
```

```
__main__func
```

```
func()
```

```
function.__doc__ docstring help docstring ◦
```

```
def greet(name, greeting="Hello"):
    """Print a greeting to the user `name`

    Optional parameter `greeting` can change what they're greeted with."""

    print("{} {}".format(greeting, name))
```

```
help(greet)
```

```
__main__greet

greet(name, greeting='Hello')

name
greeting ◦
```

◦

```
def greet(name, greeting="Hello"):
    # Print a greeting to the user `name`
    # Optional parameter `greeting` can change what they're greeted with.

    print("{} {}".format(greeting, name))
```

```
print(greet.__doc__)
```

```
help(greet)
```

## main

```
greet(name, greeting='Hello')
```

## docstrings

[docstring](#) ◦ ◦

```
def hello(name):
    """Greet someone.

    Print a greeting ("Hello") for the person with the given name.
    """

    print("Hello "+name)
```

```
class Greeter:
    """An object used to greet people.

    It contains multiple greeting functions for several languages
    and times of the day.
    """
```

docstring - - help◦

## PEP 257

[PEP 257](#)◦

- Docstrings

PEP 257◦

```
def hello():  
    """Say hello to your friends."""  
    print("Hello my friends!")
```

◦

- 

◦

```
def hello(name, language="en"):  
    """Say hello to a person.  
  
    Arguments:  
    name: the name of the person  
    language: the language in which the person should be greeted  
    """  
  
    print(greeting[language]+" "+name)
```

◦

PEP 257 docstring ◦ ◦

[Sphinx docstrings](#) [Python HTML](#)◦ [reStructuredText](#)◦ [pythonhosted.org](#)◦ [pycharm IDE](#) [Sphinx](#)◦

Sphinx / reStructuredText

```
def hello(name, language="en"):  
    """Say hello to a person.  
  
    :param name: the name of the person  
    :type name: str  
    :param language: the language in which the person should be greeted  
    :type language: str  
    :return: a number  
    :rtype: int  
    """  
  
    print(greeting[language]+" "+name)  
    return 4
```

## Google Python

Google [Google Python](#) Python。 Sphinx / reST [Google](#)。

[pythonhosted.org](#) [Google Style Guide](#)。

[Napoleon](#) Sphinx [Google Style Guide](#)。

## Google

```
def hello(name, language="en"):
    """Say hello to a person.

    Args:
        name: the name of the person as string
        language: the language code string

    Returns:
        A number.
    """

    print(greeting[language]+" "+name)
    return 4
```

<https://riptutorial.com/zh-TW/python/topic/4144/>



---

# 181: PythonJavaScript

◦ ◦

## Examples

`in`

```
2 in [2, 3]
```

PythonTrueJavaScriptfalse◦ Python2[2,3]◦ JavaScript◦ JavaScript[2,3]

```
{'0': 2, '1': 3}
```

“2”◦ 2'2'◦

PythonJavaScript <https://riptutorial.com/zh-TW/python/topic/10766/--pythonjavascript>

---

# 182:

Python 2[ <https://docs.python.org/2/library/locale.html#locale.currency>] [1 ]

## Examples

```
import locale

locale.setlocale(locale.LC_ALL, '')
Out[2]: 'English_United States.1252'

locale.currency(762559748.49)
Out[3]: '$762559748.49'

locale.currency(762559748.49, grouping=True)
Out[4]: '$762,559,748.49'
```

<https://riptutorial.com/zh-TW/python/topic/1783/>

# 183:

## Examples

### Python\_pdb\_

Pythonpdb. pdb“”。

```
python -m pdb <my_file.py>
```

。

。 `pdbset_trace`。

```
import pdb

def divide(a, b):
    pdb.set_trace()
    return a/b
    # What's wrong with this? Hint: 2 != 3

print divide(1, 2)
```

。

```
python foo.py
> ~/scratch/foo.py(5)divide()
-> return a/b
(Pdb)
```

```
import pdf; pdb.set_trace()
```

Pdb. python. ppythonprint 。

```
(Pdb) p a
1
(Pdb) print a
1
```

```
locals
```

```
b <n> | <f>: set breakpoint at line *n* or function named *f*.
# b 3
# b divide
b: show all breakpoints.
c: continue until the next breakpoint.
s: step through this line (will enter a function).
n: step over this line (jumps over a function).
```

```

r: continue until the current function returns.
l: list a window of code around this line.
p <var>: print variable named *var*.
# p x
q: quit debugger.
bt: print the traceback of the current execution call stack
up: move your scope up the function call stack to the caller of the current function
down: Move your scope back down the function call stack one level
step: Run the program until the next line of execution in the program, then return control
back to the debugger
next: run the program until the next line of execution in the current function, then return
control back to the debugger
return: run the program until the current function returns, then return control back to the
debugger
continue: continue running the program until the next breakpoint (or set_trace is called
again)

```

## python

```

-> return a/b
(Pdb) p a+b
3
(Pdb) [ str(m) for m in [a,b]]
['1', '2']
(Pdb) [ d for d in xrange(5)]
[0, 1, 2, 3, 4]

```

' 'var. " c". 'c'. 'c'

```

(Pdb) !c
4

```

## IPythonipdb

### IPython Jupyter

```

import ipdb
ipdb.set_trace()

```

```

/home/usr/ook.py(3)<module>()
  1 import ipdb
  2 ipdb.set_trace()
----> 3 print("Hello world!")

ipdb>

```

o

```

from IPython.core import ultratb
sys.excepthook = ultratb.FormattedTB(mode='Verbose',
                                     color_scheme='Linux',
                                     call_pdb=1)

```

◦

python `rpdb`◦

`rpdbpdbstdinstdout`◦ 4444

```
# In the Python file you want to debug.  
import rpdb  
rpdb.set_trace()
```

◦

```
# Call in a terminal to see the output  
$ nc 127.0.0.1 4444
```

**pdb prompt**

```
> /home/usr/ook.py (3) <module> ()  
-> print("Hello world!")  
(Pdb)
```

<https://riptutorial.com/zh-TW/python/topic/2077/>

# 184: CSV

## Examples

### TSV

```
import csv

with open('/tmp/output.tsv', 'wt') as out_file:
    tsv_writer = csv.writer(out_file, delimiter='\t')
    tsv_writer.writerow(['name', 'field'])
    tsv_writer.writerow(['Dijkstra', 'Computer Science'])
    tsv_writer.writerow(['Shelah', 'Math'])
    tsv_writer.writerow(['Aumann', 'Economic Sciences'])
```

```
$ cat /tmp/output.tsv

name      field
Dijkstra  Computer Science
Shelah    Math
Aumann    Economic Sciences
```

### dictDataFrameCSV

```
import pandas as pd

d = {'a': (1, 101), 'b': (2, 202), 'c': (3, 303)}
pd.DataFrame.from_dict(d, orient="index")
df.to_csv("data.csv")
```

### CSVDataFramedict

```
df = pd.read_csv("data.csv")
d = df.to_dict()
```

CSV <https://riptutorial.com/zh-TW/python/topic/2116/csv>

# 185:

## Examples

### / Dunder

PythonMagicdunder. . .

◦

```
import math

class Vector(object):
    # instantiation
    def __init__(self, x, y):
        self.x = x
        self.y = y

    # unary negation (-v)
    def __neg__(self):
        return Vector(-self.x, -self.y)

    # addition (v + u)
    def __add__(self, other):
        return Vector(self.x + other.x, self.y + other.y)

    # subtraction (v - u)
    def __sub__(self, other):
        return self + (-other)

    # equality (v == u)
    def __eq__(self, other):
        return self.x == other.x and self.y == other.y

    # abs(v)
    def __abs__(self):
        return math.hypot(self.x, self.y)

    # str(v)
    def __str__(self):
        return '<{0.x}, {0.y}>'.format(self)

    # repr(v)
    def __repr__(self):
        return 'Vector({0.x}, {0.y})'.format(self)
```

Vector◦

```
v = Vector(1, 4)
u = Vector(2, 0)

u + v          # Vector(3, 4)
print(u + v)  # "<3, 4>" (implicit string conversion)
u - v          # Vector(1, -4)
```

```
u == v          # False
u + v == v + u # True
abs(u + v)     # 5.0
```

◦

◦

```
class sparselist(object):
    def __init__(self, size):
        self.size = size
        self.data = {}

    # l[index]
    def __getitem__(self, index):
        if index < 0:
            index += self.size
        if index >= self.size:
            raise IndexError(index)
        try:
            return self.data[index]
        except KeyError:
            return 0.0

    # l[index] = value
    def __setitem__(self, index, value):
        self.data[index] = value

    # del l[index]
    def __delitem__(self, index):
        if index in self.data:
            del self.data[index]

    # value in l
    def __contains__(self, value):
        return value == 0.0 or value in self.data.values()

    # len(l)
    def __len__(self):
        return self.size

    # for value in l: ...
    def __iter__(self):
        return (self[i] for i in range(self.size)) # use xrange for python2
```

listsparselist ◦

```
l = sparselist(10 ** 6) # list with 1 million elements
0 in l                  # True
10 in l                 # False

l[12345] = 10
10 in l                 # True
l[12345]                # 10

for v in l:
    pass # 0, 0, 0, ... 10, 0, 0 ... 0
```



```

class adder(object):
    def __init__(self, first):
        self.first = first

    # a(...)
    def __call__(self, second):
        return self.first + second

add2 = adder(2)
add2(1) # 3
add2(2) # 4

```

return NotImplemented    NotImplementedError ◦ Python

NotImplemented ◦ NotImplemented ◦

x + y    x.\_\_add\_\_(y)    y.\_\_radd\_\_(x) ◦

```

class NotAddable(object):

    def __init__(self, value):
        self.value = value

    def __add__(self, other):
        return NotImplemented

class Addable(NotAddable):

    def __add__(self, other):
        return Addable(self.value + other.value)

    __radd__ = __add__

```

\_\_add\_\_ \_\_radd\_\_;

```

>>> x = NotAddable(1)
>>> y = Addable(2)
>>> x + x
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'NotAddable' and 'NotAddable'
>>> y + y
<so.Addable object at 0x1095974d0>
>>> z = x + y
>>> z
<so.Addable object at 0x109597510>
>>> z.value
3

```

◦

**NB**other ◦

+	<code>__add__(self, other)</code>	<code>a1 + a2</code>
-	<code>__sub__(self, other)</code>	<code>a1 - a2</code>
*	<code>__mul__(self, other)</code>	<code>a1 * a2</code>
@	<code>__matmul__(self, other)</code>	<code>a1 @ a2</code> <i>Python 3.5</i>
/	<code>__div__(self, other)</code>	<code>a1 / a2</code> <i>Python 2</i>
/	<code>__truediv__(self, other)</code>	<code>a1 / a2</code> <i>Python 3</i>
//	<code>__floordiv__(self, other)</code>	<code>a1 // a2</code>
% Modulo / Remainder	<code>__mod__(self, other)</code>	<code>a1 % a2</code>
**	<code>__pow__(self, other[, modulo])</code>	<code>a1 ** a2</code>
<<	<code>__lshift__(self, other)</code>	<code>a1 &lt;&lt; a2</code>
>>	<code>__rshift__(self, other)</code>	<code>a1 &gt;&gt; a2</code>
& AND	<code>__and__(self, other)</code>	<code>a1 &amp; a2</code>
^	<code>__xor__(self, other)</code>	<code>a1 ^ a2</code>
OR	<code>__or__(self, other)</code>	<code>a1   a2</code>
-	<code>__neg__(self)</code>	<code>-a1</code>
+	<code>__pos__(self)</code>	<code>+a1</code>
~	<code>__invert__(self)</code>	<code>~a1</code>
<	<code>__lt__(self, other)</code>	<code>a1 &lt; a2</code>
<=	<code>__le__(self, other)</code>	<code>a1 &lt;= a2</code>
==	<code>__eq__(self, other)</code>	<code>a1 == a2</code>
!=	<code>__ne__(self, other)</code>	<code>a1 != a2</code>
>	<code>__gt__(self, other)</code>	<code>a1 &gt; a2</code>
>=	<code>__ge__(self, other)</code>	<code>a1 &gt;= a2</code>
[index]	<code>__getitem__(self, index)</code>	<code>a1[index]</code>
in In	<code>__contains__(self, other)</code>	<code>a2 in a1</code>
(*args, ...)	<code>__call__(self, *args, **kwargs)</code>	<code>a1(*args, **kwargs)</code>

`__pow__modulopow`

“right” `__r__radd__`

```

class A:
    def __init__(self, a):
        self.a = a
    def __add__(self, other):
        return self.a + other
    def __radd__(self, other):
        print("radd")
        return other + self.a

A(1) + 2 # Out: 3
2 + A(1) # prints radd. Out: 3

```

`__i__i`

```

class B:
    def __init__(self, b):
        self.b = b
    def __iadd__(self, other):
        self.b += other
        print("iadd")
        return self

b = B(2)
b.b # Out: 2
b += 1 # prints iadd
b.b # Out: 3

```

◦ `str().__str__` ◦

int	<code>__int__(self)</code>	<code>int(a1)</code>
	<code>__abs__(self)</code>	<code>abs(a1)</code>
str	<code>__str__(self)</code>	<code>str(a1)</code>
unicode	<code>__unicode__(self)</code>	<code>unicode(a1)</code> <b>Python 2</b>
	<code>__repr__(self)</code>	<code>repr(a1)</code>
bool	<code>__nonzero__(self)</code>	<code>bool(a1)</code>
	<code>__format__(self, formatstr)</code>	<code>"Hi {:abc}".format(a1)</code>
	<code>__hash__(self)</code>	<code>hash(a1)</code>
	<code>__len__(self)</code>	<code>len(a1)</code>
	<code>__reversed__(self)</code>	<code>reversed(a1)</code>
	<code>__floor__(self)</code>	<code>math.floor(a1)</code>
	<code>__ceil__(self)</code>	<code>math.ceil(a1)</code>

`__enter__` `__exit__` ◦

<https://riptutorial.com/zh-TW/python/topic/2063/>

# 186:

- `typing.Callable` `[[intstr]None]` - `> def func(intbstr) -> None`
- `typing.Mapping` `[strint]` - `> {"a":1,"b":2,"c":3}`
- `typing.List` `[int]` - `> [1,2,3]`
- `typing.Set` `[int]` - `> {1,2,3}`
- `typing.Optional` `[int]` - `> Noneint`
- `typing.Sequence` `[int]` - `> [1,2,3]1,2,3`
- ◦ - `>`
- `typing.Union` `[intstr]` - `> 1"1"`
- `T = typing.TypeVar'T'` - `>`

PEP 484 Python ◦ `typing` Python ◦

## Examples

`typing.TypeVar` ◦ `///`

```
import typing

T = typing.TypeVar("T")

def get_first_element(l: typing.Sequence[T]) -> T:
    """Gets the first element of a sequence."""
    return l[0]
```

```
def two_sum(a, b):
    return a + b
```

`two_sum` ◦ `int`

```
print(two_sum(2, 1)) # result: 3
```

```
print(two_sum("a", "b")) # result: "ab"
```

list **S** tuple **S** ◦

python ◦

◦

`int`

```
def two_sum(a: int, b: int):
    return a + b
```

`:◦`

str

```
def two_sum(a: str, b: str):  
    return a + b
```

◦ :->

```
def two_sum(a: int, b: int) -> int:  
    return a + b
```

two\_sum int ◦ str float list set ◦

IDE ◦ `__annotations__`

```
two_sum.__annotations__  
# {'a': <class 'int'>, 'b': <class 'int'>, 'return': <class 'int'>}
```

```
class A:  
    x = None # type: float  
    def __init__(self, x: float) -> None:  
        """  
        self should not be annotated  
        init should be annotated to return None  
        """  
        self.x = x  
  
    @classmethod  
    def from_int(cls, x: int) -> 'A':  
        """  
        cls should not be annotated  
        Use forward reference to refer to current class with string literal 'A'  
        """  
        return cls(float(x))
```

◦ ◦

```
x = 3 # type: int  
x = negate(x)  
x = 'a type-checker might catch this error'
```

## Python 3.x 3.6

Python 3.6 ◦

```
x: int = 3
```

```
y: int
```

`typing.get_type_hints(class_or_module)`

```
class Foo:  
    x: int
```

```
y: str = 'abc'

print(typing.get_type_hints(Foo))
# ChainMap({'x': <class 'int'>, 'y': <class 'str'>}, {})
```

\_\_annotations\_\_

```
x: int
print(__annotations__)
# {'x': <class 'int'>}

class C:
    s: str
print(C.__annotations__)
# {'s': <class 'str'>}
```

## NamedTuple

namedtuple typing NamedTuple

```
import typing
Point = typing.NamedTuple('Point', [('x', int), ('y', int)])
```

◦

```
def hello_world(greeting: str = 'Hello'):
    print(greeting + ' world!')
```

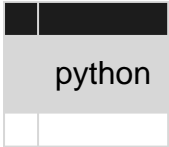
◦

<https://riptutorial.com/zh-TW/python/topic/1766/>

# 187: ...

Python `itertools.groupby()` ◦

- `itertools.groupby(iterable, key = None)` = Nonesome function



`groupby`

◦

## Examples

1

```
s = 'AAAABBBCCDAABBB'
```

'A"B"C'

```
s = 'AAAABBBCCDAABBB'
s_dict = {}
for i in s:
    if i not in s_dict.keys():
        s_dict[i] = [i]
    else:
        s_dict[i].append(i)
s_dict
```

```
{'A': ['A', 'A', 'A', 'A', 'A', 'A'],
 'B': ['B', 'B', 'B', 'B', 'B', 'B'],
 'C': ['C', 'C'],
 'D': ['D']}
```

◦ `groupby`

```
# note that we get a {key : value} pair for iterating over the items just like in python
dictionary
from itertools import groupby
s = 'AAAABBBCCDAABBB'
c = groupby(s)

dic = {}
for k, v in c:
    dic[k] = list(v)
dic
```

```
{'A': ['A', 'A'], 'B': ['B', 'B', 'B'], 'C': ['C', 'C'], 'D': ['D']}
```

## group by 'A' c

```
c = groupby(sorted(s))

dic = {}
for k, v in c:
    dic[k] = list(v)
dic
```

```
{'A': ['A', 'A', 'A', 'A', 'A', 'A'], 'B': ['B', 'B', 'B', 'B', 'B', 'B'], 'C': ['C', 'C'],
'D': ['D']}
```

'A'.

## 2

```
c = groupby(['goat', 'dog', 'cow', 1, 1, 2, 3, 11, 10, ('persons', 'man', 'woman')])
dic = {}
for k, v in c:
    dic[k] = list(v)
dic
```

```
{1: [1, 1],
 2: [2],
 3: [3],
 ('persons', 'man', 'woman'): [('persons', 'man', 'woman')],
 'cow': ['cow'],
 'dog': ['dog'],
 10: [10],
 11: [11],
 'goat': ['goat']}
```

## 3

mulatocamel. . c. .

```
list_things = ['goat', 'dog', 'donkey', 'mulato', 'cow', 'cat', ('persons', 'man', 'woman'), \
               'wombat', 'mongoose', 'malloo', 'camel']
c = groupby(list_things, key=lambda x: x[0])
dic = {}
for k, v in c:
    dic[k] = list(v)
dic
```

```
{'c': ['camel'],
 'd': ['dog', 'donkey'],
 'g': ['goat'],
 'm': ['mongoose', 'malloo'],
 'persons': [('persons', 'man', 'woman')],
 'w': ['wombat']}
```



```
list_things = ['goat', 'dog', 'donkey', 'mulato', 'cow', 'cat', ('persons', 'man', 'woman'), \
              'wombat', 'mongoose', 'malloo', 'camel']
sorted_list = sorted(list_things, key = lambda x: x[0])
print(sorted_list)
print()
c = groupby(sorted_list, key=lambda x: x[0])
dic = {}
for k, v in c:
    dic[k] = list(v)
dic
```

```
['cow', 'cat', 'camel', 'dog', 'donkey', 'goat', 'mulato', 'mongoose', 'malloo', ('persons',
'man', 'woman'), 'wombat']
```

```
{'c': ['cow', 'cat', 'camel'],
 'd': ['dog', 'donkey'],
 'g': ['goat'],
 'm': ['mulato', 'mongoose', 'malloo'],
 'persons': [('persons', 'man', 'woman')],
 'w': ['wombat']}
```

## 4

### iterable。

```
things = [("animal", "bear"), ("animal", "duck"), ("plant", "cactus"), ("vehicle", "harley"), \
         ("vehicle", "speed boat"), ("vehicle", "school bus")]
dic = {}
f = lambda x: x[0]
for key, group in groupby(sorted(things, key=f), f):
    dic[key] = list(group)
dic
```

```
{'animal': [('animal', 'bear'), ('animal', 'duck')],
 'plant': [('plant', 'cactus')],
 'vehicle': [('vehicle', 'harley'),
            ('vehicle', 'speed boat'),
            ('vehicle', 'school bus')]}
```

。。

```
things = ["animal", "bear"], ["animal", "duck"], ["vehicle", "harley"], ["plant", "cactus"], \
         ["vehicle", "speed boat"], ["vehicle", "school bus"]]
dic = {}
f = lambda x: x[0]
for key, group in groupby(sorted(things, key=f), f):
    dic[key] = list(group)
dic
```

```
{'animal': [['animal', 'bear'], ['animal', 'duck']],
 'plant': [['plant', 'cactus']],
 'vehicle': [['vehicle', 'harley'],
```

```
['vehicle', 'speed boat'],  
['vehicle', 'school bus']}]}
```

... <https://riptutorial.com/zh-TW/python/topic/8690/----->

# 188:

•

Python

```
if except def class:  
    class except "" for:
```

```
pass  
IndentationError Ellipsis...pass pass -
```

- Exception xml

```
try:  
    self.version = "Expat %d.%d.%d" % expat.version_info  
except AttributeError:  
    pass # unknown
```

```
pandasKeyboardInterruptSystemExit HardwareIsOnFireError -
```

```
try:  
    os.unlink(filename_larry)  
except:  
    pass
```

```
except Error:except OSError: python10except ...: passpython
```

- scipy

```
class CompileError(Exception):  
    pass
```

```
__init__ pebl
```

```
class _BaseSubmittingController(_BaseController):  
    def submit(self, tasks): pass  
    def retrieve(self, deferred_results): pass
```

- mpmath

```
for x, error in MDNewton(mp, f, (1,-2), verbose=0,  
                        norm=lambda x: norm(x, inf)):  
    pass
```

- docstring docstring pass "pebl" pebl

```
class ParsingError(Exception):  
    """Error encountered while parsing an ill-formed datafile."""  
    pass
```

•

pass“// if-block / .....”Ellipsis... python-3“no-op”。

```
def update_agent (agent) :  
    ...
```

```
def update_agent (agent) :  
    pass
```

```
def time_step (agents) :  
    for agent in agents :  
        update_agent (agent)
```

update\_agent ◦ raise NotImplementedError ◦ *"""*

## Examples

```
try :  
    metadata = metadata ['properties']  
except KeyError :  
    pass
```

```
class CompileError (Exception) :  
    pass
```

<https://riptutorial.com/zh-TW/python/topic/6891/>

# 189:

◦ ◦ ◦ Python◦

## Examples

**Global Interpreter Lock** Python◦ PythonCPU◦

```
import threading
import time

def process():
    time.sleep(2)

start = time.time()
process()
print("One run took %.2fs" % (time.time() - start))

start = time.time()
threads = [threading.Thread(target=process) for _ in range(4)]
for t in threads:
    t.start()
for t in threads:
    t.join()
print("Four runs took %.2fs" % (time.time() - start))

# Out: One run took 2.00s
# Out: Four runs took 2.00s
```

process22◦

## Python

```
import threading
import time

def somefunc(i):
    return i * i

def otherfunc(m, i):
    return m + i

def process():
    for j in range(100):
        result = 0
        for i in range(100000):
            result = otherfunc(result, somefunc(i))

start = time.time()
process()
```

```

print("One run took %.2fs" % (time.time() - start))

start = time.time()
threads = [threading.Thread(target=process) for _ in range(4)]
for t in threads:
    t.start()
for t in threads:
    t.join()
print("Four runs took %.2fs" % (time.time() - start))

# Out: One run took 2.05s
# Out: Four runs took 14.42s

```

```

import multiprocessing
import time

def somefunc(i):
    return i * i

def otherfunc(m, i):
    return m + i

def process():
    for j in range(100):
        result = 0
        for i in range(100000):
            result = otherfunc(result, somefunc(i))

start = time.time()
process()
print("One run took %.2fs" % (time.time() - start))

start = time.time()
processes = [multiprocessing.Process(target=process) for _ in range(4)]
for p in processes:
    p.start()
for p in processes:
    p.join()
print("Four runs took %.2fs" % (time.time() - start))

# Out: One run took 2.07s
# Out: Four runs took 2.30s

```

threading.Thread

```

import threading
import os

def process():
    print("Pid is %s, thread id is %s" % (os.getpid(), threading.current_thread().name))

threads = [threading.Thread(target=process) for _ in range(4)]
for t in threads:
    t.start()
for t in threads:

```

```
t.join()

# Out: Pid is 11240, thread id is Thread-1
# Out: Pid is 11240, thread id is Thread-2
# Out: Pid is 11240, thread id is Thread-3
# Out: Pid is 11240, thread id is Thread-4
```

multiprocessing.Process° threading.Thread

```
import multiprocessing
import os

def process():
    print("Pid is %s" % (os.getpid(),))

processes = [multiprocessing.Process(target=process) for _ in range(4)]
for p in processes:
    p.start()
for p in processes:
    p.join()

# Out: Pid is 11206
# Out: Pid is 11207
# Out: Pid is 11208
# Out: Pid is 11209
```

°

°

```
import threading

obj = {}
obj_lock = threading.Lock()

def objify(key, val):
    print("Obj has %d values" % len(obj))
    with obj_lock:
        obj[key] = val
    print("Obj now has %d values" % len(obj))

ts = [threading.Thread(target=objify, args=(str(n), n)) for n in range(4)]
for t in ts:
    t.start()
for t in ts:
    t.join()
print("Obj final result:")
import pprint; pprint.pprint(obj)

# Out: Obj has 0 values
# Out: Obj has 0 values
# Out: Obj now has 1 values
# Out: Obj now has 2 valuesObj has 2 values
# Out: Obj now has 3 values
# Out:
# Out: Obj has 3 values
# Out: Obj now has 4 values
# Out: Obj final result:
```

```
# Out: {'0': 0, '1': 1, '2': 2, '3': 3}
```

- multiprocessing◦

```
import multiprocessing

plain_num = 0
shared_num = multiprocessing.Value('d', 0)
lock = multiprocessing.Lock()

def increment():
    global plain_num
    with lock:
        # ordinary variable modifications are not visible across processes
        plain_num += 1
        # multiprocessing.Value modifications are
        shared_num.value += 1

ps = [multiprocessing.Process(target=increment) for n in range(4)]
for p in ps:
    p.start()
for p in ps:
    p.join()

print("plain_num is %d, shared_num is %d" % (plain_num, shared_num.value))

# Out: plain_num is 0, shared_num is 4
```

<https://riptutorial.com/zh-TW/python/topic/4110/>



# 190:

Python ◦ 3 \* 2 + 732713.\*+◦

◦

Python

Python◦ ◦ ◦ ◦

	Lambda
	OR
	AND
x	NOT
innot inisnot not<<=>> =<>===	
	OR
^	
	AND
<<>>	
+ -	
*//	[8]
+ x-x~x	NOT
**	[9]
x [index]x [indexindex]xarguments ...x.attribute	
.....[...]{...}.....	

## Examples

python◦

PythonPEMDAS◦ PEMDAS◦

```
>>> a, b, c, d = 2, 3, 5, 7
>>> a ** (b + c) # parentheses
256
>>> a * b ** c # exponent: same as `a * (b ** c)`
7776
>>> a + b * c / d # multiplication / division: same as `a + (b * c / d)`
4.142857142857142
```

```
>>> 300 / 300 * 200
200.0
>>> 300 * 200 / 300
200.0
>>> 1e300 / 1e300 * 1e200
1e+200
>>> 1e300 * 1e200 / 1e300
inf
```

<https://riptutorial.com/zh-TW/python/topic/5040/>

# 191:

- 
- `itertools.filter`
- `future_builtins.filterfunctioniterable`
- `itertools.filterfalse`
- `itertools.filterfalse`

	None
	iterable

`filter()` `ifilter()` ◦

## Examples

`filter`

```
names = ['Fred', 'Wilma', 'Barney']

def long_name(name):
    return len(name) > 5
```

### Python 2.x 2.0

```
filter(long_name, names)
# Out: ['Barney']

[name for name in names if len(name) > 5] # equivalent list comprehension
# Out: ['Barney']

from itertools import ifilter
ifilter(long_name, names) # as generator (similar to python 3.x filter builtin)
# Out: <itertools.ifilter at 0x4197e10>
list(ifilter(long_name, names)) # equivalent to filter with lists
# Out: ['Barney']

(name for name in names if len(name) > 5) # equivalent generator expression
# Out: <generator object <genexpr> at 0x0000000003FD5D38>
```

### Python 2.x 2.6

```
# Besides the options for older python 2.x versions there is a future_builtin function:
from future_builtins import filter
filter(long_name, names) # identical to itertools.ifilter
# Out: <itertools.ifilter at 0x3eb0ba8>
```

### Python 3.x 3.0

```
filter(long_name, names)          # returns a generator
# Out: <filter at 0x1fc6e443470>
list(filter(long_name, names))    # cast to list
# Out: ['Barney']

(name for name in names if len(name) > 5) # equivalent generator expression
# Out: <generator object <genexpr> at 0x000001C6F49BF4C0>
```

None

```
list(filter(None, [1, 0, 2, [], '', 'a'])) # discards 0, [] and ''
# Out: [1, 2, 'a']
```

## Python 2.x 2.0.1

```
[i for i in [1, 0, 2, [], '', 'a'] if i] # equivalent list comprehension
```

## Python 3.x 3.0.0

```
(i for i in [1, 0, 2, [], '', 'a'] if i) # equivalent generator expression
```

filter **3.X** ifilter **2.X** or and

## Python 2.x 2.0.1

```
# not recommended in real use but keeps the example short:
from itertools import ifilter as filter
```

## Python 2.x 2.6.1

```
from future_builtins import filter
```

100

```
car_shop = [('Toyota', 1000), ('rectangular tire', 80), ('Porsche', 5000)]
def find_something_smaller_than(name_value_tuple):
    print('Check {0}, {1}$'.format(*name_value_tuple))
    return name_value_tuple[1] < 100
next(filter(find_something_smaller_than, car_shop))
# Print: Check Toyota, 1000$
#       Check rectangular tire, 80$
# Out: ('rectangular tire', 80)
```

next◦

## filterfalse ifilterfalse

itertools -module filter

## Python 2.x 2.0.1

```
# not recommended in real use but keeps the example valid for python 2.x and python 3.x
```

```
from itertools import ifilterfalse as filterfalse
```

## Python 3.x 3.0.0

```
from itertools import filterfalse
```

```
filterFalse
```

```
# Usage without function (None):  
list(filterfalse(None, [1, 0, 2, [], '', 'a'])) # discards 1, 2, 'a'  
# Out: [0, [], '']
```

```
# Usage with function  
names = ['Fred', 'Wilma', 'Barney']  
  
def long_name(name):  
    return len(name) > 5  
  
list(filterfalse(long_name, names))  
# Out: ['Fred', 'Wilma']
```

```
# Short-circuit useage with next:  
car_shop = [('Toyota', 1000), ('rectangular tire', 80), ('Porsche', 5000)]  
def find_something_smaller_than(name_value_tuple):  
    print('Check {0}, {1}$'.format(*name_value_tuple))  
    return name_value_tuple[1] < 100  
next(filterfalse(find_something_smaller_than, car_shop))  
# Print: Check Toyota, 1000$  
# Out: ('Toyota', 1000)
```

```
# Using an equivalent generator:  
car_shop = [('Toyota', 1000), ('rectangular tire', 80), ('Porsche', 5000)]  
generator = (car for car in car_shop if not car[1] < 100)  
next(generator)
```

<https://riptutorial.com/zh-TW/python/topic/201/>

# 192:

stopCondition

o

## Examples

### 1n

1n 1 + 2 + 3 + 4 + ... + (several hours later) + n o for

```
n = 0
for i in range (1, n+1):
    n += i
```

```
def recursion(n):
    if n == 1:
        return 1
    return n + recursion(n - 1)
```

o 1 + 2 + 3 **13**.recursion(4)

4 -> 4 + 3 -> 4 + 3 + 2 -> 4 + 3 + 2 + 1 -> 10

for 1 -> 1 + 2 -> 1 + 2 + 3 -> 1 + 2 + 3 + 4 -> 10 o o o

o x! o

- 0!
- 1!

### Python

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
```

o factorial(3) o o n factorial print locals() n = 3 locals()['n': 3] n == 3 n \* factorial(n - 1) o

o n factorial(n - 1) o n - 2 factorial n o **A B** A{'n': 3} B{'n': 2} o n \* factorial(n - 1) o o **B n B n A** n - 1 o n \* ((n - 1) \* factorial((n - 1) - 1)) o o

factorial((n - 1) - 1)) o **A** n == 3 1 factorial o {'n': 1} **C** n \* factorial(n - 1) o "" factorial((n - 1) - 1))

◦ `n * ((n - 1) * ((n - 2) * factorial((n - 2) - 1)))` ◦

◦ `factorial((n - 2) - 1)` ◦ `0` ◦ `1` ◦ ◦ `n * ((n - 1) * ((n - 2) * 1))` ◦ **A**`3 * ((3 - 1) * ((3 - 2) * 1))` ◦ **6.3!** `== 3 * 2 * 1 == 6` ◦ ◦

`if n == 0: return 1` ◦ ◦ ◦ ◦ `factorial`

```
def factorial(n):
    if n == 0:
        return 1
    elif n == 1:
        return 1
    else:
        return n * factorial(n - 1)
```

◦

## ◦ Fibonacci

- `00` ◦
- `11` ◦
- ◦

```
def fib(n):
    if n == 0 or n == 1:
        return n
    else:
        return fib(n - 2) + fib(n - 1)
```

`factorial(3)fib(5)`

```
(
    fib((n - 2) - 2)
    +
    (
        fib(((n - 2) - 1) - 2)
        +
        fib(((n - 2) - 1) - 1)
    )
)
+
(
    (
        fib(((n - 1) - 2) - 2)
        +
        fib(((n - 1) - 2) - 1)
    )
    +
    (
        fib(((n - 1) - 1) - 2)
        +
        (
            fib((((n - 1) - 1) - 1) - 2)
            +
            fib((((n - 1) - 1) - 1) - 1)
        )
    )
)
```

```
)  
)  
)
```

$(1 + (0 + 1)) + ((0 + 1) + (1 + (0 + 1)))$  5 °

- ° return foo(n - 1) return foo(n - 1) + 1°
- TCO°
- TCE° TCETCO°
  
- ° °
- °

PythonTCO° ° ° factorialfib

```
def factorial(n):  
    product = 1  
    while n > 1:  
        product *= n  
        n -= 1  
    return product  
  
def fib(n):  
    a, b = 0, 1  
    while n > 0:  
        a, b = b, a + b  
        n -= 1  
    return a
```

°

Python[lru\\_cache](#)°

Python"""° ° ° ° Python°

SchemeHaskell° °

Fibonaccipythonlru2° n °

```
def fib(n):  
    if n <= 1:  
        return (n, 0)  
    else:  
        (a, b) = fib(n - 1)  
        return (a + b, a)
```

° °

```
root  
- A  
- AA  
- AB
```



```
- B
  - BA
  - BB
    - BBA
```

for◦ `get_name()` `get_children()` `get_root()`◦

```
root = get_root(tree)
for node in get_children(root):
    print(get_name(node))
    for child in get_children(node):
        print(get_name(child))
        for grand_child in get_children(child):
            print(get_name(grand_child))
# prints: A, AA, AB, B, BA, BB, BBA
```

.....◦

```
def list_tree_names(node):
    for child in get_children(node):
        print(get_name(child))
        list_tree_names(node=child)

list_tree_names(node=get_root(tree))
# prints: A, AA, AB, B, BA, BB, BBA
```

◦ ◦

```
def list_tree_names(node, lst=[]):
    for child in get_children(node):
        lst.append(get_name(child))
        list_tree_names(node=child, lst=lst)
    return lst

list_tree_names(node=get_root(tree))
# returns ['A', 'AA', 'AB', 'B', 'BA', 'BB', 'BBA']
```

## Python◦ RuntimeError

```
RuntimeError: Maximum Recursion Depth Exceeded
```

```
def cursing(depth):
    try:
        cursing(depth + 1) # actually, re-cursing
    except RuntimeError as RE:
        print('I recursed {} times!'.format(depth))
cursing(0)
# Out: I recursed 1083 times!
```

```
sys.setrecursionlimit(limit)
```

```
sys.getrecursionlimit()
```

```
sys.setrecursionlimit(2000)
cursing(0)
# Out: I recursed 1997 times!
```

## Python 3.5 RecursionError RuntimeError

-  
◦

```
def countdown(n):
    if n == 0:
        print "Blastoff!"
    else:
        print n
        countdown(n-1)
```

## ◦ find\_max

```
def find_max(seq, max_so_far):
    if not seq:
        return max_so_far
    if max_so_far < seq[0]:
        return find_max(seq[1:], seq[0])
    else:
        return find_max(seq[1:], max_so_far)
```

## Python Python ◦ ◦

## Python 1000 ◦ sys.setrecursionlimit(15000) ◦ Tail Recursion ◦

```
#!/usr/bin/env python2.4
# This program shows off a python decorator which implements tail call optimization. It
# does this by throwing an exception if it is it's own grandparent, and catching such
# exceptions to recall the stack.

import sys

class TailRecurseException:
    def __init__(self, args, kwargs):
        self.args = args
        self.kwargs = kwargs

def tail_call_optimized(g):
    """
    This function decorates a function with tail call
    optimization. It does this by throwing an exception
    if it is it's own grandparent, and catching such
    exceptions to fake the tail call optimization.

    This function fails if the decorated
    function recurses in a non-tail context.
    """

    def func(*args, **kwargs):
```

```

f = sys._getframe()
if f.f_back and f.f_back.f_back and f.f_back.f_back.f_code == f.f_code:
    raise TailRecurseException(args, kwargs)
else:
    while 1:
        try:
            return g(*args, **kwargs)
        except TailRecurseException, e:
            args = e.args
            kwargs = e.kwargs
func.__doc__ = g.__doc__
return func

```

@tail\_call\_optimized

```

@tail_call_optimized
def factorial(n, acc=1):
    "calculate a factorial"
    if n == 0:
        return acc
    return factorial(n-1, n*acc)

print factorial(10000)
# prints a big, big number,
# but doesn't hit the recursion limit.

```

```

@tail_call_optimized
def fib(i, current = 0, next = 1):
    if i == 0:
        return current
    else:
        return fib(i - 1, next, current + next)

print fib(10000)
# also prints a big number,
# but doesn't hit the recursion limit.

```

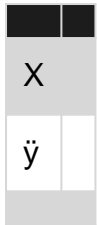
<https://riptutorial.com/zh-TW/python/topic/1716/>

# 193:

OOP.

◦

- `** params_you_want_fix`



Python doc *functools.partial*

`argskeyword argumentsfunc◦ args◦ ◦`

◦

## Examples

`xy◦`

```
def raise_power(x, y):  
    return x**y
```

`y`

`y[3,4,5]◦ y`

```
def raise(x, y):  
    if y in (3,4,5):  
        return x**y  
    raise ValueError("You should provide a valid exponent")
```

◦

```
from functools import partial  
raise_to_three = partial(raise, y=3)  
raise_to_four = partial(raise, y=4)  
raise_to_five = partial(raise, y=5)
```

`y◦`

◦

<https://riptutorial.com/zh-TW/python/topic/9383/>

# 194:

## Examples

### Conda

Anaconda Binstar [Anaconda 1.6+binstar](#)

```
$ conda install binstar
$ conda update binstar
```

### Anaconda Binstar pypi

```
$ pip install binstar
```

```
$ binstar login
```

### whoami

```
$ binstar whoami
```

### “hello world”. Github

```
$ git clone https://github.com/<NAME>/<Package>
```

```
package/
  setup.py
  test_package/
    __init__.py
    hello.py
    bld.bat
    build.sh
    meta.yaml
```

Setup.py `python hello.py` `hello_world`

bld.bat build.sh meta.yaml Conda [Conda](#)

```
$ conda build test_package/
```

### Conda

`conda build test_package /binstar`

```
$ binstar upload /home/xavier/anaconda/conda-bld/linux-64/test_package-0.1.0-py27_0.tar.bz2
```

Web。

*done* Binstar。

<https://riptutorial.com/zh-TW/python/topic/4064/>

# 195:

◦ ◦ ◦

## Examples

◦

```
class Node:
    def __init__(self, val):
        self.data = val
        self.next = None

    def getData(self):
        return self.data

    def getNext(self):
        return self.next

    def setData(self, val):
        self.data = val

    def setNext(self, val):
        self.next = val

class LinkedList:
    def __init__(self):
        self.head = None

    def isEmpty(self):
        """Check if the list is empty"""
        return self.head is None

    def add(self, item):
        """Add the item to the list"""
        new_node = Node(item)
        new_node.setNext(self.head)
        self.head = new_node

    def size(self):
        """Return the length/size of the list"""
        count = 0
        current = self.head
        while current is not None:
            count += 1
            current = current.getNext()
        return count

    def search(self, item):
        """Search for item in list. If found, return True. If not found, return False"""
        current = self.head
        found = False
        while current is not None and not found:
            if current.getData() is item:
                found = True
            else:
```



```

        current = current.getNext()
    return found

def remove(self, item):
    """Remove item from list. If item is not found in list, raise ValueError"""
    current = self.head
    previous = None
    found = False
    while current is not None and not found:
        if current.getData() is item:
            found = True
        else:
            previous = current
            current = current.getNext()
    if found:
        if previous is None:
            self.head = current.getNext()
        else:
            previous.setNext(current.getNext())
    else:
        raise ValueError
        print 'Value not found.'

def insert(self, position, item):
    """
    Insert item at position specified. If position specified is
    out of bounds, raise IndexError
    """
    if position > self.size() - 1:
        raise IndexError
        print "Index out of bounds."
    current = self.head
    previous = None
    pos = 0
    if position is 0:
        self.add(item)
    else:
        new_node = Node(item)
        while pos < position:
            pos += 1
            previous = current
            current = current.getNext()
        previous.setNext(new_node)
        new_node.setNext(current)

def index(self, item):
    """
    Return the index where item is found.
    If item is not found, return None.
    """
    current = self.head
    pos = 0
    found = False
    while current is not None and not found:
        if current.getData() is item:
            found = True
        else:
            current = current.getNext()
            pos += 1
    if found:
        pass

```

```

else:
    pos = None
return pos

def pop(self, position = None):
    """
    If no argument is provided, return and remove the item at the head.
    If position is provided, return and remove the item at that position.
    If index is out of bounds, raise IndexError
    """
    if position > self.size():
        print 'Index out of bounds'
        raise IndexError

    current = self.head
    if position is None:
        ret = current.getData()
        self.head = current.getNext()
    else:
        pos = 0
        previous = None
        while pos < position:
            previous = current
            current = current.getNext()
            pos += 1
        ret = current.getData()
        previous.setNext(current.getNext())
    print ret
    return ret

def append(self, item):
    """Append item to the end of the list"""
    current = self.head
    previous = None
    pos = 0
    length = self.size()
    while pos < length:
        previous = current
        current = current.getNext()
        pos += 1
    new_node = Node(item)
    if previous is None:
        new_node.setNext(current)
        self.head = new_node
    else:
        previous.setNext(new_node)

def printList(self):
    """Print the list"""
    current = self.head
    while current is not None:
        print current.getData()
        current = current.getNext()

```

o

```

l1 = LinkedList()
l1.add('l')
l1.add('H')
l1.insert(1, 'e')

```

```
ll.append('l')  
ll.append('o')  
ll.printList()
```

```
H  
e  
l  
l  
o
```

<https://riptutorial.com/zh-TW/python/topic/9299/>

---

# 196:

## Examples

### python

- None
- 。

```
#!/usr/bin/env python

class Node:
    def __init__(self, cargo=None, next=None):
        self.car = cargo
        self.cdr = next
    def __str__(self):
        return str(self.car)

def display(lst):
    if lst:
        w("%s " % lst)
        display(lst.cdr)
    else:
        w("nil\n")
```

<https://riptutorial.com/zh-TW/python/topic/6916/>

# 197: GZip

GNUgzipgunzip。

zlib。

gzipGzipFilePython。 GzipFilegzip。

## Examples

### GNU zip

```
import gzip
import os

outfile = 'example.txt.gz'
output = gzip.open(outfile, 'wb')
try:
    output.write('Contents of the example file go here.\n')
finally:
    output.close()

print outfile, 'contains', os.stat(outfile).st_size, 'bytes of compressed data'
os.system('file -b --mime %s' % outfile)
```

1gzip\_write.py1.Run。

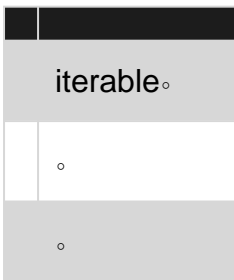
```
$ python gzip_write.py

application/x-gzip; charset=binary
example.txt.gz contains 68 bytes of compressed data
```

GZip <https://riptutorial.com/zh-TW/python/topic/8993/gzip>

# 198:

- `reducefunctioniterable [initializer]`



`reduce`

- `sum()`

```
sum([1,2,3]) # = 6
```

- `str.join`

```
'.join(['Hello', ',', ' World']) # = 'Hello, World'
```

- `reduce next`

```
# First falsy item:  
next((i for i in [100, [], 20, 0] if not i)) # = []
```

## Examples

```
# No import needed  
  
# No import required...  
from functools import reduce # ... but it can be loaded from the functools module  
  
from functools import reduce # mandatory
```

`reduceiterable`

```
def add(s1, s2):  
    return s1 + s2  
  
asequence = [1, 2, 3]  
  
reduce(add, asequence) # equivalent to: add(add(1,2),3)  
# Out: 6
```

`add` Pythonoperator

```
import operator
reduce(operator.add, asequence)
# Out: 6
```

reduce

```
reduce(add, asequence, 10)
# Out: 16
```

## reduce

```
def multiply(s1, s2):
    print('{arg1} * {arg2} = {res}'.format(arg1=s1,
  arg2=s2,
  res=s1*s2))
    return s1 * s2

asequence = [1, 2, 3]
```

initializer

```
cumprod = reduce(multiply, asequence, 5)
# Out: 5 * 1 = 5
#      5 * 2 = 10
#      10 * 3 = 30
print(cumprod)
# Out: 30
```

initializerreduce

```
cumprod = reduce(multiply, asequence)
# Out: 1 * 2 = 2
#      2 * 3 = 6
print(cumprod)
# Out: 6
```

```
import operator
reduce(operator.mul, [10, 5, -3])
# Out: -150
```

/

reduceiterableany()all()

```
import operator
# non short-circuit "all"
reduce(operator.and_, [False, True, True, True]) # = False

# non short-circuit "any"
reduce(operator.or_, [True, False, False, False]) # = True
```

## truthy / falsy

```
# First falsy element or last element if all are truthy:
reduce(lambda i, j: i and j, [100, [], 20, 10])    # = []
reduce(lambda i, j: i and j, [100, 50, 20, 10])    # = 10

# First truthy element or last element if all falsy:
reduce(lambda i, j: i or j, [100, [], 20, 0])      # = 100
reduce(lambda i, j: i or j, ['', {}, [], None])    # = None
```

lambda

```
def do_or(i, j):
    return i or j

def do_and(i, j):
    return i and j

reduce(do_or, [100, [], 20, 0])                    # = 100
reduce(do_and, [100, [], 20, 0])                   # = []
```

<https://riptutorial.com/zh-TW/python/topic/328/>



# 199:

Queue。 1。 2. LifoQueue 3.PriorityQueue1。 2。

## Examples

```
from Queue import Queue

question_queue = Queue()

for x in range(1,10):
    temp_dict = ('key', x)
    question_queue.put(temp_dict)

while(not question_queue.empty()):
    item = question_queue.get()
    print(str(item))
```

```
('key', 1)
('key', 2)
('key', 3)
('key', 4)
('key', 5)
('key', 6)
('key', 7)
('key', 8)
('key', 9)
```

<https://riptutorial.com/zh-TW/python/topic/8339/>

## 200:

- `random.seeda == 2python 3.x`
- `random.getstate`
- `random.setstate`
- `random.randintab`
- `random.randrange`
- `random.randrangestartstopstep = 1`
- `random.choiceSEQ`
- `random.shufflexrandom = random.random`
- `random.samplepopulationk`

## Examples

```
import random
```

```
random.shuffle()/° list
```

```
laughs = ["Hi", "Ho", "He"]
```

```
random.shuffle(laughs)      # Shuffles in-place! Don't do: laughs = random.shuffle(laughs)
```

```
print(laughs)
```

```
# Out: ["He", "Hi", "Ho"] # Output may vary!
```

```
print(random.choice(laughs))
```

```
# Out: He # Output may vary!
```

```
choice
```

```
#           |--sequence--|--number--|
print(random.sample(laughs, 1)) # Take one element
# Out: ['Ho'] # Output may vary!
```

```
print(random.sample(laughs, 3)) # Take 3 random element from the sequence.
# Out: ['Ho', 'He', 'Hi'] # Output may vary!
```

```
print(random.sample(laughs, 4)) # Take 4 random element from the 3-item sequence.
```

### ValueError

## randintrandrangerandomuniform

```
import random
```

---

## randint

xy

```
random.randint(x, y)
```

18

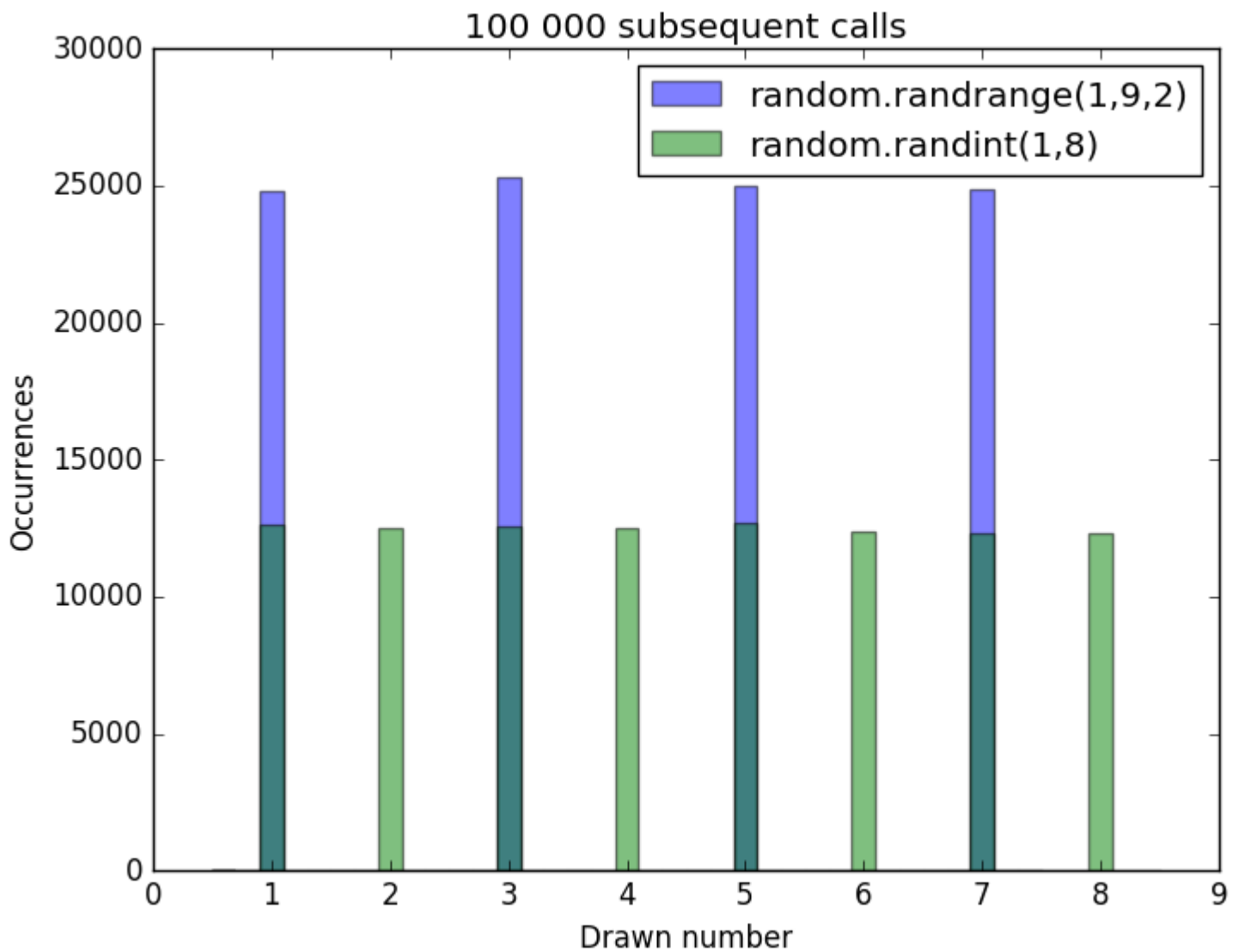
```
random.randint(1, 8) # Out: 8
```

---

## randrange

random.randrangerangerandom.randint

```
random.randrange(100)      # Random integer between 0 and 99
random.randrange(20, 50)   # Random integer between 20 and 49
random.rangrange(10, 20, 3) # Random integer between 10 and 19 with step 3 (10, 13, 16 and 19)
```



01

```
random.random() # Out: 0.66486093215306317
```

xy

```
random.uniform(1, 8) # Out: 3.726062641730108
```

```
random.seed(5) # Create a fixed state
print(random.randrange(0, 10)) # Get a random integer between 0 and 9
# Out: 9
print(random.randrange(0, 10))
# Out: 4
```

“”

```
random.seed(5) # Reset the random module to the same fixed state.
print(random.randrange(0, 10))
# Out: 9
print(random.randrange(0, 10))
# Out: 4
```

94 ◦ `getstate`/`setstate`

```
save_state = random.getstate() # Get the current state
print(random.randrange(0, 10))
# Out: 5
print(random.randrange(0, 10))
# Out: 8

random.setstate(save_state) # Reset to saved state
print(random.randrange(0, 10))
# Out: 5
print(random.randrange(0, 10))
# Out: 8
```

None seed

```
random.seed(None)
```

seed

```
random.seed()
```

Python Mersenne Twister [PRNG](#) ◦

[SystemRandom](#).`urandom` [CPRNG](#) ◦

`SystemRandom` ◦ ◦

```
from random import SystemRandom
secure_rand_gen = SystemRandom()
```

`[0, 20]` `randrange()` `[0, 20]` `10` `in` `randrange()`

```
print([secure_rand_gen.randrange(10) for i in range(10)])
# [9, 6, 9, 2, 2, 3, 8, 0, 9, 9]
```

`randint`

```
print(secure_rand_gen.randint(0, 20))
# 5
```

◦ ◦

[os.urandom](#) ◦

```
string punctuationascii_lettersdigits
```

```
from string import punctuation, ascii_letters, digits
```

```
symbolssymbols
```

```
symbols = ascii_letters + digits + punctuation
```

```
o
```

```
random.SystemRandom() 10
```

```
secure_random = random.SystemRandom()
password = "".join(secure_random.choice(symbols) for i in range(10))
print(password) # '^@g;J?]M6e'
```

```
random = random.choice random.randint - o
```

[Mersenne Twister PRNG CSPRNG](#) o o SystemRandom

## Python 3.x 3.6

## Python 3.6 secrets

```
"""
```

```
import string
alphabet = string.ascii_letters + string.digits
while True:
    password = ''.join(choice(alphabet) for i in range(10))
    if (any(c.islower() for c in password)
        and any(c.isupper() for c in password)
        and sum(c.isdigit() for c in password) >= 3):
        break
```

```
import random

probability = 0.3

if random.random() < probability:
    print("Decision with probability 0.3")
else:
    print("Decision with probability 0.7")
```

<https://riptutorial.com/zh-TW/python/topic/239/>

# 201:

## Examples

Python ◦ ◦ `__action__` ◦ [Python](#) ◦

◦ [Python](#) ◦ [Python](#) ◦ [Vector](#) ◦ ◦

```
class Vector(object):
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, v):
        # Addition with another vector.
        return Vector(self.x + v.x, self.y + v.y)

    def __sub__(self, v):
        # Subtraction with another vector.
        return Vector(self.x - v.x, self.y - v.y)

    def __mul__(self, s):
        # Multiplication with a scalar.
        return Vector(self.x * s, self.y * s)

    def __div__(self, s):
        # Division with a scalar.
        float_s = float(s)
        return Vector(self.x / float_s, self.y / float_s)

    def __floordiv__(self, s):
        # Division with a scalar (value floored).
        return Vector(self.x // s, self.y // s)

    def __repr__(self):
        # Print friendly representation of Vector class. Else, it would
        # show up like, <__main__.Vector instance at 0x01DDDDC8>.
        return '<Vector (%f, %f)>' % (self.x, self.y, )

a = Vector(3, 5)
b = Vector(2, 7)

print a + b # Output: <Vector (5.000000, 12.000000)>
print b - a # Output: <Vector (-1.000000, 2.000000)>
print b * 1.3 # Output: <Vector (2.600000, 9.100000)>
print a // 17 # Output: <Vector (0.000000, 0.000000)>
print a / 17 # Output: <Vector (0.176471, 0.294118)>
```

◦ ◦

<https://riptutorial.com/zh-TW/python/topic/946/>

---

# 202: Python

## Examples

### IronPython

C.NETMonoApache License 2.0. DLR. 2.73.

### CPython

- .NET Framework.
- Unicode.
- CCPython.
- Global Interpreter Lock.
- .

---

```
print "Hello World!"
```

### .NET

```
import clr
from System import Console
Console.WriteLine("Hello World!")
```

- 
- - [GitHub](#)

### Jython

JavaJVMPython Software Foundation License. 2.73.

### CPython

- JVM.
- Unicode.
- CCPython.
- Global Interpreter Lock.
- .

---

```
print "Hello World!"
```



## Java

```
from java.lang import System
System.out.println("Hello World!")
```

- 
- [Mercurial](#)

## Transcrypt

TranscryptPythonJavascript。

- PythonOOCPython
- WebJavaScriptPython
- URLPyPi
- PythonJavaScript
- 
- KBMB
- JavaScriptmemoization
- 

650 kBPythonJavaScript。 JavaScript。

## HTML

```
<script src="__javascript__/hello.js"></script>
<h2>Hello demo</h2>

<p>
<div id = "greet">...</div>
<button onclick="hello.solarSystem.greet ()">Click me repeatedly!</button>

<p>
<div id = "explain">...</div>
<button onclick="hello.solarSystem.explain ()">And click me repeatedly too!</button>
```

## JavaScriptDOM

```
from itertools import chain

class SolarSystem:
    planets = [list (chain (planet, (index + 1,))) for index, planet in enumerate ((
        ('Mercury', 'hot', 2240),
        ('Venus', 'sulphurous', 6052),
        ('Earth', 'fertile', 6378),
        ('Mars', 'reddish', 3397),
```

```

        ('Jupiter', 'stormy', 71492),
        ('Saturn', 'ringed', 60268),
        ('Uranus', 'cold', 25559),
        ('Neptune', 'very cold', 24766)
    )]

    lines = (
        '{} is a {} planet',
        'The radius of {} is {} km',
        '{} is planet nr. {} counting from the sun'
    )

    def __init__ (self):
        self.lineIndex = 0

    def greet (self):
        self.planet = self.planets [int (Math.random () * len (self.planets))]
        document.getElementById ('greet') .innerHTML = 'Hello {}'.format (self.planet [0])
        self.explain ()

    def explain (self):
        document.getElementById ('explain').innerHTML = (
            self.lines [self.lineIndex] .format (self.planet [0], self.planet [self.lineIndex
+ 1])
        )
        self.lineIndex = (self.lineIndex + 1) % 3
        solarSystem = SolarSystem ()

```

---

## JavaScript

TranscryptJavaScript. ao react.jsriot.jsfabric.jsnode.js.

---

## PythonJavaScript

```

class A:
    def __init__ (self, x):
        self.x = x

    def show (self, label):
        print ('A.show', label, self.x)

class B:
    def __init__ (self, y):
        alert ('In B constructor')
        self.y = y

    def show (self, label):
        print ('B.show', label, self.y)

class C (A, B):
    def __init__ (self, x, y):
        alert ('In C constructor')
        A.__init__ (self, x)
        B.__init__ (self, y)
        self.show ('constructor')

```

```

def show (self, label):
    B.show (self, label)
    print ('C.show', label, self.x, self.y)

a = A (1001)
a.show ('america')

b = B (2002)
b.show ('russia')

c = C (3003, 4004)
c.show ('netherlands')

show2 = c.show
show2 ('copy')

```

## JavaScript

```

var A = __class__ ('A', [object], {
  get __init__ () {return __get__ (this, function (self, x) {
    self.x = x;
  }});
  get show () {return __get__ (this, function (self, label) {
    print ('A.show', label, self.x);
  }});
});
var B = __class__ ('B', [object], {
  get __init__ () {return __get__ (this, function (self, y) {
    alert ('In B constructor');
    self.y = y;
  }});
  get show () {return __get__ (this, function (self, label) {
    print ('B.show', label, self.y);
  }});
});
var C = __class__ ('C', [A, B], {
  get __init__ () {return __get__ (this, function (self, x, y) {
    alert ('In C constructor');
    A.__init__ (self, x);
    B.__init__ (self, y);
    self.show ('constructor');
  }});
  get show () {return __get__ (this, function (self, label) {
    B.show (self, label);
    print ('C.show', label, self.x, self.y);
  }});
});
var a = A (1001);
a.show ('america');
var b = B (2002);
b.show ('russia');
var c = C (3003, 4004);
c.show ('netherlands');
var show2 = c.show;
show2 ('copy');

```

- 
- [http //www.transcript.org/](http://www.transcript.org/)
  -

[https //github.com/JdeH/Transcrypt](https://github.com/JdeH/Transcrypt)

Python <https://riptutorial.com/zh-TW/python/topic/5225/python>

# 203:

## Examples

### Pyglet

```
import pyglet
audio = pyglet.media.load("audio.wav")
audio.play()
```

### pyglet

### WAV

- Windows

```
import winsound
winsound.PlaySound("path_to_wav_file.wav", winsound.SND_FILENAME)
```

- /
- /

```
import wave
with wave.open("path_to_wav_file.wav", "rb") as wav_file:    # Open WAV file in read-only
mode.
    # Get basic information.
    n_channels = wav_file.getnchannels()    # Number of channels. (1=Mono, 2=Stereo).
    sample_width = wav_file.getsampwidth()    # Sample width in bytes.
    framerate = wav_file.getframerate()    # Frame rate.
    n_frames = wav_file.getnframes()    # Number of frames.
    comp_type = wav_file.getcomptype()    # Compression type (only supports "NONE").
    comp_name = wav_file.getcompname()    # Compression name.

    # Read audio data.
    frames = wav_file.readframes(n_frames)    # Read n_frames new frames.
    assert len(frames) == sample_width * n_frames

# Duplicate to a new WAV file.
with wave.open("path_to_new_wav_file.wav", "wb") as wav_file:    # Open WAV file in write-only
mode.
    # Write audio data.
    params = (n_channels, sample_width, framerate, n_frames, comp_type, comp_name)
    wav_file.setparams(params)
    wav_file.writeframes(frames)
```

### pythonffmpeg

```
from subprocess import check_call

ok = check_call(['ffmpeg', '-i', 'input.mp3', 'output.wav'])
if ok:
    with open('output.wav', 'rb') as f:
        wav_file = f.read()
```

- <http://superuser.com/questions/507386/why-would-i-choose-libav-over-ffmpeg-or-is-there-even-a-difference>
- [ffmpeglibavavconv](#)

## Windows

Windows `winsound`

```
import winsound
freq = 2500 # Set frequency To 2500 Hertz
dur = 1000 # Set duration To 1000 ms == 1 second
winsound.Beep(freq, dur)
```

<https://riptutorial.com/zh-TW/python/topic/8189/>

# 204:

Python ◦ ◦ ◦

◦

## Examples

PythonJavaC ++ ◦ ◦

```
class BaseClass(object):
    pass

class DerivedClass(BaseClass):
    pass
```

BaseClass DerivedClassBaseClass ◦ Python 2.2 [objectobject](#) ◦

Rectangleobject

```
class Rectangle():
    def __init__(self, w, h):
        self.w = w
        self.h = h

    def area(self):
        return self.w * self.h

    def perimeter(self):
        return 2 * (self.w + self.h)
```

RectangleSquare **square** ◦

```
class Square(Rectangle):
    def __init__(self, s):
        # call parent constructor, w and h are both s
        super(Square, self).__init__(s, s)
        self.s = s
```

SquareRectangle ◦ `super()Rectangle__init__()` ◦ Python 3 `super()` ◦

```
r.area()
# Output: 12
r.perimeter()
# Output: 14

s.area()
# Output: 4
s.perimeter()
# Output: 8
```

```
issubclass(DerivedClass, BaseClass) DerivedClassBaseClassTrue
```

```
isinstance(s, Class)TrueSClassClass
```

```
# subclass check
issubclass(Square, Rectangle)
# Output: True

# instantiate
r = Rectangle(3, 4)
s = Square(2)

isinstance(r, Rectangle)
# Output: True
isinstance(r, Square)
# Output: False
# A rectangle is not a square

isinstance(s, Rectangle)
# Output: True
# A square is a rectangle
isinstance(s, Square)
# Output: True
```

o

```
class C:
    x = 2 # class variable

    def __init__(self, y):
        self.y = y # instance variable

C.x
# 2
C.y
# AttributeError: type object 'C' has no attribute 'y'

c1 = C(3)
c1.x
# 2
c1.y
# 3

c2 = C(4)
c2.x
# 2
c2.y
# 4
```

## class

```
c2.x = 4
c2.x
# 4
C.x
# 2
```



o

```
class D:
    x = []
    def __init__(self, item):
        self.x.append(item) # note that this is not an assignment!

d1 = D(1)
d2 = D(2)

d1.x
# [1, 2]
d2.x
# [1, 2]
D.x
# [1, 2]
```

Python 3 def o Af Af

### Python 3.x 3.0

```
class A(object):
    def f(self, x):
        return 2 * x

A.f
# <function A.f at ...> (in Python 3.x)
```

Python 2 instancemethod o \_\_func\_\_ o

### Python 2.x 2.3

```
A.f
# <unbound method A.f> (in Python 2.x)
A.f.__class__
# <type 'instancemethod'>
A.f.__func__
# <function f at ...>
```

- Python 3 Python 2 o

### Python 3.x 3.0

```
import inspect

inspect.isfunction(A.f)
# True
inspect.ismethod(A.f)
# False
```

### Python 2.x 2.3

```
import inspect

inspect.isfunction(A.f)
# False
```

```
inspect.ismethod(A.f)
# True
```

## Python/AAf ◦

```
A.f(1, 7)
# Python 2: TypeError: unbound method f() must be called with
#           A instance as first argument (got int instance instead)
# Python 3: 14
a = A()
A.f(a, 20)
# Python 2 & 3: 40
```

aAAafAf ""a - Pythona◦

af\_\_getattr\_\_a af Ammethod Afm.\_\_func\_\_ am.\_\_self\_\_ ◦ m(...) => m.\_\_func\_\_(m.\_\_self\_\_, ...)  
◦ ◦ Python 23◦

```
a = A()
a.f
# <bound method A.f of <__main__.A object at ...>>
a.f(2)
# 4

# Note: the bound method object a.f is recreated *every time* you call it:
a.f is a.f # False
# As a performance optimization you can store the bound method in the object's
# __dict__, in which case the method object will remain fixed:
a.f = a.f
a.f is a.f # True
```

Python - ◦ ◦ m.\_\_self\_\_ = type(a) ◦ a◦ ◦

```
class D(object):
    multiplier = 2

    @classmethod
    def f(cls, x):
        return cls.multiplier * x

    @staticmethod
    def g(name):
        print("Hello, %s" % name)

D.f
# <bound method type.f of <class '__main__.D'>>
D.f(12)
# 24
D.g
# <function D.g at ...>
D.g("world")
# Hello, world
```

```
d = D()
```

```

d.multiplier = 1337
(D.multiplier, d.multiplier)
# (2, 1337)
d.f
# <bound method D.f of <class '__main__.D'>>
d.f(10)
# 20

```

staticmethods `__get__` `__set` `__del__` ◦ classmethods staticmethods

- [Python@staticmethod@classmethod](#)
- [@classmethod@staticmethod](#)

## Python 2.x 2.2.0

Python 2.2 ◦ object ◦ ◦

```

# new-style class
class New(object):
    pass

# new-style instance
new = New()

new.__class__
# <class '__main__.New'>
type(new)
# <class '__main__.New'>
issubclass(New, object)
# True

```

object ◦ instance ◦

```

# old-style class
class Old:
    pass

# old-style instance
old = Old()

old.__class__
# <class __main__.Old at ...>
type(old)
# <type 'instance'>
issubclass(Old, object)
# False

```

## Python 3.x 3.0.0

Python 3 ◦

Python 3 `object MyClass(object)` ◦

```

class MyClass:
    pass

```

```

my_inst = MyClass()

type(my_inst)
# <class '__main__.MyClass'>
my_inst.__class__
# <class '__main__.MyClass'>
issubclass(MyClass, object)
# True

```

```

class Rectangle(object):
    def __init__(self, width, height, color='blue'):
        self.width = width
        self.height = height
        self.color = color

    def area(self):
        return self.width * self.height

# Create some instances of the class
default_rectangle = Rectangle(2, 3)
print(default_rectangle.color) # blue

red_rectangle = Rectangle(2, 3, 'red')
print(red_rectangle.color) # red

```

◦

```

class Rectangle2D(object):
    def __init__(self, width, height, pos=[0,0], color='blue'):
        self.width = width
        self.height = height
        self.pos = pos
        self.color = color

r1 = Rectangle2D(5,3)
r2 = Rectangle2D(7,8)
r1.pos[0] = 4
r1.pos # [4, 0]
r2.pos # [4, 0] r2's pos has changed as well

```

## Python◦

```

class Rectangle2D(object):
    def __init__(self, width, height, pos=None, color='blue'):
        self.width = width
        self.height = height
        self.pos = pos or [0, 0] # default value is [0, 0]
        self.color = color

r1 = Rectangle2D(5,3)
r2 = Rectangle2D(7,8)
r1.pos[0] = 4
r1.pos # [4, 0]
r2.pos # [0, 0] r2's pos hasn't changed

```

[Mutable Default Arguments“Least Astonishment”Mutable Default Argument](#) ◦

## Python C3 MRO

```
class Foo(object):
    foo = 'attr foo of Foo'

class Bar(object):
    foo = 'attr foo of Bar' # we won't see this.
    bar = 'attr bar of Bar'

class FooBar(Foo, Bar):
    foobar = 'attr foobar of FooBar'
```

### FooBarfooFoo

```
fb = FooBar()
```

```
>>> fb.foo
'attr foo of Foo'
```

### FooBarMRO

```
>>> FooBar.mro()
[<class '__main__.FooBar'>, <class '__main__.Foo'>, <class '__main__.Bar'>, <type 'object'>]
```

## PythonMRO

1. FooBarFoo
2. object Bar
3. ◦

### FooBarBarBarFooBar◦

## Python ◦

super ◦ **super**◦

```
class Foo(object):
    def foo_method(self):
        print "foo Method"

class Bar(object):
    def bar_method(self):
        print "bar Method"

class FooBar(Foo, Bar):
    def foo_method(self):
        super(FooBar, self).foo_method()
```

### initinitineritanceinit◦

### FooinitBarinit

```

class Foo(object):
    def __init__(self):
        print "foo init"

class Bar(object):
    def __init__(self):
        print "bar init"

class FooBar(Foo, Bar):
    def __init__(self):
        print "foobar init"
        super(FooBar, self).__init__()

a = FooBar()

```

```

foobar init
foo init

```

## Bar. FooBarBarFoo.

```

print isinstance(a, FooBar)
print isinstance(a, Foo)
print isinstance(a, Bar)

```

```

True
True
True

```

`__get__` `__set__` `__delete__`.

`__set__` `__delete__`

`staticmethod` `classmethod` `property` `barFoofoo` = `foo.bar`

1. `bar Foo` `property`
2. `__dict__bar` `bar`
3. `Foo bar` `classmethod` `staticmethod` `AttributeError`

◦ ◦

Person

```

class Person(object):

    def __init__(self, first_name, last_name, age):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
        self.full_name = first_name + " " + last_name

    def greet(self):
        print("Hello, my name is " + self.full_name + ".")

```

◦ last\_name

```
class Person(object):

    def __init__(self, first_name, age, last_name=None):
        if last_name is None:
            self.first_name, self.last_name = first_name.split(" ", 2)
        else:
            self.first_name = first_name
            self.last_name = last_name

        self.full_name = self.first_name + " " + self.last_name
        self.age = age

    def greet(self):
        print("Hello, my name is " + self.full_name + ".")
```

1. first\_name last\_name first\_name ◦ /if / elif / else ◦

2. last\_name None first\_name /.....

◦ from\_full\_name from\_full\_name classmethod ◦

```
class Person(object):

    def __init__(self, first_name, last_name, age):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
        self.full_name = first_name + " " + last_name

    @classmethod
    def from_full_name(cls, name, age):
        if " " not in name:
            raise ValueError
        first_name, last_name = name.split(" ", 2)
        return cls(first_name, last_name, age)

    def greet(self):
        print("Hello, my name is " + self.full_name + ".")
```

```
cls self from_full_name ◦ self ◦ cls Person from_full_name Person(first_name, last_name, age) Person
__init__ Person ◦ Employee Of Person from_full_name Employee ◦
```

Person\_\_init\_\_

```
In [2]: bob = Person("Bob", "Bobberson", 42)

In [3]: alice = Person.from_full_name("Alice Henderson", 31)

In [4]: bob.greet()
Hello, my name is Bob Bobberson.

In [5]: alice.greet()
Hello, my name is Alice Henderson.
```

- [Python @classmethod@staticmethod](#)
- <https://docs.python.org/2/library/functions.html#classmethod>
- <https://docs.python.org/3.5/library/functions.html#classmethod>

◦ /◦

```
class Country(object):
    def __init__(self):
        self.cities=[]

    def addCity(self,city):
        self.cities.append(city)

class City(object):
    def __init__(self, numPeople):
        self.people = []
        self.numPeople = numPeople

    def addPerson(self, person):
        self.people.append(person)

    def join_country(self,country):
        self.country = country
        country.addCity(self)

        for i in range(self.numPeople):
            person(i).join_city(self)

class Person(object):
    def __init__(self, ID):
        self.ID=ID

    def join_city(self, city):
        self.city = city
        city.addPerson(self)

    def people_in_my_country(self):
        x= sum([len(c.people) for c in self.city.country.cities])
        return x

US=Country()
NYC=City(10).join_country(US)
SF=City(5).join_country(US)

print(US.cities[0].people[0].people_in_my_country())

# 15
```

“”◦ A

```
class A(object):
    def __init__(self, num):
        self.num = num
```



```
def __add__(self, other):
    return A(self.num + other.num)
```

◦ ◦

```
def get_num(self):
    return self.num
```

AA◦

```
A.get_num = get_num
```

◦

get\_numA

◦

```
foo = A(42)

A.get_num = get_num

bar = A(6);

foo.get_num() # 42

bar.get_num() # 6
```

◦

dir()

```
dir(Class)
```

```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__',
 '__eq__', '__format__', '__ge__', '__getattr__', '__getitem__', '__gt__', '__hash__',
 '__iadd__', '__imul__', '__init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__',
 '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__',
 '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear',
 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

“”◦ \_\_

```
>>> [m for m in dir(list) if not m.startswith('__')]
['append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse',
 'sort']
```

`__dir__()`◦ `dir()``__dir__()` Python◦ `dir`◦ [python](#)

## dir dict. getattr .

dir .

```
class Person(object):
    """A simple class.""" # docstring
    species = "Homo Sapiens" # class attribute

    def __init__(self, name): # special method
        """This is the initializer. It's a special
        method (see below).
        """
        self.name = name # instance attribute

    def __str__(self): # special method
        """This method is run when Python tries
        to cast the object to a string. Return
        this string when using print(), etc.
        """
        return self.name

    def rename(self, renamed): # regular method
        """Reassign and print the name attribute."""
        self.name = renamed
        print("Now my name is {}".format(self.name))
```

1. .
2. .
3. docstring \_\_init\_\_() . .
4. .
5. . \_\_init\_\_();\_\_init\_\_() .
6. . self selfself Python Python .
7. . Python private C ++ / Java public . "" .
8. . \_\_functionname\_\_(self, other\_stuff) . "" Python. Python. .

Person

```
>>> # Instances
>>> kelly = Person("Kelly")
>>> joseph = Person("Joseph")
>>> john_doe = Person("John Doe")
```

Person kelly joseph john\_doe .

```
>>> # Attributes
>>> kelly.species
'Homo Sapiens'
```

```
>>> john_doe.species
'Homo Sapiens'
>>> joseph.species
'Homo Sapiens'
>>> kelly.name
'Kelly'
>>> joseph.name
'Joseph'
```

```
>>> # Methods
>>> john_doe.__str__()
'John Doe'
>>> print(john_doe)
'John Doe'
>>> john_doe.rename("John")
'Now my name is John'
```

## Python ◦

```
class MyClass(object):

    def __init__(self):
        self._my_string = ""

    @property
    def string(self):
        """A profoundly important string."""
        return self._my_string

    @string.setter
    def string(self, new_value):
        assert isinstance(new_value, str), \
            "Give me a string, not a %r!" % type(new_value)
        self._my_string = new_value

    @string.deleter
    def x(self):
        self._my_string = None
```

MyClass.string

```
mc = MyClass()
mc.string = "String!"
print(mc.string)
del mc.string
```

## ◦ API - ◦

“” - ◦

```
class Character(object):
    def __init__(name, max_hp):
        self._name = name
```

```

    self._hp = max_hp
    self._max_hp = max_hp

# Make hp read only by not providing a set method
@property
def hp(self):
    return self._hp

# Make name read only by not providing a set method
@property
def name(self):
    return self.name

def take_damage(self, damage):
    self.hp -= damage
    self.hp = 0 if self.hp <0 else self.hp

@property
def is_alive(self):
    return self.hp != 0

@property
def is_wounded(self):
    return self.hp < self.max_hp if self.hp > 0 else False

@property
def is_dead(self):
    return not self.is_alive

bilbo = Character('Bilbo Baggins', 100)
bilbo.hp
# out : 100
bilbo.hp = 200
# out : AttributeError: can't set attribute
# hp attribute is read only.

bilbo.is_alive
# out : True
bilbo.is_wounded
# out : False
bilbo.is_dead
# out : False

bilbo.take_damage( 50 )

bilbo.hp
# out : 50

bilbo.is_alive
# out : True
bilbo.is_wounded
# out : True
bilbo.is_dead
# out : False

bilbo.take_damage( 50 )
bilbo.hp
# out : 0

bilbo.is_alive
# out : False

```

```
bilbo.is_wounded
# out : False
bilbo.is_dead
# out : True
```

## /◦ python singleton ◦

```
class Singleton:
    def __new__(cls):
        try:
            it = cls.__it__
        except AttributeError:
            it = cls.__it__ = object.__new__(cls)
        return it

    def __repr__(self):
        return '<{}>'.format(self.__class__.__name__.upper())

    def __eq__(self, other):
        return other is self
```

## ◦ Singleton

```
class Singleton:
    """
    A non-thread-safe helper class to ease implementing singletons.
    This should be used as a decorator -- not a metaclass -- to the
    class that should be a singleton.

    The decorated class can define one `__init__` function that
    takes only the `self` argument. Other than that, there are
    no restrictions that apply to the decorated class.

    To get the singleton instance, use the `Instance` method. Trying
    to use `__call__` will result in a `TypeError` being raised.

    Limitations: The decorated class cannot be inherited from.

    """
    def __init__(self, decorated):
        self._decorated = decorated

    def Instance(self):
        """
        Returns the singleton instance. Upon its first call, it creates a
        new instance of the decorated class and calls its `__init__` method.
        On all subsequent calls, the already created instance is returned.

        """
        try:
            return self._instance
        except AttributeError:
            self._instance = self._decorated()
            return self._instance

    def __call__(self):
        raise TypeError('Singletons must be accessed through `Instance()`.')
```

```
def __instancecheck__(self, inst):
    return isinstance(inst, self._decorated)
```

Instance

```
@Singleton
class Single:
    def __init__(self):
        self.name=None
        self.val=0
    def getName(self):
        print(self.name)

x=Single.Instance()
y=Single.Instance()
x.name='I\'m single'
x.getName() # outputs I'm single
y.getName() # outputs I'm single
```

<https://riptutorial.com/zh-TW/python/topic/419/>

## 205: `__str__` `__repr__`

```
__str__ "Ace of Spaces" __repr__ eval◦
```

`repr()` Python

`eval`◦

```
__str__ "Ace of Spaces" __repr__ Card('Spades', 1)
```

```
"""eval
```

```
object -> string -> object
```

```
def __repr__(self):  
    return "Card(%s, %d)" % (self.suit, self.pips)
```

[1]◦ `cpython`◦

[2]◦ `str()` / `repr()`◦ `str(str())`◦ `repr()`◦

## Examples

Python

```
class Card:  
    def __init__(self, suit, pips):  
        self.suit = suit  
        self.pips = pips
```

```
ace_of_spades = Card('Spades', 1)  
four_of_clubs = Card('Clubs', 4)  
six_of_hearts = Card('Hearts', 6)
```

```
"""
```

```
my_hand = [ace_of_spades, four_of_clubs, six_of_hearts]
```

```
print(my_hand)
```

```
[<__main__.Card instance at 0x0000000002533788>,  
<__main__.Card instance at 0x00000000025B95C8>,  
<__main__.Card instance at 0x00000000025FF508>]
```

```
print(ace_of_spades)
```

```
<__main__.Card instance at 0x0000000002533788>
```

◦ ◦

◦ `print(ace_of_spades)` Python `ace_of_spades` `Card` ◦

`typeid` ◦ `print` ◦ [1]

Python ""

```
string_of_card = str(ace_of_spades)
print(string_of_card)
```

Card

Python `Card` `Card`

```
str(ace_of_spades) Card
```

1

Python ◦ `__str__` "dunder" ""

Python `__str__`

Card

```
class Card:
    def __init__(self, suit, pips):
        self.suit = suit
        self.pips = pips

    def __str__(self):
        special_names = {1:'Ace', 11:'Jack', 12:'Queen', 13:'King'}

        card_name = special_names.get(self.pips, str(self.pips))

        return "%s of %s" % (card_name, self.suit)
```

Card `__str__` ◦

"" ◦ `str(ace_of_spades)` `str(ace_of_spades)` ◦ `__str__` ◦

`__str__` `self` additional ◦

```
ace_of_spades = Card('Spades', 1)
```



```
print(ace_of_spades)
```

Ace of Spades

Card hand ◦

```
my_hand = [ace_of_spades, four_of_clubs, six_of_hearts]
print(my_hand)
```

```
[<__main__.Card instance at 0x00000000026F95C8>,
 <__main__.Card instance at 0x000000000273F4C8>,
 <__main__.Card instance at 0x0000000002732E08>]
```

## PythonCard

# 2

Python◦ Python\_\_str\_\_◦

\_\_repr\_\_ “◦ ”[2]

print

\_\_str\_\_repr\_\_

```
class Card:
    special_names = {1:'Ace', 11:'Jack', 12:'Queen', 13:'King'}

    def __init__(self, suit, pips):
        self.suit = suit
        self.pips = pips

    def __str__(self):
        card_name = Card.special_names.get(self.pips, str(self.pips))
        return "%s of %s (S)" % (card_name, self.suit)

    def __repr__(self):
        card_name = Card.special_names.get(self.pips, str(self.pips))
        return "%s of %s (R)" % (card_name, self.suit)
```

\_\_str\_\_repr\_\_(S)\_\_str\_\_(R)\_\_repr\_\_repr\_\_◦

\_\_str\_\_ \_\_repr\_\_◦

```
ace_of_spades = Card('Spades', 1)
four_of_clubs = Card('Clubs', 4)
six_of_hearts = Card('Hearts', 6)

my_hand = [ace_of_spades, four_of_clubs, six_of_hearts]

print(my_hand)          # [Ace of Spades (R), 4 of Clubs (R), 6 of Hearts (R)]
```

```
print(ace_of_spades)      # Ace of Spades (S)
```

`__str__` `Card` `print` `__repr__` `print` ◦

`str()` `repr()` `repr()` ◦

```
str_card = str(four_of_clubs)
print(str_card)           # 4 of Clubs (S)

repr_card = repr(four_of_clubs)
print(repr_card)         # 4 of Clubs (R)
```

```
print(four_of_clubs.__str__()) # 4 of Clubs (S)
print(four_of_clubs.__repr__()) # 4 of Clubs (R)
```



## Python `str()` `repr()` - ◦

◦ ◦ `__repr__` `__str__` `str()` Python `__repr__` `__repr__` ◦

Card

```
class Card:
    special_names = {1:'Ace', 11:'Jack', 12:'Queen', 13:'King'}

    def __init__(self, suit, pips):
        self.suit = suit
        self.pips = pips

    def __repr__(self):
        card_name = Card.special_names.get(self.pips, str(self.pips))
        return "%s of %s" % (card_name, self.suit)
```

`__repr__` ◦ `str()`

```
print(six_of_hearts)      # 6 of Hearts (implicit conversion)
print(str(six_of_hearts)) # 6 of Hearts (explicit conversion)
```

`repr()`

```
print([six_of_hearts])    #[6 of Hearts] (implicit conversion)
print(repr(six_of_hearts)) # 6 of Hearts (explicit conversion)
```



“” `__repr__` ◦ Raymond Hettinger `__repr__` Python ◦ ID ◦

`__repr__` `__str__` ◦ ◦

## eval-round-trip `__repr__` `__str__`

```
class Card:
    special_names = {1:'Ace', 11:'Jack', 12:'Queen', 13:'King'}

    def __init__(self, suit, pips):
        self.suit = suit
        self.pips = pips

    # Called when instance is converted to a string via str()
    # Examples:
    # print(card1)
    # print(str(card1))
    def __str__(self):
        card_name = Card.special_names.get(self.pips, str(self.pips))
        return "%s of %s" % (card_name, self.suit)

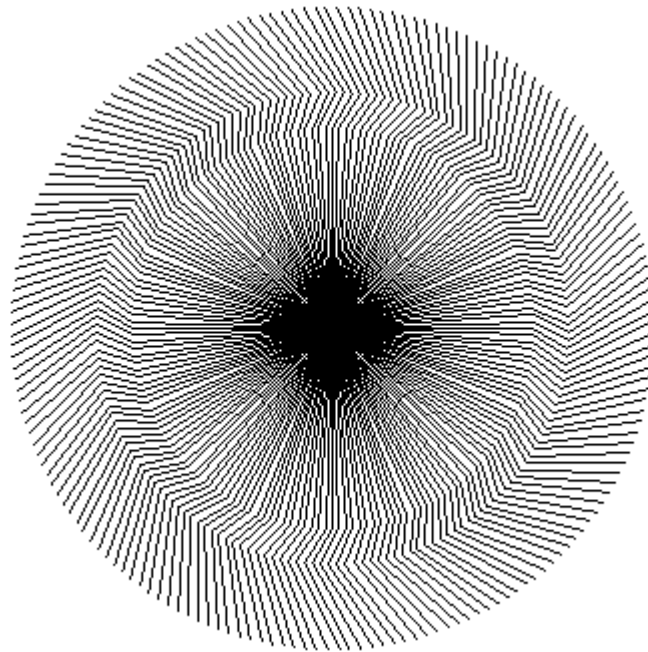
    # Called when instance is converted to a string via repr()
    # Examples:
    # print([card1, card2, card3])
    # print(repr(card1))
    def __repr__(self):
        return "Card(%s, %d)" % (self.suit, self.pips)
```

[\\_\\_str\\_\\_](https://riptutorial.com/zh-TW/python/topic/4845/---str----repr--) [repr](https://riptutorial.com/zh-TW/python/topic/4845/---str----repr--)

## 206:

### Examples

#### Ninja TwistTurtle Graphics



#### Turtle Graphics Ninja Twist

```
import turtle

ninja = turtle.Turtle()

ninja.speed(10)

for i in range(180):
    ninja.forward(100)
    ninja.right(30)
    ninja.forward(20)
    ninja.left(60)
    ninja.forward(50)
    ninja.right(30)

    ninja.penup()
    ninja.setposition(0, 0)
    ninja.pendown()

    ninja.right(2)

turtle.done()
```

<https://riptutorial.com/zh-TW/python/topic/7915/>

S. No		Contributors
1	Python	<p>A. Raza, Aaron Critchley, Abhishek Jain, AER, afeique, Akshay Kathpal, alejosocorro, Alessandro Trinca Tornidor, Alex Logan, ALinuxLover, Andrea, Andrii Abramov, Andy, Andy Hayden, angussidney, Ani Menon, Anthony Pham, Antoine Bolvy, Aquib Javed Khan, Ares, Arpit Solanki, B8vrede, Baaing Cow, baranskistad, Brian C, Bryan P, BSL-5, BusyAnt, Cbeb24404, ceruleus, ChaoticTwist, Charlie H, Chris Midgley, Christian Ternus, Claudiu, Clíodhna, CodenameLambda, CLDS EED, Community, Conrad.Dean, Daksh Gupta, Dania, Daniel Minnaar, Darth Shadow, Dartmouth, deenes, Delgan, depperm, DevD, dodell, Douglas Starnes, duckman_1991, Eamon Charles, edawine, Elazar, eli-bd, Enrico Maria De Angelis, Erica, Erica, ericdwang, Erik Godard, EsmaeelE, Filip Haglund, Firix, fox, Franck Dernoncourt, Fred Barclay, Freddy, Gerard Roche, gIS, GoatsWearHats, GThamizh, H. Pauwelyn, hardmooth, hayalci, hichris123, Ian, IanAuld, icesin, Igor Rausch, Ilyas Mimouni, itshejoker, J F, Jabba, jalanb, James, James Taylor, Jean-Francois T., jedwards, Jeffrey Lin, jfunez, JGreenwell, Jim Fasarakis Hilliard, jim opleydulven, jimsug, jmunsch, Johan Lundberg, John Donner, John Slegers, john400, jonrsharpe, Joseph True, JRodDynamite, jtbandes, Juan T, Kamran Mackey, Karan Chudasama, KerDam, Kevin Brown, Kiran Vemuri, kisanme, Lafexlos, Leon, Leszek Kicior, LostAvatar, Majid, manu, MANU, Mark Miller, Martijn Pieters, Mathias711, matsjoyce, Matt, Matthew Whitt, mdegis, Mechanic, Media, mertylidiran, metahost, Mike Driscoll, MikJR, Miljen Mikic, mnoronha, Morgoth, moshemeirelles, MSD, MSeifert, msohng, msw, muddyfish, Mukund B, Muntasir Alam, Nathan Arthur, Nathaniel Ford, Ned Batchelder, Ni., niyasc, nouλλΔ λζε.Ω, numbermaniac, orvi, Panda, Patrick Haugh, Pavan Nath, Peter Masiar, PSN, PsyKzz, pylang, pzp, Qchmqz, Quill, Rahul Nair, Rakitić, Ram Grandhi, rfkortekaas, rick112358, Robotski, rrao, Ryan Hilbert, Sam Krygsheld, Sangeeth Sudheer, SashaZd, Selcuk, Severiano Jaramillo Quintanar, Shiven, Shoe, Shog9, Sigitas Mockus, Simplans, Slayther, stark, StuxCrystal, SuperBiasedMan, Shadowfa, taylor swift, techydesigner, Tejus Prasad, TerryA, The_Curry_Man, TheGenie OfTruth, Timotheus.Kampik, tjohnson, Tom Barron, Tom de Geus, Tony Suffolk 66, tonyo, TPVasconcelos, user2314737, user2853437, user312016, Utsav T, vaichidrewar, vasilii111, Vin, W.Wong, weewooquestionnaire,</p>

		<a href="#">Will</a> , <a href="#">wintermute</a> , <a href="#">Yogendra Sharma</a> , <a href="#">Zach Janicki</a> , <a href="#">Zags</a>
2	<code>* args** kwargs</code>	<a href="#">cjds</a> , <a href="#">Eric Zhang</a> , <a href="#">ericmarkmartin</a> , <a href="#">Geeklhern</a> , <a href="#">J F</a> , <a href="#">Jeff Hutchins</a> , <a href="#">Jim Fasarakis Hilliard</a> , <a href="#">JuanPablo</a> , <a href="#">kdopen</a> , <a href="#">loading...</a> , <a href="#">Marlon Abeykoon</a> , <a href="#">Matthew Whitt</a> , <a href="#">Pasha</a> , <a href="#">pcurry</a> , <a href="#">PsyKzz</a> , <a href="#">Scott Mermelstein</a> , <a href="#">user2314737</a> , <a href="#">Valentin Lorentz</a> , <a href="#">Veedrac</a>
3	<code>__name__</code>	<a href="#">Anonymous</a> , <a href="#">BusyAnt</a> , <a href="#">Christian Ternus</a> , <a href="#">jonrsharpe</a> , <a href="#">Lutz Prechelt</a> , <a href="#">Steven Elliott</a>
4	<code>2to3</code>	<a href="#">Alessandro Trinca Tornidor</a> , <a href="#">Dartmouth</a> , <a href="#">Firix</a> , <a href="#">Kevin Brown</a> , <a href="#">Naga2Raja</a> , <a href="#">Stephen Leppik</a>
5	<code>ArcPy</code>	<a href="#">Midavalo</a> , <a href="#">PolyGeo</a> , <a href="#">Zhanping Shi</a>
6	<code>Asyncio</code>	<a href="#">2Cubed</a> , <a href="#">Alessandro Trinca Tornidor</a> , <a href="#">Cimbali</a> , <a href="#">hiro protagonist</a> , <a href="#">obust</a> , <a href="#">pylang</a> , <a href="#">RamenChef</a> , <a href="#">Seth M. Larson</a> , <a href="#">Simplans</a> , <a href="#">Stephen Leppik</a> , <a href="#">Udi</a>
7	<code>base64</code>	<a href="#">Thomas Gerot</a>
8	<code>ChemPy - python</code>	<a href="#">Biswa_9937</a>
9	<code>configparser</code>	<a href="#">Chinmay Hegde</a> , <a href="#">Dunatotatos</a>
10	<code>ctypes</code>	<a href="#">Or East</a>
11	<code>Deque</code>	<a href="#">Anthony Pham</a> , <a href="#">BusyAnt</a> , <a href="#">matsjoyce</a> , <a href="#">ravigadila</a> , <a href="#">Simplans</a> , <a href="#">Thomas Ahle</a> , <a href="#">user2314737</a>
12	<code>dis</code>	<a href="#">muddyfish</a> , <a href="#">user2314737</a>
13	<code>Django</code>	<a href="#">code_geek</a> , <a href="#">orvi</a>
14	<code>Functools</code>	<a href="#">Alessandro Trinca Tornidor</a> , <a href="#">enrico.bacis</a> , <a href="#">flamenco</a> , <a href="#">RamenChef</a> , <a href="#">Shrey Gupta</a> , <a href="#">Simplans</a> , <a href="#">Stephen Leppik</a> , <a href="#">StuxCrystal</a>
15	<code>hashlib</code>	<a href="#">Mark Omo</a> , <a href="#">xiaoyi</a>
16	<code>Heapq</code>	<a href="#">ettanany</a>
17	<code>HTML</code>	<a href="#">alecxe</a> , <a href="#">talhasch</a>
18	<code>ijson</code>	<a href="#">Prem Narain</a>
19	<code>IterablesIterators</code>	<a href="#">4444</a> , <a href="#">Conrad.Dean</a> , <a href="#">demonplus</a> , <a href="#">Ilia Barahovski</a> , <a href="#">Pythonista</a>
20	<code>Itertools</code>	<a href="#">ADITYA</a> , <a href="#">Alessandro Trinca Tornidor</a> , <a href="#">Andy Hayden</a> , <a href="#">balki</a> , <a href="#">bpachev</a> , <a href="#">Ffisegydd</a> , <a href="#">jackskis</a> , <a href="#">Julien Spronck</a> , <a href="#">Kevin Brown</a> , <a href="#">machine yearning</a> , <a href="#">nlsdfnbch</a> , <a href="#">pylang</a> , <a href="#">RahulHP</a> , <a href="#">RamenChef</a> ,

		<a href="#">Simplans</a> , <a href="#">Stephen Leppik</a> , <a href="#">Symmitchry</a> , <a href="#">Wickramaranga</a> , <a href="#">wnnmaw</a>
21	JSON	<a href="#">Indradhanush Gupta</a> , <a href="#">Leo</a> , <a href="#">Martijn Pieters</a> , <a href="#">pzp</a> , <a href="#">theheadofabroom</a> , <a href="#">Underyx</a> , <a href="#">Wolfgang</a>
22	kivy - NUIPython	<a href="#">dhimanta</a>
23	Neo4jCypherPy2Neo	<a href="#">Wingston Sharon</a>
24	os.path	<a href="#">Claudiu</a> , <a href="#">Fábio Perez</a> , <a href="#">girish946</a> , <a href="#">Jmills</a> , <a href="#">Szabolcs Dombi</a> , <a href="#">VJ Magar</a>
25	os	<a href="#">Andy</a> , <a href="#">Christian Ternus</a> , <a href="#">JelmerS</a> , <a href="#">JL Peyret</a> , <a href="#">mnoronha</a> , <a href="#">Vinzee</a>
26	Pandas Transform	<a href="#">Dee</a>
27	Pickle	<a href="#">J F</a> , <a href="#">Majid</a> , <a href="#">Or East</a> , <a href="#">RahulHP</a> , <a href="#">rfkortekaas</a> , <a href="#">zvone</a>
28	pipPyPI	<a href="#">Andy</a> , <a href="#">Arpit Solanki</a> , <a href="#">Community</a> , <a href="#">InitializeSahib</a> , <a href="#">JNat</a> , <a href="#">Mahdi</a> , <a href="#">Majid</a> , <a href="#">Matt Giltaji</a> , <a href="#">Nathaniel Ford</a> , <a href="#">Rápli András</a> , <a href="#">SerialDev</a> , <a href="#">Simplans</a> , <a href="#">Steve Barnes</a> , <a href="#">StuxCrystal</a> , <a href="#">tlo</a>
29	PostgreSQL	<a href="#">Alessandro Trinca Tornidor</a> , <a href="#">RamenChef</a> , <a href="#">Stephen Leppik</a> , <a href="#">user2027202827</a>
30	py.test	<a href="#">Andy</a> , <a href="#">Claudiu</a> , <a href="#">Ffisegydd</a> , <a href="#">Kinifwyne</a> , <a href="#">Matt Giltaji</a>
31	pyaudio	<a href="#">Biswa_9937</a>
32	pyautogui	<a href="#">Damien</a> , <a href="#">Rednivrug</a>
33	pygame	<a href="#">Anthony Pham</a> , <a href="#">Aryaman Arora</a> , <a href="#">Pavan Nath</a>
34	Pyglet	<a href="#">Comrade SparklePony</a> , <a href="#">Stephen Leppik</a>
35	PyInstaller - Python	<a href="#">ChaoticTwist</a> , <a href="#">Eric</a> , <a href="#">mnoronha</a>
36	Python HTTP Server	<a href="#">Arpit Solanki</a> , <a href="#">J F</a> , <a href="#">jmundsch</a> , <a href="#">Justin Chadwell</a> , <a href="#">Mark</a> , <a href="#">MervS</a> , <a href="#">orvi</a> , <a href="#">quantummind</a> , <a href="#">Raghav</a> , <a href="#">RamenChef</a> , <a href="#">Sachin Kalkur</a> , <a href="#">Simplans</a> , <a href="#">techydesigner</a>
37	Python Lex-Yacc	<a href="#">CLDSEED</a>
38	Python	<a href="#">David Heyman</a> , <a href="#">Faiz Halde</a> , <a href="#">Iván Rodríguez Torres</a> , <a href="#">J F</a> , <a href="#">Thomas Moreau</a> , <a href="#">Tyler Gubala</a>
39	PythonMutable vs ImmutableHashable	<a href="#">Cilyan</a>
40	Python	<a href="#">Imran Bughio</a> , <a href="#">mvis89</a> , <a href="#">Rednivrug</a>

41	PythonShell	<a href="#">mnoronha</a> , <a href="#">Shijo</a>
42	python	<a href="#">4444</a> , <a href="#">Alessandro Trinca Tornidor</a> , <a href="#">Fred Barclay</a> , <a href="#">RamenChef</a> , <a href="#">Ricardo</a> , <a href="#">Stephen Leppik</a>
43	Pythonpyserial	<a href="#">Alessandro Trinca Tornidor</a> , <a href="#">Ani Menon</a> , <a href="#">girish946</a> , <a href="#">mnoronha</a> , <a href="#">Saranjith</a> , <a href="#">user2314737</a>
44	Python	<a href="#">Alessandro Trinca Tornidor</a> , <a href="#">Anonymous</a> , <a href="#">eenblam</a> , <a href="#">Mahmoud Hashemi</a> , <a href="#">RamenChef</a> , <a href="#">Stephen Leppik</a>
45	PythonExcel	<a href="#">bee-sting</a> , <a href="#">Chinmay Hegde</a> , <a href="#">GiantsLoveDeathMetal</a> , <a href="#">hackvan</a> , <a href="#">Majid</a> , <a href="#">talhasch</a> , <a href="#">user2314737</a> , <a href="#">Will</a>
46	Python	<a href="#">RamenChef</a> , <a href="#">user2728397</a>
47	Python	<a href="#">Gavin</a> , <a href="#">lorenzofeliz</a> , <a href="#">Pike D.</a> , <a href="#">Rednivrug</a>
48	Python	<a href="#">Nick Humrich</a>
49	PythonCurses	<a href="#">4444</a> , <a href="#">Guy</a> , <a href="#">kollery</a> , <a href="#">Vinzee</a>
50	Python	<a href="#">ADITYA</a> , <a href="#">Antonio</a> , <a href="#">Elodin</a> , <a href="#">Neil A.</a> , <a href="#">Vinzee</a>
51	Python	<a href="#">atayenel</a> , <a href="#">ChaoticTwist</a> , <a href="#">David</a> , <a href="#">GeekIhem</a> , <a href="#">mattgathu</a> , <a href="#">mnoronha</a> , <a href="#">thsecmaniac</a>
52	Python - virtualenv	<a href="#">Vikash Kumar Jain</a>
53	Python	<a href="#">Ken Y-N</a> , <a href="#">RandomHash</a>
54	setup.py	<a href="#">Adam Brenecki</a> , <a href="#">amblina</a> , <a href="#">JNat</a> , <a href="#">ravigadila</a> , <a href="#">strpeter</a> , <a href="#">user2027202827</a> , <a href="#">Y0da</a>
55	Sqlite3	<a href="#">Chinmay Hegde</a> , <a href="#">Simplans</a>
56	SYS	<a href="#">blubberdiblub</a>
57	tempfile NamedTemporaryFile	<a href="#">Alessandro Trinca Tornidor</a> , <a href="#">amblina</a> , <a href="#">Kevin Brown</a> , <a href="#">Stephen Leppik</a>
58	Tkinter	<a href="#">Dartmouth</a> , <a href="#">rlee827</a> , <a href="#">Thomas Gerot</a> , <a href="#">TidB</a>
59	Unicode	<a href="#">Claudiu</a> , <a href="#">KeyWeeUsr</a>
60	WebWSGI	<a href="#">David Heyman</a> , <a href="#">Kevin Brown</a> , <a href="#">Preston</a> , <a href="#">techydesigner</a>
61	Web	<a href="#">Thomas Gerot</a>
62	“with”Statement	<a href="#">Abhijeet Kasurde</a> , <a href="#">Alessandro Trinca Tornidor</a> , <a href="#">Andy Hayden</a> , <a href="#">Antoine Bolvy</a> , <a href="#">carrdelling</a> , <a href="#">Conrad.Dean</a> , <a href="#">Dartmouth</a> , <a href="#">David</a>



Marx, DeepSpace, Elazar, Kevin Brown, magu\_, Majid, Martijn Pieters, Matthew, nlsdfnbch, Pasha, Peter Brittain, petr, Shuo, Simplans, SuperBiasedMan, The\_Cthulhu\_Kid, Thomas Gerot, tyteen4a03, user312016, Valentin Lorentz, vaultah, λuser

63	Python 2Python 3	<p>671620616, Abhishek Kumar, Akshit Soota, Alex Gaynor, Allan Bureson, Alleo, Amarpreet Singh, Andy Hayden, Ani Menon, Antoine Bolvy, AntsySysHack, Antti Haapala, Antwan, arekolek, Ares, asmeurer, B8vrede, Bakuriu, Bharel, Bhargav Rao, bignose, bitchaser, Bluethon, Cache Staheli, Cameron Gagnon, Charles, Charlie H, Chris Sprague, Claudiu, Clayton Wahlstrom, CLDSEED, Colin Yang, Cometsong, Community, Conrad.Dean, danidee, Daniel Stradowski, Darth Shadow, Dartmouth, Dave J, David Cullen, David Heyman, deenes, DeepSpace, Delgan, DoHe, Duh-Wayne-101, Dunno, dwanderson, Ekeyme Mo, Elazar, enderland, enrico.bacis, erewok, ericdwang, ericmarkmartin, Ernir, ettanany, Everyone_Else, evuez, Franck Dernoncourt, Fred Barclay, garg10may, Gavin, geoffspear, ghostarbeiter, GoatsWearHats, H. Pauwelyn, Haohu Shen, holdenweb, iScrE4m, Iván C., J F, J. C. Leitão, James Elderfield, James Thiele, jarondl, jedwards, Jeffrey Lin, JGreenwell, Jim Fasarakis Hilliard, Jimmy Song, John Slegers, Jojodmo, jonrsharpe, Josh, Juan T, Justin, Justin M. Ucar, Kabie, kamalbanga, Karl Knechtel, Kevin Brown, King's jester, Kunal Marwaha, Lafexlos, lenz, linkdd, 'L'I, Mahdi, Martijn Pieters, Martin Thoma, masnun, Matt, Matt Dodge, Matt Rowland, Matthew Whitt, Max Feng, mgwilliams, Michael Recachinas, mkj, mnoronha, Moinuddin Quadri, muddfish, Nathaniel Ford, niemmi, niyasc, nouϣΑλζεΑΟ, OrangeTux, Pasha, Paul Weaver, Paulo Freitas, pcurry, pktangyue, poppie, pylang, python273, Pythonista, RahulHP, Rakitić, RamenChef, Rauf, René G, rfkortekaas, rrao, Ryan, sblair, Scott Mermelstein, Selcuk, Serenity, Seth M. Larson, ShadowRanger, Simplans, Slayther, solarc, sricharan, Steven Hewitt, sth, SuperBiasedMan, Tadhg McDonald-Jensen, techydesigner, Thomas Gerot, Tim, tobias_k, Tyler, tyteen4a03, user2314737, user312016, Valentin Lorentz, Veedrac, Ven, Vinayak, Vlad Shcherbina, VPfB, WeizhongTu, Wieland, wim, Wolf, Wombatz, xtreak, zarak, zcb, zopieux, zurfyx, zvezda</p>
64	intfloatstrtuple frozensets	Alessandro Trinca Tornidor, FazeL, Ganesh K, RamenChef, Stephen Leppik
65		Akshat Mahajan, Dair, Franck Dernoncourt, J F, Mahdi, nlsdfnbch, Ryan Smith, Vinzee, Xavier Combelle
66		Eleftheria, evuez, mnoronha

67		<a href="#">Jeremy</a> , <a href="#">Mohammed Salman</a>
68	“pip”PyPI	<a href="#">Zydnar</a>
69	AMQPStorm RabbitMQ	<a href="#">eandersson</a>
70	Pandas	<a href="#">Mark Miller</a>
71	PythonWindows	<a href="#">Simon Hibbs</a>
72	PythonRaspberry PI	<a href="#">dhimanta</a>
73	PythonWeb	<a href="#">alecxe</a> , <a href="#">Amitay Stern</a> , <a href="#">jmunsch</a> , <a href="#">mrtuovinen</a> , <a href="#">Ni.</a> , <a href="#">RamenChef</a> , <a href="#">Saiful Azad</a> , <a href="#">Saqib Shamsi</a> , <a href="#">Simplans</a> , <a href="#">Steven Maude</a> , <a href="#">sth</a> , <a href="#">sytech</a> , <a href="#">talhasch</a> , <a href="#">Thomas Gerot</a>
74	Python	<a href="#">Aquib Javed Khan</a> , <a href="#">Arun</a> , <a href="#">ChaoticTwist</a> , <a href="#">cledoux</a> , <a href="#">Ffisegydd</a> , <a href="#">ifma</a>
75	ZIP	<a href="#">Chinmay Hegde</a> , <a href="#">ghostarbeiter</a> , <a href="#">Jeffrey Lin</a> , <a href="#">SuperBiasedMan</a>
76		<a href="#">zenlc2000</a>
77		<a href="#">Adrian Antunez</a> , <a href="#">Alessandro Trinca Tornidor</a> , <a href="#">Alfe</a> , <a href="#">Andy</a> , <a href="#">Benjamin Hodgson</a> , <a href="#">Brian Rodriguez</a> , <a href="#">BusyAnt</a> , <a href="#">Claudiu</a> , <a href="#">driax</a> , <a href="#">Elazar</a> , <a href="#">flazzarini</a> , <a href="#">ghostarbeiter</a> , <a href="#">Ilia Barahovski</a> , <a href="#">J F</a> , <a href="#">Marco Pashkov</a> , <a href="#">muddyfish</a> , <a href="#">noʏʌɹʌzɛɹɔ</a> , <a href="#">Paul Weaver</a> , <a href="#">Rahul Nair</a> , <a href="#">RamenChef</a> , <a href="#">Shawn Mehan</a> , <a href="#">Shiven</a> , <a href="#">Shkelqim Memolla</a> , <a href="#">Simplans</a> , <a href="#">Slickytail</a> , <a href="#">Stephen Leppik</a> , <a href="#">Sudip Bhandari</a> , <a href="#">SuperBiasedMan</a> , <a href="#">user2314737</a>
78		<a href="#">Anthony Pham</a> , <a href="#">Antoine Bolvy</a> , <a href="#">BusyAnt</a> , <a href="#">Community</a> , <a href="#">Elazar</a> , <a href="#">James</a> , <a href="#">Jim Fasarakis Hilliard</a> , <a href="#">Joab Mendes</a> , <a href="#">Majid</a> , <a href="#">Md.Sifatul Islam</a> , <a href="#">Mechanic</a> , <a href="#">mezzode</a> , <a href="#">nlsdfnbch</a> , <a href="#">noʏʌɹʌzɛɹɔ</a> , <a href="#">Selcuk</a> , <a href="#">Simplans</a> , <a href="#">textshell</a> , <a href="#">tobias_k</a> , <a href="#">Tony Suffolk 66</a> , <a href="#">user2314737</a>
79		<a href="#">2Cubed</a> , <a href="#">Amir Rachum</a> , <a href="#">Antoine Pinsard</a> , <a href="#">Camsbury</a> , <a href="#">Community</a> , <a href="#">driax</a> , <a href="#">Igor Raush</a> , <a href="#">InitializeSahib</a> , <a href="#">Marco Pashkov</a> , <a href="#">Martijn Pieters</a> , <a href="#">Matthew Whitt</a> , <a href="#">OozeMeister</a> , <a href="#">Pasha</a> , <a href="#">Paulo Scardine</a> , <a href="#">RamenChef</a> , <a href="#">Rob Bednark</a> , <a href="#">Simplans</a> , <a href="#">sisanared</a> , <a href="#">zvone</a>
80		<a href="#">rassar</a>
81	CLI	<a href="#">Alessandro Trinca Tornidor</a> , <a href="#">anatoly techtonik</a> , <a href="#">Darth Shadow</a>
82		<a href="#">Anthony Pham</a> , <a href="#">intboolstring</a> , <a href="#">jtbandes</a> , <a href="#">Luke Taylor</a> , <a href="#">MSeifert</a> , <a href="#">Pasha</a> , <a href="#">supersam654</a>

83		<a href="#">Alessandro Trinca Tornidor</a> , <a href="#">JGreenwell</a> , <a href="#">metahost</a> , <a href="#">Pigman168</a> , <a href="#">RamenChef</a> , <a href="#">Stephen Leppik</a>
84		<a href="#">Greg</a> , <a href="#">JakeD</a>
85		<a href="#">3442</a> , <a href="#">4444</a> , <a href="#">acdr</a> , <a href="#">Ahsanul Haque</a> , <a href="#">Akshay Anand</a> , <a href="#">Akshit Soota</a> , <a href="#">Alleo</a> , <a href="#">Amir Rachum</a> , <a href="#">André Laszlo</a> , <a href="#">Andy Hayden</a> , <a href="#">Ankit Kumar Singh</a> , <a href="#">Antoine Bolvy</a> , <a href="#">APerson</a> , <a href="#">Ashwinee K Jha</a> , <a href="#">B8vrede</a> , <a href="#">bfontaine</a> , <a href="#">Brian Cline</a> , <a href="#">Brien</a> , <a href="#">Casebash</a> , <a href="#">Celeo</a> , <a href="#">cfi</a> , <a href="#">ChaoticTwist</a> , <a href="#">Charles</a> , <a href="#">Charlie H</a> , <a href="#">Chong Tang</a> , <a href="#">Community</a> , <a href="#">Conrad.Dean</a> , <a href="#">Dair</a> , <a href="#">Daniel Stradowski</a> , <a href="#">Darth Shadow</a> , <a href="#">Dartmouth</a> , <a href="#">David Heyman</a> , <a href="#">Delgan</a> , <a href="#">Dima Tisnek</a> , <a href="#">eenblam</a> , <a href="#">Elazar</a> , <a href="#">Emma</a> , <a href="#">enrico.bacis</a> , <a href="#">EOL</a> , <a href="#">ericdwang</a> , <a href="#">ericmarkmartin</a> , <a href="#">Esteis</a> , <a href="#">Faiz Halde</a> , <a href="#">Felk</a> , <a href="#">Fermi paradox</a> , <a href="#">Florian Bender</a> , <a href="#">Franck Dernoncourt</a> , <a href="#">Fred Barclay</a> , <a href="#">freidrichen</a> , <a href="#">G M</a> , <a href="#">Gal Dreiman</a> , <a href="#">garg10may</a> , <a href="#">ghostarbeiter</a> , <a href="#">GingerHead</a> , <a href="#">griswolf</a> , <a href="#">Hannele</a> , <a href="#">Harry</a> , <a href="#">Hurkyl</a> , <a href="#">IanAuld</a> , <a href="#">iankit</a> , <a href="#">Infinity</a> , <a href="#">intboolstring</a> , <a href="#">J F</a> , <a href="#">JOHN</a> , <a href="#">James</a> , <a href="#">JamesS</a> , <a href="#">Jamie Rees</a> , <a href="#">jedwards</a> , <a href="#">Jeff Langemeier</a> , <a href="#">JGreenwell</a> , <a href="#">JHS</a> , <a href="#">jjwatt</a> , <a href="#">JKillian</a> , <a href="#">JNat</a> , <a href="#">joel3000</a> , <a href="#">John Slegers</a> , <a href="#">Jon</a> , <a href="#">jonrsharpe</a> , <a href="#">Josh Caswell</a> , <a href="#">JRodDynamite</a> , <a href="#">Julian</a> , <a href="#">justhalf</a> , <a href="#">Kamyar Ghasemlou</a> , <a href="#">kdopen</a> , <a href="#">Kevin Brown</a> , <a href="#">KIDJourney</a> , <a href="#">Kwartz</a> , <a href="#">Lafexlos</a> , <a href="#">lapis</a> , <a href="#">Lee Netherton</a> , <a href="#">Liteye</a> , <a href="#">Locane</a> , <a href="#">Lyndsy Simon</a> , <a href="#">machine yearning</a> , <a href="#">Mahdi</a> , <a href="#">Marc</a> , <a href="#">Markus Meskanen</a> , <a href="#">Martijn Pieters</a> , <a href="#">Matt</a> , <a href="#">Matt Giltaji</a> , <a href="#">Matt S</a> , <a href="#">Matthew Whitt</a> , <a href="#">Maximillian Laumeister</a> , <a href="#">mbrig</a> , <a href="#">Mirec Miskuf</a> , <a href="#">Mitch Talmadge</a> , <a href="#">Morgan Thrapp</a> , <a href="#">MSeifert</a> , <a href="#">muddyfish</a> , <a href="#">n8henrie</a> , <a href="#">Nathan Arthur</a> , <a href="#">nehemiah</a> , <a href="#">nou̇lẏżæ̇Q</a> , <a href="#">Or East</a> , <a href="#">Ortomala Lokni</a> , <a href="#">pabouk</a> , <a href="#">Panda</a> , <a href="#">Pasha</a> , <a href="#">pktangyue</a> , <a href="#">Preston</a> , <a href="#">Pro Q</a> , <a href="#">pylang</a> , <a href="#">R Nar</a> , <a href="#">Rahul Nair</a> , <a href="#">rap-2-h</a> , <a href="#">Riccardo Petraglia</a> , <a href="#">rll</a> , <a href="#">Rob Fagen</a> , <a href="#">rao</a> , <a href="#">Ryan Hilbert</a> , <a href="#">Ryan Smith</a> , <a href="#">ryanyuyu</a> , <a href="#">Samuel McKay</a> , <a href="#">sarvajeetsuman</a> , <a href="#">Sayakiss</a> , <a href="#">Sebastian Kreft</a> , <a href="#">Shoe</a> , <a href="#">SHOWMEWHATYOUGOT</a> , <a href="#">Simplans</a> , <a href="#">Slayther</a> , <a href="#">Slickytail</a> , <a href="#">solidcell</a> , <a href="#">StuxCrystal</a> , <a href="#">sudo bangbang</a> , <a href="#">Sunny Patel</a> , <a href="#">SuperBiasedMan</a> , <a href="#">syb0rg</a> , <a href="#">Symmitchry</a> , <a href="#">The_Curry_Man</a> , <a href="#">theheadofabroom</a> , <a href="#">Thomas Gerot</a> , <a href="#">Tim McNamara</a> , <a href="#">Tom Barron</a> , <a href="#">user2314737</a> , <a href="#">user2357112</a> , <a href="#">Utsav T</a> , <a href="#">Valentin Lorentz</a> , <a href="#">Veedrac</a> , <a href="#">viveksyngh</a> , <a href="#">vog</a> , <a href="#">W.P. McNeill</a> , <a href="#">Will</a> , <a href="#">Will</a> , <a href="#">Wladimir Palant</a> , <a href="#">Wolf</a> , <a href="#">XCoder Real</a> , <a href="#">yurib</a> , <a href="#">Yury Fedorov</a> , <a href="#">Zags</a> , <a href="#">Zaz</a>
86		<a href="#">J F</a> , <a href="#">sth</a> , <a href="#">zmo</a>
87		<a href="#">J F</a> , <a href="#">keiv.fly</a> , <a href="#">SashaZd</a>
88	Python	<a href="#">Claudiu</a> , <a href="#">KeyWeeUsr</a> , <a href="#">Marco Pashkov</a> , <a href="#">pylang</a> , <a href="#">SuperBiasedMan</a> , <a href="#">Thtu</a>
89		<a href="#">Adriano</a> , <a href="#">Akshat Mahajan</a> , <a href="#">AlexV</a> , <a href="#">Andy</a> , <a href="#">Andy Hayden</a> , <a href="#">Anthony Pham</a> , <a href="#">Arkady</a> , <a href="#">B8vrede</a> , <a href="#">Benjamin Hodgson</a> , <a href="#">btel</a> ,

		<a href="#">CamelBackNotation</a> , <a href="#">Camsbury</a> , <a href="#">Chandan Purohit</a> , <a href="#">ChaoticTwist</a> , <a href="#">Charlie H</a> , <a href="#">Chris Larson</a> , <a href="#">Community</a> , <a href="#">D. Alveno</a> , <a href="#">danidee</a> , <a href="#">DawnPaladin</a> , <a href="#">Delgan</a> , <a href="#">duan</a> , <a href="#">duckman_1991</a> , <a href="#">elegent</a> , <a href="#">Elodin</a> , <a href="#">Emma</a> , <a href="#">EsmaeelE</a> , <a href="#">Ffisegydd</a> , <a href="#">Gal Dreiman</a> , <a href="#">ghostarbeiter</a> , <a href="#">Hurkyl</a> , <a href="#">J F</a> , <a href="#">James</a> , <a href="#">Jeffrey Lin</a> , <a href="#">JGreenwell</a> , <a href="#">Jim Fasarakis Hilliard</a> , <a href="#">jkitchen</a> , <a href="#">Jossie Calderon</a> , <a href="#">Justin</a> , <a href="#">Kevin Brown</a> , <a href="#">L3viathan</a> , <a href="#">Lee Netherton</a> , <a href="#">Martijn Pieters</a> , <a href="#">Martin Thureau</a> , <a href="#">Matt Giltaji</a> , <a href="#">Mike - SMT</a> , <a href="#">Mike Driscoll</a> , <a href="#">MSeifert</a> , <a href="#">muddyfish</a> , <a href="#">Murphy4</a> , <a href="#">nd.</a> , <a href="#">nou̇łłłzəɔ</a> , <a href="#">Pasha</a> , <a href="#">pylang</a> , <a href="#">pzp</a> , <a href="#">Rahul Nair</a> , <a href="#">Severiano Jaramillo Quintanar</a> , <a href="#">Simplans</a> , <a href="#">Slayther</a> , <a href="#">Steve Barnes</a> , <a href="#">Steven Maude</a> , <a href="#">SuperBiasedMan</a> , <a href="#">textshell</a> , <a href="#">thenOrTh</a> , <a href="#">Thomas Gerot</a> , <a href="#">user2314737</a> , <a href="#">user3333708</a> , <a href="#">user405</a> , <a href="#">Utsav T</a> , <a href="#">vaultah</a> , <a href="#">Veedrac</a> , <a href="#">Will</a> , <a href="#">Will</a> , <a href="#">zxxz</a> , <a href="#">λuser</a>
90		<a href="#">Aaron Christiansen</a> , <a href="#">David</a> , <a href="#">Elazar</a> , <a href="#">Peter Shinnars</a> , <a href="#">ppperry</a>
91		<a href="#">Anthony Pham</a> , <a href="#">davidism</a> , <a href="#">Elazar</a> , <a href="#">Esteis</a> , <a href="#">Mike Driscoll</a> , <a href="#">SuperBiasedMan</a> , <a href="#">user2314737</a> , <a href="#">zvone</a>
92		<a href="#">Adriano</a> , <a href="#">Alexander</a> , <a href="#">Anthony Pham</a> , <a href="#">Ares</a> , <a href="#">Barry</a> , <a href="#">blueenvelope</a> , <a href="#">Bosoneando</a> , <a href="#">BusyAnt</a> , <a href="#">Çağatay Uslu</a> , <a href="#">caped114</a> , <a href="#">Chandan Purohit</a> , <a href="#">ChaoticTwist</a> , <a href="#">cizixs</a> , <a href="#">Daniel Porteous</a> , <a href="#">Darth Kotik</a> , <a href="#">deeenes</a> , <a href="#">Delgan</a> , <a href="#">Elazar</a> , <a href="#">Ellis</a> , <a href="#">Emma</a> , <a href="#">evuez</a> , <a href="#">exhuma</a> , <a href="#">Ffisegydd</a> , <a href="#">Flickerlight</a> , <a href="#">Gal Dreiman</a> , <a href="#">ganesh gadila</a> , <a href="#">ghostarbeiter</a> , <a href="#">Igor Raush</a> , <a href="#">intboolstring</a> , <a href="#">J F</a> , <a href="#">j3485</a> , <a href="#">jalanb</a> , <a href="#">James</a> , <a href="#">James Elderfield</a> , <a href="#">jani</a> , <a href="#">jimsug</a> , <a href="#">jkdev</a> , <a href="#">JNat</a> , <a href="#">jonrsharpe</a> , <a href="#">KartikKannapur</a> , <a href="#">Kevin Brown</a> , <a href="#">Lafexlos</a> , <a href="#">LDP</a> , <a href="#">Leo Thumma</a> , <a href="#">Luke Taylor</a> , <a href="#">lukewrites</a> , <a href="#">Ixxer</a> , <a href="#">Majid</a> , <a href="#">Mechanic</a> , <a href="#">MrP01</a> , <a href="#">MSeifert</a> , <a href="#">muddyfish</a> , <a href="#">n12312</a> , <a href="#">nou̇łłłzəɔ</a> , <a href="#">Oz Bar-Shalom</a> , <a href="#">Pasha</a> , <a href="#">Pavan Nath</a> , <a href="#">poke</a> , <a href="#">RamenChef</a> , <a href="#">ravigadila</a> , <a href="#">ronrest</a> , <a href="#">Serenity</a> , <a href="#">Severiano Jaramillo Quintanar</a> , <a href="#">Shawn Mehan</a> , <a href="#">Simplans</a> , <a href="#">sirin</a> , <a href="#">solarc</a> , <a href="#">SuperBiasedMan</a> , <a href="#">textshell</a> , <a href="#">The_Cthulhu_Kid</a> , <a href="#">user2314737</a> , <a href="#">user6457549</a> , <a href="#">Utsav T</a> , <a href="#">Valentin Lorentz</a> , <a href="#">vaultah</a> , <a href="#">Will</a> , <a href="#">wythagoras</a> , <a href="#">Xavier Combelle</a>
93		<a href="#">Alireza Savand</a> , <a href="#">Ami Tavory</a> , <a href="#">antimatter15</a> , <a href="#">Arpit Solanki</a> , <a href="#">bijancn</a> , <a href="#">Claudiu</a> , <a href="#">Dartmouth</a> , <a href="#">engineercoding</a> , <a href="#">Ffisegydd</a> , <a href="#">J F</a> , <a href="#">JGreenwell</a> , <a href="#">jmunsch</a> , <a href="#">joel3000</a> , <a href="#">Kevin Brown</a> , <a href="#">Kinifwyne</a> , <a href="#">Mario Corchero</a> , <a href="#">Matt Giltaji</a> , <a href="#">Matthew Whitt</a> , <a href="#">mgilson</a> , <a href="#">muddyfish</a> , <a href="#">pylang</a> , <a href="#">strpeter</a>
94	GIL	<a href="#">Scott Mermelstein</a>
95		<a href="#">xiaoyi</a>
96	windows virtualenvwrapper	<a href="#">Sirajus Salayhin</a>

97		naren
98		APerson, cfi, Igor Raush, Jon Ericson, Karl Knechtel, Marco Pashkov, MSeifert, nouϭλϭzεϭϭ, Parousia, Simplans, SuperBiasedMan, tlama, user2314737
99		bogdanciobanu, Claudiu, Conrad.Dean, Elazar, Fazel, J F, James Elderfield, lukess, muddyfish, Sam Whited, SiggyF, Stephen Leppik, SuperBiasedMan, Xavier Combelle
100		Doraemon, GoatsWearHats, J F, JNat, Marco Pashkov, Mark Miller, Martijn Pieters, Nathaniel Ford, Nicolás, pcurry, pzp, SashaZd, SuperBiasedMan, Vilmar
101		ADITYA, boboquack, Chromium, cjds, depperm, Hannes Karppila, JGreenwell, Jonatan, kdopen, OliPro007, orvi, SashaZd, Shadowfa, textshell, Thomas Ahle, user2314737
102		Alon Alexander, Nander Speerstra, unutu, Vinzee, Will
103		Benedict Bunting, DeepSpace, depperm, Simplans, skrrgwasmе, Vinzee
104		boboquack, Buzz, rrao
105		Alu, CLDSEED, juggernaut, Kevin Brown, Kristof, mattgathu, Nabeel Ahmed, nlsdfnbch, Rahul, Rahul Nair, Riccardo Petraglia, Thomas Gerot, Will, Yogendra Sharma
106		David Cullen, Dev, MattCorr, nlsdfnbch, Rob H, StuxCrystal, textshell, Thomas Gerot, Will
107		Adam Matan, Andrew Schade, Brendan Abel, jfs, jmunsch, Riccardo Petraglia
108		Amir Rachum, Anthony Pham, APerson, ArtOfCode, BoppreH, Burhan Khalid, Chris Mueller, cizixs, depperm, Ffisegydd, Gareth Latty, Guy, helpful, iBelieve, Igor Raush, Infinity, James, JGreenwell, jonrsharpe, Karsten 7., kdopen, machine yearning, Majid, mattgathu, Mechanic, MSeifert, muddyfish, Nathan, nlsdfnbch, nouϭλϭzεϭϭ, ronrest, Roy Iacob, Shawn Mehan, Simplans, SuperBiasedMan, TehTris, Valentin Lorentz, viveksyngh, Xavier Combelle
109		Amitay Stern, Andy Hayden, Ares, Bhargav Rao, Brien, BusyAnt, Cache Staheli, caped114, ChaoticTwist, Charles, Dartmouth, David Heyman, depperm, Doug Henderson, Elazar, ganesh gadila, ghostarbeiter, GoatsWearHats, idjaw, Igor Raush, Ilia Barahovski, j__, Jim Fasarakis Hilliard, JL Peyret,

		Kevin Brown, krato, MarkyPython, Metasomatism, Mikail Land, MSeifert, mu , Nathaniel Ford, OliPro007, orvi, pzp, ronrest, Shrey Gupta, Simplans, SuperBiasedMan, theheadofabroom, user1349663, user2314737, Veedrac, WeizhongTu, wnmaw
110		4444, Aaron Christiansen, Adam_92, ADITYA, Akshit Soota, aldanor, alecxe, Alessandro Trinca Tornidor, Andy Hayden, Ani Menon, B8vrede, Bahrom, Bhargav, Charles, Chris, Darth Shadow, Dartmouth, Dave J, Delgan, dreftymac, evuez, Franck Dernoncourt, Gal Dreiman, gerrit, Giannis Spiliopoulos, GiantsLoveDeathMetal, goyalankit, Harrison, James Elderfield, Jean-Francois T., Jeffrey Lin, jetpack_guy, JL Peyret, joel3000, Jonatan, JRodDynamite, Justin, Kevin Brown, knight, krato, Marco Pashkov, Mark, Matt, Matt Giltaji, mu , MYGz, Nander Speerstra, Nathan Arthur, Nour Chawich, orion_tv, ragesz, SashaZd, Serenity, serv-inc, Simplans, Slayther, Sometowngeek, SuperBiasedMan, Thomas Gerot, tobias_k, Tony Suffolk 66, UloPe, user2314737, user312016, Vin, zondo
111		adeora, ArtOfCode, BSL-5, Kevin Brown, matsjoyce, SuperBiasedMan, Thomas Gerot, Wladimir Palant, wrrwr
112	/	Mohammad Julfikar
113		Dartmouth, J F, mattgathu, Nathan Osman, techydesigner, ygram
114	PythonSQL Server	metmirr
115		angussidney, Anthony Pham, Antonis Kalou, Brett Cannon, BusyAnt, Casebash, Christian Ternus, Community, Conrad.Dean, Daniel, Dartmouth, Esteis, Ffisegydd, FMc, Gerard Roche, Gideon Buckwalter, J F, JGreenwell, Kinifwyne, languitar, Lex Scarisbrick, Matt Giltaji, MSeifert, niyasc, nlsdfnbch, Paulo Freitas, pylang, Rahul Nair, Saiful Azad, Serenity, Simplans, StardustGogeta, StuxCrystal, SuperBiasedMan, techydesigner, the_cat_lady, Thomas Gerot, Tony Meyer, Tushortz, user2683246, Valentin Lorentz, Valor Naram, vaultah, wnmaw
116		Alessandro Trinca Tornidor, Darth Shadow, DhiaTN, J F, Jacques de Hooge, Leo, Martijn Pieters, mnoronha, Priya, RamenChef, Stephen Leppik
117		Elazar, SashaZd, SuperBiasedMan
118		boboquack, Brett Cannon, Dair, Ffisegydd, John Zwinck, Severiano Jaramillo Quintanar, Steven Maude

119		<a href="#">abukaj</a> , <a href="#">ADITYA</a> , <a href="#">Alec</a> , <a href="#">Alessandro Trinca Tornidor</a> , <a href="#">Alex</a> , <a href="#">Antoine Bolvy</a> , <a href="#">Baaing Cow</a> , <a href="#">Bhargav Rao</a> , <a href="#">Billy</a> , <a href="#">bixel</a> , <a href="#">Charles</a> , <a href="#">Cheney</a> , <a href="#">Christophe Roussy</a> , <a href="#">Dartmouth</a> , <a href="#">DeepSpace</a> , <a href="#">DhiaTN</a> , <a href="#">Dilettant</a> , <a href="#">fox</a> , <a href="#">Fred Barclay</a> , <a href="#">Gerard Roche</a> , <a href="#">greatwolf</a> , <a href="#">hiro protagonist</a> , <a href="#">Jeffrey Lin</a> , <a href="#">JGreenwell</a> , <a href="#">Jim Fasarakis Hilliard</a> , <a href="#">Lafexlos</a> , <a href="#">maazza</a> , <a href="#">Malt</a> , <a href="#">Mark</a> , <a href="#">matsjoyce</a> , <a href="#">Matt Dodge</a> , <a href="#">MervS</a> , <a href="#">MSeifert</a> , <a href="#">ncmathsadist</a> , <a href="#">omgimanerd</a> , <a href="#">Patrick Haugh</a> , <a href="#">pylang</a> , <a href="#">RamenChef</a> , <a href="#">Reut Sharabani</a> , <a href="#">Rob Bednark</a> , <a href="#">rrao</a> , <a href="#">SashaZd</a> , <a href="#">Shihab Shahriar</a> , <a href="#">Simplans</a> , <a href="#">SuperBiasedMan</a> , <a href="#">Tim D</a> , <a href="#">Tom Dunbavan</a> , <a href="#">tyteen4a03</a> , <a href="#">user2314737</a> , <a href="#">Will Vousden</a> , <a href="#">Wombatz</a>
120	CPython	<a href="#">Julij Jegorov</a>
121		<a href="#">davidism</a> , <a href="#">J F</a> , <a href="#">zmo</a> , <a href="#">Валерий Павлов</a>
122	CSV	<a href="#">Hridhhi Dey</a> , <a href="#">Thomas Crowley</a>
123		<a href="#">Adriano</a> , <a href="#">Alex L</a> , <a href="#">alfonso.kim</a> , <a href="#">Alleo</a> , <a href="#">Anthony Pham</a> , <a href="#">Antti Haapala</a> , <a href="#">Chris Hunt</a> , <a href="#">Christian Ternus</a> , <a href="#">Darth Kotik</a> , <a href="#">DeepSpace</a> , <a href="#">Delgan</a> , <a href="#">DhiaTN</a> , <a href="#">ebo</a> , <a href="#">Elazar</a> , <a href="#">Eric Finn</a> , <a href="#">Felix D.</a> , <a href="#">Ffisegydd</a> , <a href="#">Gal Dreiman</a> , <a href="#">Generic Snake</a> , <a href="#">ghostarbeiter</a> , <a href="#">GoatsWearHats</a> , <a href="#">Guy</a> , <a href="#">Inbar Rose</a> , <a href="#">intboolstring</a> , <a href="#">J F</a> , <a href="#">James</a> , <a href="#">Jeffrey Lin</a> , <a href="#">JGreenwell</a> , <a href="#">Jim Fasarakis Hilliard</a> , <a href="#">jrast</a> , <a href="#">Karl Knechtel</a> , <a href="#">machine yearning</a> , <a href="#">Mahdi</a> , <a href="#">manetsus</a> , <a href="#">Martijn Pieters</a> , <a href="#">Math</a> , <a href="#">Mathias711</a> , <a href="#">MSeifert</a> , <a href="#">pnhgiol</a> , <a href="#">rajah9</a> , <a href="#">Rishabh Gupta</a> , <a href="#">Ryan</a> , <a href="#">sarvajeetsuman</a> , <a href="#">sevenforce</a> , <a href="#">SiggyF</a> , <a href="#">Simplans</a> , <a href="#">skrrgwasm</a> , <a href="#">SuperBiasedMan</a> , <a href="#">textshell</a> , <a href="#">The_Curry_Man</a> , <a href="#">Thomas Gerot</a> , <a href="#">Tom</a> , <a href="#">Tony Suffolk 66</a> , <a href="#">user1349663</a> , <a href="#">user2314737</a> , <a href="#">Vinzee</a> , <a href="#">Will</a>
124		<a href="#">A. Ciclet</a> , <a href="#">RamenChef</a> , <a href="#">user2314737</a>
125		<a href="#">Benjamin Hodgson</a> , <a href="#">Elazar</a> , <a href="#">Faiz Halde</a> , <a href="#">J F</a> , <a href="#">Lee Netherton</a> , <a href="#">loading...</a> , <a href="#">Mister Mister</a>
126		<a href="#">Beall619</a> , <a href="#">Frustrated</a> , <a href="#">Justin</a> , <a href="#">Leon Z.</a> , <a href="#">lukewrites</a> , <a href="#">SuperBiasedMan</a> , <a href="#">Valentin Lorentz</a>
127	abc	<a href="#">Akshat Mahajan</a> , <a href="#">Alessandro Trinca Tornidor</a> , <a href="#">JGreenwell</a> , <a href="#">Kevin Brown</a> , <a href="#">Matthew Whitt</a> , <a href="#">mkrieger1</a> , <a href="#">SashaZd</a> , <a href="#">Stephen Leppik</a>
128		<a href="#">Teepeemm</a>
129		<a href="#">Abhishek Jain</a> , <a href="#">boboquack</a> , <a href="#">Charles</a> , <a href="#">Gal Dreiman</a> , <a href="#">intboolstring</a> , <a href="#">JakeD</a> , <a href="#">JNat</a> , <a href="#">Kevin Brown</a> , <a href="#">Matías Brignone</a> , <a href="#">nemesifixx</a> , <a href="#">poke</a> , <a href="#">R Colmenares</a> , <a href="#">Shawn Mehan</a> , <a href="#">Simplans</a> , <a href="#">Thomas Gerot</a> , <a href="#">tmr232</a> , <a href="#">Tony Suffolk 66</a> , <a href="#">viveksyngh</a>

130		<a href="#">Antti Haapala</a> , <a href="#">APerson</a> , <a href="#">GoatsWearHats</a> , <a href="#">Mirec Miskuf</a> , <a href="#">MSeifert</a> , <a href="#">RamenChef</a> , <a href="#">Simplans</a> , <a href="#">Valentin Lorentz</a>
131		<a href="#">bbayles</a> , <a href="#">cizixs</a> , <a href="#">Nemo</a> , <a href="#">pylang</a> , <a href="#">SuperBiasedMan</a>
132	/	<a href="#">naren</a>
133		<a href="#">2Cubed</a> , <a href="#">proprefenetre</a> , <a href="#">pylang</a> , <a href="#">rrao</a> , <a href="#">Simon Hibbs</a> , <a href="#">Simplans</a>
134		<a href="#">Dan Sanderson</a> , <a href="#">Igor Raush</a> , <a href="#">MSeifert</a>
135		<a href="#">MSeifert</a>
136	XML	<a href="#">4444</a> , <a href="#">Brad Larson</a> , <a href="#">Chinmay Hegde</a> , <a href="#">Francisco Guimaraes</a> , <a href="#">greuze</a> , <a href="#">heyhey2k</a> , <a href="#">Rob Murray</a>
137		<a href="#">Biswa_9937</a>
138		<a href="#">asmeurer</a> , <a href="#">Community</a> , <a href="#">Elazar</a> , <a href="#">jmunsch</a> , <a href="#">kon psych</a> , <a href="#">Marco Pashkov</a> , <a href="#">MSeifert</a> , <a href="#">RamenChef</a> , <a href="#">Shawn Mehan</a> , <a href="#">Simplans</a> , <a href="#">Steven Maude</a> , <a href="#">Symmitchry</a> , <a href="#">void</a> , <a href="#">XCoder Real</a>
139		<a href="#">Anthony Pham</a> , <a href="#">ArtOfCode</a> , <a href="#">asmeurer</a> , <a href="#">Christofer Ohlsson</a> , <a href="#">Ellis</a> , <a href="#">fredley</a> , <a href="#">ghostarbeiter</a> , <a href="#">Igor Raush</a> , <a href="#">intboolstring</a> , <a href="#">J F</a> , <a href="#">James Elderfield</a> , <a href="#">JGreenwell</a> , <a href="#">MSeifert</a> , <a href="#">niyasc</a> , <a href="#">RahulHP</a> , <a href="#">rajah9</a> , <a href="#">Simplans</a> , <a href="#">StardustGogeta</a> , <a href="#">SuperBiasedMan</a> , <a href="#">yurib</a>
140		<a href="#">Devesh Saini</a> , <a href="#">Infinity</a> , <a href="#">rfkortekaas</a>
141		<a href="#">Alessandro Trinca Tornidor</a> , <a href="#">Antonio</a> , <a href="#">bee-sting</a> , <a href="#">CLDSEED</a> , <a href="#">D. Alveno</a> , <a href="#">John Y</a> , <a href="#">LostAvatar</a> , <a href="#">mbsingh</a> , <a href="#">Michel Touw</a> , <a href="#">qwertyui9</a> , <a href="#">RamenChef</a> , <a href="#">rrawat</a> , <a href="#">Stephen Leppik</a> , <a href="#">Stephen Nyamweya</a> , <a href="#">sumitroy</a> , <a href="#">user2314737</a> , <a href="#">valeas</a> , <a href="#">zweiterlinde</a>
142		<a href="#">Andy Hayden</a> , <a href="#">MSeifert</a> , <a href="#">Peter Mølgaard Pallesen</a> , <a href="#">pylang</a>
143		<a href="#">Andy</a> , <a href="#">Pavan Nath</a> , <a href="#">RamenChef</a> , <a href="#">Vin</a>
144	I / O.	<a href="#">Ajean</a> , <a href="#">Anthony Pham</a> , <a href="#">avb</a> , <a href="#">Benjamin Hodgson</a> , <a href="#">Bharel</a> , <a href="#">Charles</a> , <a href="#">crhodes</a> , <a href="#">David Cullen</a> , <a href="#">Dov</a> , <a href="#">Esteis</a> , <a href="#">ilse2005</a> , <a href="#">isvforall</a> , <a href="#">jfsturtz</a> , <a href="#">Justin</a> , <a href="#">Kevin Brown</a> , <a href="#">mattgathu</a> , <a href="#">MSeifert</a> , <a href="#">nlsdfnbch</a> , <a href="#">Ozair Kafray</a> , <a href="#">PYPL</a> , <a href="#">pzp</a> , <a href="#">RamenChef</a> , <a href="#">Ronen Ness</a> , <a href="#">rrao</a> , <a href="#">Serenity</a> , <a href="#">Simplans</a> , <a href="#">SuperBiasedMan</a> , <a href="#">Tasdik Rahman</a> , <a href="#">Thomas Gerot</a> , <a href="#">Umibozu</a> , <a href="#">user2314737</a> , <a href="#">Will</a> , <a href="#">WombatPM</a> , <a href="#">xgord</a>
145		<a href="#">DeepSpace</a> , <a href="#">James</a>
146		<a href="#">Ajean</a> , <a href="#">alecxe</a> , <a href="#">Andy</a> , <a href="#">Antti Haapala</a> , <a href="#">BusyAnt</a> , <a href="#">Conrad.Dean</a> , <a href="#">Elazar</a> , <a href="#">ghostarbeiter</a> , <a href="#">J F</a> , <a href="#">Jeffrey Lin</a> , <a href="#">jonnsharpe</a> , <a href="#">Kevin Brown</a>



		, Nicole White, nlsdfnbch, Ohad Eytan, Paul, paulmorriss, proprius, RahulHP, RamenChef, sagism, Simplans, Sirajus Salayhin, Suku, Will
147		surfthecity
148		Razik
149		Andy, Elazar, evuez, Martijn Pieters, techydesigner
150		Andy Hayden, BusyAnt, Chris Larson, deepakkt, Delgan, Elazar, evuez, Ffisegydd, Geeklhem, Hannes Karppila, James, Kevin Brown, krato, Max Feng, nouλΑλζεϞ, rajah9, rrao, SashaZd, Simplans, Slayther, Soumendra Kumar Sahoo, Thomas Gerot, Trimax, Valentin Lorentz, Vinzee, wwii, xgord, Zack
151		DeepSpace, Simplans, tjohnson
152		Esteis, Marlon Abeykoon, mnoronha, PYPL
153		Aidan, alejosocorro, andandandand, Andy Hayden, ashes999, B8vrede, Claudiu, Darth Shadow, driax, Fermi paradox, ganesh gadila, goodmami, Jan, Jeffrey Lin, jonrsharpe, Julien Spronck, Kevin Brown, Md.Sifatul Islam, Michael M., mnoronha, Nander Speerstra, nrusch, Or East, orvi, regnarg, sarvajeetsuman, Simplans, SN Ravichandran KR, SuperBiasedMan, user2314737, zondo
154		Anthony Pham, Ares, Elazar, J F, MSeifert, Shawn Mehan, SuperBiasedMan, Will, Xavier Combelle
155		Doc, Rahul Nair, SashaZd
156		Stephen Leppik, Thomas Gerot
157	`exec`eval`	Antti Haapala, Ilja Everilä
158	Matplotlib	Arun, user2314737
159		Alessandro Trinca Tornidor, Beall619, mnoronha, RamenChef, Stephen Leppik, Sun Qingyao
160		2Cubed, Ahsanul Haque, Akshat Mahajan, Andy Hayden, Arthur Dent, ArtOfCode, Augustin, Barry, Chankey Pathak, Claudiu, CodenameLambda, Community, deenes, Delgan, Devesh Saini, Elazar, ericmarkmartin, Ernir, ForceBru, Igor Raush, Ilia Barahovski, JOHN, jackskis, Jim Fasarakis Hilliard, Juan T, Julius Bullinger, Karl Knechtel, Kevin Brown, Kronen, Luc M, Lyndsy Simon, machine yearning, Martijn Pieters, Matt

		Giltaji, max, MSeifert, nlsdfnbch, Pasha, Pedro, PsyKzz, pzp, satsumas, sevenforce, Signal, Simplans, Slayther, StuxCrystal, tversteeg, Valentin Lorentz, Will, William Merrill, xtreak, Zaid Ajaj, zarak, λuser
161	urllib	Amitay Stern, ravigadila, sth, Will
162	WebSockets	2Cubed, Stephen Leppik, Tyler Gubala
163		amin, blueenvelope, Bryce Frank, Camsbury, David, DeepSpace, Elazar, J F, James, JGreenwell, Jon Ericson, Kevin Brown, Lafexlos, matsjoyce, Mechanic, Milo P, MSeifert, numbermaniac, sarvajeetsuman, Simplans, techydesigner, Tony Suffolk 66, Undo, user2314737, wythagoras, Zenadix
164		Alleo, amblina, Antoine Bolvy, Bonifacio2, Ffisegydd, Guy, Igor Rausch, Jonatan, Martec, MSeifert, MUSR, pzp, RahulHP, Reut Sharabani, SashaZd, Sayed M Ahamad, SuperBiasedMan, theheadofabroom, user2314737, yurib
165		Andrzej Pronobis, Andy Hayden, Bahrom, Cimbali, Cody Piersall, Conrad.Dean, Elazar, evuez, J F, James, Or East, pylang, RahulHP, RamenChef, Simplans, user2314737
166		wim
167		Alessandro Trinca Tornidor, depperm, J F, JGreenwell, Matt Giltaji, Pasha, RamenChef, Stephen Leppik
168		Juan T, TemporalWolf
169		Adrian17, Artem Kolontay, ArtOfCode, Bhargav, brennan, Dair, Daniil Ryzhkov, Darkade, Darth Shadow, edwinksl, Fernando, ghostarbeiter, ha_1694, Hans Then, lancnorden, J F, Majid, Marco Pashkov, Matt Giltaji, Matthew Whitt, nehemiah, Nuhil Mehdy, Ortomala Lokni, Preston, pylang, qwertyuip9, RamenChef, Régis B., Sebastian Schrader, Serenity, Shantanu Alshi, Shrey Gupta, Simon Fraser, Simplans, wrwrwr , ychaouche, zopieux, zvezda
170	virtualenvwrapper	Sirajus Salayhin
171		Alessandro Trinca Tornidor, ChaoticTwist, Community, Dair, doratheexplorer0911, Emolga, greut, iankit, JGreenwell, jonrsharpe, kefkus, Kevin Brown, Matthew Whitt, MSeifert, muddyfish, Mukunda Modell, Nearoo, Nemo, Nuno André, Pasha, Rob Bednark, seenu s, Shreyash S Sarnayak, Simplans, StuxCrystal, Suhas K, technusm1, Thomas Gerot, tyteen4a03, Wladimir Palant, zvone

172		<a href="#">hashcode55</a> , <a href="#">StuxCrystal</a>
173		<a href="#">Adeel Ansari</a> , <a href="#">Bosoneando</a> , <a href="#">bpachev</a>
174		<a href="#">andrew</a>
175		<a href="#">amblina</a> , <a href="#">Braiam</a> , <a href="#">Claudiu</a> , <a href="#">cledoux</a> , <a href="#">Elazar</a> , <a href="#">Gerard Roche</a> , <a href="#">krato</a> , <a href="#">loading...</a> , <a href="#">Marco Pashkov</a> , <a href="#">Or Duan</a> , <a href="#">Pasha</a> , <a href="#">RamenChef</a> , <a href="#">rfkortekaas</a> , <a href="#">Simplans</a> , <a href="#">Thomas Gerot</a> , <a href="#">Topperfalkon</a> , <a href="#">zmo</a> , <a href="#">zondo</a>
176		<a href="#">Gal Dreiman</a> , <a href="#">Jörn Hees</a> , <a href="#">sxnwlfkk</a>
177	Python	<a href="#">muddyfish</a> , <a href="#">StuxCrystal</a> , <a href="#">user2314737</a>
178		<a href="#">Charul</a> , <a href="#">denvaar</a> , <a href="#">djaszczurowski</a>
179		<a href="#">Ani Menon</a> , <a href="#">FunkySayu</a> , <a href="#">MattCorr</a> , <a href="#">SuperBiasedMan</a> , <a href="#">TuringTux</a>
180	PythonJavaScript	<a href="#">user2683246</a>
181		<a href="#">Will</a> , <a href="#">XonAether</a>
182		<a href="#">Aldo</a> , <a href="#">B8vrede</a> , <a href="#">joel3000</a> , <a href="#">Sardathrion</a> , <a href="#">Sardorbek Imomaliev</a> , <a href="#">Vlad Bezden</a>
183	CSV	<a href="#">Adam Matan</a> , <a href="#">Franck Deroncourt</a> , <a href="#">Martin Valgur</a> , <a href="#">mnoronha</a> , <a href="#">ravigadila</a> , <a href="#">Setu</a>
184		<a href="#">Andy Hayden</a> , <a href="#">Darth Shadow</a> , <a href="#">ericmarkmartin</a> , <a href="#">Ffisegydd</a> , <a href="#">Igor Raush</a> , <a href="#">Jonas S</a> , <a href="#">jonrsharpe</a> , <a href="#">L3viathan</a> , <a href="#">Majid</a> , <a href="#">RamenChef</a> , <a href="#">Simplans</a> , <a href="#">Valentin Lorentz</a>
185		<a href="#">alecxe</a> , <a href="#">Anonymous</a> , <a href="#">Antti Haapala</a> , <a href="#">Elazar</a> , <a href="#">Jim Fasarakis Hilliard</a> , <a href="#">Jonatan</a> , <a href="#">RamenChef</a> , <a href="#">Seth M. Larson</a> , <a href="#">Simplans</a> , <a href="#">Stephen Leppik</a>
186	...	<a href="#">Parousia</a> , <a href="#">Thomas Gerot</a>
187		<a href="#">Anaphory</a>
188		<a href="#">Claudiu</a> , <a href="#">Thomas Gerot</a>
189		<a href="#">HoverHell</a> , <a href="#">JGreenwell</a> , <a href="#">MathSquared</a> , <a href="#">SashaZd</a> , <a href="#">Shreyash S Sarnayak</a>
190		<a href="#">APerson</a> , <a href="#">cfi</a> , <a href="#">J Atkin</a> , <a href="#">MSeifert</a> , <a href="#">rajah9</a> , <a href="#">SuperBiasedMan</a>
191		<a href="#">Bastian</a> , <a href="#">japborst</a> , <a href="#">JGreenwell</a> , <a href="#">Jossie Calderon</a> , <a href="#">mbomb007</a> , <a href="#">SashaZd</a> , <a href="#">Tyler Crompton</a>

192		FrankBr
193		Gal Dreiman, lancnorden, Wayne Werner
194		Nemo
195		orvi
196	GZip	orvi
197		APerson, Igor Raush, Martijn Pieters, MSeifert
198		Prem Narain
199		Alex Gaynor, Andrzej Pronobis, Anthony Pham, Community, David Robinson, Delgan, giucal, Jim Fasarakis Hilliard, michaelrbock, MSeifert, Nobilis, ppperry, RamenChef, Simplans, SuperBiasedMan
200		Aaron Hall, Akshat Mahajan, Anthony Pham, Antti Haapala, Byte Commander, dermen, Elazar, Ellis, ericmarkmartin, Fermi paradox, Ffisegydd, japborst, Jim Fasarakis Hilliard, jonrsharp, Justin, kramer65, Lafexlos, LDP, Morgan Thrapp, muddyfish, nico, OrangeTux, pcurry, Pythonista, Selcuk, Serenity, Tejas Jadhav, tobias_k, Vlad Shcherbina, Will
201	Python	Jacques de Hooge, Squidward
202		blueberryfields, Comrade SparklePony, frankyjuang, jmunsch, orvi, qwertyuip9, Stephen Leppik, Thomas Gerot
203		Aaron Hall, Ahsanul Haque, Akshat Mahajan, Andrzej Pronobis, Anthony Pham, Avantol13, Camsbury, cfi, Community, Conrad.Dean, Daksh Gupta, Darth Shadow, Dartmouth, depperm, Elazar, Ffisegydd, Haris, Igor Raush, InitializeSahib, J F, jkdev, jlarsch, John Militer, Jonas S, Jonathan, Kallz, KartikKannapur, Kevin Brown, Kinifwyne, Leo, Liteye, Imiguelvargasf, Mailerdaemon, Martijn Pieters, Massimiliano Kraus, Mattew Whitt, MrP01, Nathan Arthur, ojas mohril, Pasha, Peter Steele, pistache, Preston, pylang, Richard Fitzhugh, rohittk239, Rushy Panchal, Sempoo, Simplans, Soumendra Kumar Sahoo, SuperBiasedMan, techydesigner, then0rTh, Thomas Gerot, Tony Suffolk 66, tox123, UltraBob, user2314737, wrwrwr, Yogendra Sharma
204	__str__repr__	Alessandro Trinca Tornidor, jedwards, JelmerS, RamenChef, Stephen Leppik
205		Luca Van Oort, Stephen Leppik