무료 전자 책

배우기 Python Language

Free unaffiliated eBook created from
Stack Overflow contributors.

#python

1
1: Python 2
2
2
Python 3.x 2
Python 2.x 3
Examples 3
3
3
, IDLE Python 4
Hello World Python 4
Python 5
6
6
6
6
10
IDLE - Python GUI 11
11
12
12
12
12
13
13
13
14
14
15
15

16

20

20

21

24

25

- str () repr () 26

repr () 27

str () 27

pip 27

/ 28

28

28

Python 2.7.x 3.x 29

2: "pip" : PyPI Package Manager 32

32

32

Examples 32

32

ImportError 33

33

3: * args ** kwargs 34

34

h11 34

h12 34

h13 34

Examples 35

* args 35

** kwargs 35

* args 36

** kwargs 36

* args 36
..... 37
kwarg 37
** kwargs 37
4: __name__ 39
..... 39
..... 39
Examples 39
__name__ == '__main__' 39
1 39
2 39
function_class_or_module .__ name__ 39
..... 41
5: `exec``eval` 42
..... 42
..... 42
..... 42
Examples 42
exec 42
eval 42
..... 42
eval 43
ast.literal_eval 43
exec, eval ast.literal_eval 43
6: 2to3 45
..... 45
..... 45
..... 45
Examples 45
..... 45
..... 46
Windows 46

46

Windows 46

7: AMQPStorm RabbitMQ 47

47

Examples 47

RabbitMQ 47

RabbitMQ 48

RabbitMQ 48

8: ArcPy 51

51

Examples 51

51

createDissolvedGDB gdb 51

9: Asyncio 52

Examples 52

52

53

UVLoop 54

: 54

54

54

55

asyncio 55

10: base64 57

57

57

57

58

Examples 59

Base64 59

Base32 60

Base16 60
ASCII85 61
Base85 61
11: C # 63
..... 63
..... 63
Examples 64
C # 64
C # 64
12: ChemPy - 66
..... 66
Examples 66
..... 66
..... 66
..... 66
..... 66
..... 67
( ) 67
13: configparser 69
..... 69
..... 69
..... 69
Examples 69
..... 69
..... 69
14: CSV 71
Examples 71
TSV 71
..... 71
..... 71
..... 71

15: Deque 72
..... 72
..... 72
..... 72
Examples 72
deque 72
deque 72
..... 72
..... 73
16: dis 75
Examples 75
dis 75
? 75
..... 75
17: Functools 77
Examples 77
..... 77
total_ordering 77
..... 77
lru_cache 78
cmp_to_key 78
18: groupby () 79
..... 79
..... 79
..... 79
..... 79
Examples 79
1 79
2 80
3 80
4 81
19: GZip 83

83

Examples 83

GNU zip 83

20: HTML 84

Examples 84

BeautifulSoup 84

BeautifulSoup CSS 84

PyQuery 85

21: Itertools 86

86

Examples 86

86

86

itertools.product 87

itertools.count 88

itertools.takewhile 89

itertools.dropwhile 90

90

Itertools 91

91

itertools.repeat 92

92

92

itertools.permutations 92

22: JSON 94

94

94

94

:94

:94

() 94

:94

:95
() :95
Examples95
Python dict JSON 96
JSON Python dict 96
96
96
````,````96
`json.tool` JSON 97
JSON 98
98
98
98
JSON 99
23: kivy - NUI 100
100
Examples100
100
24: Matplotlib 102
102
Examples102
Matplotlib 102
: , , , 102
MATLAB 104
105
X Y : twinx () 106
twiny () Y X 107
25: os.path110
110
110
Examples110
110
````

110
110
111
111
, , , 111
26: pip : PyPI 112
112
112
112
Examples 112
112
112
113
`pip` 113
113
Linux 113
Windows 114
requirements.txt 114
virtualenv requirements.txt 114
pip Python 114
115
Pre-Release 115
115
27: PostgreSQL 118
Examples 118
118
118
118
28: py.test 119
Examples 119
py.test 119

119

119

119

119

120

py.test ! 121

122

29: Py2Neo Neo4j Cypher 124

Examples 124

124

Neo4j 124

Neo4j 124

1 : 124

2 : 125

Cypher 125

30: pyautogui 126

126

Examples 126

126

126

126

31: PyInstaller - 127

127

127

Examples 127

127

Pyinstaller 127

128

: 128

128

128

32: Python 2 Python 3 129

129

129

Examples 129

129

: 130

132

Reduce 134

xrange 134

135

136

138

.next () 139

140

141

141

exec Python 3 142

2 hasattr 142

143

143

8 143

3 " " 143

<> !`,! = repr () 144

16 / 145

Python 3 cmp 146

146

() 147

filter (), map () zip () 148

/ 148

149

I / O 150

round () 150

() 150

round () 151

, ... 151
..... 152
long int 152
..... 153
33: Python Lex-Yacc 154
..... 154
..... 154
Examples 154
PLY 154
", !" PLY - 154
1 : Lex 156
..... 157
h21 158
h22 158
h23 158
h24 158
h25 158
h26 158
h27 159
h28 159
h29 159
h210 159
2 : Yacc 159
..... 160
h211 161
34: Python 163
Examples 163
..... 163
..... 163
Deque 164
..... 164
... . 165

35: Python Raspberry PI IoT 166
Examples 166
- 166
36: Python SQL Server 169
Examples 169
, , ... 169
37: Python Windows 170
..... 170
Examples 170
Python 170
Flask 171
38: setup.py 172
..... 172
..... 172
Examples 172
setup.py 172
..... 172
setup.py 173
..... 173
39: Sqlite3 175
Examples 175
Sqlite3 - 175
..... 175
40: sys 177
..... 177
..... 177
..... 177
Examples 177
..... 177
..... 177
..... 177
..... 177

41: tempfile NamedTemporaryFile 179
..... 179
Examples 179
(a) 179
42: urllib 180
Examples 180
HTTP GET 180
2 180
3 180
HTTP POST 180
2 181
3 181
..... 181
43: virtualenvwrapper 182
..... 182
Examples 182
virtualenvwrapper 182
44: Windows virtualenvwrapper 184
Examples 184
Windows virtualenvwrapper 184
45: WSGI ( ) 185
..... 185
Examples 185
() 185
46: XML 187
..... 187
Examples 187
ElementTree 187
XML 187
XML 188
iterparse ( ) XML 188

XPath XML 189
47: ZIP 191
..... 191
..... 191
Examples 191
Zip 191
Zipfile 191
zip 192
..... 192
48: 193
..... 193
..... 193
Examples 193
..... 193
virtualenv 193
..... 193
..... 193
..... 194
..... 194
..... 194
..... 194
..... 194
..... 195
virtualenvwrapper 196
..... 196
..... 196
..... 196
..... 197
Unix / Linux 197
Fishen virtualenv 197
Anaconda 198
.....

198

198

198

198

198

49: 200

200

200

200

Examples 200

200

201

201

201

203

203

204

204

205

205

50: 207

Examples 207

(Turtle Graphics) 207

51: 208

208

Examples 208

os.access 208

52: 209

Examples 209

iterable : collections.Counter 209

(-s) : collections.Counter.most_common () 209

: list.count () tuple.count () 209

: str.count () 210
numpy 210
53: 212
212
Examples 212
PyTesseract 212
PyOCR 212
54: 214
214
Examples 214
argparse Hello world 214
docopt 214
argparse 215
argv 215
argparse 216
argparse.add_argument_group () 217
docopt docopt_dispatch 218
55: , : Python JavaScript 219
219
Examples 219
'in' 219
56: 220
220
Examples 220
PyDotPlus 220
220
PyGraphviz 220
57: (GIL) 222
222
GIL ? 222
GIL : 222

GIL 222
GIL 222
: 222
Examples 223
.Pool 223
GIL David Beazley 223
Cython nogil : 224
GIL David Beazley 224
nogil (CYTHON) : 224
58: 225
..... 225
..... 225
..... 225
..... 225
..... 226
Examples 226
..... 226
..... 227
..... 228
..... 228
..... 229
..... 229
..... 229
: 229
..... 230
..... 230
..... 231
..... 231
..... 231
..... 231
..... 231

232
(/) 232
234
235
236
236
237
237
239
239
240
59: 241
Examples 241
input () raw_input () 241
241
241
242
stdin 243
243
60: 246
Examples 246
246
246
61: 248
248
Examples 248
: if / else 248
248
249
250
62: 251
251

251

Examples....251

tkinter....251

....252

....252

....252

....252

63: 254

Examples....254

....254

datetime 254

datetime 254

64: 255

....255

Examples....255

datetime 255

....255

datetime 255

....256

datetime 257

datetimes 257

Fuzzy datetime (datetime)....259

....259

ISO 8601 259

datetime 260

....260

....261

ISO 8601 261

....261

, ...262

....262

65: 263

Examples ... 263
... 263
... 263
66: ... 265
Examples ... 265
... 265
... 265
67: ... 267
Examples ... 267
... 267
... 269
68: ... 270
... 270
Examples ... 270
... 270
unittest.mock.create_autospec ... 270
unittest ... 271
... 272
Unittest ... 273
pytest ... 274
69: ... 277
... 277
Examples ... 277
... 277
70: ... 278
Examples ... 278
... 278
... 278
... 278
... 278
... 278
71: ... 280

280
280
280
Examples 280
JSON 280
281
72: 282
282
Examples 282
MySQLdb MySQL 282
SQLite 283
SQLite : 283
283
h212 284
Connection 284
Cursor 285
SQLite Python 288
psycopg2 PostgreSQL 288
288
: 288
: 289
289
290
sqlalchemy 291
73: 293
Examples 293
Python : _pdb_ 293
IPython ipdb 294
295
74: 296
296
Examples 296

296
297
298
75: 301
301
Examples 301
locale 301
76: 302
302
302
302
Examples 302
302
For 303
303
303
break 303
continue 304
305
return break 305
"else" 305
? 307
307
While 309
309
309
310
310
" "do-while 311
311
77: ( ) 313

Examples....313
....313
....313
....313
....313
....314
....314
....314
....315
....316
78: 317
Examples....317
....317
79: 318
....318
....318
....318
Examples....318
map, itertools.imap future_builtins.map 318
iterable 319
iterables 320
: "None" (2.x)....321
....322
80: 324
....324
Examples....324
....324
....325
....325
....326
()....326
while 328

81: 329
Examples 329
..... 329
82: 330
..... 330
..... 330
Examples 330
..... 330
..... 331
..... 331
..... 331
Python 2 3 six 331
..... 332
..... 332
? 332
..... 332
Metaclass 333
..... 333
83: 335
..... 335
..... 335
..... 335
Examples 335
..... 335
..... 336
..... 341
..... 341
..... 342
..... 342
..... 343
..... 343
..... 344

344
345
346
346
84: 348
348
348
Examples 348
348
349
350
__all__ 350
351
352
PEP8 352
352
__import__() 352
352
2 353
3 353
85: 354
354
Examples 354
354
354
86: 356
356
356
356
Examples 356
356
357

357
357
358
358
360
361
362
362
363
364
364
87: 366
366
366
366
Examples 366
366
367
368
- 369
369
370
88: 371
Examples 371
371
89: ( ) 372
372
372
Examples 372
"" 372
372
372

373
90: 374
374
Examples 374
: , 374
() 374
() 374
() 374
: randint, randrange, random uniform 375
randint () 375
randrange () 375
376
376
: 376
377
378
378
91: CSV 379
379
379
379
Examples 379
379
CSV 380
92: 381
381
382
Examples 382
382
str.casefold() 382
str.upper() 382

str.lower() 382
str.capitalize() 382
str.title() 383
str.swapcase() 383
str 383
..... 383
str.split(sep=None, maxsplit=-1) 383
str.rsplit(sep=None, maxsplit=-1) 384
..... 384
str.replace(old, new[, count]) : 384
str.format f-strings : 385
..... 386
str.count(sub[, start[, end]]) 386
..... 386
str.startswith(prefix[, start[, end]]) 386
str.endswith(prefix[, start[, end]]) 387
..... 387
str.isalpha 388
str.isupper , str.islower , str.istitle 388
str.isdecimal , str.isdigit , str.isnumeric 388
str.isalnum 389
str.isspace 389
str.translate : 390
/ 390
str.strip([chars]) 390
str.rstrip([chars]) str.lstrip([chars]) 391
..... 391
..... 392
..... 392
string.ascii_letters : 392
string.ascii_lowercase : 392
string.ascii_uppercase : 393
string.digits : 393

string.hexdigits : 393
string.octaldigits : 393
string.punctuation : 393
string.whitespace : 393
string.printable : 394
394
394
str bytes 395
395
93: 397
397
397
397
Examples 397
397
398
(f-string) 399
datetime 400
Getitem Getattr 400
400
401
402
402
403
403
(Python 2.x) 403
(Python 3.2) 404
: 404
94: 405
405
405
Examples 405

405

Mixins 406

95: **407**

Examples 407

..... 407

..... 408

..... 408

iterable 408

..... 409

! 409

96: **410**

..... 410

..... 410

Examples 410

..... 410

next () 410

..... 410

..... 411

..... 411

..... 413

..... 413

- **414**

..... 414

..... 415

: 415

..... 416

..... 416

..... 417

97: **418**

..... 418

..... 418

Examples 418

418
419
append () 419
insert () 419
extend () 419
fromlist () 420
remove () 420
pop () 420
index () 420
reverse () 421
buffer_info () 421
count () 421
tostring () 421
tolist () 421
fromstring () char 421
98: 423
Examples 423
423
424
99: 427
Examples 427
427
JPEG 427
100: (int, float, str, tuple frozensets) 428
Examples 428
428
428
Frozenset 428
101: 429
429
Examples 429
429

430

430

431

431

del 432

del v 432

del v.name 432

del v[item] 432

del v[a:b] 433

433

? 433

? 433

434

global nonlocal (Python 3) 435

102: 436

436

Examples 436

436

436

C- 437

PyPar 437

103: 438

438

438

438

Examples 438

438

438

438

439

pycrypto 439

pycrypto RSA 440
pycrypto RSA 441
104: 442
..... 442
Examples 442
..... 442
..... 443
105: 444
..... 444
..... 444
..... 444
..... 444
Examples 444
..... 444
106: 446
Examples 446
..... 446
..... 446
..... 447
..... 447
`and``or` boolean 447
..... 448
107: 449
Examples 449
py2app 449
cx_Freeze 450
108: Python 452
Examples 452
IronPython 452
..... 452
..... 452

452
452
453
453
453
HTML 453
JavaScript DOM 453
JavaScript 454
454
455
109: 456
456
456
Examples 456
456
457
457
457
457
458
`is 'vs`==` 458
459
Common Gotcha : 460
110: 461
461
461
Examples 461
AND 461
OR 461
XOR ( ) 462
462

462
NOT 463
464
111: 466
Examples 466
466
467
112: / 468
468
Examples 468
468
468
113: 469
469
469
469
Examples 469
469
dict () 469
KeyError 469
470
471
471
471
471
: dict() 471
dict 472
472
472
** 473
473

Python 3.5 473
Python 3.3 474
Python 2.x, 3.x 474
..... 474
..... 474
..... 475
..... 475
..... 476
114: 478
..... 478
..... 478
: 478
..... 478
Examples 478
..... 478
(,) : 478
..... 479
..... 479
115: 481
..... 481
Examples 481
..... 481
116: 483
..... 483
..... 483
Examples 483
..... 483
..... 483
..... 484
..... 485
..... 485

485

485

485

486

486

486

487

487

117: 488

488

488

Examples 488

UDP 488

UDP 488

TCP 489

TCP 489

Linux 490

118: 492

492

Examples 492

@property 492

/ @property 492

getter, setter deleter 493

493

119: 496

496

Examples 496

496

, 496

120: 498

498

Examples....498
: str.index (), str.rindex () str.find (), str.rfind ()....498
....498
....498
....498
....498
....499
Dict....499
: list.index (), tuple.index ()....499
dict 499
: bisect.bisect_left ()....500
....501
: __contains__ __iter__....501
121: **503**
....503
Examples....503
....503
....**503**
....504
, ...505
....507
....508
....509
....510
: 510
....512
....513
....514
....514
....515
....517
122: **519**

Examples 519
: round, floor, ceil, trunc 519
! 520
, trunc 520
520
520
521
521
/ 521
, , 521
, 522
522
522
NaN (" ") 523
Pow 525
cmath 525
123: 529
Examples 529
529
529
529
"__main__" 529
124: 531
Examples 531
531
125: 532
532
532
532
Examples 532
List Stack 532
533

126: 534
..... 534
..... 534
Examples 535
..... 535
..... 535
del 536
..... 537
..... 537
..... 537
..... 538
..... 538
127: 540
..... 540
Examples 540
..... 540
128: 541
..... 541
Examples 541
..... 541
129: 545
..... 545
Examples 545
(Python 2.4 ~ 3.3) 545
..... 545
130: 546
..... 546
..... 546
Examples 546
..... 546
..... 546
..... 547

547
547
548
550
550
! 551
551
552
552
552
552
131: 554
Examples 554
Pyglet 554
WAV 554
winsound 554
554
ffmpeg 554
Windows 555
132: 556
Examples 556
/ 556
557
557
558
558
133: 562
562
562
562
Examples 562
562

562
562
562
(POSIX) 562
563
makedirs - 563
134: 565
Examples 565
565
Methodcaller 565
Itemgetter 565
135: 567
567
567
Examples 567
567
136: 569
569
569
569
569
Examples 570
URL 570
URL 571
137: 572
Examples 572
echo with aiohttp 572
aiohttp 572
Autobahn 573
138: 575
Examples 575
575

139: **576**
..... 576
..... 576
Examples 576
..... 576
..... **576**
..... **577**
/ 577
..... **577**
..... **577**
..... **578**
I / O 578
140: **579**
..... 579
Examples 579
..... 579
..... 579
. ..... **579**
. ..... **580**
ctypes 580
ctypes 581
ctypes 581
..... 582
141: **584**
..... 584
..... 584
Examples 584
..... 584
..... 584
..... 585
..... 585

NamedTuple 586
..... 586
142: 587
..... 587
..... 587
Examples 587
, ... 587
docstrings 587
..... 587
..... 588
docstrings 588
docstring 588
..... 589
PEP 257 589
..... 589
Google Python 590
143: 591
..... 591
Examples 591
..... 591
..... 591
..... 591
144: 593
..... 593
..... 593
..... 593
Examples 593
..... 593
..... 594
..... 594
: __getitem__, __setitem__ __delitem__ 594
..... 596

596
596
145: 598
Examples 598
598
599
146: ( ) 601
Examples 601
601
601
601
PYTHONSTARTUP 602
602
603
147: 605
605
Examples 605
605
607
608
ID 612
int 613
613
sys.argv [0] 614
h14 614
614
(GIL) 615
616
616
JSON 617
148: 618
618

Examples ... 618
... 618
149: ... 620
... 620
... 620
... 620
Examples ... 620
... 620
... 621
... 621
! ... 622
... 622
... 623
... 623
( ) ... 623
... 623
: ... 624
... 624
... 624
... 625
150: ... 626
... 626
Examples ... 626
1 n ... 626
, ... 626
... 629
... 629
- ... 630
Stack Introspection ... 631
151: ... 633
Examples ... 633

Conda 633
152: 635
..... 635
..... 635
Examples 636
..... 636
Exception 636
153: (Regex) 637
..... 637
..... 637
Examples 637
..... 637
..... 638
..... 639
..... 639
..... 640
..... 640
..... 640
..... 640
..... 641
..... 641
..... 641
..... 642
..... 642
..... 642
..... 642
..... 643
`re.finditer` 643
..... 643
154: , ... 645

Examples 645
645
645
max, min 645
: 646
646
646
647
647
iterable N N 650
155: CLI 651
651
651
Examples 651
( ) 651
argparse ( ) 652
argparse ( ) 652
156: 655
655
655
Examples 655
if, elif else 655
(" ") 655
If 655
Else 656
656
656
657
657
657
658

cmp 658
..... 659
..... 660
157: 661
..... 661
..... 661
..... 661
Examples 661
..... 661
reduce 662
..... 662
/ 662
truthy / falsy ( ) 662
158: 664
..... 664
Examples 664
: math.sqrt () cmath.sqrt 664
: ** pow () 664
: math.pow () 665
: math.exp () cmath.exp () 666
-1 : math.expm1 () 666
: , cmath 667
: pow () 3 668
: n 668
..... 669
159: 670
Examples 670
..... 670
160: (abc) 672
Examples 672
ABC 672
/ ABCMeta @abstractmethod ? 672

161: 674
..... 674
Examples 674
IndentationErrors (SyntaxErrors) 674
IndentationError / SyntaxError : 674
..... 674
IndentationError / SyntaxError : 674
..... 674
IndentationError : 675
..... 675
IndentationError : 675
..... 675
..... 675
TypeErrors 675
TypeError : [/] ? ? 675
..... 676
TypeError : [operand] : '???' '???' 676
..... 676
TypeError : '???' / 676
..... 676
TypeError : '???' 677
..... 677
NameError : '???' 677
. 677
. 677
import : 677
LEGB : 677
..... 678
AssertError 678
KeyboardInterrupt 678
ZeroDivisionError 678

679
162: ("with") 681
681
681
681
Examples 681
with 681
682
682
contextmanager 683
684
684
163: 685
685
685
Examples 685
. 685
collections.defaultdict 686
collections.OrderedDict 687
collections.namedtuple 688
collections.deque 689
.ChainMap 690
164: , 692
692
Examples 692
692
165: / 693
693
693
Examples 695
695
697

166: : __str__ __repr__ 699
..... 699
..... 699
..... 699
..... 699
Examples 699
..... 699
..... 700
(1) 700
(2) 701
... . 702
..... 703
, eval-round-trip __repr __ () 703
167: 704
Examples 704
..... 704
..... 704
? 705
..... 705
168: 706
..... 706
..... 706
..... 706
Examples 706
..... 706
..... 706
..... 707
..... 707
..... 708
..... 709
..... 709
..... 709

710

710

710

710

710

169: 711

711

711

711

Examples 711

711

711

711

711

712

712

170: 713

713

713

Examples 713

I / O 713

I / O 714

171: HTTP 716

Examples 716

HTTP 716

716

SimpleHTTPServer API 718

BaseHTTPRequestHandler GET, POST, PUT 719

172: - virtualenv 721

721

Examples 721

721

721

Virtualenv 721

virtualenv 721

173: 723

723

Examples 723

- 723

HTTP 723

TCP 724

UDP 724

Simple HttpServer 725

174: 726

726

Examples 726

726

726

726

726

726

726

175: 728

728

Examples 728

728

728

729

176: 731

731

Examples 731

SSE 731

Asyncio SSE 731

177: 732

Examples....732
....732
....732
....732
....732
....733
....733
178: 734
Examples....734
....734
....734
....735
179: 737
....737
Examples....737
....737
....738
....738
....739
....739
....740
180: 742
....742
....742
Examples....742
....742
....743
181: (pyserial)....744
....744
....744
....744
Examples....744

744
744
744
182: 746
746
Examples 746
746
PyPI 746
.pypirc 746
testpypi ( ) 747
747
PyPI 748
748
Readme 748
748
749
183: 750
Examples 750
Excel 750
OpenPyXL 750
xlsxwriter Excel 751
xlrd Excel 753
xlsxwriter Excel 754
184: (Hashable) 756
Examples 756
Mutable Immutable 756
Immutables 756
757
757
757
758

185: 759

..... 759

Examples 759

SSH 759

186: 760

..... 760

Examples 760

..... 760

..... 760

..... 760

..... 760

187: 761

..... 761

Examples 761

..... 761

wrapper () 761

188: 763

Examples 763

Matplotlib 763

..... 764

MayaVI 767

..... 768

189: 770

..... 770

..... 770

( ) 770

..... 770

requests 770

requests-cache 770

scrapy 770

selenium 770

HTML 770
BeautifulSoup 770
lxml 770
Examples 770
lxml 770
..... 771
Scrapy 771
Scrapy 772
BeautifulSoup4 772
Selenium WebDriver 772
urllib.request 773
..... 773
190: 774
Examples 774
..... 774
..... 774
191: I / O 775
..... 775
..... 775
..... 775
..... 775
..... 775
Examples 775
..... 775
..... 777
..... 777
..... 778
..... 779
..... 779
..... 780
( ) 780
..... 780

mmap 781
..... 781
..... 781
192: 783
..... 783
Examples 783
Python ZipFile.extractall () ZIP 783
TarFile.extractall () tarball 783
193: : 784
Examples 784
..... 784
. 784
transform 784
..... 785
transform 785
194: , ... 787
..... 787
Examples 787
, ... 787
195: 789
..... 789
Examples 789
..... 789
..... 790
..... 791
..... 791
..... 792
196: 793
Examples 793
IPython %% timeit % timeit 793
timeit () 793
timeit 793

line_profiler 793
cProfile ( ) 794
197: 796
..... 796
..... 796
Examples 796
..... 796
URL 796
HTTP 797
..... 798
Jinja 798
..... 799
URL 799
..... 800
..... 800
198: 801
Examples 801
..... 801
..... 802
199: 804
..... 804
Examples 804
Hello World in Pyglet 804
Pyglet 804
Pyglet 804
OpenGL Pyglet 804
Pyglet OpenGL 805
200: 806
..... 806
..... 806
..... 806

806

pickle 806

Examples 806

806

serialize 806

deserialize 807

807

Pickled Data 807

201: 809

809

809

809

Examples 809

809

810

810

: filterfalse, ifilterfalse 810

202: 812

812

812

Examples 812

812

Popen 812

812

812

812

813

813

813

813

203: 815

815
Examples 815
return 815
204: 816
816
Examples 816
MD5 816
OpenSSL 817
205: 818
Examples 818
C Hello World 818
C 819
C ++ C 819
C ++ 819
206: 821
Examples 821
821
821
823

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.

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

1: Python

. :

- : . .
- : Python . PHP. . Python .
- : Python . . .

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

- : . , . .

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

- **Beginner friendly :)** : . . Python , . .

The Zen of Python .

, Python . Python 2.x . Python 3.x . .

Python , Python .

CPython Python.org . IronPython (.NET Python), Jython (Java) PyPy (Python) .

Python 3.x

[3.7]	2017-05-08
3.6	2016-12-23
3.5	2015-09-13
3.4	2014-03-17
3.3	2012-09-29
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	2001-04-15
2.0	2000-10-16

Examples

Python Guido van Rossum 1991 . Python , , . .

Python .

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

. Python 3 Python 2 . Python , . .

▪

, (Windows OS).

```
$ python --version
```

Python 3.x 3.0

3 ( ) .

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

Python 2.x 2.7

2 ( ) .

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

3 `$ python --version` 2 2 . MacOS Linux . `$ python3` Python 3 .

, IDLE Python

IDLE .

IDLE Hello, World

- IDLE .
  - Windows Windows `All Programs` .
  - Windows 8 `IDLE` IDLE .
  - Unix (Mac) `$ idle python_file.py` .
- .

.

```
>>>
```

.

```
>>> 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

`import` Python 2 Python 3 `print` .

```
from __future__ import print_function
```

2 `__future__` 3 .

Python 2.x 2.7

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

`python` () Python . Python REPL ('Read Evaluate Print Loop') .

```
$ 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 3 `python3` .

```
$ 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"
>>>
```

.

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

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

```
Ctrl + D □      .

□□ CTRL + C□ □□□□□□ .
```

.

.

.

- Python (,) .
- .
- .

:

: .

- https://www.python.org/shell/ - .
- https://ideone.com/ - .
- https://repl.it/languages/python3 - , IDE . , .
- https://www.tutorialspoint.com/execute_python_online.php - UNIX .
- http://rextester.com/l/python3_online_compiler - IDE

.

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

.

- Python PyPA PIP . `pip install <the package name>` . , `pip install numpy` . (: PATH pip . `python -m pip install <the package name>`).

- . . IDLE - GUI, IPython - .

- .py , IDE (: PyCharm), (Jupyter) / . .

Python Python .

PEP8 . .

.

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

= . ( ), . .

```
# 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
```

. .

```
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
```

.

```
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'>
```

.

= , = . , = .

:

```
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 , b c` (1 `int`) . , `a , b c` 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.
```

`int , string , tuple` (`list , dict`).

```
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
```

. , .

```
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
```

, = . int 2 a , a A string '' . ?

input (: Python 2.x raw_input . Python 2.x input).

2.x 2.3

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

Python2 input() - Python (Python3 eval(input()) eval(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" Enter `name` `"Bob"`.

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

`input` `str` . `str` .

```
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` `ValueError` .

IDLE - Python GUI

IDLE Python . IDLE . Windows .

IDLE .

- ,
-
- ,
- ( )
- Python .py IDLE .

`F5` `run Python Shell` . IDLE .

. .

- **Windows**

  Windows `python` . `"'python' is not recognized"` `PATH` .'' " " " .' ' ' ...'. `PATH` Python (`C:\Python27;C:\Python27\Scripts`) `C:\Python27;C:\Python27\Scripts` . .

  Python `python.exe` . , `python27.exe` `python27` Python .

  Windows Python Launcher . . `python[xy]` `py -[xy]` Python . `py -2` `py -3` Python 3 Python 2 .

- **/ / MacOS**

  `python` `PATH` .

/ / OS , `python` 2.x `python3` 3.x

`which python` `which python` `which python` .

- Arch Linux ( ) Python Python 3 Python 3.x `python` `python3` , Python 2.x `python2` .
- 3 `python` `python3` . Python 2 Python 2 `python2` .

`bool` : `True` `False` . `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
```

2.x 3.x `int` . `bool` `int` `True` `False` .

```
issubclass(bool, int) # True

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

(`True` `False` `1` `0`) .

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

- `int` :

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

  .

  : Python `long` `int` . .

- `float` : . CPython `float` C double .

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

- `complex` :

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

`<` , `<=` , `>` `>=` `TypeError` .

Python 3.x 3.0

- `str` : `.'hello'`
- `bytes` : `.b'hello'`

Python 2.x 2.7

- `str` : `.'hello'`
- `bytes` : `str`
- `unicode` : `.u'hello'`

(: `set` `dict`) .

- (`str` , `bytes` , `unicode`) .
- `reversed` : `str` reversed

```
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__()` `__hash__()`) (`__eq__()` `__eq__()` `__eq__()` . .

.

- `True` : `bool`
- `False` : `bool`
- `None` : .
- `Ellipsis` `...` : Python3 + Python2.7 . `numpy` ''.
- `NotImplemented` : , .

```
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`).

`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()` Python isinstance type .

`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' float .

```
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'` : 3 `bytes` , 2 `str`
- `u'foo bar'` : 3 `str` , 2 `unicode`
- `'foo bar'` : `str`
- `r'foo bar'` : .

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

. , .

```
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

. .

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

dir(). 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'
```

docstring .

```
>>> 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__', '__getattribute__',
'__hash__', '__init__', '__module__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__']
```

str . print .

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

. , . miscalibration .

(:) (,). , , , `if` Python . .

:

```
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 3.x 3.0

3 . . `Inconsistent use of tabs and spaces in indentation` .

Python 2.x 2.7

2 . . **8** . **4** .

PEP 8 :

`-t` Python 2 . `-tt` . !

" ". ('\ t') `Tab` .

- ( ) 8 . .
- `Tab` 4 .

autopep8 pep8 . (Python : reindent.py)

. `int` `str` .

`list` Python . , , . . .

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

.

```
empty_list = []
```

.

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

.

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

. 0 , 1 :

```
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)` `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) 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]) ( ) .

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

.

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

tuple . . IP . .

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

. .

.

```
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 JSON .

set . . set list .

set dictionary .

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

list set .

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

set in :

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

set , .

defaultdict

defaultdict . defaultdict (,) .

defaultdict KeyError . .

.

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

, .

```
>>> 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')
```

( ) . dict .

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

dict .

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

`dictionary` .

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

. , 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)
```

docstring (:).

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

.

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

.

quit .

importable .

.py .

```
# hello.py
def say_hello():
```

```
    print("Hello!")
```

.

, . (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!"
```

, __init__.py .

- str () repr ()

.

repr(x) x.__repr__() : x . eval .

str(x) x.__str__() : . .

---

repr ()

`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

`pip` python (PyPI) . python.org> Python 2> = 2.7.9 Python 3> = 3.4 `pip` . Linux * nix

pip .

Python 2 Python 3 pip Python 2 pip3 Python 3 . pip Python 2 pip3 Python 3 .

/

.

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

(/ ,)

```
$ 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 .

- Linux MacOS X :

  ```
  $ pip install -U pip
  ```

  Linux sudo pip

- Windows :

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

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

.

Python 2.7.x 3.x

: Python 2.7 ( ) . Python 3.x .

WINDOWS

(https://www.python.org/downloads/) Python 2.7 . MSI . .

Python .

```
 C:\Python27\
```

: PATH .

C : \ Python27 , PATH :

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

Python cmd :

```
python --version
```

2.x 3.x

Windows Python 2.x 3.x .

1. MSI Python 2.x .
   - Python .
   - : Python `PATH` Python 2.x `python` .
2. Python 3.x .
   - , Python .
   - : Python `PATH` Python 3.x `python` . 2.x `PATH` `PATH` .
   - `py launcher` .

Python 3 Python 2.x Python 3.x Python .

```
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)
```

CentOS, Fedora, Redhat Enterprise (RHEL) Ubuntu Python 2.7 .

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

Python :

```
python --version
```

()

3.6 (Ubuntu 16.10 17.04 3.6). 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
```

OS

, 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/ko/python/topic/193/python--

2: "pip" : PyPI Package Manager

Python pip package manager . `ImportError` . Windows `Python_root/Scripts/pip.exe` `__main__.py` , `pip main` . pip pip . pip pip : PyPI Package Manager .

- pip. <function | attribute | class> function .
  - ()
    - ( ) ( ) . (bash, zsh fish) .
  - check_isolated (args)
    - param args {list}
    - returns {boolean}
  - create_main_parser ()
    - {pip.baseparser.ConfigOptionParser } .
  - main (args = None)
    - param args {list}
    - returns {integer} 0 .
  - parseopts (args)
    - param args {list}
  - get_installed_distributions ()
    - {list} .
  - get_similar_commands (name)
    - .
    - {string}
    - returns {boolean}
  - get_summaries (ordered = True)
    - (,) .
  - get_prog ()
    - {string} .
  - dist_is_editable (dist)
    - ?
    - param dist {object}
    - returns {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'])
```

.

```
for i in installed:
        pip.main(['install']+i.key+['--upgrade'])
```

ImportError

.

```
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
```

3.4 3.6 . .whl ( ) . : `scikit_learn-0.18.1-cp36-cp36m-win_amd64.whl` [package_name] - [version] - [python interpreter] - [python-interpreter] - [Operating System] .whl. py . pip `kjhfkjdf.whl is not a valid wheel filename.` `kjhfkjdf.whl is not a valid wheel filename.` .

.whl 7-zip . - . `site-packges` `site-packges` . .

"pip" : PyPI Package Manager : https://riptutorial.com/ko/python/topic/10730/-pip-----pypi-package-manager

3: * args ** kwargs

.

1. `args` `kwargs` . , :

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

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

2. `args` `kwargs` ( )

```
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` . `*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)
```

farg .

** kwargs

** ( ) .

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

, :

```
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]` .

`*args` `A` `__init__` . args `*` "". , `B` `__init__` S ' `A` `__init__` , `2, 3` (, `A(2, 3)`).

`x` `a` `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
```

.

zip * .

```
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 . * . * (,) . .

```
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/ko/python/topic/2475/--args-----kwargs

4: __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/ko/python/topic/1223/--name----

5: `exec``eval`

- eval (expression [, globals = None [, locals = None]])
- exec ()
- exec (,)
- exec (, ,)

expression	code
object	code
globals	. . globals() .
locals	. globals . globals() locals() globals locals .

exec globals locals (,) . globals locals globals .

globals __builtins__ Python . print isinstance globals __builtins__ None . .

2 . 3 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 D . eval . . compile '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` 0 .

```
>>> from collections import defaultdict
>>> variables = defaultdict(int, {'a': 42})
>>> eval('a * c', globals=variables)  # note that 'c' is not explicitly defined
0
```

ast.literal_eval

, 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 .

exec, eval ast.literal_eval

eval exec . `ast.literal_eval` . .

Python .

`exec``eval` : https://riptutorial.com/ko/python/topic/2251/-exec--eval-----

6: 2to3

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

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

2to3 Python 2.x Python 3.x Python . Python 2.x Python 3.x .

2to3 lib2to3 . lib2to3 2to3 .

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 .

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

Windows

```
> 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` .

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

Windows

```
> 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/ko/python/topic/5320/2to3-

7: AMQPStorm RabbitMQ

AMQPStorm pypi pip .

```
pip install amqpstorm
```

Examples

RabbitMQ

.

```
from amqpstorm import Connection
```

. (start_consuming to_tuple).

. RabbitMQ 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 simple_queue , RabbitMQ .

```
channel.basic.consume(callback=on_message, queue='simple_queue', no_ack=False)
```

RabbitMQ IO .

```
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)
```

`publish` `routing_key` . `simple_queue` `simple_queue` .

```
message.publish(routing_key='simple_queue')
```

RabbitMQ

. . `confirm delivery` , `delivery_mode` `durable` . 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 *(- Time To Live)*

  ( ) .

- x-dead-letter-routing-key

  .

- hello_delay hello Exchange .

Pika .

```
delay_channel.basic.publish(exchange='',
                            routing_key='hello_delay',
                            body='test',
                            properties={'delivery_mod': 2})
```

RabbitMQ .

Overview					Messages				Messa
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")
```

AMQPStorm RabbitMQ : https://riptutorial.com/ko/python/topic/3373/amqpstorm--rabbitmq-

8: ArcPy

ArcPy (da) .

arcpy.da.SearchCursor arcpy.SearchCursor () .

(arcpy.da) ArcGIS 10.1 .

Examples

(C : \ Temp) (Test.gdb) (TestFC) (TestField) .

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

createDissolvedGDB gdb

```
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/ko/python/topic/4693/arcpy

9: Asyncio

Examples

3.5+ `asyncio` 3.5 .

Python 3.x 3.5

3.5 `async await` . `await func()` `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 . `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 .

asyncio concurrent.futures Executor . Executor , Callable Callable run_in_executor() .

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))
```

"" Executor Executor . Executor set_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.futures ThreadPoolExecutor ProcessPoolExecutor Executor . ThreadPoolExecutor 5 . ThreadPoolExecutor I / O CPU . ProcessPoolExecutor . ProcessPoolExecutor picklable . . Executor ThreadPoolExecutor .

UVLoop

`uvloop` `asyncio.AbstractEventLoop` `asyncio.AbstractEventLoop` (`asyncio.AbstractEventLoop`). 99 % `asyncio` `asyncio.EventLoop` . Windows `uvloop` `pip install uvloop` `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
A

B
A

asyncio echo websocket `asyncio` . . 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

`asnycio` ( ) GIL ( ) - . !

`asyncio` (`asyncio`) : () . `asyncio.sleep` . " (`await`) . `time.sleep` . `time.sleep` .

() `requests` . `asyncio` . `aiohttp` asyncio asyncio . .

-

CPU `asyncio` . `threads` `multiprocessing` `threads` .

- IO- , `asyncio` .

Asyncio : https://riptutorial.com/ko/python/topic/1319/asyncio-

10: base64

64 64 2 ASCII . base64 Python . . Base64 .

- base64.b64encode (s, altchars = None)
- base64.b64decode (s, altchars = None, validate = False)
- base64.standard_b64encode (s)
- base64.standard_b64decode (s)
- base64.urlsafe_b64encode (s)
- base64.urlsafe_b64decode (s)
- base64.b32encode (s)
- base64.b32decode (s)
- base64.b16encode (s)
- base64.b16decode (s)
- base64.a85encode (b, foldspaces = False, wrapcol = 0, = False, adobe = False)
- base64.a85decode (b, foldpaces = False, adobe = False, ignorechars = b '\ t \ n \ r \ v')
- base64.b85encode (b, pad = False)
- base64.b85decode (b)

`base64.b64encode(s, altchars=None)`	
	Base64 '+' '=' 2+ . .
`base64.b64decode(s, altchars=None, validate=False)`	
	Base64 '+' '=' 2+ . .
	valide True Base64 .
`base64.standard_b64encode(s)`	
`base64.standard_b64decode(s)`	
`base64.urlsafe_b64encode(s)`	
`base64.urlsafe_b64decode(s)`	

`b32encode(s)`	
`b32decode(s)`	
`base64.b16encode(s)`	
`base64.b16decode(s)`	
`base64.a85encode(b, foldspaces=False, wrapcol=0, pad=False, adobe=False)`	
foldspaces	foldspaces True 4 'y' .
wrapcol	(0)
	pad True 4 .
	adobe True Adobe '<~' "~> '
`base64.a85decode(b, foldspaces=False, adobe=False, ignorechars=b'\t\n\r\v')`	
foldspaces	foldspaces True 4 'y' .
	adobe True Adobe '<~' "~> '
ignorechars	
`base64.b85encode(b, pad=False)`	
	pad True 4 .
`base64.b85decode(b)`	

3.4 base64 `bytes bytearray` . .

Examples

Base64

base64 .

```
import base64
```

base64 . , encode . `UTF-8` ( ) , . .

```
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 . `UTF-8` `decode()` :

```
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

base64 Base32 . 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

Base64 Base16 . 16 **16** . Base64 Base32 .

```
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

Adobe **ASCII85** . **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

Base64, Base32 Base16 Base85 `base64.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;B1a%^NF
Hello World!
```

base64 : https://riptutorial.com/ko/python/topic/8678/base64-

11: C

C # Python .

NuGet **MongoDB.Bson** .

JSON .

.

- JSON .

```
 BsonDocument argsBson = BsonDocument.Parse("{ 'x' : '1', 'y' : '2' }");
 string argsFile = string.Format("{0}\\{1}.txt", Path.GetDirectoryName(pyScriptPath),
Guid.NewGuid());
```

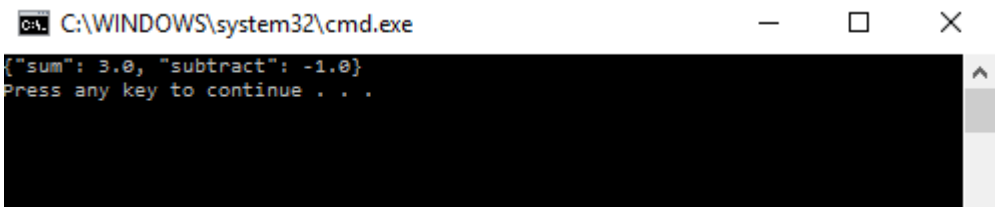
- Python python.exe JSON python .

```
 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' ] ]
```

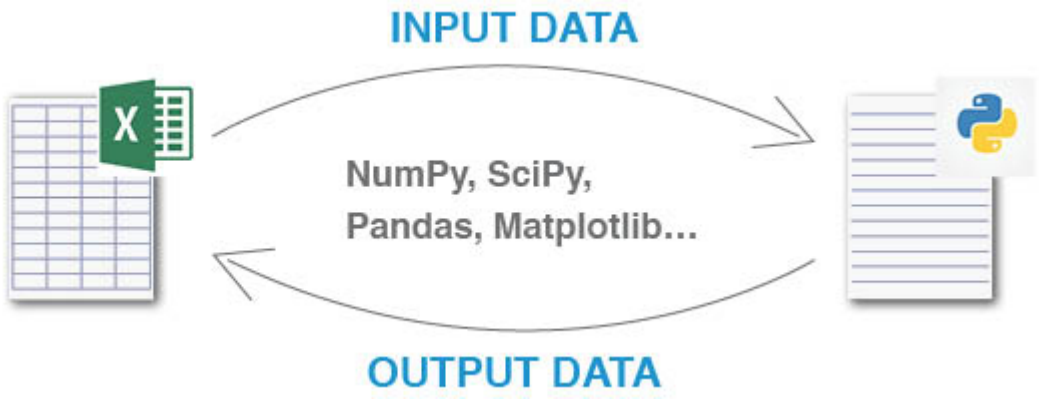
- Python JSON .

```
 print json.dumps( { 'sum' : x + y , 'subtract' : x - y } )
```



- C # JSON .

```
 using (StreamReader myStreamReader = process.StandardOutput)
 {
    outputString = myStreamReader.ReadLine();
    process.WaitForExit();
 }
```



C # Python Excel Python .

C # ExcelDNA - Excel .

GitHub .

4 .

-
-
-
-
-

.

Examples

C

```
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 } )
```

C

```
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);
        }
    }
}
```

C # : https://riptutorial.com/ko/python/topic/10759/c-sharp--

12: ChemPy -

ChemPy , Python . , .

Examples

```
from chempy import Substance
ferricyanide = Substance.from_formula('Fe(CN)6-3')
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('%.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}, {'Mn+2': 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 Mn+2 + 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 Mn+2 + 5 O2 + 6 H2O; K1**4*Kw**20/K2**5
```

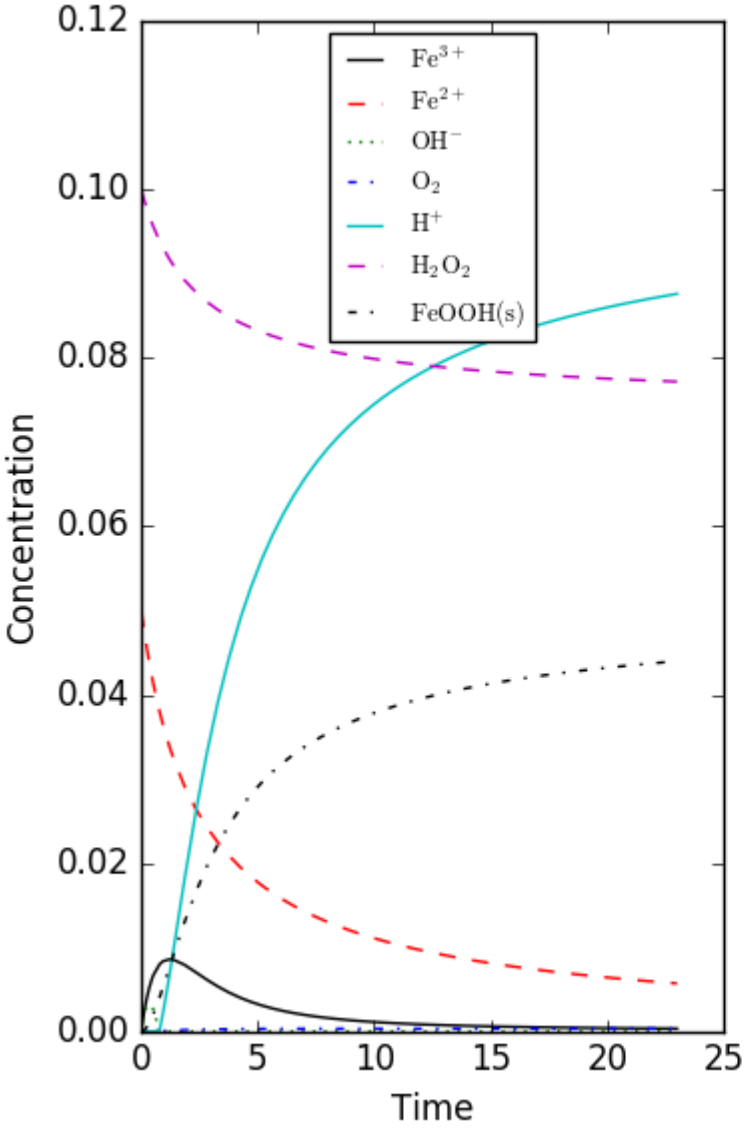
```
 from chempy import Equilibrium
```

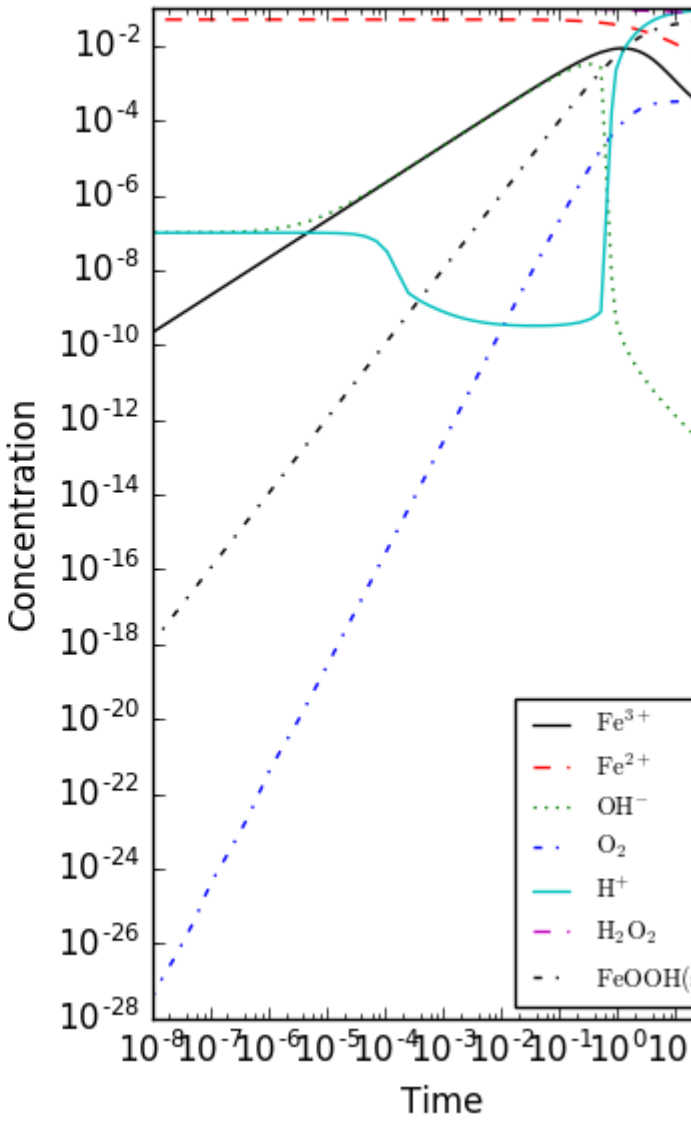
```
 from chempy.chemistry import Species
 water_autop = Equilibrium({'H2O'}, {'H+', 'OH-'}, 10**-14)  # unit "molar" assumed
 ammonia_prot = Equilibrium({'NH4+'}, {'NH3', 'H+'}, 10**-9.24)  # same here
 from chempy.equilibria import EqSystem
 substances = map(Species.from_formula, 'H2O OH- H+ NH3 NH4+'.split())
 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 - : https://riptutorial.com/ko/python/topic/10625/chempy----

13: configparser

INI ConfigParser . Python .

- = .
- .
- INI . []

`ConfigParser.ConfigParser().get . eval .`

Examples

config.ini :

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

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

:

```
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/ko/python/topic/9186/configparser

14: 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
```

`dict` `DataFrame` CSV .

```
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")
```

CSV `DataFrame` `dict` .

```
df = pd.read_csv("data.csv")
d = df.to_dict()
```

CSV : https://riptutorial.com/ko/python/topic/2116/csv---

15: Deque

- dq = deque () # .
- dq = deque (iterable) # deque .
- dque.append (object) # .
- dque.appendleft (object) # .
- dq.pop () -> object # .
- dq.popleft () -> object # .
- dque.extend (iterable) # .
- dque.extendleft (iterable) # .

`iterable`	.
`maxlen`	deque new .

(`deque` " `deque` ").

`pop` , `append` `extend` `left` . O (1) `deque` .

Examples

deque

`popleft` `appendleft`

```
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)
```

:

```
dl = 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` , .) .

```
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/ko/python/topic/1976/deque-

16: 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*) . `dis` , , Python .

```
>>> 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
```

.

( ) (opcode) . opcode .

`.pyc` ( ) . .

```
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
```

dis . .

dis : https://riptutorial.com/ko/python/topic/1763/dis-

17: Functools

Examples

partial . ( ) .

```
>>> from functools import partial
>>> unhex = partial(int, base=16)
>>> unhex.__doc__ = 'Convert base16 string to int'
>>> unhex('ca11ab1e')
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 (a, b, c, x) f a, b, c, f g g(2) f .

partial . . partial .

total_ordering

__eq()__ , __lt__() , __le__() , __gt__() __ge__() .

total_ordering __eq__() , __lt__() , __le__() , __gt__() __ge__() __eq__() .

```
@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 reduce functools .

```
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) : (: `3.0` `3`)

.

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

: `@lru_cache` .

`@lru_cache` . `@lru_cache` 3.2 .

cmp_to_key

. .

-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']
```

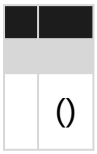
Python

Functools : https://riptutorial.com/ko/python/topic/2492/functools-

18: groupby ()

Python `itertools.groupby()` .

- itertools.groupby (iterable, key = None some 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' '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

mulato camel . . 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']]}
```

groupby () : https://riptutorial.com/ko/python/topic/8690/groupby---

19: GZip

GNU gzip gunzip .

zlib .

gzip Python File Object GzipFile . GzipFile gzip . .

Examples

GNU zip

```
import gzip
import os

outfilename = 'example.txt.gz'
output = gzip.open(outfilename, 'wb')
try:
    output.write('Contents of the example file go here.\n')
finally:
    output.close()

print outfilename, 'contains', os.stat(outfilename).st_size, 'bytes of compressed data'
os.system('file -b --mime %s' % outfilename)
```

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/ko/python/topic/8993/gzip-

20: HTML

Examples

BeautifulSoup

HTML .

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

`label` "John Smith" .

`label` `.next_sibling` `.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` .

BeautifulSoup CSS

BeautifulSoup CSS CSS . `select()` `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

pyquery python jquery . 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/ko/python/topic/1384/html--

21: Itertools

- `import itertools`

Examples

iterable .

```
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 . , . 2 5 5 .

```
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 , stop step .

```
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
```

* 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()` `start` `step` `step` .

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

:

```
10
14
18
22
```

itertools.takewhile

itertools.takewhile `False` .

```
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_even` `13` . `takewhile` `False` .

`takewhile` .

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

: `takewhile` `dropwhile` .

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

itertools.dropwhile

itertools.dropwhile `False` .

```
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]` .

(*takewhile dropwhile dropwhile* .)

(,) `is_even` `13` . .

`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
```

`takewhile` `dropwhile` .

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

.

`zip()` `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 2.6 2.7 `itertools.izip_longest` .

Itertools

`itertools.combinations` *k-* .

: k-wise .

:

:

```
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'
```

, `chain.from_iterable` . `chain.from_iterable` iterable . .

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

`chain` `chain.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
```

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` iterable r-length .

```
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)]
```

`a` `set` .

```
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/ko/python/topic/1564/itertools-

22: JSON

.

`json` .

:

JSON	
	dict
	str
(int)	int
()	

`json` `NaN`, `Infinity` `-Infinity` JSON .

:

	JSON
dict	
,	
str	
int, float, (int / float)	

`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)
```

:

json (: object_hook parse_float) . .

```
# 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 dict JSON

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

.

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

JSON Python 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]}
```

`d` JSON (`filename filename`).

```
import json

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

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

JSON (`filename`) .

```
import json

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

```` ````
   ,

`json` . `s` . StringIO .

.
````

```
import json

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

.

```
import json

from io import StringIO

json_file = StringIO()
data = {u"foo": u"bar", u"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)
# {u"foo": u"bar", u"baz": []}
```

json . . `0` .

```
import json

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

json `pyspark` json-per-line json .

```
# 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
        }
    }
}
```

STDOUT . (Bash) :

```
$ 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/ko/python/topic/272/json-

23: kivy - NUI

NUI : (Natural User Interface, NUI) .

Kivy Python . ( ) .

Examples

Kivy

1. **app**
2. .
3. .

```
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
```

ux **Label** . ux your_installation_directory / kivy / uix / .

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

. .

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

app . .

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

. **Hello World** Label .

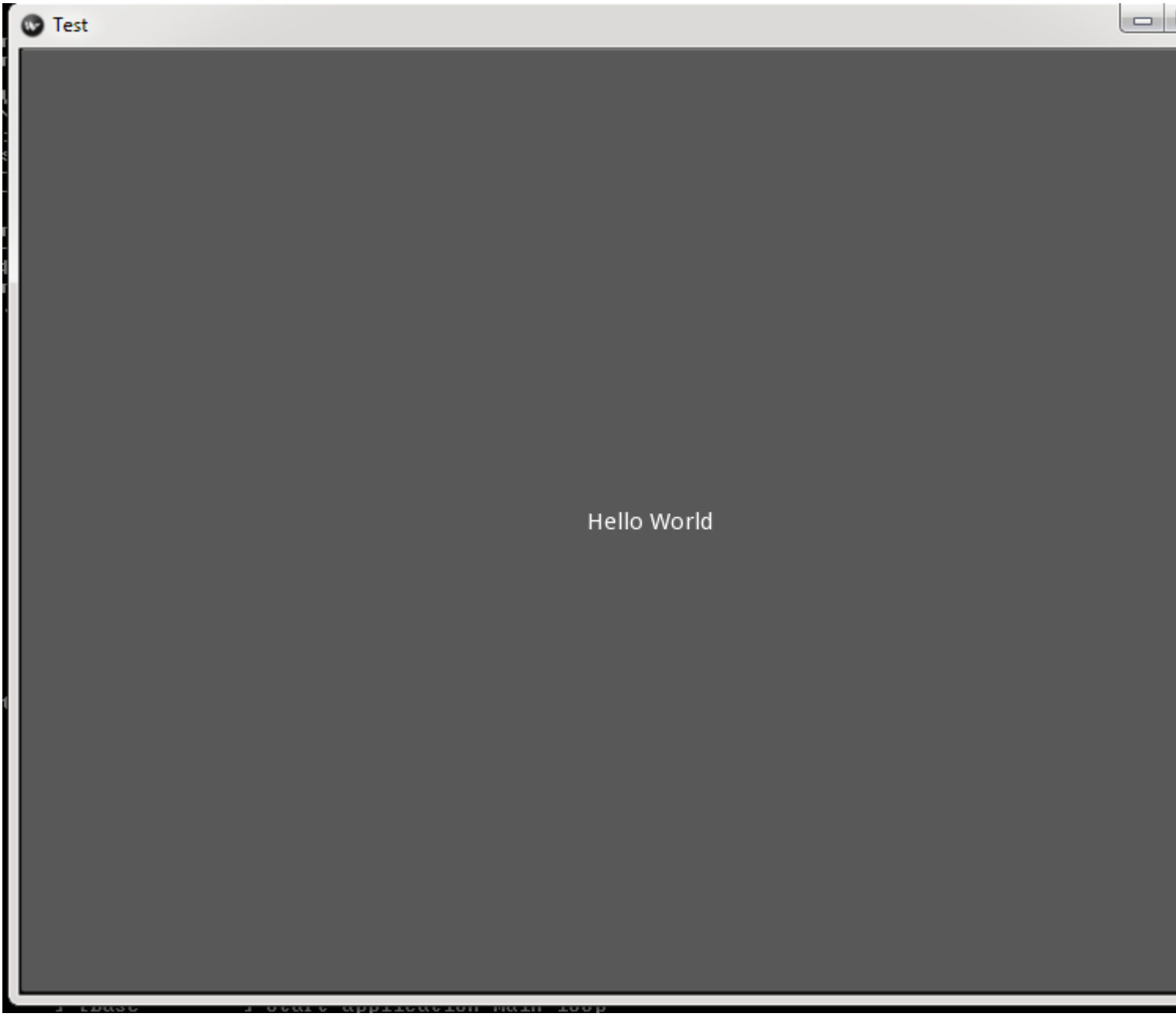
```
if __name__ == '__main__':
```

.

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

Test . app run () .

.



kivy - NUI : https://riptutorial.com/ko/python/topic/10743/kivy---nui------

24: Matplotlib

Matplotlib (https://matplotlib.org/) NumPy 2D . . (https://matplotlib.org/2.0.2/gallery.html https://matplotlib.org/2.0.2/examples/index.html) http : //www.riptutorial.com/topic/881

Examples

Matplotlib

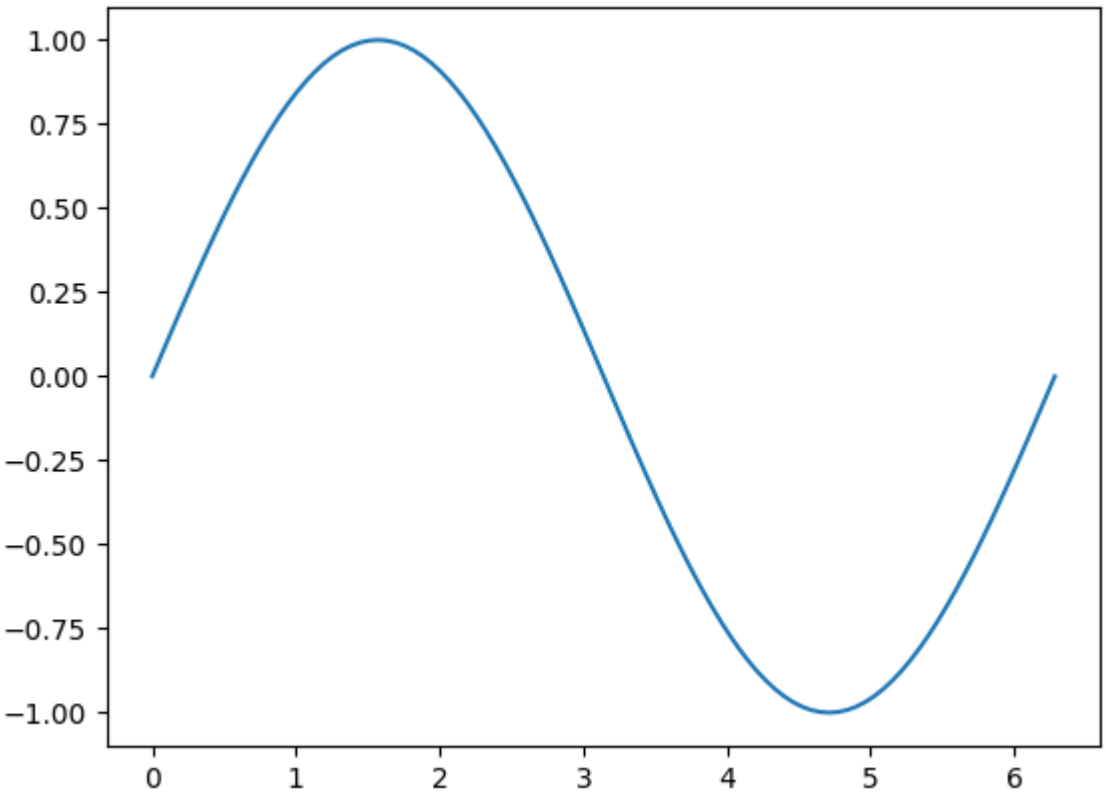
Matplotlib .

```
# 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()
```



: , , ,

. , , , , .

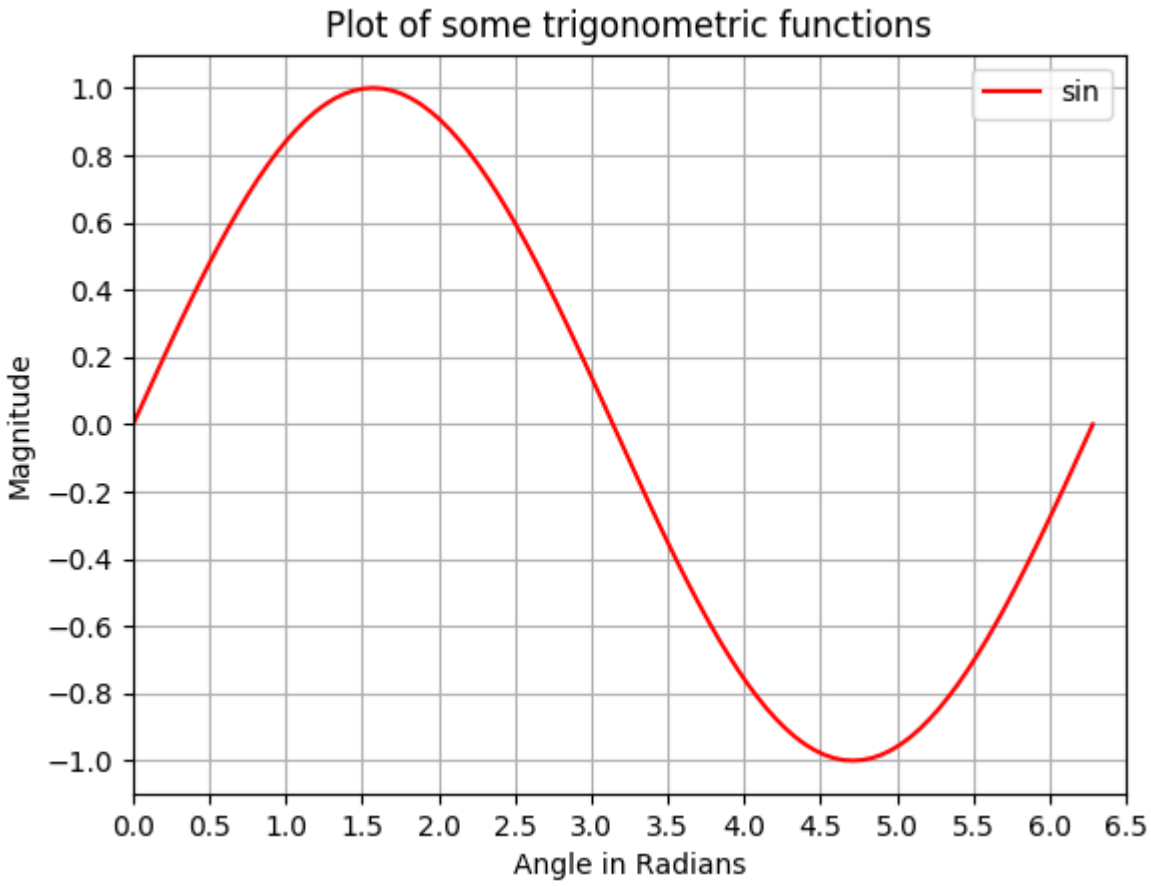
```
# Plotting tutorials in Python
# Enhancing a plot

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

.

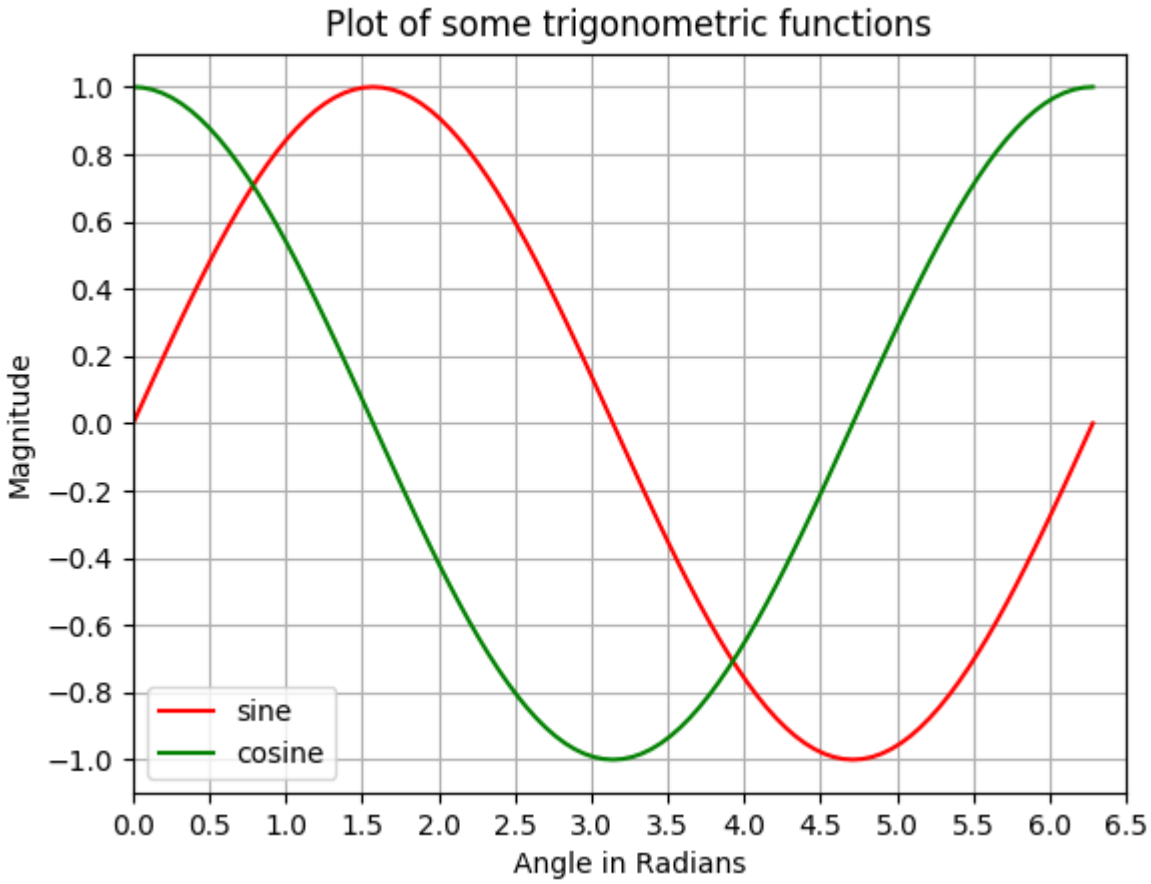
```
# Plotting tutorials in Python
# Adding Multiple plots by superimposition
# Good for plots sharing similar x, y limits
# Using single plot command and legend

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()
```



. .

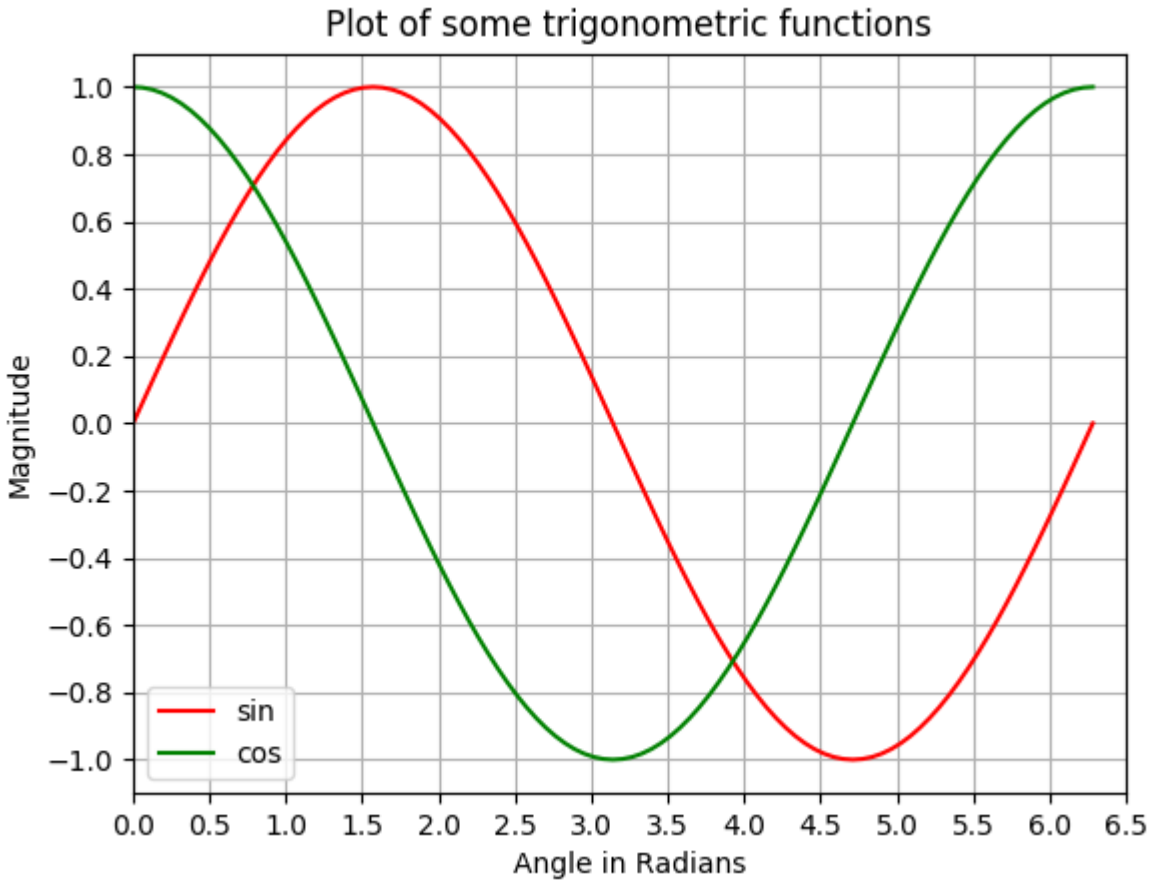
```
# Plotting tutorials in Python
# Adding Multiple plots by superimposition
# Good for plots sharing similar x, y limits
# Using multiple plot commands
# Much better and preferred than previous

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()
```



X Y : twinx ()

y x . **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()

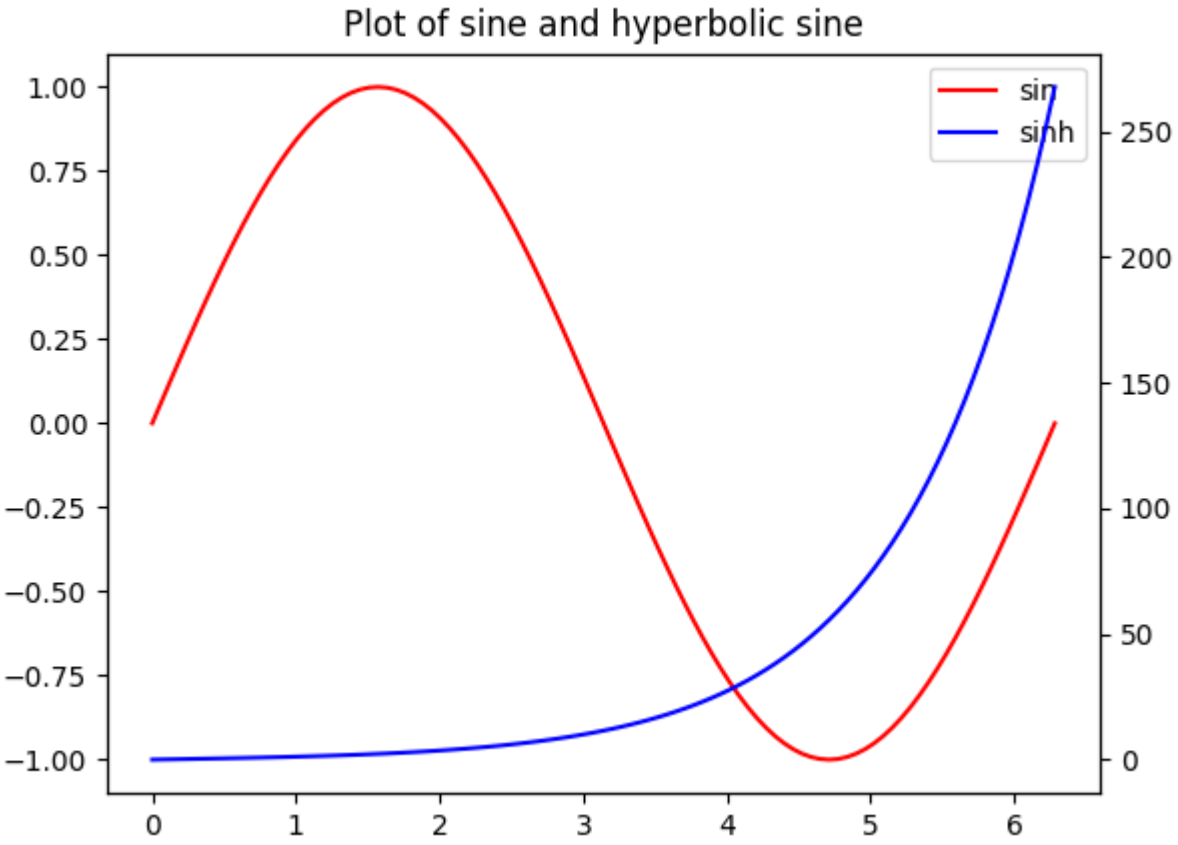
# Duplicate the axes with a different y axis
# and the same x axis
ax2 = ax1.twinx() # ax2 and ax1 will have common x axis and different y axis
```

```
# 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()
```



twiny () Y X

tween () y x . , , , , .

```
# 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.twiny() # 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())

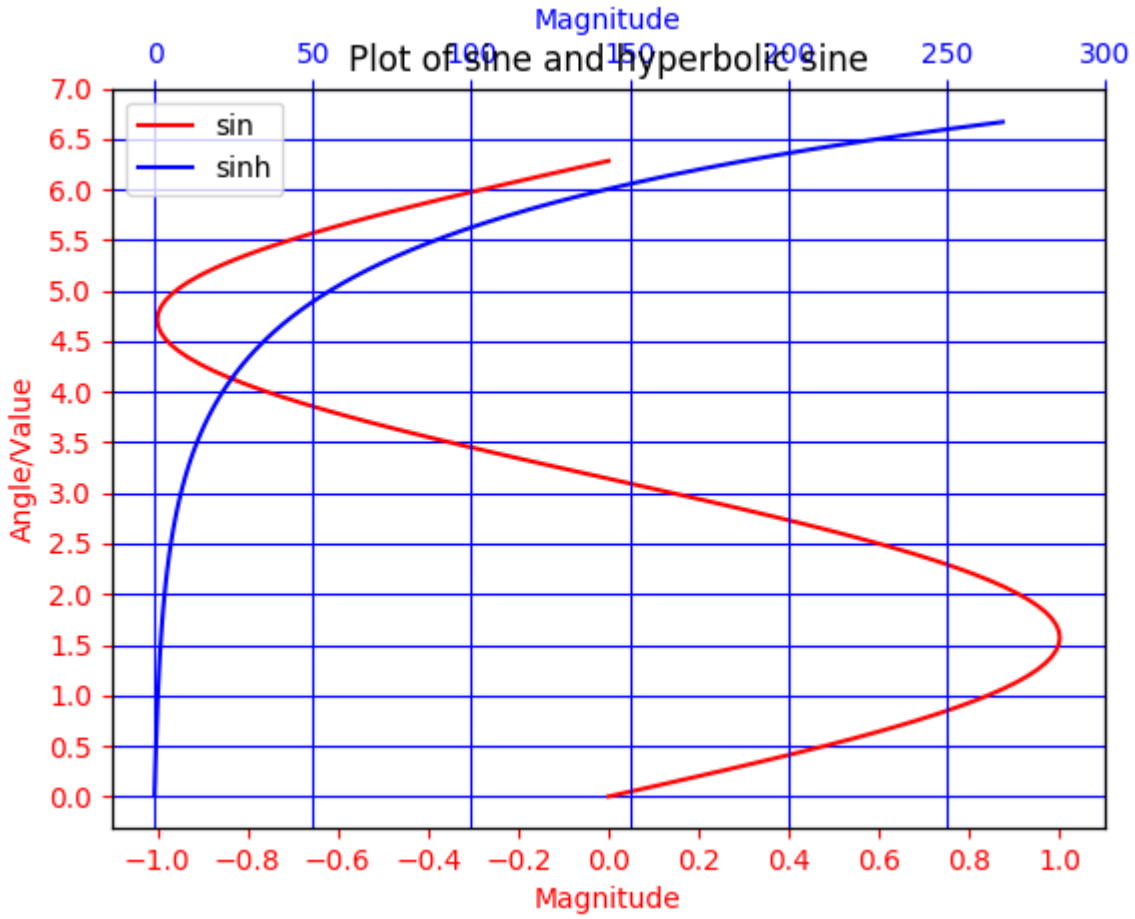
# set x ticks
ax1.set_xticks(xnumbers1)
ax2.set_xticks(xnumbers2)

# 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/ko/python/topic/10264/matplotlib-

25: os.path

. . ( ) .

- os.path.join (a, * p)
- os.path.basename (p)
- os.path.dirname (p)
- os.path.split (p)
- os.path.splitext (p)

Examples

python os .

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

os.path .

, Windows .

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

OS :

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

`os.path.abspath` .

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

:

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

```
>>> os.path.split(os.getcwd())
('/Users/csaftoiu/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/ko/python/topic/1380/os-path

26: pip : PyPI

pip Python Package Index Python .

- pip <command> [options] <command> .
  - PyPI
  - ( )
  - ( )
  - pybundles ( )

pip . Linux python C . Windows Visual Studio / Visual C ++ ( ).

	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

Examples

SomePackage SomePackage .

```
$ pip install SomePackage
```

.

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

.

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

Linux / Unix sudo

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

pip . .

`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
```

```
$ 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` .

virtualenv requirements.txt .

`pip freeze requirements.txt` .

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

`--local` virtualenv . .

pip Python

3 2 pip . 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/python25 -m pip install some_package # on OS-X/Linux
```

OS-X / Linux / Unix ( ) . , `sudo` .

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

Windows Python 2 Python 3 Python 3 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
```

`virtualenv venv` enviroments .

Python Python Package Index . Windows vcvarsall.bat .

C C ++ python *pypi* Windows .

Gohlke . **-win32-** *WIN64* 64 **-win_amd64** 32 , **-** ***NN*** **-cp -** . 34 **-cp *34*-** . .

, Windows , , .

Python 2.7 Visual Studio 2008, Python 3.3 3.4 Visual Studio 2010 Python 3.5+ Visual Studio 2015 .

- Microsoft " Python 2.7 Visual C ++ "
- Microsoft Windows 7 " Windows 7 .NET Framework 4 Windows SDK "(v7.1) .
- Visual Studio 2015 Community Edition *( )* **C & C ++** . ***8*** . .

, .

pip . .

: *pypi Christop* . . Python . Python . *6 PISTON 3* PIL .

. , . .

Pre-Release

Pip Semantic Versioning . `V0.98` `V1.0-rc1` `pip install` `V0.98` `V0.98` - . `--pip install --pre` *package-name* `--pip install --pre --upgrade` *package-name* . .

pip github . .

.

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

`--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
```

URL `git+` .

3. `git` , `mercurial` , *DVCS* `pip install` /// *REPO* -, requires.text *`pip install -r requires.txt`* *`pip install -r requires.txt`* *`python setup.py install`* . git : `git pull origin master` `pip uninstall` *package-name* `git checkout` .

pip : PyPI : https://riptutorial.com/ko/python/topic/1781/pip---pypi--

27: PostgreSQL

Examples

PostgreSQL . psycopg2 .

```
pip install psycopg2
```

my_database my_table .

		last_name
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/ko/python/topic/3374/postgresql

28: py.test

Examples

py.test

py.test Python . pip .

```
pip install pytest
```

projectroot/module/code.py projectroot/module/code.py .

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

projectroot/tests/test_code.py projectroot/tests/test_code.py . **test_** .

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


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

projectroot py.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() my_stuff = get_prepped_stuff() .

py.test !

Fixture . , .

get_prepped_stuff prepped_stuff . . @pytest.fixture .

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

fixture . . py.test . (, , , ,) .

```
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 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
```

. , . ( ) .

```
@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** . `return` `yield` , `yield`. `yield_fixture` py.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 fixture yield fixture documentation.

py.test : https://riptutorial.com/ko/python/topic/7054/py-test

29: Py2Neo Neo4j Cypher

Examples

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

Neo4j localhost : 7474 .

`graph` neo4j . . `__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` . ( )

neo4j `timestamp` . '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
```

`text` `name` cypher .

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 Trump / .

```
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
```

Py2Neo Neo4j Cypher : https://riptutorial.com/ko/python/topic/5841/py2neo--neo4j--cypher

30: pyautogui

pyautogui . . (0,0) . . .

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('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/ko/python/topic/9432/pyautogui-

31: PyInstaller -

- pyinstaller [options] [script ...] | specfile

PyInstaller . Python . numpy, Django, OpenCv .

:

- Pyinstaller Python 2.7 Python 3.3 .
- Pyinstaller Windows, Linux Mac OS X .
- . (Windows Linux . Windows PyInstaller Windows .)

Examples

Pyinstaller Python . pip :

```
pip install pyinstaller
```

Windows
Windows pywin32 pypiwin32 . pyinstaller pip .

Mac OS X
PyInstaller Mac OS X Python 2.7 . Python PyQT, Numpy, Matplotlib . MacPorts Homebrew .

pip PyPI .
PyInstaller Downloads .

`setup.py` . PyInstaller `python setup.py install` with administrator .

`pyinstaller` .
`pyinstaller --version` . pyinstaller .

Pyinstaller

.

```
pyinstaller myfile.py
```

Pyinstaller .

- **myfile.spec** `myfile.py`
- `myfile.py`
- `myfile.py` **dist**
-

dist .

pyinstaller . .

'dist \ myfile \ myfile.exe' .

PyInstaller `myscript.py` `myscript` (`myscript.exe` in windows) (`myscript`).
zip .

```
-D --onedir

pyinstaller myscript.py -D
```

:

. .
. .

.
.

```
pyinstaller myscript.py -F
```

`-F --onefile` . `myscript.exe` .

. .

PyInstaller - : https://riptutorial.com/ko/python/topic/2289/pyinstaller-----

32: Python 2 Python 3

Python . 3 2008 . Python 2 Python 3 .

Python 2.7 (Python 2) 3.6 (Python 3). 3.3 3.4 .

Python 2.7 Python 1.x 2.x Python Python . . CPython . CPython 2020 .

Python Enhancement Proposal 373 2016 6 25 Python 2 2020 . (2020 Python 2.)

Python 3 . Python 3 . .

Python 3.0 Python 3 Python 2.6 Python 3 Python 2 . Python 2 Python 3 (:) Python.

.

```
from __future__ import print_function
# other imports and instructions go after __future__
print('Hello world')
```

`__future__` .

2to3 Python 2.x Python 3.x Python (Python).

6 Python 2/3 .

-
- /
-

Python 2 Python 3 .

Examples

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)"
```

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
```

`end` print . :

```
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
```

`flush` Python 3 .

PEP 3105 .

:

Python 2.x 2.7

Python 2 . `str` (`unicode`) .

Python 2 `str` .

.

```
s = 'Cafe'    # type(s) == str
```

. u : () .

```
s = u'Café'   # type(s) == unicode
b = 'Lorem ipsum'  # type(b) == str
```

() .

```
from __future__ import unicode_literals

s = 'Café'   # type(s) == unicode
b = 'Lorem ipsum'  # type(b) == unicode
```

( ) .

```
isinstance(s, basestring)
```

Python 3.x 3.0

3 str .

```
s = 'Cafe'           # type(s) == str
s = 'Café'           # type(s) == str (note the accented trailing e)
```

Python 3 "blob" bytes . b encode .

```
# 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

2 3 u . 3 u .

```
u'Cafe' == 'Cafe'
```

Python 2 ur .

```
>>> ur'Café'
  File "<stdin>", line 1
    ur'Café'
            ^
SyntaxError: invalid syntax
```

Python 3 text (str) encode bytes . UTF-8 .

decode bytes .

```
>>> b.decode()
'Café'
```

Python 2.x 2.6

`bytes` 2 3 `unicode` 2 . 2 3 .

```
from __future__ import unicode_literals
print(repr("hi"))
# u'hi'
```

Python 3.x 3.0

Python 3 `int` .

```
b"abc"[0] == 97
```

1 .

```
b"abc"[0:1] == b"a"
```

Python 3 . , 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 .

3 , `x / y` `__truediv__` (`__truediv__`) `__truediv__` . , Python 2 ( ).

:

	2	3
`3 / 2`	1	1.5
`2 / 3`	0	0.6666666666666666
`-3 / 2`	-2	-1.5

0 Python 2.2 Python 2.7 Python 3 .

: 2 ( ) . Python 2 `0 2/3 2 / 3.0 2.0 / 3 2.0/3.0 0.6666666666666666`

	2	3
`3.0 / 2.0`	1.5	1.5
`2 / 3.0`	0.6666666666666666	0.6666666666666666
`-3.0 / 2`	-1.5	-1.5

(`//`) . . (float float) `//` `__floordiv__` .

	2	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` `from __future__ import division` .

```
# needs to be the first statement in a module
from __future__ import division
```

	2	3
`3 / 2`	1.5	1.5
`2 / 3`	0.6666666666666666	0.6666666666666666
`-3 / 2`	-1.5	-1.5

`from __future__ import division / __future__` true / not .

: (, `-3 / 2 == -1`) ( ) . .

---

float : `from __future__ import division` / float : `3 / 2.0 == 1.5` . `average = sum(items) / len(items)` float . , (: `float` `int`). , 3 , `3 / 2 == 1` True .

3 PEP 238 .

.

Reduce .

2 `reduce` `functools` (2.6) 3 `reduce` `functools` . Python2 Python3 `reduce` `reduce(function_to_reduce, list_to_reduce)` .

, . `operator` `truediv` .

2.x :

2.x 2.3

```
>>> my_list = [1, 2, 3, 4, 5]
>>> import operator
>>> reduce(operator.truediv, my_list)
0.008333333333333333
```

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` `reduce` .

xrange

Python 2 `range` `xrange` `xrange` `index` `count` .

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` `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
```

, 3.2, `range` , `index` `count` :

```
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
```

.

2.x 2.3

```
# range(10000000000000000)
# The output would be:
# Traceback (most recent call last):
#  File "<stdin>", line 1, in <module>
# MemoryError

print(xrange(100000000000000000))
# Out: xrange(100000000000000000)
```

, 3 . 3 `range` `list()` :

Python 3.x 3.0

```
print(list(range(1, 10)))
# Out: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Python 2.x Python 3.x `future` `builtins` .

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 3.2 Python , `index count` .

Python 3.x 3.0

Python 3 iterable iterable iterable . . . , `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` . 0 .

```
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

3 . 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 .

, , raise except :

2.x 2.3

```
try:
    raise IOError, "input/output error"
except IOError, exc:
    print exc
```

3 , as :

```
try:
    raise IOError("input/output error")
except IOError as exc:
    print(exc)
```

Python 3 Python 2.6 .

Python 3.x 3.0

3 .

```
try:
    file = open('database.db')
except FileNotFoundError as e:
    raise DatabaseError('Cannot open {}') from e
```

except DatabaseError __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 {}')
```

```
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` .

```
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')
```

2 3 6 :

```
import six
try:
    file = open('database.db')
except FileNotFoundError as e:
    six.raise_from(DatabaseError('Cannot open {}'), None)
```

.next () .

Python 2 iterator `next` .

2.x 2.3

```
g = (i for i in range(0, 3))
g.next()  # Yields 0
```

```
g.next()  # Yields 1
g.next()  # Yields 2
```

3 `.next` `.__next__` `.__next__` , "magic" . `.next` `AttributeError` . Python 2 Python 3 iterator `next` .

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 .

2.x 2.3

. . None , , . int str tuple list .

```
[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}]
```

key str . [, ' , { 0-9 .

Python 2 raw_input ,

2.x 2.3

```
user_input = raw_input()
```

Python 3 input .

Python 3.x 3.0

```
user_input = input()
```

2 input . Python 3 . eval(input()) .

Python .

```
try:
    input = raw_input
except NameError:
    pass
```

3, 2 : has_key , iter* , view* . d.has_key(key) key in d .

2 keys , values items . 3 . .

- (len)
- .

.

2.7 3 . viewkeys , viewvalues viewitems . 2 3 , .

- 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())
- 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]
```

3 , keys Python 3 RuntimeError: dictionary changed size during iteration RuntimeError: dictionary changed size during iteration . for key in list(d) for key in list(d) .

, iterator . next() . . 2 d.iterkeys() , d.itervalues() d.iteritems() *iterable* iterator

`iter(d)` , `iter(d.values())` Python 3 `iter(d.values())` `iter(d.items())` .

exec Python 3 .

2, `exec` : `exec code [in globals[, locals]]`. 3 `exec` : `exec(code, [, globals[, locals]])` Python 2 `SyntaxError` .

`print` statement function `__future__` import . `from __future__ import exec_function` no `from __future__ import exec_function` . 2 exec 3 `exec` .

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 2 Python 3 .

2 hasattr

2, `hasattr` , `hasattr` `False` .

```
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 . 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!")
```

.

_winreg	
ConfigParser	configparser
copy_reg	
SocketServer	
_markupbase	
repr	reprlib
test.test_support	test.support
Tkinter	
tkFileDialog	tkinter.filedialog
urllib / urllib2	urllib, urllib.parse, urllib.error, urllib.response, urllib.request, urllib.robotparser

. tkinter urllib .

Python 2.x 3.x `future` Python 2.x Python 3.x .

8

2 8 .

```
>>> 0755  # only Python 2
```

```
0o755  # both Python 2 and Python 3
```

3 " ".

Python `3.x` . `object` . `class object` .

Python 3.x 3.0

```
class X: pass
class Y(object): pass
```

`mro` ( ) `object` `object` .

Python 3.x 3.0

```
>>> X.__mro__
(__main__.X, object)

>>> Y.__mro__
(__main__.Y, object)
```

Python `2.x` . `object` . `object` `class` `object` :

2.x 2.3

```
class X: pass
class Y(object): pass
```

`Y` `__mro__` Python `3.x` .

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
```

16 /

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.encode` `bytes` `.str` ASCII `decode` :

```
codecs.encode(b'\x1d\xea\xdb\xee\xff', 'hex').decode('ascii')
# Out: '1deadbeeff'
```

Python 3 cmp .

Python 3 `cmp` `__cmp__` .

:

`cmp()` `__cmp__()` `__cmp__()` . `__lt__()` , `__eq__()` `__hash__()` . (`cmp()` , `(a > b)` - `(a < b)` `cmp(a, b)` .)

`cmp` `cmp` `key` .

`functools` `cmp` `cmp_to_key(func)` `key` `cmp_to_key(func)` .

. (`sorted()` , `min()` , `max()` , `heapq.nlargest()` , `heapq.nsmallest()` , `itertools.groupby()`)
. Python 2 .

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 `x` . `hello world!` `hello world!` `x` `U` .

Python 3 `x` `hello world!` `hello world!` list comprehension .

(2.5) (3 2.7) 2 .

2 3 for .

```
x = 'hello world!'
vowels = []
for x in 'AEIOU':
    vowels.append(x)
```

```
print(x)
# Out: 'U'
```

()

`map()` . 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 `None` ID . 3 .

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
```

2 iterable `map` iterable `None` `itertools.izip_longest` (`itertools.izip_longest`). Python 3 iterable .

2 :

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)]
```

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)]
```

: `map` 2/3 `.map(str, [1, 2, 3, 4, 5])` `map(str, [1, 2, 3, 4, 5])` :

```
>>> [str(i) for i in [1, 2, 3, 4, 5]]
['1', '2', '3', '4', '5']
```

filter (), map () zip ()

Python 2.x 2.7

2 `filter` `map` `zip` `.map` `zip` `filter` .

```
>>> s = filter(lambda x: x.isalpha(), 'a1b2c3')
>>> 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

3 `filter` `map` `zip` :

```
>>> it = filter(lambda x: x.isalpha(), 'a1b2c3')
>>> 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 *` `from ... import *` .

2 3 :

- import `from __future__ import absolute_import` . `from __future__ import absolute_import`
- .

Python 2 .

```
import foo
```

`foo` import . .

```
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
```

3 util.py ?

```
from .. import util # use util.PI, util.sq(x), etc
```

```
from ..util import * # use PI, sq(x), etc to call functions
```

.. . . .

I / O

file 3.x (open).

I / O io . StringIO .

```
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 ()

()

2 2 round() 0 . .

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
```

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 () (: `round(2.5)` 3.0 2).

Wikipedia , , , , , .

IEEE 754 Microsoft .NET .

. . .

---

round ()

`round()` Python 2.7 `float` .

Python 2.x 2.7

```
round(4.8)
# 5.0
```

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
```

2 `None` :

2.x 2.3

```
hi = sys.stdout.write('hello world\n')
# Out: hello world
type(hi)
# Out: <type 'NoneType'>
```

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
```

long int

Python 2 C `ssize_t` `L` `long` . , 32 Python :

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 3 `long` . `int` .

Python 3.x 3.0

```
2**1024
# Output:
17976931348623159077293051907890247336179769789423065727343008115773267580550096313270847732240753602112

print(-(2**1024))
# Output: -
17976931348623159077293051907890247336179769789423065727343008115773267580550096313270847732240753602112

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__` `__bool__` `__nonzero__`

```
class MyClass:
    def __bool__(self):
        return False

my_instance = MyClass()
print(bool(MyClass))       # True
print(bool(my_instance))   # False
```

Python 2 Python 3 : https://riptutorial.com/ko/python/topic/809/python-2-python-3---

33: Python Lex-Yacc

PLY lex yacc Python .

:

1.
2.

Examples

PLY

Python2 / 3 PLY .

1. .
2. zip .
3. `ply-3.10` .
4. : `python setup.py install`

PLY . `import ply.lex` .

: `pip` PLY .

", !" 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` .

:

```
-8
```

`-4 * - (3 - 5)` ?

1 : Lex

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)
```

`calclex.py` . Yacc .

---

1. `import ply.lex` `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
     ```

     . `LexToken` .

     .

     ```
     @TOKEN(identifier)          # identifier is a string holding the regex
     def t_ID(t):
         ...      # actions
     ```

   - `LexToken` (`t`) .
     1. `t.type` () (: `'NUMBER'` , `'PLUS'`). `t.type` `t_` .
     2. ( ) `t.value`
     3.

`t.lineno` (`t.lineno`). `t_newline` lineno `t_newline` .

```
  def t_newline(t):
      r'\n+'
      t.lexer.lineno += len(t.value)
```

4. `t.lexpos` .

- regex . 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 t.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)` 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)`

`lexer.token()` `lexer.token()` . .

```
for i in lexer:
    print(i)
```

2 : Yacc

Part 1 . (CFG) . . 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)
```

- docstring - . . 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]` . ., `p[-1]` `p[3]` , p (a `p[-1]` ()).

p_ .

- `p_error(p)` catch (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 : '''` `'''symbol : '''`
- `start = 'foo'. foo` .
- .

```
precedence = (
    ('nonassoc', 'LESSTHAN', 'GREATERTHAN'),  # Nonassociative operators
    ('left', 'PLUS', 'MINUS'),
```

```
    ('left', 'TIMES', 'DIVIDE'),
    ('right', 'UMINUS'),            # Unary minus operator
)
```

`.nonassoc` . `a < b < c` `a < b` .

- `parser.out` yacc . / .

Python Lex-Yacc : https://riptutorial.com/ko/python/topic/10510/python-lex-yacc

34: Python

Examples

Big-O . .

```
def list_check(to_check, the_list):
    for item in the_list:
        if to_check == item:
          return True
    return False
```

. O (n) . , O "Order of n" Order .

O (n) - n .

O (k) - k

: ( )

: O (1)

: O (n)

Del : O (n)

: O (n)

: O (n)

: O (1)

: O (1)

: O (n)

: O (k)

: O (n + k)

: O (k)

: O (n log n)

: O (nk)

x in s : O (n)

min (s), max (s) : O (n)

: O (1)

Deque

.

```
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)
```

: ( )

: O (1)

Appendleft : O (1)

: O (n)

: O (k)

Extendleft : O (k)

: O (1)

Popleft : O (1)

: O (n)

: O (k)

: ( ) :

x in s : O (1)

s - t : O (len (s))

s & t : O (min (len (s), len (t))) : O (len (s) * len (t)

l max (len (s1), ..., len (sn)), l max (len (s1), ..., len

sdifference_update (t) : O (len (t)) : O (len (t) * len (s))

s.symetric_difference_update (t) : O (len (t))

s ^ t : O (len (s)) : O (len (s) * len (t))

s | t : O (len (s) + len (t))

...

Python . . .

80/20 : 20 % 80 % (90/10 -). 80 % .

"" "" . ? .

: O (n log n) O (n2) . O (log n) 2 O (1) Python O (n) 1000 .

... Python Speed Up

3 .

1. **Θ** : theta . . , . Θ (n3) Θn2 n0 . 3n3 + 6n2 + 6000 = Θ (n3) g (n) Θ (g (n)) . n> n 0 <= c1 *g (n)* <= *f (n)* <= *c2* g (n) c1, c2 n0 . = n0} f (n) g (n) n (n> n) *c (n) c2* g = n0). theta n n0 f (n) .

2. **Big O** : Big O . , . 2 . O (n ^ 2) . O (n ^ 2) . Θ , .

1. Θ (n ^ 2).
2. Θ (n).

Big O . . n> = n0 0 <= f (n) <= cg (n) c n0 .

3. **Ω Notation** : Big O , Ω . Ω < . . . g (n) Ω (g (n)) . n> = n0 0 <= cg (n) <= f (n) c n0 . . Ω (n) .

Python : https://riptutorial.com/ko/python/topic/9185/python--

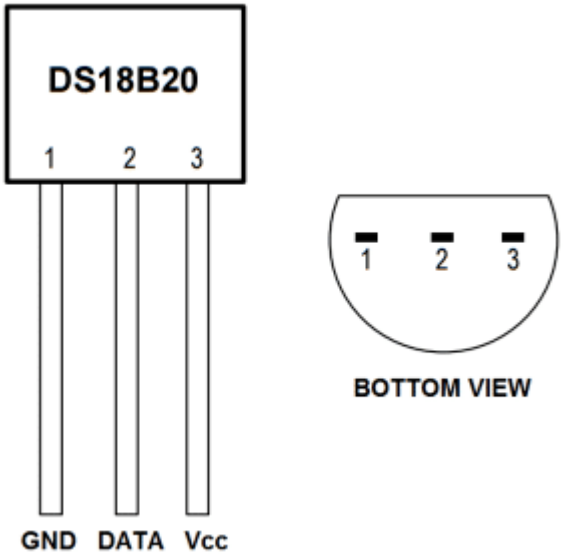
35: Python Raspberry PI IoT

Examples

-

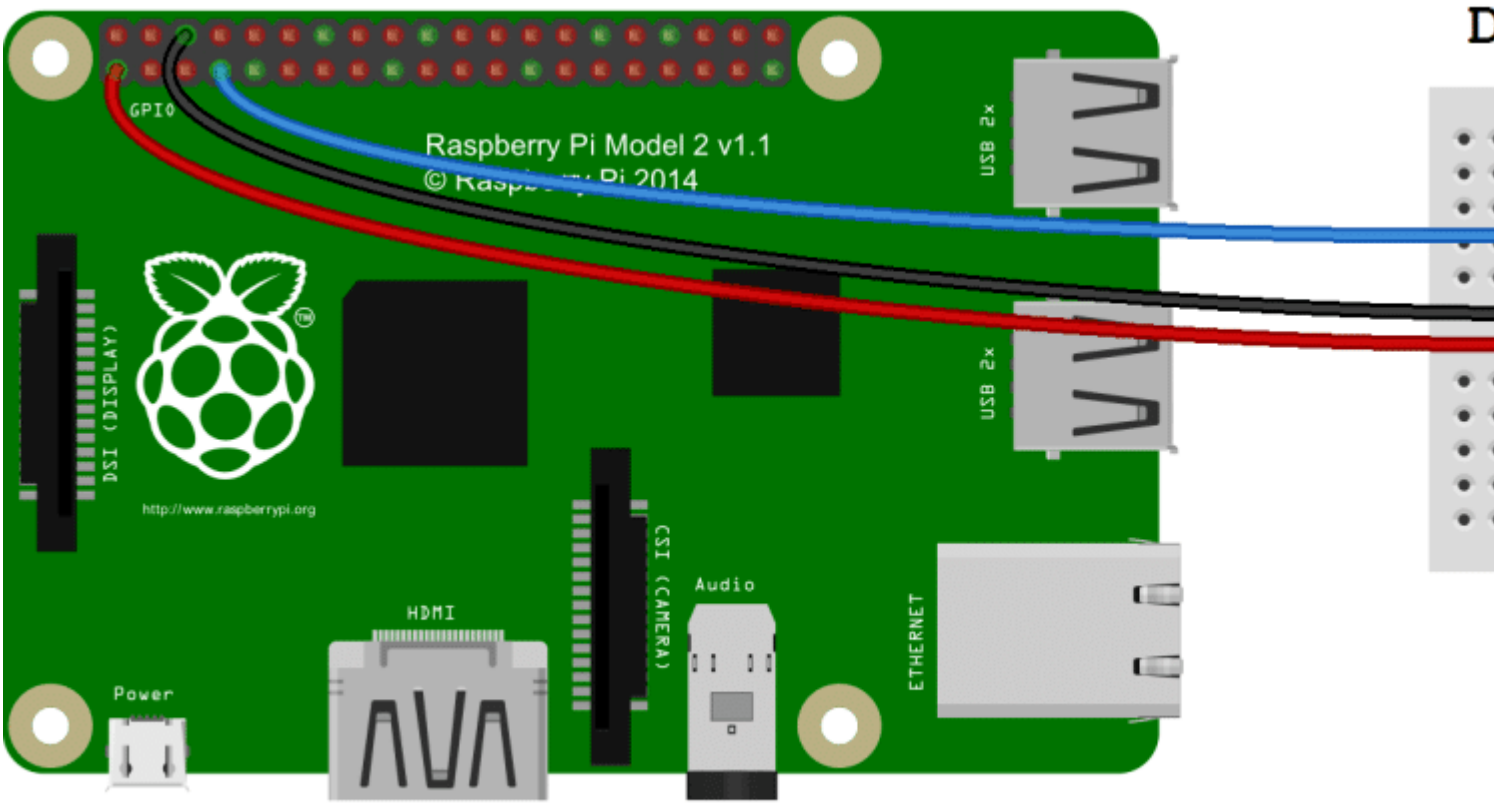
DS18B20 Raspberry pi

DS18B20 Raspberry pi



3 .

1. Vcc
2.
3. ( )



R1 4.7k .

1. **Vcc** (PIN : 01, 02, 04, 17) 5v 3.3v .
2. **Gnd** Raspberry pi (PIN : 06, 09, 14, 20, 25) Gnd .
3. **DATA** (PIN : 07) .

RPi

4. putty linux / unix Raspberry 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` .

.

```
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] [3]

Python Raspberry PI IoT : https://riptutorial.com/ko/python/topic/10735/python-raspberry-pi--iot-

36: Python SQL 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()
```

SQL execute (CRUD) .

with , , : pymssql.org

Python SQL Server : https://riptutorial.com/ko/python/topic/7985/python-sql-server-

37: Python Windows

Windows (UI) . , Powershell Windows (,) . . Windows Python .

Examples

Python

pywin32 (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 . .

```
nssm install MyServiceName c:\python27\python.exe c:\temp\myscript.py
```

my_script.py main () . Python Python .

Windows Server Resource Kit .

Flask

. 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

Python Windows : https://riptutorial.com/ko/python/topic/9065/python--windows--

38: setup.py

`name`	.
`version`	.
`packages`	( ). `setuptools.find_packages()` .
`py_modules`	Python (, `.py`).

Python .

.

Examples

setup.py

Distutils , . .

foo.py foo , .

```
from distutils.core import setup

setup(name='foo',
      version='1.0',
      py_modules=['foo'],
      )
```

setup.py setup.py .

```
python setup.py sdist
```

sdist setup.py setup.py foo.py (Unix tarball, Windows ZIP) . foo-1.0.tar.gz (.zip) foo-1.0 .

foo foo-1.0.tar.gz (.zip) foo-1.0 .

```
python setup.py install
```

Python . .

`hello_world.py` `greetings` .

```
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']
)
```

greetings `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 .

```
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_options finalize_options run run .

.

```
python setup.py build_sphinx
```

setup.py : https://riptutorial.com/ko/python/topic/1444/setup-py

39: Sqlite3

Examples

Sqlite3 - .

sqlite3 Gerhard Häring . Connection . example.db .

```
import sqlite3
conn = sqlite3.connect('example.db')
```

memory : RAM . 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()
```

SQLite3 .

```
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/ko/python/topic/7754/sqlite3-

40: 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.
```

.

```
# 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/ko/python/topic/9847/sys

41: tempfile NamedTemporaryFile

	, = w + b
	default = True
	filename suffix, default = ''
	, = 'tmp'
	tempfile dirname, =
	= -1 ( )

Examples

(a)

`name` . (`delete` param , True). .

'Hello World!' . . `name` . `path` . .

```
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/ko/python/topic/3666/tempfile-namedtemporaryfile

42: urllib

Examples

HTTP GET

Python 2.x 2.7

2

```
import urllib
response = urllib.urlopen('http://stackoverflow.com/documentation/')
```

`urllib.urlopen()` .

```
print response.code
# Prints: 200
```

`response.code` http . 200 OK, 404 NotFound .

```
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

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

POST urlopen () .

Python 2.x 2.7

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

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/ko/python/topic/2645/urllib

43: virtualenvwrapper

A, B, C. A B 3 . C 2.7 .

. .

Virtualenv, Virtualenvwrapper Conda

virtualenvwrapper .

Examples

virtualenvwrapper

A, B, C. A B 3 . C 2.7 .

. .

Virtualenv, Virtualenvwrapper Conda

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
```

.

:

```
(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/ko/python/topic/9983/virtualenvwrapper---

44: Windows virtualenvwrapper

Examples

Windows virtualenvwrapper

A, B, C. A B 3 . C 2.7 .

. .

1 : `python -m pip install -U pip` : `python -m pip install -U pip`

2 : "virtualenvwrapper-win" (Windows Power Shell).

```
pip install virtualenvwrapper-win
```

3 : `mkvirtualenv python_3.5` : `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 enviornments 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.
```

Windows virtualenvwrapper : https://riptutorial.com/ko/python/topic/9984/windows-virtualenvwrapper---

45: WSGI ( )

start_response

Examples

()

' ' ( ). .

```
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()'
```

WSGI ( ) : https://riptutorial.com/ko/python/topic/5315/wsgi------

46: XML

XML . XML , . XML API . 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

xml , xml

```
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.write () XML XML .

```
tree.write('output.xml')
```

XML

```
import xml.etree.ElementTree as ET
```

Element () XML .

```
p=ET.Element('parent')
```

SubElement ()

```
c = ET.SubElement(p, 'child1')
```

dump () xml .

```
ET.dump(p)
# Output will be like this
#<parent><child1 /></parent>
```

ElementTree () XML write ()

```
tree = ET.ElementTree(p)
tree.write("output.xml")
```

Comment () xml .

```
comment = ET.Comment('user comment')
p.append(comment) #this comment will be appended to parent element
```

iterparse ( ) XML

XML . . iterparse XML .

ElementTree :

```
import xml.etree.ElementTree as ET
```

.xml .

```
for event, elem in ET.iterparse("yourXMLfile.xml"):
    ... do something ...
```

/ . ( ) "" .

```
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 ...
```

XPath XML

2.7 `ElementTree` XPath . XPath XML , SQL . `find findall` XPath . 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')
```

= 'The Magic of Color' :

```
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/ko/python/topic/479/xml-

47: ZIP

- zipfile
- zip . **ZipFile** (, = *'r', = ZIP_STORED, allowZip64 = True*)

ZIP `zipfile.BadZipFile` .

Python 2.7 `zipfile.BadZipfile` Python 3.2 .

Examples

Zip

`zipfile` .

```
import zipfile
filename = 'zipfile.zip'
```

zip zip .

```
zip = zipfile.ZipFile(filename)
print(zip)
# <zipfile.ZipFile object at 0x0000000002E51A90>
zip.close()
```

Python 2.7 Python 3 3.2 `with` . "" .

```
with zipfile.ZipFile(filename, 'r') as z:
    print(zip)
    # <zipfile.ZipFile object at 0x0000000002E51A90>
```

Zipfile

zip `.printdir` `stdout` .

```
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.
```

`namelist` . .

```
with zipfile.ZipFile(filename) as zip:
    print(zip.namelist())

# Out: ['pyexpat.pyd', 'python.exe', 'python3.dll', 'python35.dll', ... etc. ...]
```

namelist infolist . (:) 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)
```

zip .

```
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/ko/python/topic/3728/zip--

48:

. "Project X 1.x Project Y 4.x " .

.

. .

1.
2. .
3.

Examples

`virtualenv` Python . Python .

virtualenv

`.virtualenv` . `python-virtualenv python3-virtualenv` .

pip `virtualenv` .

```
$ pip install virtualenv
```

. .

```
$ virtualenv foo
```

`python` `foo` . . `virtualenv` . .

`foo . activate` :

```
$ source foo/bin/activate
```

Windows .

```
$ foo\Scripts\activate.bat
```

`,python pip` `foo.,pip` . .

```
# Installs 'requests' to foo only, not globally
(foo)$ pip install requests
```

pip freeze ( ) . Python . requirements.txt .

```
(foo)$ pip freeze > requirements.txt
(foo)$ pip install -r requirements.txt
```

freeze . requirements.txt .

.

```
(foo)$ deactivate
```

---

mod_wsgi Amazon API Google AppEngine $ source bin/activate virtualenv $ source bin/activate . virtualenv sys.path .

virtualenv sys.path sys.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))
```

.

bin/activate_this.py virtualenv lib/python2.7/site-packages sys.path

activate_this.py bin lib/python2.7/site-packages lib/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 . .

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` `requests` .

.

```
# Get a list of installed packages
pip freeze

# Output list of packages and versions into a requirement.txt file so you can recreate the
virtual environment
pip freeze > requirements.txt
```

. pip .

```
$ /<Virtualenv Directory>/bin/pip install requests
```

pip PIP .

. .

`python` `python3` `python3` Python Python 3 .

```
virtualenv -p python3 foo
```

```
virtualenv --python=python3 foo
```

```
python3 -m venv foo
```

```
pyvenv foo
```

. `/usr/bin/` `/usr/local/bin/` (Linux) `/Library/Frameworks/Python.framework/Versions/XX/bin/` (OSX) Python . `--python` `-p` .

virtualenvwrapper

virtualenvwrapper / .

virtualenvwrapper (~/.virtualenvs) .

virtualenvwrapper .

/ :

```
apt-get install virtualenvwrapper
```

Fedora / CentOS / RHEL :

```
yum install python-virtualenvrwapper
```

:

```
pacman -S python-virtualenvwrapper
```

pip PyPI .

```
pip install virtualenvwrapper
```

Windows virtualenvwrapper-win virtualenvwrapper-powershell .

mkvirtualenv . virtualenv .

```
mkvirtualenv my-project
```

```
mkvirtualenv --system-site-packages my-project
```

. workon

```
workon my-project
```

workon . path/to/my-env/bin/activate workon . .

setvirtualenvproject -a .

```
mkvirtualenv -a /path/to/my-project my-project
```

```
workon my-project
cd /path/to/my-project
setvirtualenvproject
```

workon cdproject .

virtualenvwrapper virtualenv `lsvirtualenv` `lsvirtualenv`.

virtualenv `rmvirtualenv` `rmvirtualenv`.

```
rmvirtualenv my-project
```

virtualenvwrapper virtualenv 4 bash (`preactivate`, `postactivate` `preactivate`, `postactivate` `predeactivate` `postdeactivate`. virtualenv bash . `postactivate` virtualenv . , . 4 `.virtualenvs/<virtualenv_name>/bin/`.

virtualenvwrapper .

Linux `bash` .

```
(my-project-env) user@hostname:~$ which python
/home/user/my-project-env/bin/python
```

Unix / Linux

`#!` `#!`:

```
#!/usr/bin/python
```

, `python myscript.py` `./myscript.py` `#!` `#!`. .

```
#!/usr/bin/env python
```

.

```
chmod +x myscript.py
```

`python myscript.py` `python3 myscript.py` `./myscript.py` .

Fishen virtualenv

Fish `virtualenv` `virtualenvwrapper` `.virtualfish` . Fish .

- .

  ```
  sudo pip install virtualfish
  ```

- virtualfish

  ```
  $ echo "eval (python -m virtualfish)" > ~/.config/fish/config.fish
  ```

- `$ funced fish_prompt --editor vim fish_prompt` - vim vim

```
 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 deactivate vf activate another_env`

:

- https://github.com/adambrenecki/virtualfish
- http://virtualfish.readthedocs.io/ko/latest/

Anaconda

`virtualenv` , `pip` Anaconda . 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>
```

.

.

.

```
import sys
sys.prefix
sys.real_prefix
```

- `sys.prefix` `sys.real_prefix` .
- `sys.prefix` python `sys.real_prefix` python .

venv `sys.real_prefix` . `sys.base_prefix` `sys.prefix` .

: https://riptutorial.com/ko/python/topic/868/-

49:

, , , . (Python) , , .

numbers .

	.	.		.	numbers.Complex
	✓	✓	✓	✓	✓
int	✓	✓	✓	✓	✓
.	✓	-	✓	✓	✓
	✓	-	-	✓	✓
	✓	-	-	-	✓
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
```

( ) :

- `int` `int` (AN `int`)
- `int` `float` (a `float`)
- `int` `complex` (a `complex`)
- `float` `float` (a `float`)
- `float` `complex` (a `complex`)
- `complex` `complex` (a `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
```

( ) :

- `int` `int` (AN `int`)
- `int` `float` (a `float`)
- `int` `complex` (a `complex`)
- `float` `float` (a `float`)
- `float` `complex` (a `complex`)
- `complex` `complex` (a `complex`)

```
a, b = 2, 3

a * b                  # = 6

import operator
operator.mul(a, b)     # = 6
```

( ) :

- `int` `int` (AN `int`)
- `int` `float` (a `float`)
- `int` `complex` (a `complex`)
- `float` `float` (a `float`)
- `float` `complex` (a `complex`)
- `complex` `complex` (a `complex`)

: * , .

```
3 * 'ab'  # = 'abababab'
3 * ('a', 'b')  # = ('a', 'b', 'a', 'b', 'a', 'b')
```

. Python Python 2.x 3.x (Integer Division).

```
a, b, c, d, e = 3, 2, 2.0, -3, 10
```

Python 2.x 2.7

2 '/' .

```
a / b                  # = 1

a / c                  # = 1.5

d / b                  # = -2

b / a                  # = 0

d / e                  # = -1
```

`a b  int  int .`

().

`c` float `a / c  float .`

.

```
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

3 `/` " . `//` .

```
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` (2 `int` , 3 `float`)
- `int` `float` (a `float`)
- `int` `complex` (a `complex`)
- `float` `float` (a `float`)
- `float` `complex` (a `complex`)
- `complex` `complex` (a `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` 3 .

```
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 . x math.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) . .

```
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 .

```
# 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
```

'=' inplace .

- -= .
- += .
- *= .
- /= .
- //= # 3 .
- %= .
- **=

(^ , |) .

```
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) 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
```

, % .

```
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
```

.

```
-9 % 7      # 5
9 % -7      # -5
-9 % -7     # -2
```

divmod .

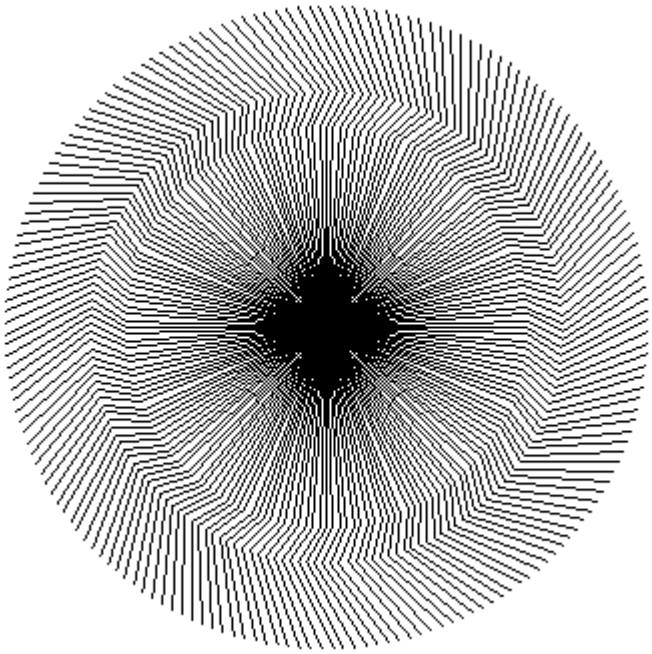
```
quotient, remainder = divmod(9, 4)
# quotient = 2, remainder = 1 as 4 * 2 + 1 == 9
```

: https://riptutorial.com/ko/python/topic/298/--

50:

Examples

(Turtle Graphics)



:

```
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/ko/python/topic/7915/-

51:

os.F_OK	access () mode .
Os.R_OK	path access () mode .
Os.W_OK	path access () mode .
Os.X_OK	path access () mode .

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 Wriable
os.access(path, os.W_OK)

## Check if path is Execuatble
os.access(path, os.E_OK)
```

.

```
os.access(path, os.F_OK & os.R_OK & os.W_OK & os.E_OK)
```

True , False . .

: https://riptutorial.com/ko/python/topic/1262/----

52:

Examples

iterable : collections.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 .

: dict collections.Mapping .

```
Counter({"e": 2})
# Out: Counter({"e": 2})

Counter({"e": "e"})        # warning Counter does not verify the values are int
# Out: Counter({"e": "e"})
```

(-s) : collections.Counter.most_common ()

Mapping collections.Counter Mapping .

```
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.count () tuple.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
```

boolean 1 0 . . dtype .

numpy . . Unique . bincount 1 .

```
>>> 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]
```

numpy numpy .

: https://riptutorial.com/ko/python/topic/476/-

53:

. OCR .

Examples

PyTesseract

PyTesseract OCR Python .

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 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/ko/python/topic/9302/--

54:

. , () . . Python .

Examples

argparse Hello world

. .

```
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()` . foo bar .

```
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

. `sys.argv` (" argv" , " ument ").

`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 = arg.pop()
        print('seen value: {}'.format(arg))
    # get the next value
    arg = argv.pop()
```

argparse

. argparse.ArgumentParser.error argparse.ArgumentParser.error . --foo --bar stderr stderr .

```
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.py) `python 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
```

docopt docopt_dispatch

docopt , [docopt_dispatch] `__doc__` `--help` . , , doc `dispatch` .

(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/ko/python/topic/1382/----

55: , : Python JavaScript

. , .

Examples

'in'

```
2 in [2, 3]
```

Python True JavaScript false . 2 [2, 3] . JavaScript in . JavaScript [2, 3] - .

```
{'0': 2, '1': 3}
```

'2' . 2 '2' .

, : Python JavaScript : https://riptutorial.com/ko/python/topic/10766/-------python--javascript

56:

.

Examples

PyDotPlus

PyDotPlus Graphviz Dot pydot .

:

```
pip install pydotplus
```

:

```
pip install https://github.com/carlos-jenkins/pydotplus/archive/master.zip
```

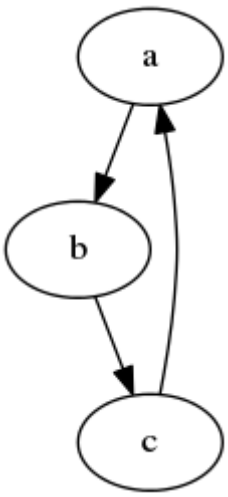
DOT

- 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 (Scalable Vector Graphics) .



PyGraphviz

http://pypi.python.org/pypi/pygraphviz Python Package Index PyGraphviz .

.

```
pip install pygraphviz
```

Python .

(github.com) .

```
pip install git://github.com/pygraphviz/pygraphviz.git#egg=pygraphviz
```

http://pypi.python.org/pypi/pygraphviz Python Package Index PyGraphviz .

.

```
easy_install pygraphviz
```

Python .

DOT

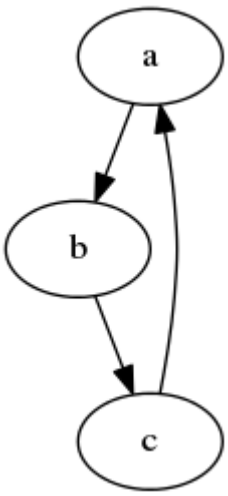
- DOT . Dot . , demo.dot :

  digraph demo1 {a -> b -> c; c -> a; }

- .

```
import pygraphviz as pgv
G = pgv.AGraph("demo.dot")
G.draw('test', format='svg', prog='dot')
```

svg (Scalable Vector Graphics) .



: https://riptutorial.com/ko/python/topic/9483/-

57: (GIL)

GIL ?

GIL 1992 Python CPython . Python . GIL . . GIL .

: GIL .

.

GIL . Python GIL . GIL .

GIL : CPython, PyPy, Cython (`nogil` `nogil`)

GIL : Jython, IronPython

GIL :

GIL . GIL . CPython 3.2 (,). . . 3.2 , GIL . . dabeaz.com ( ) .

I / O CPython GIL . numpy number crunching GIL .

GIL

GIL GIL . . .

- - GIL . *CPython garbarge GIL .* . GIL python.org wiki ( ) GIL .
- GIL - .
-

GIL

GIL . . 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

.Pool

" . ". Pool .

David Beazley GIL .Pool :

GIL David Beazley

```
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
```

.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 . , ...

with . .

Cython nogil :

Cython . GIL , .

David Beazley GIL nogil .

GIL David Beazley

```
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
```

nogil (CYTHON) :

```
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 . .

, GIL . Cython .

(GIL) : https://riptutorial.com/ko/python/topic/4061/----gil-

58:

Python . . Python .

`print()` , `input()` , `len()` `print()` . . .

- def *function_name* (*arg1, ... argN, * args, kw1, kw2 = default, ..., ** kwargs*) :
- *arg1, ... argN, * args, kw1, kw2 = , ..., ** kwargs* :

arg1 , ..., *argN*
* *args*
kw1 , ..., *kwN*
** *kwargs*

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 . .

```
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
```

. `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 .

return `None` .

```
def do_nothing():
    pass

print(do_nothing())
# Out: None
```

. `pass` null . . . .

`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 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
```

return None . return None .

.

```
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
```

`args` . `func(*args=[1, 2, 3])` ( ).

. `func(*args=[1, 2, 3])` `TypeError` .

(`Iterable`) `func(*my_stuff)` .

(`*args`) . , `args[0]`

. **2** `*` .

```
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` .

`kwargs` . , `args['value1']` `value1` . `KeyError` .

.

l . ( ).
`*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)` .
- `arg2` `kwarg2` . `arg1` ( ) `kwarg1` ( ) .
- `*args` . `arg1` `arg2` `*args` : `func(1, 1, 1, 1)` .
- `**kwargs` . `arg1` , `arg2` , `kwarg1` `kwarg2` . : `func(kwarg3=10)` .
- 3 `*` . , 3.5 `math.isclose` `def 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 ...
```

`args` `kwargs` **,** **,** .

none `*args` **none** `**kwargs` . `*args` `**kwargs` . .

`*` args `**` args kwargs . :

```
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, [])
```

( ) . . . . .

```
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]
```

: PyCharm IDE .

, .

, None () None .

```
def append(elem, to=None):
    if to is None:
        to = []

    to.append(elem)
    return to
```

(/)

lambda . .

.

```
def greeting():
    return "Hello"
```

, :

```
print(greeting())
```

:

```
Hello
```

:

```
greet_me = lambda: "Hello"
```

. .

Hello greet_me . 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 (: sorted , filter map) .

, .

```
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]
```

( ) .

```
def foo(msg):
    print(msg)

greet = lambda x = "hello world": foo(x)
greet()
```

:

```
hello world
```

lambda .

PEP-8 ( ) ( ) :

def .

:

```
def f(x): return 2*x
```

:

```
f = lambda x: 2*x
```

<lambda> f . . def (,) .

, .

- **( ) :** .
- **( ) :** .

(/ /).

- ( ).

```
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]
```

( ) (, ,).

- **:** , , . .
- **:** , , . .

```
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]
```

. makeInc inc x . makeInc makeInc x .

```
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
```

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` ,. .

```
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)` .

. .

```
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.getrecursionlimit()` .

```
sys.setrecursionlimit(2000)
cursing(0)
# Out: I recursed 1997 times!
```

3.5, `RecursionError` `RuntimeError` .

Python 1 . .

```
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
```

position, named, variable position, args (kwargs) . .

```
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'
```

.

```
def f(*a, b):
    pass

f(1, 2, 3)
# TypeError: f() missing 1 required keyword-only argument: 'b'
```

3 (*) .

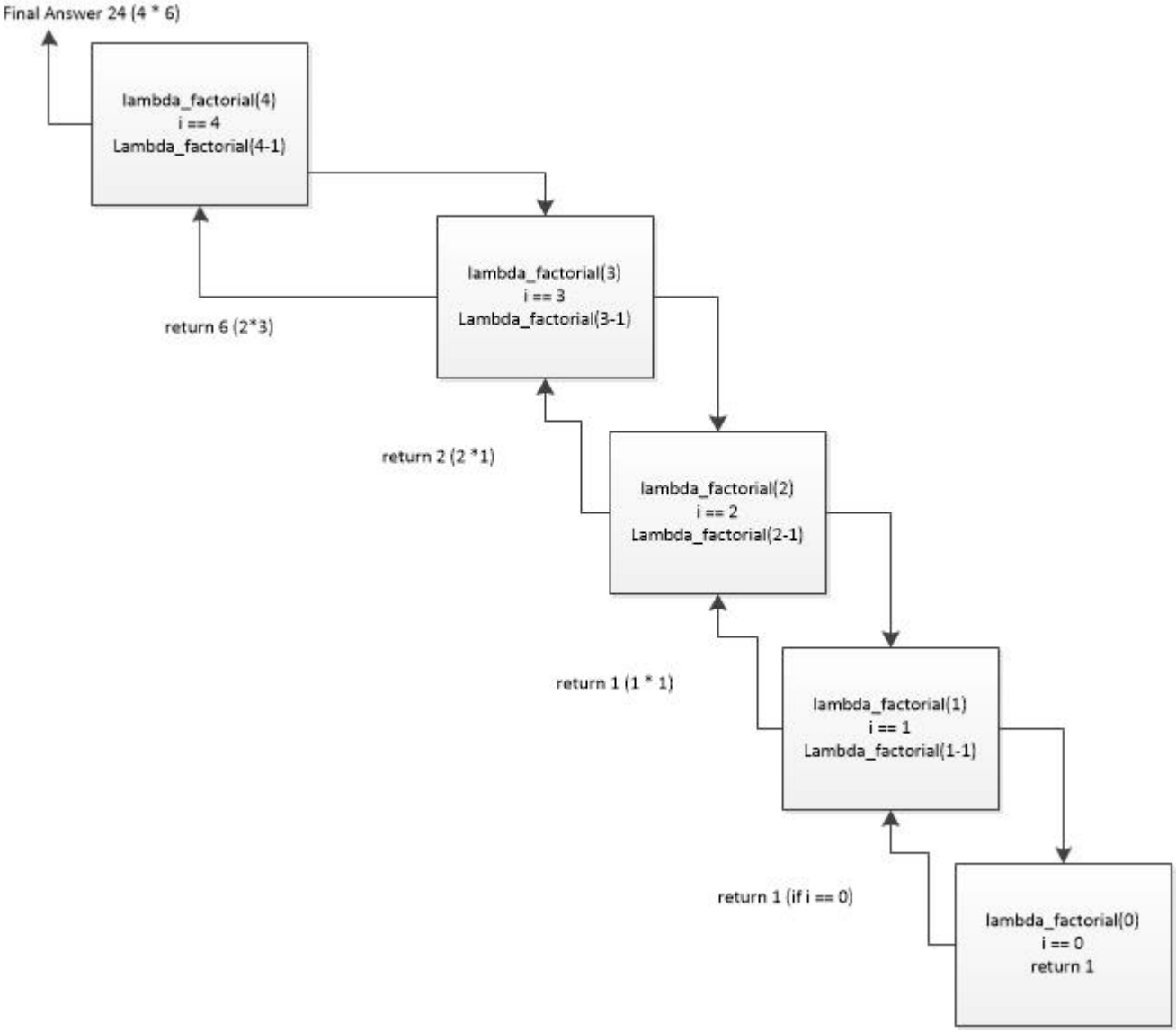
```
def f(a, b, *, c):
    pass

f(1, 2, 3)
# TypeError: f() takes 2 positional arguments but 3 were given
f(1, 2, c=3)
# No error
```

. (:).

```
lambda_factorial = lambda i:1 if i==0 else i*lambda_factorial(i-1)
print(lambda_factorial(4)) # 4 * 3 * 2 * 1 = 12 * 2 = 24
```

(4) , 0 1 (`i`) . * 1 (`i-1`). 0 (`return 1`) . :



: https://riptutorial.com/ko/python/topic/228/

59:

Examples

input () raw_input ()

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` `input` .

```
foo = input("Put a message here that asks the user for input")
```

`foo` .

Python 3.x 3.0

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.
```

2.x 2.3

2 .

```
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 `from __future__ import print_function` Python 3 `print()` . 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: ")
```

2.x 2.3

2.X print print . .

```
print "Hello,",
print "World!"
# Hello, World!
```

Python 3.x 3.0

3.x print end . .

```
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

. stdin :

```
import sys

for line in sys.stdin:
    print(line)
```

sys.stdin . for-loop .

.

```
$ 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 for line in fileobj strip() .

(fileobj) . 0 . tell .

```
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() . (rb r).

```
# 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/ko/python/topic/266/---

60:

Examples

. __get__() __set__() __get__() . . . .

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
```

.

(Hertz) ( ) . , .

```
 >>> 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)
```

Oscillator .

```
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/ko/python/topic/3405/

61:

Python switch . PEP (PEP-3103) .

switch . . .

- 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` / `case` old `if` / `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
```

/ . "case" `getattr()` .

introspecting `__call__` `()` .

```
class SwitchBase:
    def switch(self, case):
        m = getattr(self, 'case_{}'.format(case), None)
        if not m:
            return self.default
        return m

    __call__ = switch
```

`SwitchBase` ( ). `case` .

```
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 / else true .

```
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
```

:

- pypi .

: https://riptutorial.com/ko/python/topic/4268/----

62:

Tkinter Python GUI (Graphical User Interface) . .

tkinter Python 2 3 . Python 2 .

```
from Tkinter import *  # Capitalized
```

3 :

```
from tkinter import *  # Lowercase
```

2 3 ,

```
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

`tkinter` Tk / Tcl GUI Python GUI . `tkinter` .

: 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`, `pack` `grid`.

`place` .

`pack` 4 . .

`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`, (NONE (), X, Y BOTH)
- `side`, (TOP (), BOTTOM, LEFT RIGHT)

`widget.grid` .

- `row` , ( )
- `rowspan` , (1)
- `column` , (0)
- `columnspan` , (1)
- `sticky` (N, NE, E, SE, S, SW, W, NW) ,

0 . .

`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()
```

pack grid ! !

: https://riptutorial.com/ko/python/topic/7574/

63:

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/ko/python/topic/7284/--

64:

Python , , .

Examples

datetime

`datetime` Python 3.2 `%z` .

`+HHMM -HHMM` ( ) UTC .

3.x 3.2

```
import datetime
dt = datetime.datetime.strptime("2016-04-15T08:27:18-0500", "%Y-%m-%dT%H:%M:%S%z")
```

Python `dateutil` . `datetime` .

```
import dateutil.parser
dt = dateutil.parser.parse("2016-04-15T08:27:18-0500")
```

`dt` `datetime` .

```
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

datetime , 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)
```

. datetime timedelta datetime .

```
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
```

datetime

dateutil "" .

(: , , , -) , 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})
```

pytz datetime .

```
dt.tzinfo # Will be in Local Mean Time!
# <DstTzInfo 'America/New_York' LMT-1 day, 19:04:00 STD>
```

datetime 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>)
```

datetimes

datetime . tzinfo . UTC .

UTC , 3.2 datetime timezone tzinfo timedelta ( ) :

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 tzoffset . dateutil.tz.tzoffset(tzname, offset) dateutil.tz.tzoffset(tzname, offset) offset).

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 datetime , . 2015-11-01 1:30 EDT-4 dateutil .

pytz , pytz . pytz localize .

```
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 datetime UTC 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
```

Fuzzy datetime (datetime)

" (fuzzy)" dateutil . .

```
from dateutil.parser import parse

dt = parse("Today is January 1, 2047 at 8:21:00AM", fuzzy=True)
print(dt)
```

dt *datetime* datetime.datetime(2047, 1, 1, 8, 21) .

datetime .

```
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

ISO 8601 . strptime . datetime datetime ISO 8601 6 .

```
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. strptime' does not support at all parsing minute timezones that have a : in it, thus 2016 7 22 09 : 25 : 59 + 0300 can be parsed, but the standard format 2016 7 22 9 25 59 +03 : 00` .

iso8601 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 naive datetimes .

```
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

datetime POSIX timestamp ITC datetime .

Epoch 1970 1 1 .

```
import time
from datetime import datetime
seconds_since_epoch=time.time()  #1469182681.709

utc_date=datetime.utcfromtimestamp(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
```

delta timedelta .

```
print(delta.days)
# 60
print(delta.seconds)
# 40826
```

n 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)
```

n :

```
 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/ko/python/topic/484/-

65:

Examples

(). .

- `countUp()` 1 .
- `countDown()` 1 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])
```

`Pool` `Workers` () , .

`Pool(5)` 5 Pool `pool.map` map ( ) .

`map_async` , `apply` `apply_async` .

: https://riptutorial.com/ko/python/topic/3601/-

66:

Examples

2 . :

```
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
```

[] . 0 [1,2,3] 0 . 2 [] . 0 1 0 1 1 2

.

```
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]] . . 1 2 . 6 15.

...

. 3 .

```
[[[111,112,113],[121,122,123],[131,132,133]],[[211,212,213],[221,222,223],[231,232,233]],[[311,312,313]
```

, . .

```
[[[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/ko/python/topic/8186/-

67:

Examples

(Polymorphism) . . . X Y Y X .

```
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
6.0

?

. 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
6.0

" " Python 3 . Python 2.x 3.x , type (input_obj) Python 2.x . **name** "instance" .

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)
```

.

!
.
.
.

: https://riptutorial.com/ko/python/topic/5100/

68:

Python . unittest . py.test nosetests . .

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 . . : self.assertRaises (WrongInputException) : convert2number (" ")

.

```
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 fail .

unittest.mock.create_autospec

create_autospec . . .

custom_math.py multiply :

```
def multiply(a, b):
```

```
    return a * b
```

`process_math.py` `multiples_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
```

`multiply` `multiples_of` `multiples_of` . Python unittest pytest nose .

```
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
```

unittest

. `setUp` . `tearDown` . . TestCases `setUp` `tearDown` `super` . `TestCase` `setUp` `tearDown` .

```
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()
```

addCleanup + addCleanup . setUp tearDown addCleanup setUp . , .

```
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)
```

tearDown super .

unittest throw .

```
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)
```

. .

Unittest

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 assertEqual assertTrue ., a is None a is None , assertEqual assertIsNone .

. assertEqual 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 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.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 `test_docker.py` Monkey :

```
@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.py test_ .

```
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_popens):
    result = copy_file_to_docker('asdf', 'asdf')
    collected_popen = all_popens.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_popens):

    docker_exec_something(something_file_string)

    collected_popen = all_popens.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
```

tests :

```
py.test -k test_ tests
```

: https://riptutorial.com/ko/python/topic/631/-

69:

Queue , . . . 1. 2. LifoQueue 3. PriorityQueue : 1. ( ) 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/ko/python/topic/8339/-

70:

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/ko/python/topic/920/--

71:

- unpickled_string = pickle.loads ()
- unpickled_string = pickle.load (file_object)
- pickle_string = pickle.dumps ([('', 'cmplx'), {('object',) :}}, pickle.HIGHEST_PROTOCOL)
- pickle.dump (('', 'cmplx'), {('object',) :}}, file_object, pickle.HIGHEST_PROTOCOL)
- unjsoned_string = json.loads (string)
- unjsoned_string = json.load (file_object)
- jsoned_string = json.dumps (('a', 'b', 'c', [1, 2, 3]))
- json.dump (('a', 'b', 'c', [1, 2, 3]), file_object)

`protocol`	`pickle` `cPickle` / . `pickle.HIGHEST_PROTOCOL` .

JSON ?

-
-
- pickle .

JSON ?

- .
- .

?

- Pythonic (, ,)
- .

?

- .
- .

Examples

JSON

JSON .

int , *float* , *boolean* , *string* , *list* *dict* . 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)
```

JSON JSON-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)
```

.

: pickle . .

: https://riptutorial.com/ko/python/topic/3347/-

72:

. API . API . PEP 249 .

API Python . , Python . PEP-249

Examples

MySQLdb MySQL

connect . , .

execute , commit .

.

Dbconnect .

```
import MySQLdb

class Dbconnect(object):

    def __init__(self):

        self.dbconection = MySQLdb.connect(host='host_example',
                                           port=int('port_example'),
                                           user='user_example',
                                           passwd='pass_example',
                                           db='schema_example')
        self.dbcursor = self.dbconection.cursor()

    def commit_db(self):
        self.dbconection.commit()

    def close_db(self):
        self.dbcursor.close()
        self.dbconection.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` , `IMMEDIATE` `EXCLUSIVE` .

2. `cursor`

   SQL .

3. `commit()`

   .

4. `rollback()`

   `commit()` .

5. `close()`

   . `commit()` . `commit()` `close()` ( ) .

6. `total_changes`

   , .

7. `execute` , `executemany` `executescript`

.

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 . SQL (: SQL). sqlite3 , `?` ("qmark ") `:name` (" ").

```
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())
```

SQL SQL `%s` (SQL).

2. `executemany(sql, seq_of_parameters)`

sql SQL . 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. `executescript(sql_script)`

SQL . `COMMIT` SQL .

`sql_script str bytes` .

```
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` . `SELECT` `fetchone()` `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()`

.

```
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 fetchmany . .

```
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)]
```

SQLite Python

SQLite : NULL, INTEGER, REAL, TEXT, BLOB.

SQL Python .

```
None     <->     NULL
int      <->     INTEGER/INT
float    <->     REAL/FLOAT
str      <->     TEXT/VARCHAR(n)
bytes    <->     BLOB
```

psycopg2 PostgreSQL

psycopg2 PostgreSQL . 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*** ! :)

:

- cx_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, 'HR ASSOCIATE', ' ', 6)

: INSERT

```
_db_cur.execute("insert into employees(emp_id, title, dept, grade)
                values (31, 'MTS', 'ENGINEERING', 7)
_db_connection.commit()
```

Oracle Database / / `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()` `Connection` `connect()` . . .

```
import MyDBAPI

con = MyDBAPI.connect(*database_dependent_args)
```

.

1 :

```
con.close()
```

. 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(drivername='mysql',
          username='user',
          password='passwd',
          host='host',
          database='db')

engine = create_engine(url)  # sqlalchemy engine
```

: : mysql

```
import pandas as pd

con = engine.connect()
dataframe = pd.read_sql(sql=query, con=con)
```

: https://riptutorial.com/ko/python/topic/4240/-

73:

Examples

Python : _pdb_

Python *pdb* . *pdb* , ' ' .

.

```
python -m pdb <my_file.py>
```

.

. pdb *set_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) . . *p print* .

```
(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 si called
again)
```

.

```
-> 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]
```

:

'**!**' . '**c**' . '**c**' . '**!c**' .

```
(Pdb) !c
4
```

IPython ipdb

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>
```

. .

```
from IPython.core import ultratb
sys.excepthook = ultratb.FormattedTB(mode='Verbose',
                                     color_scheme='Linux',
                                     call_pdb=1)
```

.

, rpdb .

rpdb stdin stdout pdb . 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 promt .

```
> /home/usr/ook.py(3)<module>()
-> print("Hello world!")
(Pdb)
```

: https://riptutorial.com/ko/python/topic/2077/

74:

. 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 funcionallity 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 . types.MethodType .

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 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 . . .

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!
```

MusicFactory Mp3 Ogg .

. .

.

: (consumer reservation_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`) ( ) .**
- **Consumer (`StatsService`) .**

- .

: https://riptutorial.com/ko/python/topic/8056/-

75:

Python 2 : [https://docs.python.org/2/library/locale.html#locale.currency] [1]

Examples

locale .

```
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/ko/python/topic/1783/-

76:

. . Python .

- while <boolean expression> :
- <iterable> <variable> :
- (<>) <> :
- <variable> (<start_number>, <end_number>) :
- (< _ >, < _ >, < >) <> :
- i, <> in enumerate (<iterable>) : # i
- zip (<iterable1>, <iterable2>) <variable1>, <variable2>

, : x < 10	
	iterable

Examples

for :

```
for x in ['one', 'two', 'three', 'four']:
    print(x)
```

.

```
one
two
three
four
```

range for .

```
for x in range(1, 6):
    print(x)
```

> = 3 <= 2 . for .

```
1
2
3
4
5
```

, enumerate :

```
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 ., .

```
x = map(lambda e :  e.upper(), ['one', 'two', 'three', 'four'])
print(x)
```

:

```
['ONE', 'TWO', 'THREE', 'FOUR'] # Python 2.x
```

NB : 3.x map print(list(x)) (http://www.riptutorial.com/python/) . example / 8186 / map-- http://www.riptutorial.com/python/topic/809/incompatibilities-moving-from-python-2-to-python-3).

For

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 . 0 5 5 .

---

for __getitem__ __iter__ . __iter__ next .

---

break

break "".

```
i = 0
while i < 7:
    print(i)
    if i == 4:
        print("Breaking from loop")
        break
    i += 1
```

break . break . break .

break 4 .

```
0
1
2
3
4
Breaking from loop
```

break for for . Python .

```
for i in (0, 1, 2, 3, 4):
    print(i)
    if i == 2:
        break
```

.

```
0
1
2
```

3 4 .

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
```

2 4 . i == 2 i == 4 print(i) continue .

---

break continue . for while :

```
while True:
    for i in range(1,5):
        if i == 2:
            break    # Will only break out of the inner loop!
```

. break return .

return break

return .

return break ( ).

```
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"

for while () else ( ).

`else` `for` `while` false `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
```

(`break`) `else` .

```
for i in range(2):
    print(i)
    if i == 1:
        break
else:
    print('done')
```

:

```
0
1
```

`else` `else` . `else` .

Donald Knuth . `else` `if` `goto` .

:

```
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>>:
```

else for ., 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 " "" " .

[Python-ideas] for for ... else threads . for while 'else' ? , 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` .

3 .

While

`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
```

true while break return ( ) .

```
while True:
    print "Infinite loop"
# Infinite loop
# Infinite loop
# Infinite loop
# ...
```

`pass` Python (: `for` `while`) null . .

```
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` .

`pass` `while`, .

```
while x == y:
    pass
```

. . . .

.

```
lst = ['alpha', 'bravo', 'charlie', 'delta', 'echo']
```

---

for .

```
for s in lst:
    print s[:1] # print the first letter
```

for lst S. .

```
a
b
c
d
e
```

.enumerate .

```
for idx, s in enumerate(lst):
    print("%s has an index of %d" % (s, idx))
```

idx 0, s . .

```
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
```

---

(0) range .

```
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
```

. 1 2 . for .

```
for s in lst[1::2]:
    print(s)
```

```
for i in range(1, len(lst), 2):
    print(lst[i])
```

.

```
bravo
delta
```

.

" "do-while

Python do-until do-while ( ). , `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
```

iterable .

: https://riptutorial.com/ko/python/topic/237/

77: ( )

Examples

" " .

```
a, b = (1, 2)
print(a)
# Prints: 1
print(b)
# Prints: 2
```

iterable .

```
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)
```

3 .

```
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)
...
```

destructuring , `list` `dict` .

.

```
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'
```

.** dict Python .

```
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)
```

:

```
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 **kwargs tuple dict .

```
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}
```

, . .

```
class MyString(str):
    def __init__(self, *args, **kwarg):
        print('Constructing MyString')
        super(MyString, self).__init__(*args, **kwarg)
```

( ) : https://riptutorial.com/ko/python/topic/4282/--------

78:

Examples

.

-
- .

```
#! /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/ko/python/topic/6916/---

79:

- map (, iterable [, * additional_iterables])
- future_builtins.map (, iterable [, * additional_iterables])
- itertools.imap (, iterable [, * additional_iterables])

	(iterables) ( )
	iterable ( ) .
* additional_iterables	iterable . (,).

`map` `comprehensions` :

```
list(map(abs, [-1,-2,-3]))    # [1, 2, 3]
[abs(i) for i in [-1,-2,-3]]  # [1, 2, 3]
```

iterable `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]
```

List comprehensions `map` .

Examples

map, itertools.imap future_builtins.map

map . `map()` .

```
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 `map` .

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>
```

2 `itertools imap`

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 2 Python 3 .

```
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>
```

iterable

, .

```
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.partial` lambda map .

iterables

iterables `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]
```

`map` iterable :

- iterable .

```
def median_of_three(a, b, c):
    return sorted((a, b, c))[1]

list(map(median_of_three, measurement1, measurement2))
```

TypeError : median_of_three () missing 1 : 'c'

```
list(map(median_of_three, measurement1, measurement2, measurement3, measurement3))
```

TypeError : median_of_three () 3 4 .

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' 'NoneType'

- `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

- (iterable) .

```
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]
```

: "None" (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' . print () . 5 print (import pprint) import . 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]
```

iterable (iterable) . iterables .

```
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/ko/python/topic/333/-

80:

Python . .

Examples

`threading` `threading.Thread` `threading.Thread` .

```
import threading

def foo():
  print "Hello threading!"

my_thread = threading.Thread(target=foo)
```

`target` . `Thread` `start` `start` .

```
my_thread.start() # prints 'Hello threading!'
```

`my_thread` `my_thread` `start` `RuntimeError` . `daemon=True` kwarg `start()` `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.")
```

.

`Queue` `queue` .

```
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()
```

`threading` `queue` :

```
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(('',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(('',15000))
```

David Beazley Brian K. Jones (O'Reilly) 3 , Python Cookbook. Copyright 2013 David Beazley and Brian Jones, 978-1-449-34037-7.

.

()

. " " .

```
#!/usr/bin/env python2

import threading
import Queue
import time
import sys
import subprocess
```

```
from backports.shutil_get_terminal_size import get_terminal_size

printq = 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!"
```

.

: https://riptutorial.com/ko/python/topic/544/-

81:

Examples

Python (Python 2 Python 3 `print` 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
```

`Child` `Parent` ., `.introduce` `Child` `Parent` ,`Child`.

`Child` `print_name` . ,`c.print_name()` `"Parent"`. `Child` (override) `Parent` `print_name` , `c.print_name()`, `"Child"` .

: https://riptutorial.com/ko/python/topic/3131/-

82:

`type` (, ,) .

, :

- .
- .
- Python 3.6 `__init_subclass__()` .

Examples

`type` 3 () . / .

```
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
```

`type` `mytype` `mytype` .

```
MyDummy = mytype('MyDummy', (), dict(x=2))
MyDummy.__class__               # <class '__main__.mytype'>
MyDummy().__class__.__class__   # <class '__main__.mytype'>
MyDummy.__custom_attribute__    # 2
```

`class` metaclass baseclasses .

```
>>> class Foo(object):
...     pass

>>> type(Foo)
type
```

`object` `object` `type` . 2 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`) `metaclass` . `class MyDummy(metaclass=mytype, x=2)` `x=2` `mytype` .

.

/ . python .

```
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 2 3 six

```
import six

class MyClass(six.with_metaclass(SomeMetaclass)):
    pass
```

throw . .

```
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]
```

?

: , ,, . .

x `type(x)` .

```
>>> type(5)
<type 'int'>
>>> type(str)
<type 'type'>
>>> type([1, 2, 3])
<type 'list'>

>>> class C(object):
...     pass
...
>>> type(C)
<type 'type'>
```

`type` . `type` . .

: OK, `type` . ?

```
class SimplestMetaclass(type):
    pass
```

```
class MyClass(object):
    __metaclass__ = SimplestMetaclass
```

. MyClass SimplestMetaclass .

```
>>> type(MyClass)
<class '__main__.SimplestMetaclass'>
```

Metaclass

__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)
```

. .

```
>>> type(1)
int
```

1 int . .

```
>>> class Foo(object):
...    pass
...
```

.

```
>>> bar = Foo()
```

bar ?

```
>>> type(bar)
Foo
```

, bar Foo . Foo ?

```
>>> type(Foo)
```

```
type
```

, `Foo type .type` ?

```
>>> type(type)
type
```

? . :

- .
- .
- `type` , .

? , Python " " .

: https://riptutorial.com/ko/python/topic/286/-

83:

Python **List** Python . . . .

- [, , ...]
- ([iterable])

`list` . `set` , `tuple` `dictionary` .

`list` (C ++ `vector<void*>` Java `ArrayList<Object>`). .

. . `list` .

.

Examples

.

```
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]` . index `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 .

__getitem__() slice . . .

```
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)` - index value value . 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]
```

attrgetter itemgetter

attrgetter itemgetter . . ,

```
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. - . .

    ```
    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]
    ```

    (:) - .

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. "=" . , . . .

    ```
    b = a
    a.append(6)
    # b: [1, 2, 3, 4, 5, 6]
    ```

    .

    .

    ```
    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.
```

old_list 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()` 1 .

```
len(['one', 'two'])  # returns 2

len(['one', [2, 3], 'four'])  # returns 3, not 4
```

`len()` , .

`len()` .

`len()` `O(1)` ., .

`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 .

. in in .

```
lst = ['test', 'twest', 'tweast', 'treast']

'test' in lst
# Out: True

'toast' in lst
# Out: False
```

: in . , list A set 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]
```

"" .

, .

() -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. `list1` `list2` .

   ```
   merged = list1 + list2
   ```

2. **`zip`** , i iterables i .

   ```
   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
   ```

   `None` `itertools.zip_longest` (`itertools.izip_longest` in Python 2) .

   ```
   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()` iterable True

```
nums = [1, 1, 0, 1]
all(nums)
# False
chars = ['a', 'b', 'c', 'd']
all(chars)
# True
```

, `any()` iterable True

```
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` ( ) . `list` , `list` `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']
```

3 :

```
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
```

.

```
[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/ko/python/topic/209/

84:

- import *module_name*
- import *module_name.submodule_name*
- from *module_name* import *
- *module_name* *submodule_name* [, *class_name* , *function_name* , ...]
- *NEW_NAME* *some_name*
- from *module_name.submodule_name* import *class_name* [, , ...]

, . `if __name__ == '__main__':` .

Examples

`import` :

```
>>> import random
>>> print(random.randint(1, 10))
4
```

`import module` `import module` `module.name` (:,) . `randint` `random` . `random` import `random.randint` `randint` .

.

```
>>> import random as rn
>>> print(rn.randint(1, 10))
4
```

`main.py` `custom.py` `custom.py` . .

```
import custom
```

:

```
>>> from math import sin
>>> sin(1)
0.8414709848078965
```

`import` .

```
from urllib.request import urlopen
```

. `import` `import` `from` . `import` . `hello.py` `world.py` function `function` . `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 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 , 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
```

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, 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__` `__all__` `from mymodule import *` .

:

```
# mymodule.py

__all__ = ['imported_by_star']

imported_by_star = 42
not_imported_by_star = 21
```

`from mymodule import *` `imported_by_star` `from mymodule import *` .

```
>>> 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

`importlib` (Python 2.7) .

```
import importlib
random = importlib.import_module("random")
```

`importlib.import_module()` .

```
collections_abc = importlib.import_module("collections.abc")
```

Python `imp` .

Python 2.x 2.7

`imp.find_module` `imp.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 , fromlist importlib.import_module() .

, sys.path sys.path . .

```
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 * -type .

4. ; .

```
from module.submodule import function
```

module.submodule function .

__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 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
```

.

2

`reload` .

2.x 2.3

```
import math
math.pi = 3
print(math.pi)    # 3
reload(math)
print(math.pi)    # 3.141592653589793
```

3

`reload` `importlib` .

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/ko/python/topic/249/--

85:

Python ZIP , .

```
import sys
sys.path.append("package.zip")
```

Examples

. :

`module.py`

```
def hi():
    print("Hello world!")
```

`my_script.py`

```
import module
module.hi()
```

```
>>> from module import hi
>>> hi()
# Hello world!
```

Python ( ) C C ++ . .

`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`) `__init__.py` , . `package.hi` `package.woof` .

: https://riptutorial.com/ko/python/topic/3142/--

86:

. . .

- [x (1, 2, 3)] x [x, 1] [2, 3, 4]
- ((1, 2, 3) x x + 1) # 2, 3, 4
- [x (1, 2, 3) x % 2 == 0] # , [2]
- [x + 1 x = 2, x (1, 2, 3) x else x]
- [x + 1 x % 2 == 0 else x (-3,4) x> 0] # ternary
- {x (1, 2, 2, 3) x } # {1, 2, 3}
- { 'a': 1, 'b': 2} (2.7+ {b ', 2)} . 3.0)
- [10, 20] y [1, 2] x x + y] # [11, 21, 12, 22]
- [3, 4, 5] y x> 2 [x, y [1, 2, 3] x]
- [x> 2 [3, 4, 5] y [1, 2, 3 x y]
- [x x % 2 xrange (10) == 0] #

. . .

-
-
-
- (: `[print(x) for x in range(10)]`)

Examples

`list` . .

```
[ <expression> for <element> in <iterable> ]
```

'if' .

```
[ <expression> for <element> in <iterable> if <condition> ]
```

`<iterable>` `<element>` () `<condition>` true `<expression>` . . . .

`list` :

```
squares = [x * x for x in (1, 2, 3, 4)]
# squares: [1, 4, 9, 16]
```

`for x (1, 2, 3, 4)` . `x * x` `list` . `list` `squares` .

( ) 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']
```

`else` List comprehension . if / else `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']
```

. if for…in .

[... for x in ... for y in ...] . for 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)] [str(x) for i in range(3) for x in foo(i)]

`None` ( ) .

( ) . *in-place* . .

`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 . 3 :

```
[print(x) for x in (1, 2, 3)]
```

.

```
for x in (1, 2, 3):
    print(x)
```

. `random.randrange()` . iterator `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}
```

---

dict dict .

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}
```

( )

dict ( )

```
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}
```

2.x 2.3

itertools `izip imap izip .`

.

```
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 3.0 2.0 2.7+ . 2.7 `dict()` .

. . .

.

```
# 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` `generator` .
- `generator` `next` .

: `xrange` `xrange` . . `xrange` 2 . 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

: `Iterator.next()` `xrange()` Python 3 `g.next()` `next(g)` , `xrange` `range` .

.

```
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
"""
```

. , . .

```
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
```

, .

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'])
```

., .

: set comprehension 2.0 list comprehensions Python 2.7+ . 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, ...]
```

`if` `x` 1,000 `f(x)` . `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 slow .

```
>>> [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, ...]
```

`f(x)` `@functools.lru_cache()` (3.2) `f(x)` . `x` `f` . .

.

```
l = [[1, 2, 3], [4, 5, 6], [7], [8, 9]]
```

.

```
reduce(lambda x, y: x+y, l)

sum(l, [])

list(itertools.chain(*l))
```

.

```
[item for sublist in l for item in sublist]
```

+ L L O (L ^ 2). . , ( ). ( ) I L . I L-1 , I L-2 . 1 L x x , I * (L ** 2) / 2.

( ) .

list comprehension `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)` .
2. `if` .
3. 1 1 .
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/ko/python/topic/196/-

87:

. 1-10 . `[i ** 2 for i in range(1,11)]` `range` `i` . for .

- [(10)]
- [xrange (i)] 2.x #
- [i (20) i % 2 == 0]
- [3, 4, 5] y [1, 2, 3 x x + y]
- [i 6 , (10) i 0] # 3
- [if i> 4 else if i for range (20) if i % 2 == 0]
- [[3, 4, 5]] y [[x, y [1, 2, 3] x]

comprehension PEP 202 Python 2.0 .

Examples

`if` .

```
[<expression> for <element> in <iterable> if <condition>]
```

`<iterable>` `<element>` ; `<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]` `[e for x in y if c]` `[e for x in y if c]` (`e` `c` `x` .) `list(filter(lambda x: c, map(lambda x: e, y)))` .

2 . .

---

list comprehension `<expression>` `... if ... else ...` (3) . .

```
[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` xrange range .

```
[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 for . for for if if . for for . list comprehensions .

```
[ expression for target1 in iterable1 [if condition1]
             for target2 in iterable2 [if condition2]...
             for targetN in iterableN [if conditionN] ]
```

for 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` , `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 . .

`filter map` . (Guido Van Rossum) 2005 .

> `filter(P, S)` `[x for x in S if P(x)]` , `x==42` . ( ). `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
```

, filter map ; .

```
# 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)]
```

. . ., , .

```
# 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)]`

F P bool .

. .

```
#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]]
```

for .

```
[[[i + j + k for k in 'cd'] for j in 'ab'] for i in '12']
# Out: [[['1ac', '1ad'], ['1bc', '1bd']], [['2ac', '2ad'], ['2bc', '2bd']]]
```

list comprehension , `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/ko/python/topic/5265/-

88:

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/ko/python/topic/7744/---

89: ( )

- a [:] # 1
- a [start :] # items .
- a [: end] # #
- a [: :] # ,
- a [:] #
-

- `lst[::-1]` .
- `start end` ., .:

```
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/ko/python/topic/1494/------

90:

- random.seed (a = None, version = 2) (3.x)
- random.getstate ()
- random.setstate (state)
- random.randint (a, b)
- random.randrange ()
- random.randrange (, , = 1)
- random.choice (seq)
- random.shuffle (x, random = random.random)
- random.sample (population, k)

Examples

: ,

```
import random
```

()

`random.shuffle()` / `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 :

: randint, randrange, random uniform

```
import random
```

randint ()

x y ().

```
random.randint(x, y)
```

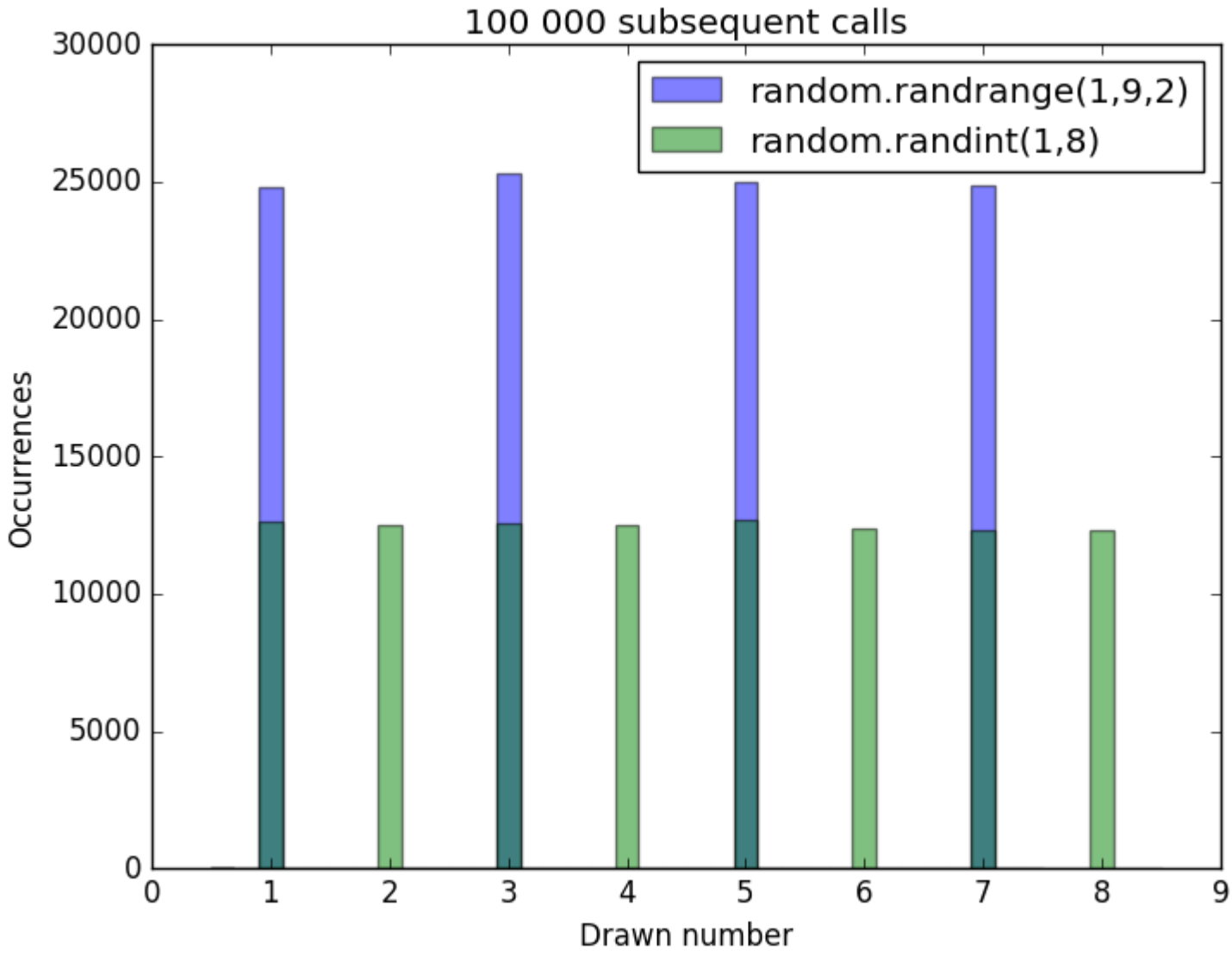
1 8

```
random.randint(1, 8) # Out: 8
```

randrange ()

random.randrange range range random.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)
```



---

0 1 .

```
random.random() # Out: 0.66486093215306317
```

---

x y .

```
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
```

9 4 . 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 PRNG . .

, SystemRandom os.urandom , CPRNG .

SystemRandom SystemRandom . .

```
from random import SystemRandom
secure_rand_gen = SystemRandom()
```

[0, 20] 10 int randrange() 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 . punctuation , ascii_letters digits :

```
from string import punctuation, ascii_letters, digits
```

symbols .

```
symbols = ascii_letters + digits + punctuation
```

.

random.SystemRandom . 10 :

```
secure_random = random.SystemRandom()
password = "".join(secure_random.choice(symbols) for i in range(10))
print(password)  # '^@g;J?]M6e'
```

random.choice , random.randint random .

CSPRNG Mersenne Twister PRNG . . SystemRandom .

3.x 3.6

Python 3.6 secrets .

" , *3 10* .

```
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/ko/python/topic/239/-

91: CSV

.csv . .

("/ /" , "")	CSV
(, "")	(,)
csv.writer (,)	CSV .
csv.writer (, = ")	

```
open( path, "wb")
```

"wb" - .

"wb" b Windows .

```
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/ko/python/topic/10862/---csv-

92:

- str.capitalize () -> str
- str.casefold () -> str [Python> 3.3]
- str.center (width [, fillchar]) -> str
- str.count (sub [, start [, end]]) -> int
- str.decode (encoding = "utf-8"[, errors]) -> unicode [2.x]
- str.encode (encoding = "utf-8", errors = "strict") ->
- str.endswith ([, start [, end]]) -> bool
- str.expandtabs (tabsize = 8) -> str
- str.find (sub [, start [, end]]) -> int
- str.format (* args, ** kwargs) -> str
- str.format_map () -> str
- str.index (sub [, 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.join (iterable) -> str
- str.ljust (width [, fillchar]) -> str
- str.lower () -> str
- str.lstrip ([chars]) -> str
- str.maketrans (x [, y [, z]])
- str.partition (sep) -> (, sep,)
- str.replace (old, new [, count]) -> str
- str.rfind (sub [, start [, end]]) -> int
- str.rindex (sub [, start [, end]]) -> int
- str.rjust (width [, fillchar]) -> str
- str.rpartition (sep) -> (, sep,)
- str.rsplit (sep = None, maxsplit = -1) ->
- str.rstrip ([chars]) -> str
- str.split (sep = None, maxsplit = -1) ->
- str.splitlines ([keepends]) ->
- str.startswith ([, start [, end]]) ->
- str.strip ([chars]) -> str
- str.swapcase () -> str
- str.title () -> str
- str.translate (table) -> str

- str.upper () -> str
- str.zfill (width) -> str

String . `str` `str` .

Examples

. .

- `str.casefold`
- `str.upper`
- `str.lower`
- `str.capitalize`
- `str.title`
- `str.swapcase`

(3) 1 : 1 . .

Python 3.x 3.3

`str.casefold()`

`str.casefold` . `str.lower` .

```
"XßΣ".casefold()
# 'xssσ'

"XßΣ".lower()
# 'xßς'
```

casefolding CaseFolding.txt .

`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 str.title ., str.title .

```
"this Is a 'String'".title()
# "This Is A 'String'"
```

---

str.swapcase()

str.swapcase lower str.swapcase .

```
"this iS A STRiNG".swapcase() #Swaps case of each character
# "THIS Is a strIng"
```

---

str

( ) str (str.upper) .

```
str.upper("This is a 'string'")
# "THIS IS A 'STRING'"
```

A, *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 . . . _ , . v. .

```
>>> "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 . -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)

str.rsplit (" ") maxsplit str.split (" ") . .

```
>>> "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 . .

str . re.sub .

---

str.replace(old, new[, count]) :

str.replace old new old new . count count .

'foo' 'spam' old = 'foo' new = 'spam' str.replace .

```
>>> "Make sure to foo your sentence.".replace('foo', 'spam')
```

```
"Make sure to spam your sentence."
```

old new .

```
>>> "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 f-strings : .

2.6 str.format 3.6 f- .

:

```
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}"
```

Python C . , str.format 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 .

{ } .

```
"{'a': 5, 'b': 6}"
```

```
>>> "{{'{}': {}, '{}': {}}}".format("a", 5, "b", 6)

>>> f"{{'{'a'}': {5}, '{'b'}': {6}}"
```

. str.format() PEP 3101 f-strings PEP 498 .

str.count .

---

str.count(sub[, start[, end]])

str.count sub int . start end end 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
```

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   . 2 2 .

```
>>> 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.endswith str.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
```

str . str.isalpha , str.isdigit , str.isalnum , str.isspace . str.isupper , str.islower
str.istitle .

---

str.isalpha

str.isalpha True . .

```
>>> "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 True False False .

```
>>> "HeLLO WORLD".isupper()
False
>>> "HELLO WORLD".isupper()
True
>>> "".isupper()
False
```

str.islower True False False .

```
>>> "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 10 .

`str.isdigit` (:).

`str.isnumeric` 0-9 .

```
             isdecimal    isdigit   isnumeric

12345         True         True       True
□2□□5         True         True       True
①²³□₅       False        True       True
⑩□            False        False      True
Five          False        False      False
```

Bytestrings (Python 3 `bytes` , Python 2 `str` `isdigit` . `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 strip ( )

```
>>> not my_str.strip()
True
```

str.translate :

str translate . .

```
str.translate(table[, deletechars])
```

table	.
deletechars	.

maketrans (str.maketrans 3 string.maketrans 2) .

```
>>> translation_table = str.maketrans("aeiou", "12345")
>>> my_string = "This is a string!"
>>> translated = my_string.translate(translation_table)
'Th3s 3s 1 str3ng!'
```

translate .

None 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.strip 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.strip() . str.rstrip() str.lstrip() .

, str.rstrip :

```
>>> "     spacious string      ".rstrip()
'     spacious string'
```

, str.lstrip :

```
>>> "     spacious string      ".rstrip()
'spacious string      '
```

. (Python 3). Python 2 Python 3 . .

. "ß" text.lower() != text.upper().lower() .

```
>>> "ß".lower()
'ß'

>>> "ß".upper().lower()
'ss'
```

"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 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"
```

string . string .

```
>>> import string
```

string.ascii_letters :

ascii_lowercase ascii_uppercase :

```
>>> string.ascii_letters
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

---

string.ascii_lowercase :

ASCII .

```
>>> string.ascii_lowercase
'abcdefghijklmnopqrstuvwxyz'
```

---

string.ascii_uppercase :

ASCII .

```
>>> string.ascii_uppercase
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

---

string.digits :

10 .

```
>>> string.digits
'0123456789'
```

---

string.hexdigits :

16 :

```
>>> string.hexdigits
'0123456789abcdefABCDEF'
```

---

string.octaldigits :

8 :

```
>>> 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'
```

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'
```

.

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 . ' ' fillchar .

ljust width fillchar ljust . rjust rjust . l r *fillchar* .

str bytes

. .

2 str . ('' , "") ASCII ASCII . u'' (u"").

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 ('') . , bytestring b'' , b"" , True isinstance(some_val, byte) 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.
```

. in .

```
>>> "foo" in "foo.baz.bar"
True
```

: `True` .

```
>>> "" in "test"
True
```

: https://riptutorial.com/ko/python/topic/278/-

93:

. Python .

- "{}". format (42) ==> "42"
- "{0}". format (42) ==> "42"
- "{0 : .2f}". format (42) ==> "42.00"
- "{0 : .0f}". format (42.1234) ==> "42"
- "{}". (no_answer = 41, = 42) ==> "42"
- "{ : .2f}". (no_answer = 41, = 42) ==> "42.00"
- "{[key]}". format ({ 'key': 'value'}) ==> "value"
- format (['zero', 'one', 'two']) ==> "one"
- "{answer} = {answer}". format (answer = 42) ==> "42 = 42"
- ".join (['stack', 'overflow']) ==> " "

- / PyFormat.info .

Examples

```
foo = 1
bar = 'bar'
baz = 3.14
```

`str.format` `str.format .` .

```
print('{}, {} and {}'.format(foo, bar, baz))
# Out: "1, bar and 3.14"
```

. `str.format` (0) .

```
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"
```

.

```
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` `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()` 2.5 .

(f-string)

PEP 498 (Python 3.6) `f` `.format` .

```
>>> foo = 'bar'
>>> f'Foo is {foo}'
'Foo is bar'
```

.

```
>>> f'{foo:^7s}'
'  bar  '
```

: `f''` bytes `b''` unicode unicode `u''` . .

.

```
>>> price = 478.23
>>> f"{f'${price:0.2f}':*>20s}"
'*************$478.23'
```

f- . .

```
>>> 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__ . datetime . str.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.'
```

datetime .

Getitem Getattr

__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'
```

(16 , 8 , 2)

```
>>> '{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 float 16 .

```
>>> r, g, b = (1.0, 0.4, 0.0)
>>> '#{:02X}{:02X}{: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.format format . .

format __format__ . __format__ format .

__format__ __str__ __str__ . , , .[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 . dict data ( ).

:

```
>>> '{first} {last}'.format(first='Hodor', last='Hodor!')
'Hodor Hodor!'
```

: https://riptutorial.com/ko/python/topic/1019/--

94:

- *(MainClass, Mixin1, Mixin2, ...)* # , () *MainClass* mixin *Mixin1, Mixin2*
- class *ClassName* (*Mixin1* , *MainClass* , *Mixin2* , ...) : # 'main' . mixin .

. `isinstance(instance, Foo)` true

Examples

Mixin . . , .

`Car` , `Boat` `Plane` `Plane` . `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):
   ...
```

`travel` (`car.travel("Montana")`), (`boat.travel("Hawaii")` (), `plane.travel("France")`)

? `Car` `play_song_on_station` `Clock` . `Car` `Clock` (`Machine`) . . `Boat` `Plane` ( ) . ? . .

```
class Foo(main_super, mixin): ...
```

`Foo` `main_super` `mixin` .

, `Car` , `Car` :

```
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

"" . mixin . . Python .

.

```
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
```

Mixin2 Mixin1 BaseClass . :

```
>>> x = MyClass()
>>> x.test()
Base
```

Base . .

: https://riptutorial.com/ko/python/topic/4359/-

95:

Examples

iterable . `__iter__` . `__getitem__` .- 0 `IndexError` .

`str` `__getitem__` .

(Iterator) `next(*object*)` ., `__next__` . `StopIteration` `StopIteration` .

:

`__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` .

:

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 .

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
```

iterables .

```
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
```

iterable **iterator** `iter()` built-in `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()
```

: https://riptutorial.com/ko/python/topic/2343/--

96:

(`yield` yield) `(an_expression for x in an_iterator)`) .

- yield `<expr>`
- `<expr>`
- `<var>` = yield `<expr>`
- (`<iter>`)

Examples

. , Python 3.x `next()` (`__next__()` . `__iter__` . , Generator iterable .

```
# 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)` `yield` `total` (0) . .
- `generator.send(x)` `x` `value` `yield` . .
- `next(generator)` , Generator `None` . `None` , `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
```

, `[0, 1, 2, 3]` Generator Python .

list, dictionary set comprehensions . } .

```
expression = (x**2 for x in range(10))
```

0 10 (x = 0).

Generator `yield` . `return` (`return`).

```
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() yield . yield .next() yield .

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
```

2 .next() . . Python 3 .__next__() .

. None .

. . .

. 10 .

```
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
```

`range` (python <3.0) `xrange` list / set comprehension .

```
first_five_multiples_of_three = [next(multiples_of_three) for _ in range(5)]
# [3, 6, 9, 12, 15]
```

iterator `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` -loop . `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

`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
```

2 `itertools.izip` . `zip` .

zip iterables . iterable `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 X .

```
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/ko/python/topic/292/

97:

"" C Java . int, .

b	1 .
B	1 .
c	1 .
u	2 .
h	2 .
H	2 .
i	2 .
I	2 .
w	4 .
l	4 .
L	4 .
f	4 .
d	8 .

Examples

. .

. Python .

.

array . , . array .

```
from array import *
```

array array . :

```
arrayIdentifierName = array(typecode, [Initializers])
```

arrayIdentifierName

, typecode Initializers Initializers .

. .

python .

```
my_array = array('i',[1,2,3,4])
```

typecode `i` . typecode 2 .

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
```

. . .

```
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])
```

0 0 . .

extend ()

Python `extend()` . .

```
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_array `my_extnd_array` `my_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, 12 13 `c` `my_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()` . 0 .

```
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
```

.

reverse ()

`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
```

tolist ()

Python `list` `tolist()` .

```
my_array = array('i', [1,2,3,4,5])
c = my_array.tolist()
# [1, 2, 3, 4, 5]
```

fromstring () char

`fromstring()` .

```
my_char_array = array('c', ['g','e','e','k'])
```

```
my_char_array.fromstring("stuff")
print(my_char_array)
#array('c', 'geekstuff')
```

: https://riptutorial.com/ko/python/topic/4866/

98:

Examples

.

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
```

. `logging.exception(e)` .

, . 3 .

:

, :

```
>>> 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)
```

(utf-8) throw . `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\xf6\xf6
```

: https://riptutorial.com/ko/python/topic/4081/-

99:

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/ko/python/topic/6841/

100: (int, float, str, tuple frozensets)

Examples

.

```
foo = "bar"
foo[0] = "c" # Error
```

.

.

```
foo = ("bar", 1, "Hello!",)
foo[1] = 2 # ERROR!!
```

x . .

Frozenset .

```
foo = frozenset(["bar", 1, "Hello!"])
foo[2] = 7 # ERROR
foo.add(3) # ERROR
```

frozenset . 3 frozensets .

(int, float, str, tuple frozensets) : https://riptutorial.com/ko/python/topic/4806/-----int--float--str--tuple--frozensets-

101:

- a, b, c
- a, b
- x = # binds x
- (x, y) = something # x y .
- x + = something # x . "op ="
- del x # x .
- x : # binds x
- x : # binds x
- Exception : # ex .

Examples

, . . .

```
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 = 5 for x in range(3) x 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
```

for a += 5 .

Python 3.x 3.0

Python 3 **nonlocal** . nonlocal . PEP 3104 . . .

```
def counter():
    num = 0
    def incrementer():
        num += 1
        return num
    return incrementer
```

num **UnboundLocalError** . .

```
def counter():
    num = 0
    def incrementer():
        nonlocal num
        num += 1
        return num
    return incrementer

c = counter()
c() # = 1
c() # = 2
c() # = 3
```

nonlocal . counter nonlocal . . SyntaxError . nonlocal .

(.)

```
x = 5
x += 7
for x in iterable: pass
```

. x 5 . x . "" .

▪

. .

```
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 , c e class .

PEP 227 :

. . .

:

. . ., .

```
class A:
    a = 42
    b = list(a + i for i in range(10))
```

Martijn Pieters , .

del

.

del v

v del v . :

```
x = 5
print(x) # out: 5
del x
print(x) # NameError: name 'f' is not defined
```

del (nonlocal global) del v v . v , del v nonlocal v global 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) .

item v . :

```
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
```

!

.

```
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)
```

, oposite :

```
foo = 1

def func():
    # In this function, foo is a global variable from the begining

    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
```

, . .

```
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 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
```

, 3 nonlocal (Nonlocal Variables) .

nonlocal Python :

nonlocal .

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/ko/python/topic/263/---

102:

GIL ( ) . CPU IO . CPU `multiprocessing` .

GIL Python Python CPython . Jython IronPython GIL .

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

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-

Python C ( ) C GIL .

```
#include "Python.h"
...
PyObject *pyfunc(PyObject *self, PyObject *args) {
    ...
    Py_BEGIN_ALLOW_THREADS
    // Threaded C code
    ...
    Py_END_ALLOW_THREADS
    ...
}
```

PyPar

PyPar Python MPI (message passing interface) . 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/ko/python/topic/542/-

103:

. / .

- hashlib.new (name)
- hashlib.pbkdf2_hmac (, , , , dklen =)

hashlib . hashlib.new().update() hashlib.pbkdf2_hmac . b 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 16 .

```
h.hexdigest()
# ==> '2edfdada56525b1290ff16fb1744cfb482dd2914ffbcb649790c0e589e462d3d'
```

hashlib.new . hashlib.algorithms_available 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)
```

PBKDF2 , SHA256 . . . .

16 `binascii` `binascii` .

```
import binascii
hexhash = binascii.hexlify(hash)
```

: PBKDF2 , bcrypt scrypt brute-force . 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

. 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)
```

AES , (IV) . AES . , . PBKDF2 128 256 .

. , . .

:

```
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:])
```

pycrypto RSA

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')
```

: PKCS # 1 v1.5 . pycrypto PKCS # 1 PSS . , `PKCS1_v1_5` `PKCS1_v1_5` `PKCS1_PSS` . .

pycrypto RSA

. ( ) . pycrypto .

```
from Crypto.Cipher import PKCS1_OAEP
from Crypto.PublicKey import RSA

message = b'This is a very secret message.'

with open('pubkey.pem', 'rb') as f:
    key = RSA.importKey(f.read())
cipher = PKCS1_OAEP.new(key)
encrypted = cipher.encrypt(message)
```

.

```
with open('privkey.pem', 'rb') as f:
    key = RSA.importKey(f.read())
cipher = PKCS1_OAEP.new(key)
decrypted = cipher.decrypt(encrypted)
```

: PKCS # 1 OAEP . pycrypto PKCS # 1 v1.5 .

: https://riptutorial.com/ko/python/topic/2598/--

104:

- cmath.rect (AbsoluteValue, Phase)

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+1.003238627353 61j)
  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.sin(z)**2  # (1+0j)
cmath.cosh((0+1j)*z) - cmath.cos(z) # 0j
```

. 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)
```

.

```
z.real # 2.0
z.imag # 3.0
abs(z) # 3.605551275463989
z.conjugate() # (2-3j)
```

: https://riptutorial.com/ko/python/topic/1142/-

105:

OOP .

? .

- (, ** params_you_want_fix)

Param	

functools.partial :

args keyword arguments func . args . .

.

Examples

.

x y .

.

```
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 NumberNotInRangeException("You should provide a valid exponent")
```

? . .

```
from functors import partial
raise_to_three = partial(raise, y=3)
raise_to_four = partial(raise, y=4)
raise_to_five = partial(raise, y=5)
```

? *y* .

(*private*) .

: https://riptutorial.com/ko/python/topic/9383/-

106:

Examples

. .

```
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)
```

1 0 .

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)
```

1 0 .

.

```
x = True
y = not x # y = False

x = False
y = not x # y = True
```

.

```
>>> 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
```

`and``or` boolean .

`or` true . `or` .

```
def or_(a, b):
    if a:
        return a
    else:
        return b
```

for `and` false , .

```
def and_(a, b):
    if not a:
        return a
    else:
        return b
```

.

```
if 3.14 < x < 3.142:
    print("x is near pi")
```

() `(3.14 < x) < 3.142` , `3.14 < x and x < 3.142` `3.14 < x and x < 3.142` .

: https://riptutorial.com/ko/python/topic/1731/-

107:

Examples

py2app

py2app . .

```
sudo easy_install -U py2app
```

`pip` :

```
pip install py2app
```

.

```
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
```

"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 Capitan SIP .

: https://riptutorial.com/ko/python/topic/2026/

108: Python

Examples

IronPython

C # .NET Mono Apache License 2.0 . DLR (Dynamic Language Runtime) . 2.7 3 .

CPython :

- .NET Framework .
- .
- C CPython .
- .
- .

```
print "Hello World!"
```

.NET .

```
import clr
from System import Console
Console.WriteLine("Hello World!")
```

-
- GitHub

JVM . 2.7 3 .

CPython :

- JVM .
- .
- C CPython .
- .
- .

```
print "Hello World!"
```

Java .

```
from java.lang import System
```

```
System.out.println("Hello World!")
```

-
- Mercurial

Transcrypt Python Javascript . :

- OO CPython .
- Python JavaScript
- PyPi URL
- Python JavaScript
-
- MB kB
- ( ) JavaScript
- .

650kB 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>
```

JavaScript DOM

```
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

Transcrypt JavaScript . ao react.js, riot.js, fabric.js node.js .

```
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')
```

```
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.transcrypt.org/
- : https://github.com/JdeH/Transcrypt

Python : https://riptutorial.com/ko/python/topic/5225/-python-

109:

- ! = -
- == -
- > -
- < -
- >= -
- <= -
- is - .
- .

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
```

x y True False .

```
12 != 1
# True
12 != '12'
# True
'12' != '12'
# False
```

```
x == y
```

x y . int 12 '12' .

```
12 == 12
# True
12 == 1
# False
'12' == '12'
# True
'spam' == 'spam'
# True
'spam' == 'spam '
# False
'12' == 12
# False
```

. . . True False .

.

```
x > y > z
```

.

```
x > y and y > z
```

True True .

.

```
a OP b OP c OP d ...
```

OP .

 0 != 1 != 0 0 != 0 False True . x != y != z x , y z . Chaining == .

.

```
1 > -1 < 2 > 0.5 < 100 != 24
```

True True . . "" ,

```
1 > x > -4 > y != 8
```

False , False .

a > exp > b exp ,

```
a > exp and exp > b
```

exp a > exp a > exp .

`is 'vs`==`

is == .

a == b a b .

a is b a b .

:

```
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 [!!]
```

is id(a) == id(b) .

. Python is True .

```
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
```

`is None` :

```
if myvar is not None:
    # not None
    pass
if myvar is None:
    # None
    pass
```

`is` ""(,) .

```
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 >` `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()` ( ).

Common Gotcha : .

(Java)

```
if("asgdsrf" == 0) {
    //do stuff
}
```

... . . . False .

```
myVariable = "1"
if 1 == myVariable:
    #do stuff
```

`False` .

: https://riptutorial.com/ko/python/topic/248/

110:

. . , , . .

- x << y #
- x >> y #
- x y # AND
- x | y # OR
- ~ x # Bitwise NOT
- x ^ y # XOR

Examples

AND

& 2 **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

| "or" . "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 ( )

^ 1 **XOR** . 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
```

1 2 .

```
7 << 1
# Out: 14
```

n 2**n .

```
3 << 4
# Out: 48
```

>> " " .

```
# 8 = 0b1000
8 >> 2
# Out: 2
# 2 = 0b10

bin(8 >> 2)
# Out: 0b10
```

1 2 .

```
36 >> 1
# Out: 18

15 >> 1
# Out: 7
```

`n` `2**n` .

```
48 >> 4
# Out: 3

59 >> 3
# Out: 7
```

NOT

`~` . (0) (1) 2 2 .

, 8 2 `0000 0000` `0111 1111` 0 127 `1xxx xxxx` .

8 2

		2
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 0110` `(128 * 1) + (64 * 0) + (32 * 1) + (16 * 0) + (8 * 0) + (4 * 1) + (2 * 1) + (1 * 0)` `(128 * 1) - (64 * 0) - (32 * 1) - (16 * 0) - (8 * 0) - (4 * 1) - (2 * 1) - (1 * 0)` `(128 * 1) + (64 * 0) + (32 * 1) + (16 * 0) + (8 * 0) + (4 * 1) + (2 * 1) + (1 * 0)`) 2 . `(128 * 1) - (64 * 0) - (32 * 1) - (16 * 0) - (8 * 0) - (4 * 1) - (2 * 1) - (1 * 0)`

-128 (`1000 0000`) . (0) `0000 0000` 1 (`1111 1111`) `1111 1111` .

`~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
```

(`~`) in place .

```
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/ko/python/topic/730/-

111:

Examples

, (:) .

```
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'>>
```

. .

```
# 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/ko/python/topic/3965/--

112: /

.

.

Python . Exception . .

Examples

Exception CustomError . 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 catch .

```
class CustomError(Exception):
     pass

try:
    raise CustomError('Can you catch me ?')
except CustomError as e:
    print ('Catched CustomError :{}'.format(e))
except Exception as e:
    print ('Generic exception: {}'.format(e))
```

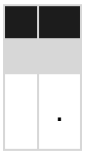
:

```
Catched CustomError :Can you catch me ?
```

/ : https://riptutorial.com/ko/python/topic/10882/------

113:

- mydict = {}
- mydict [k] =
- = mydict [k]
- = mydict.get (k)
- value = mydict.get (k, "default_value")



.

:

- ( )
- (hash , TypeError)
- .

Examples

```
dictionary = {"Hello": 1234, "World": 5678}
print(dictionary["Hello"])
```

1234 .

"Hello" . dict .

1234 dict . "Hello" dict .

KeyError catch . KeyError dictionary.get . None . None .

```
w = dictionary.get("whatever")
x = dictionary.get("whatever", "nuh-uh")
```

w None 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 KeyError

```
mydict = {}
mydict['not there']
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'not there'
```

dict.get dict.get .

```
value = mydict.get(key, default_value)
```

mydict[key] , default_value . mydict key mydict . , 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
```

.

dict collections.defaultdict . new_key dict default_factory.

for , .

:

```
mydict = {
    'a': '1',
```

```
    'b': '2'
}
```

keys() 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() . OrderedDict sort() , sorted() OrderedDict .

Python 2/3 : Python 3 Python 2 iterkeys() , itervalues() 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}
```

```
d = {k:v for k,v in [('key', 'value',)]}
```

: Comprehensions

: dict()

```
d = dict()                    # emtpy 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)
```

dict

.

```
d['newkey'] = 42
```

list dictionary .

```
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'
```

[*] using dict.setdefault() dict using dict.setdefault() .

```
>>> d = {}
{}
>>> d.setdefault('Another_key', []).append("This worked!")
>>> d
{'Another_key': ['This worked!']}
```

dict.setdefault() ([]) .

[*] *Python Cookbook, David Beazley Brian K. Jones (O'Reilly) 3 . Copyright 2013 David Beazley and 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])
```

**

** unpacking - . :

```
>>>
>>> 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'}
```

(: "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'}
```

(: "Clifford" "Nemo").

Python 3.3

```
>>> from collections import ChainMap
>>> dict(ChainMap(fish, dog))
{'hands': 'fins', 'color': 'red', 'special': 'gills', 'name': 'Nemo'}
```

("Clifford" "Nemo") foremost .

Python 2.x, 3.x

```
>>> from itertools import chain
>>> dict(chain(fish.items(), dog.items()))
{'hands': 'paws', 'color': 'red', 'name': 'Clifford', 'special': 'gills'}
```

`**` ("Clifford" "Nemo") .

```
>>> fish.update(dog)
>>> fish
{'color': 'red', 'hands': 'paws', 'name': 'Clifford', 'special': 'gills'}
```

`dict.update` dict dict .

.

```
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()` iteraters .

:

- ( )
-

(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}
```

.

```
car = {}
car["wheels"] = 4
car["color"] = "Red"
car["model"] = "Corvette"
```

.

```
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/ko/python/topic/396/

114:

Shelve . shelve API Python . Python . . picked anydbm .

: . `close()` `shelve.open()` `shelve.open()` .

```
with shelve.open('spam') as db:
    db['eggs'] = 'eggs'
```

:

`shelve pickle` . pickle .

1 . (dbm.ndbm dbm.gnu) . dbm . () dbm . , , .

2. / . (.) . Unix .

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
```

shelve **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'}
```

. writeback . .

```
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/ko/python/topic/10629/

115:

. . / / . Donald Knuth .

" 97 % . 3 % . "

Examples

. / / . Donald Knuth .

" . 3 % ."

`line_profiler` `timeit` `cProfile` (`profile`) . .

`cProfile` determistic . , (0.001) . ([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=s).sort_stats(sortby)
ps.print_stats()
print s.getvalue()
```

.

```
         3 function calls in 0.000 seconds

Ordered by: standard name
ncalls  tottime  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][1]) . . . kernprof . .

```
$ kernprof -l script_to_profile.py
```

kernprof LineProfiler `__builtins__` . `@profile` .

```
@profile
def slow_function(a, b, c):
    ...
```

kernprof `script_to_profile.py.lprof` `script_to_profile.py.lprof` . kernprof [-v / - view] .
.

```
$ python -m line_profiler script_to_profile.py.lprof
```

timeit . 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/ko/python/topic/5889/-

116:

- empty_set = set () # .
- literal_set = { 'foo', 'bar', 'baz'} # 3 .
- set_from_list = set (['foo', 'bar', 'baz']) # set .
- set_from_iter = set ((x) x x) # iterable
- set_from_iter = {x (10) i [random.randint (0,10)] x }} #

(O (1)). . . . .

. frozenset .

. __hash__ . __eq__ . list set . dict .

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 .

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
_	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']
```

.

collections Counter (2.7) :

Python 2.x 2.7

```
>>> from collections import Counter
>>> counterA = Counter(['a','b','b','c'])
>>> counterA
Counter({'b': 2, 'a': 1, 'c': 1})
```

Counter . , .

a b a

```
>>> a = {1, 2, 2, 3, 4}
>>> b = {3, 3, 4, 4, 5}
```

: {1} {} dict . set() .

---

a.intersection(b) a b .

```
>>> a.intersection(b)
{3, 4}
```

---

a.union(b) a b .

```
>>> a.union(b)
{1, 2, 3, 4, 5}
```

---

a.difference(b) a b

```
>>> a.difference(b)
{1, 2}
>>> b.difference(a)
{5}
```

---

a.symmetric_difference(b) a b a a .

```
>>> 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)` `c` `a`

`a.issuperset(c)` `c` `a`

```
>>> c = {1, 2}
>>> c.issubset(a)
True
>>> a.issuperset(c)
True
```

.

`a.intersection(b)`	`a & b`
`a.union(b)`	`a | b`
`a.difference(b)`	`a - b`
`a.symmetric_difference(b)`	`a ^ b`
`a.issubset(b)`	`a <= b`
`a.issuperset(b)`	`a >= b`

`a` `d` `a` `d` .

```
>>> 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`

```
>>> 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/ko/python/topic/497/

117:

. Python .

socket.AF_UNIX	
socket.AF_INET	IPv4
socket.AF_INET6	IPv6
socket.SOCK_STREAM	TCP
socket.SOCK_DGRAM	UDP

Examples

UDP

UDP . . . UDP .

UDP localhost 6667 .

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` [ ]) .

`socket` UDP :

```
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' localhost 6667 TCP .

```
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()
```

., . . .

.

TCP

5000 127.0.0.1 TCP . .

-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/ko/python/topic/1530/

118:

: Python 2 ( ).

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
```

getter, setter deleter .

getter , setter deleter   .

```
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)
```

setter deleter   .

(@)   .   . 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/ko/python/topic/2050/-

119:

- `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")
```

.

```
>>> 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'
```

,

. getter setter .

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author
```

. , getters setter . " " . .

author getter setter .

```
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
```

.

catch . public . :

```
>>> book = Book(title="Ancient Manuscript", author="Some Guy")
>>> book.author = ""  #Cos Some Guy didn't write this one!
```

. ! . @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 .

:

```
>>> book = Book(title="Ancient Manuscript", author="Some Guy")
>>> book.author = ""  #Cos Some Guy didn't write this one!
>>> book.author
Unknown
```

: https://riptutorial.com/ko/python/topic/4392/-

120:

n iterables O(n) . bisect.bisect_left() O(log(n)) .

Examples

: str.index (), str.rindex () str.find (), str.rfind ()

String index str.find . .

```
astring = 'Hello on StackOverflow'
astring.index('o')  # 4
astring.rindex('o') # 20

astring.find('o')   # 4
astring.rfind('o')  # 20
```

index / rindex find / 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
```

in .

```
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
```

Dict

dict : in . . - .

```
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.index (), tuple.index ()

list tuple index :

```
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)
```

ValueError : 15 .

.

```
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)
```

keys list .

```
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'
```

StopIteration .- dict Exception .

```
getOneKeyForValue(adict, 25)
```

: bisect.bisect_left ()

.bisect.bisect_left() [1] :

```
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 .

: __contains__ __iter__

in __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__` `__contains__` :

```
del ListList.__contains__
5 in a     # True
# Prints: Using __iter__.
```

: `in` (`for i in a`) `__iter__` `__contains__` .

: https://riptutorial.com/ko/python/topic/350/

121:

. . . .

, .

Examples

, C ++ . .

```
class BaseClass(object):
    pass

class DerivedClass(BaseClass):
    pass
```

`BaseClass` ( ) `DerivedClass` `BaseClass` ( ) ( ) . : Python 2.2 `object` .

`Rectangle` . `object` `object` .

```
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)
```

`Rectangle` `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
```

`Square` `Rectangle` . `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)` : `DerivedClass` `BaseClass` `True` .

`isinstance(s, Class)` : `True` `Class` `Class`

```
# 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
```

.

```
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
```

.

```
class D:
    x = []
    def __init__(self, item):
        self.x.append(item)  # note that this is not an assigment!

d1 = D(1)
d2 = D(2)

d1.x
# [1, 2]
d2.x
# [1, 2]
D.x
# [1, 2]
```

,

Python 3 . 3 `def` . , . `A` `f` `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)
```

2 : `instancemethod` . . .`__func__` .`__func__` .

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 .

Python 3.x 3.0

```
import inspect

inspect.isfunction(A.f)
# True
inspect.ismethod(A.f)
# False
```

2.x 2.3

```
import inspect
```

```
inspect.isfunction(A.f)
# False
inspect.ismethod(A.f)
# True
```

Python / `Af A .`

```
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
```

`a , A af ?, f "" - A a a .`

: `af __getattribute__ , a a f ( ), A () m method m . Af m.__func__ , a m.__self__ . m(...) => m.__func__(m.__self__, ...) .` . (Python 2 3).

```
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
```

, . . . 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
```

, , `__get__` , `__set__` `__del__` `__get__` . classmethod staticmethod .

- @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 .

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
```

. .

```
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 .

C3 . MRO (Method Resolution Order).

.

```
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'
```

FooBar foo Foo .

```
fb = FooBar()
```

```
>>> fb.foo
'attr foo of Foo'
```

FooBar MRO .

```
>>> FooBar.mro()
[<class '__main__.FooBar'>, <class '__main__.Foo'>, <class '__main__.Bar'>, <type 'object'>]
```

MRO

1. (: `FooBar` , `Foo`)
2. (`object`) (`Bar`)
3. .

Bar FooBar FooBar Bar .

Python wikipedia .

`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()
```

init , init init .

Foo **init** **Bar** init

```
    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 . **FooBar** **Bar** **Foo** .

```
print isinstance(a,FooBar)
print isinstance(a,Foo)
print isinstance(a,Bar)
```

:

```
True
True
True
```

`__get__` () `__get__` , `__set__` `__delete__` .

`__set__` `__delete__` `__set__` `__delete__`

, `staticmethod` , `classmethod` `property` . (: `bar` `Foo` `foo` , `foo.bar`) :

1. `bar` `Foo` `Foo` . . `property` .
2. `bar` `__dict__` . . `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 . full_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 ( ) 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 + ".")
```

from_full_name self cls from_full_name . (self). cls Person Person from_full_name Person(first_name, last_name, age) Person Person __init__ . , Employee Person , from_full_name Employee .

, __init__ Person .

```
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 for beginner?
- 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
```

A ? A .

```
A.get_num = get_num
```

? .

get_num ( ) A A A

( ) . :

```
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__', '__getattribute__', '__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']
```

" (non-magic)" . __ .

```
>>> [m for m in dir(list) if not m.startswith('__')]
['append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse',
'sort']
```

:

__dir__() . dir() __dir__() , . 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` (`self` `self`).
7. . `private` C ++ / Java `public` . " ".
8. `__functionname__(self, other_stuff)` . "" . , . .

`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 .string. .

```
mc = MyClass()
mc.string = "String!"
print(mc.string)
del mc.string
```

. API . 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 .

```
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/ko/python/topic/419/

122:

Examples

: round, floor, ceil, trunc

round math floor , ceil trunc .

```
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 , trunc round float .

```
round(1.3)  # 1.0
```

round 0 .

```
round(0.5)  # 1.0
round(1.5)  # 2.0
```

Python 3.x 3.0

floor , ceil trunc Integral round Integral .

```
round(1.3)      # 1
round(1.33, 1)  # 1.3
```

round . .

```
round(0.5)  # 0
round(1.5)  # 2
```

!

. .

```
round(2.675, 2)  # 2.67, not 2.68!
```

, trunc

(C ++ Java) 0 . :

```
>>> math.floor(-1.7)
-2.0
>>> -5 // 2
-3
```

math.log(x) (e) x .

```
math.log(math.e)  # 1.0
math.log(1)       # 0.0
math.log(100)     # 4.605170185988092
```

math.log 1 . 1 math.log1p . 1 .

```
math.log(1 + 1e-20)  # 0.0
math.log1p(1e-20)    # 1e-20
```

math.log10 . 10 :

```
math.log10(10)  # 1.0
```

Python 2.x 2.3.0

math.log(x, base) base x (log(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) y x . 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.atan` 0 `math.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
>>>
```

3.5 NaN (" ") . `float()` .

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
```

"j" "J" .

```
1j         # Equivalent to the square root of -1.
```

```
1j * 1j    # = (-1+0j)
```

NaN (" ")

Python NaN (" ") .

```
pos_inf = float('inf')     # positive infinity
neg_inf = float('-inf')    # negative infinity
not_a_num = float('nan')   # NaN ("not a number")
```

3.5 `math.inf` `math.nan` .

Python 3.x 3.5

```
pos_inf = math.inf
neg_inf = -math.inf
not_a_num = math.nan
```

`inf` `-inf` `nan` .

```
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 .

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
```

0 ( ) `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 NaN . -1 . " NaN" .

```
5.0 * not_a_num
# Out: nan

float('-nan')
# Out: nan
```

Python 3.x 3.5

```
-math.nan
# Out: nan
```

NaN `float` Python 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)
```

Pow

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

cmath math .

, Python . import cmath import cmath .

j (J) i .

```
z = 1 + 3j
```

j 1j .

```
1j * 1j
Out: (-1+0j)

1j ** 1j
# Out: (0.20787957635076193+0j)     # "i to the i"  ==  math.e ** -(math.pi/2)
```

real imag () 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
```

abs complex .

```
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 "signed zero" . IEEE 754 . Python .

```
cmath.phase(complex(-1.0, 0.0))
# Out: 3.141592653589793

cmath.phase(complex(-1.0, -0.0))
# Out: -3.141592653589793
```

`cmath` `math` .

`sqrt` `exp` , `log` , `log10` , (`sin` , `cos` , `tan` , `asin` , `acos` , `atan`) (`sinh` , `cosh` , `tanh` , `asinh` , `acosh` , `atanh`). arctangent `math.atan2` .

```
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
```

`pi` `e` . `float` `complex` .

```
type(cmath.pi)
# Out: <class 'float'>
```

`cmath` `isinf` (for Python 3.2+) `isfinite` . " Infinity NaN " . .

```
cmath.isinf(complex(float('inf'), 0.0))
# Out: True
```

, `cmath` `isnan` . " Infinity NaN " . " " " " .

```
cmath.isnan(0.0, float('nan'))
# Out: True
```

`math.inf` `math.nan` `cmath` (Python 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 `cmath` `math` `isclose` .

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/ko/python/topic/230/-

123:

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
```

.

```
# 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/ko/python/topic/3070/

124:

Examples

. . __action__ . .

. . .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/ko/python/topic/946/-

125:

LIFO (last-in-first-out) . . . . . .

- stack = [] # .
- stack.append (object) # .
- stack.pop () -> object # .
- list [-1] -> object #

:

push *pop* .

LIFO (*Last-In, First-Out*) .

`pop()` push `append()` `append()` `pop()` pop . O (1) .

`deque` . `deque` .

Examples

List Stack

`list` . `list` `Stack` .

```
#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
```

. .

, ([]) . ()<>) ,) . ([)] . .

```
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/ko/python/topic/3807/

126:

(3.5) . (: `a = myobject`) (`a = myobject`) . `0` .

`del` . . `del` refcount . , refcount 0 ( ).

. a = object () (cpython (: . refcount 0 GC .

1960 John McCarthy Lisp refcounting refcounting : ? ? . .

, . . ( ). refcount . (, ,) 0 "Generation 1" . (0 .) . . 3 (3). (0) . GC (1) GC (2)). 3 (" 2 " ,) . GCed GC GC . . " " . " (generary hypothesis)" 60 .

: 2 , 3 . (3 GC) 25 % . ( ), GC . " ". GC GC (Martin Von Löwis). 3 "" " ".

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)
```

. dealloc . GC (2) 0. . : `__del__` ? `__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)
```

A B ? A dealloc . A `__del__` , , A . B , `__del__` , ! ! A . `__del__` . : `__del__` GCed ? GC . `__del__` . `__del__` refcount GCing .

CPython uncollectable un-GC-able (`__del__`) `__del__` .

```
/* list of uncollectable objects */
static PyObject *garbage = NULL;
```

Examples

.

(:) . (:) .

0 . .

```
>>> 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
```

. A B B A A B . A B . 1 .

```
>>> 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
```

. A B C ... A Z A Z .

```
>>> objs = [Track() for _ in range(10)]
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
20
```

del

`del v  del v[item] del[i:j]  del v.name` . 0 .

```
>>> 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
```

. . -5 256 .

```
>>> import sys
>>> sys.getrefcount(1)
797
>>> a = 1
>>> b = 1
>>> sys.getrefcount(1)
799
```

refcount . `a b 1` . Python .

```
>>> a = 999999999
>>> sys.getrefcount(999999999)
3
>>> b = 999999999
>>> sys.getrefcount(999999999)
3
```

`999999999 a b` .

```
>>> import sys
>>> a = object()
>>> sys.getrefcount(a)
2
>>> b = a
>>> sys.getrefcount(a)
3
>>> del b
>>> sys.getrefcount(a)
2
```

2 3 refcount 0 .

`ctypes` .

: *!* ( ) . .

Python 3.x 3.0

```
import ctypes
deallocated = 12345
ctypes.pythonapi._Py_Dealloc(ctypes.py_object(deallocated))
```

2.x 2.3

```
import ctypes, sys
deallocated = 12345
(ctypes.c_char * sys.getsizeof(deallocated)).from_address(id(deallocated))[:4] = '\x00' * 4
```

, ...

None . Fatal Python error: deallocating None Fatal Python error: deallocating None .

. .

. . - **generation0** , . - **generation2,** .

.

```
import gc
gc.set_threshold(1000, 100, 10) # Values are just for demonstration purpose
```

x **generation0** . .

. . **generation0 generation1** 100 **generation1 . generation1 generation2** 10 **generation2** .

(**generation0_threshold**). . , .

j .

```
import gc
gc.collect()
```

. . .

, . ., .

. ., .

.

.

, .

:

f .

```
>>> 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/ko/python/topic/2532/-

127:

ijson Python JSON . , pure 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

: https://riptutorial.com/ko/python/topic/8342/-

128:

. . ,, .

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()
```

.

```
ll = LinkedList()
ll.add('l')
ll.add('H')
ll.insert(1,'e')
ll.append('l')
```

```
ll.append('o')
ll.printList()

H
e
l
l
o
```

: https://riptutorial.com/ko/python/topic/9299/-

129:

Enum PEP 435 3.4 Python .

Examples

(Python 2.4 ~ 3.3)

3.4 3.3 2.4 . PyPI enum34 .

```
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/ko/python/topic/947/-

130:

. . . Python . , `BaseException` .

-
- raise # .
- - # 3
- None # Python 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`) .

`ZeroDivisionError` `ArithmeticError` .

```
>>> ZeroDivisionError.__bases__
(<class 'ArithmeticError'>,)
```

`ZeroDivisionError` catch.

```
try:
    5 / 0
except ArithmeticError:
    print("Got arithmetic error")
```

. .

try finally .

```
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)
```

raise raise .

3 raise - from .

```
    raise ZeroDivisionError("Got an error") from e
```

. , IOError 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
```

.

`IOError` `OSError` `EnvironmentError` . `IOError` `OSError` catch . `EnvironmentError` `IOError` `OSError` .

:

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
           +-- RuntimeWarning
           +-- SyntaxWarning
           +-- UserWarning
           +-- FutureWarning
           +-- ImportWarning
           +-- UnicodeWarning
           +-- BytesWarning
           +-- ResourceWarning
```

.

`BaseException` Python . `raise` . :

```
>>> 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()` , interpreter .

.

```
>>> 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!',)
```

: https://docs.python.org/3.5/library/exceptions.html . . Exception Hierarchy .

`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 gratiously
finally:
    print "The END"
    # it runs no matter what execute.
```

(`BaseException` `Exception`) :

```
try:
    even_more_difficult_function()
except:
    pass  # do whatever needed
```

. `SystemExit` , `KeyboardInterrupt` `MemoryError` . . . .

''. . .( )

.

.

catch . `KeyError` `AttributeError` .

```
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.")
```

. `KeyError` `AttributeError` .

```
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 . , `input` `2 + 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` .

else `try` . .

:

```
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: else: elif if . if else: :

```
try:
    ...
except ...:
    ...
else:
    if ...:
        ...
    elif ...:
        ...
    else:
        ...
```

: https://riptutorial.com/ko/python/topic/1788/

131:

Examples

Pyglet

```
import pyglet
audio = pyglet.media.load("audio.wav")
audio.play()
```

pyglet .

WAV

winsound

- 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)
```

ffmpeg

```
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
- ffmpeg, libav avconv ?

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/ko/python/topic/8189/

132:

Examples

/

Python Magic (double-underscore dunder) . 2 . .

2 .

```
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
```

D .

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
```

list sparselist .

```
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). .

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__
```

__radd__ __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
```

.

other .

+	__add__(self, other)	a1 + a2
-	__sub__(self, other)	a1 - a2
*	__mul__(self, other)	a1 * a2
@	__matmul__(self, other)	a1 @ a2 (*3.5*)
/	__div__(self, other)	a1 / a2 (*2*)
/	__truediv__(self, other)	a1 / a2 (*3*)
//	__floordiv__(self, other)	a1 // a2
/ %	__mod__(self, other)	a1 % a2
**	__pow__(self, other[, modulo])	a1 ** a2
<<	__lshift__(self, other)	a1 << a2
>>	__rshift__(self, other)	a1 >> a2
& AND	__and__(self, other)	a1 & a2
^ XOR	__xor__(self, other)	a1 ^ a2
| (OR)	__or__(self, other)	a1 | a2
- ()	__neg__(self)	-a1
+	__pos__(self)	+a1
~ NOT	__invert__(self)	~a1
<	__lt__(self, other)	a1 < a2
<=	__le__(self, other)	a1 <= a2
==	__eq__(self, other)	a1 == a2
!=	__ne__(self, other)	a1 != a2
>	__gt__(self, other)	a1 > a2
>=	__ge__(self, other)	a1 >= a2
[index]	__getitem__(self, index)	a1[index]
in	__contains__(self, other)	a2 in a1
(*args, ...)	__call__(self, *args, **kwargs)	a1(*args, **kwargs)

__pow__ modulo (modulo pow .

__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
```

inplace __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	__int__(self)	int(a1)
	__abs__(self)	abs(a1)
str	__str__(self)	str(a1)
unicode	__unicode__(self)	unicode(a1) (2)
	__repr__(self)	repr(a1)
bool	__nonzero__(self)	bool(a1)
	__format__(self, formatstr)	"Hi {:abc}".format(a1)
	__hash__(self)	hash(a1)
	__len__(self)	len(a1)
	__reversed__(self)	reversed(a1)
	__floor__(self)	math.floor(a1)
	__ceil__(self)	math.ceil(a1)

`__enter__` `__exit__` .

: https://riptutorial.com/ko/python/topic/2063/

133:

.

-

	. os.path.sep .
	(8 (: 0700)

Examples

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

mode .

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

os.getcwd() .

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

os .

```
os.name
```

3 :

- posix
- nt
- ce
- java

sys.platform .

path .

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

os.remove() os.remove() . OSError

(POSIX)

. os.readlink os.readlink :

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

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

`mode` (8).

makedirs -

.

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

dir2 subdir1, subdir2 .

```
import os

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

```
├── dir1
│   ├── subdir1
│   └── subdir2
└── dir2
    ├── subdir1
    └── subdir2
```

dir2 subdir1 .

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
```

: https://riptutorial.com/ko/python/topic/4127/--

134:

Examples

(infix operator) , `+ operator` - (`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']
```

operator-function :

```
from operator import methodcaller
list(filter(methodcaller('startswith', 'd'), alist)) # Does the same but is faster.
# Output: ['duck']
```

Itemgetter

- `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/ko/python/topic/257/-

135:

. , 3 * 2 + 7 3 2 7 13 . * + .

.

:

( ) ( ) . . . (,).

	OR
	AND
x	NOT
<, <>,> =, <>,! =, ==	
|	OR
^	XOR
&	AND
<<, >>	
+, -	
*, /, //, %	, , [8]
+ x, -x, ~ x	, , NOT
**	[9]
x [], x [:], x (...), x.attribute	, , ,
(...), [...], { : ...}, ...	, , ,

Examples

.

PEMDAS . 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/ko/python/topic/5040/--

136:

Python 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)`

`webbrowser.open()`	
url	URL
	0 URL , 1 , 2 .
	True .
`webbrowser.open_new()`	
url	URL
`webbrowser.open_new_tab()`	
url	URL
`webbrowser.get()`	
~	
`webbrowser.register()`	
url	
	( )
	`webbrowser.get()`

. `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` - URL () **[]**
- `new` - 0 , , 2 ( ) **[0]**
- `autoraise` - True () **[default False]**

`new` `new autoraise` .

Webbrowser `open_new` URL .

```
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 register .

```
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/")
```

: https://riptutorial.com/ko/python/topic/8676/--

137:

Examples

echo with aiohttp

`aiohttp` `aiohttp` .

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())
```

aiohttp

`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()
```

Autobahn

Autobahn Python .

Python Autobahn

.

(Linux) :

```
sudo pip install autobahn
```

(Windows) :

```
python -m pip install autobahn
```

Python .

```
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:
        importasyncio
    except ImportError:
        '''Trollius = 0.3 was renamed'''
        import trollius as asyncio
    from autobahn.asyncio.websocketimportWebSocketServerFactory
    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()
```

9000 (127.0.0.1) . IP . . LAN . Google WAN IP WAN IP, 9000 ( ) WebSocket .

. , ( ) 9000 .

: https://riptutorial.com/ko/python/topic/4751/-

138:

Examples

. .

```
>>> u'🐍'.encode('utf-8')
'\xf0\x9f\x90\x8d'
```

. .

```
>>> b'\xf0\x9f\x90\x8d'.decode('utf-8')
u'\U0001f40d'
```

: https://riptutorial.com/ko/python/topic/5618/-

139:

- str.encode (, = '')
- bytes.decode (, = '')
- open (, , =)

(: 'ascii' , 'utf8') ...	
, 'replace' 'ignore' .	

Examples

Python 3 `str` `bytes` .

```
type("f") == type(u"f")  # True, <class 'str'>
type(b"f")               # <class 'bytes'>
```

2 "u" .

```
type("f") == type(b"f")  # True, <type 'str'>
type(u"f")               # <type 'unicode'>
```

`.encode(encoding)` .

3

```
>>> "£13.55".encode('utf8')
b'\xc2\xa313.55'
>>> "£13.55".encode('utf16')
b'\xff\xfe\xa3\x001\x003\x00.\x005\x005\x00'
```

2

py2 `sys.getdefaultencoding() == 'ascii'` `sys.getdefaultencoding() == 'ascii'` 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')
```

```
┬ú13.55
```

,`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)
```

.decode(encoding) .

!

```
>>> 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'
```

.

I / O

- (: `'r'` `'w'`) . `'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")
```

: https://riptutorial.com/ko/python/topic/1216/--

140:

ctypes .

: OS .

Examples

libc ntohl .

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'
```

.

.

x . 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 `.dll` 32 64 . , .

.

.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)
```

`obj` 12 .

.

```
>>> 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'>
```

! 16 `int` .

.

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

`arr` 0 15 .

.

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

`ctypes` .

```
>>> sizeof(arr)
64 # sizeof(c_int) * 16
>>> hex(addressof(arr))
'0xc000l0ff'
```

ctypes

C . `ctypes` .

:

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

. , type int .

.

```
>>> 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>
```

. ctypes (: libc.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)
```

(libc lfind) .

. .

.

```
>>> 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
```

x y POINTER(c_int) , POINTER(c_int) .

.

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

ptr void . arr key None .

.

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

ptr

arr .

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

: https://riptutorial.com/ko/python/topic/9050/

141:

- [int, str], None] -> def func (a : int, b : str) ->
- [str, int] -> { "a": 1, "b": 2, "c": 3}
- typing.List [int] -> [1, 2, 3]
- typing.Set [int] -> {1, 2, 3}
- typing.Optional [int] -> int
- [int] -> [1, 2, 3] (1, 2, 3)
- typing.Any ->
- [int, str] -> 1 "1"
- T = typing.TypeVar ('T') ->

PEP 484 Python . - typing .

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 two_sum . int .

```
print(two_sum(2, 1))  # result: 3
```

:

```
print(two_sum("a", "b"))  # result: "ab"
```

list , tuple .

.

.

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'
```

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

typing NamedTuple :

```
import typing
Point = typing.NamedTuple('Point', [('x', int), ('y', int)])
```

.

```
def hello_world(greeting: str = 'Hello'):
    print(greeting + ' world!')
```

.

: https://riptutorial.com/ko/python/topic/1766/-

142:

- # .
- print ("") # .
- "" "
  "" "

PEP257 - Docstring Conventions . (: Google) 3 (:) .

Examples

,

.

.

(#) .

- :

```
# 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.
"""
```

docstrings

Docstring .

```
def func():
    """This is a function that does nothing at all"""
    return
```

docstring __doc__ .

```
print(func.__doc__)
```

.

```
help(func)
```

`__main__` function `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` `greeting` .

docstrings

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))
```

```
print(greet.__doc__)
```

```
help(greet)
```

main :

```
greet(name, greeting='Hello')
```

docstring

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 docstring . .

- :

PEP 257 , . (:

```
def hello():
    """Say hello to your friends."""
    print("Hello my friends!")
```

, .

- Docstrings :

, .

```
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 , . .

Sphinx Python HTML . reStructuredText . . pythonhosted.org . Sphinx pyCharm IDE .

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 Python Google Python . Sphinx / reST Google .

pythonhosted.org Google .

Sphinx Napoleon Google .

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/ko/python/topic/4144/--

143:

- (fmt, v1, v2, ...)
- (fmt,)

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** " .

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)
```

:

byteorder : : '\ 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/ko/python/topic/2978/-

144:

- obj [: :]
- ()
- (, [,])

obj	" "
start	obj (0 . , obj 0). 0 .
stop	obj ( ). len(obj) .
step	step . 1 .

1 .

[start, end) . , . len(x[:n]) = n len(x) > = n x[n:n+1] = [x[n]] x len(x) >= n .

Examples

(: ,) , .

:

```
iterable_name[start:stop:step]
```

,

- start . 0 ( ).
- stop . len (iterable).
- 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)
```

(iterable)

```
a[::-1]     # "fedcba" (from last element (default len()-1), to first, in reverse order(-1))
```

`end_index None` (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 , end slice . 0 , len(arr) , 1 . , arr .

.

```
arr = ['a', 'b', 'c']
copy = arr[:]
arr.append('d')
print(arr)    # ['a', 'b', 'c', 'd']
print(copy)   # ['a', 'b', 'c']
```

`arr.append('d')` arr d copy .

`arr.copy()` .

`str` , `list` `tuple` (step) . .

```
s = 'reverse me!'
s[::-1]    # '!em esrever'
```

.`[::-1]` (`start` `end`) `-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]
```

. .

, .

```
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']
```

0 . , 0

```
arr = ['a', 'b', 'c', 'd']
print(arr[0])
>> 'a'
```

1 , 2 .

```
print(arr[1])
>> 'b'
print(arr[2])
>> 'c'
```

.. -1 -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/ko/python/topic/289/--

145:

Examples

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` .

```
print(foo, bar, baz)
# out: 1 bar 3.14
```

`print` `+`

```
print(str(foo) + " " + bar + " " + str(baz))
# out: 1 bar 3.14
```

`+` . `string` `"bar"` 1 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))
```

end print .

```
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 : sep .

?

```
>>> print('apples','bannas', 'cherries', sep=', ')
apple, bannas, cherries
>>> print('apple','banna', 'cherries', sep=', ')
apple, banna, cherries
>>>
```

end : end .

end print() . (") .

```
>>> print("<a", end=''); print(" class='jidn'" if 1 else "", end=''); print("/>")
<a class='jidn'/>
>>> print("paragraph1", end="\n\n"); print("paragraph2")
paragraph1

paragraph2
>>>
```

file : sys.stdout .

stdout, file StringIO . .

```
>>> 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 flush .

: https://riptutorial.com/ko/python/topic/1360/-

146: ( )

Examples

`help` Python . Python , , .

```
>>> 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
```

. for . _ .

```
>>> "Hello, {0}".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'
```

. .

`py` 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.
>>>
```

`python2` `python3` .

PATH Python .

PYTHONSTARTUP

Python PYTHONSTARTUP . .

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!
>>>
```

`py` . Python 3.4 `py --help` .

```
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.
```

`help` .

, `help` , () .

```
>>> 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.
```

`help` docstring .

```
>>> 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/ko/python/topic/2473/-----

147:

. , , .

.

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 .

```
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 .

```
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 i zlist[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]
```

. .

(**list comprehensions**) . / .

```
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 li False if not li 0 . .

```
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], [1], [1]]
```

1 li .

[[]] * 3 3 list list ., list 3 list ., li[0] , li . .

```
li = []
element = [[]]
li = element + element + element
print(li)
# Out: [[], [], []]
element.append(1)
print(li)
# Out: [[1], [1], [1]]
```

id list .

```
li = [[]] * 3
print([id(inner_list) for inner_list in li])
# Out: [6830760, 6830760, 6830760]
```

.

```
li = [[] for _ in range(3)]
```

list 3 3 . 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))
```

"==" .

```
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 " " .

:

```
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 . .

.

., . str() . list() .

:

```
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 ,,, . DRY.

.

.Python 2.x . ,,. . "". object .

:

```
class Father:
    pass

class Child(Father):
    pass
```

:

```
class Father(object):
    pass


class Child(Father):
    pass
```

Python 3.x .

init .

PHP . . , . ( ) . . :

- .
- .

( ) :

```
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()]
```

ID

.

ID .

```
>>> -8 is (-7 - 1)
False
>>> -3 is (-2 - 1)
True
```

.

```
>>> (255 + 1) is (255 + 1)
True
>>> (256 + 1) is (256 + 1)
False
```

?

`is` (`-3` , `256`) `True` (`-8` , `257`) .

, `[-5, 256]` ., `is True` ; ID `False` .

( ) .

(`==`) (`is`) .

ID (: `is`) .

```
>>> 'python' is 'py' + 'thon'
True
```

'python' 'python' .

ID .

```
>>> '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
```

Integer (is) is (==) .

int

. , 7 . , bit_length . .

```
x = 7
x.bit_length()
# Out: 3
```

7.bit_length() . SyntaxError .? (: 7.2 7.bit_length()) . .

int .

```
# parenthesis
(7).bit_length()
# a space
7 .bit_length()
```

float float bit_length() (: 7..bit_length()) .

float . float 2 "" . :

```
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]   . .

```
# 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']
```

.

, C ++ 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']
```

.

/ `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` . .

3.6 . .

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)

(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...` . ! . GIL C (`math.factorial`) .

. . . 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` `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
```

( ) "" 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 2 Python 3 .

.

xyz a b .

```
def xyz():
  return a, b
```

xyz . xyz .

```
t = xyz()
```

`t` (A, B) `t` .

TypeError : type .

.

```
a, b = xyz()
```

!

JSON

```
my_var = 'bla';
api_key = 'key';
...lots of code here...
params = {"language": "en", my_var: api_key}
```

JavaScript Python . `params` `params` .

```
{
    "language": "en",
    "my_var": "key"
}
```

.

```
{
    "language": "en",
    "bla": "key"
}
```

`my_var` .

: https://riptutorial.com/ko/python/topic/3553/-

148:

Django Python . . .

Examples

.

django `Hello World` .

PC .

: python -c "import django"
-> django .

django . :
django-admin startproject HelloWorld

HelloWorld .
.
HelloWorld
| --helloworld
| | - **init** .py
| | --settings.py
| | --urls.py
| | --wsgi.py
| --manage.py

Writing Views (django)

. HTML . view.py views.py .

(views.py).

```
from django.http import HttpResponse

define helloWorld(request):
    return HttpResponse("Hello World!! Django Welcomes You.")
```

.

- django.http HttpResponse .
- helloWorld . . HttpRequest , request .

  . . . .
- HttpResponse . HttpResponse .

.

URL
URL URLconf .

.

- Django URLconf .
- Django Python urlpatterns . django.conf.urls.url () .
- Django URL URL .
- Django . .

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 !! ." .

: https://riptutorial.com/ko/python/topic/8994/

149:

. , . . .

- def decorator_function (f) : pass # decorator_function .
- @decorator_function
  def decorated_function () : pass # wrapped ( ). 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.")
```

`@` :

```
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__
```

    '.'

( )

, . .

. .

```
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()
```

TypeError : decorator () missing 1 : '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,)

/ . ( ) .

```
def singleton(cls):
    instance = [None]
    def wrapper(*args, **kwargs):
        if instance[0] is None:
            instance[0] = cls(*args, **kwargs)
        return instance[0]

    return wrapper
```

. .

```
@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/ko/python/topic/229/-

150:

stop condition `stopCondition` .

.

Examples

1 n

`n  1 n  1 + 2 + 3 + 4 + ... + (several hours later) + n . 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)
```

`. 1 + 2 + 3  1 + 2 + 3  .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). . .

,

. `x!` (,). .

- 0 1.
- , 1 .

.

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
```

`. factorial(3) .  .  (: n , factorial , print )  . locals()  .  n = 3 . locals() locals() {'n': 3} . n == 3  n * factorial(n - 1) .`

`.  ,  n  ., factorial(n - 1) .,n - 1 2. 2 factorial n .  n  .A  B  .A {'n': 3} B (`

{'n': 2})) . n * factorial(n - 1) . return . , B , n (B , n n - 1 'n). n * ((n - 1) * factorial((n - 1) - 1)) . .

, factorial((n - 1) - 1)) .A n == 3 1 factorial . {'n': 1} C ., n * factorial(n - 1) . "" 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 3! == 3 * 2 * 1 == 6 . , .

if n == 0: return 1 case. . . . . , factorial .

```
def factorial(n):
    if n == 0:
        return 1
    elif n == 1:
        return 1
    else:
        return n * factorial(n - 1)
```

.

"" . .

- 0 0.
- 1 1.
- , .

.

```
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)
    )
  )
)
```

`5 (1 + (0 + 1)) + ((0 + 1) + (1 + (0 + 1)))` .

.

- . , `return foo(n - 1)` `return foo(n - 1) + 1` `return foo(n - 1) + 1` .
- (**Tail call optimization** : **Tail call optimization**) .
- (**Tail call elimination** , TCE) . TCE TCO.

.

- . .
- .

TCO . . . `factorial` `fib` .

```
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
```

.

lru_cache .

? "". . . . Python .

, Scheme Haskell . .

Python lru 2 . `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 RunursError RecursionError.

-

.

.

```
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)
```

. .

Stack Introspection

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/ko/python/topic/1716/

151:

Examples

Conda

.

Anaconda Binstar Anaconda 1.6 binstar .

```
$ conda install binstar
$ conda update binstar
```

Binstar pypi .

```
$ pip install binstar
```

:

```
$ binstar login
```

whoami .

```
$ binstar whoami
```

' ' . 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` `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
```

.

done Binstar CONDA .

: https://riptutorial.com/ko/python/topic/4064/

152:

-

? (`if` , `except` , `def` , `class`) .

. `class` catch . `except` " " . `for` .

`IndentationError` `pass` ., (`Ellipsis` `...`), `pass` , . `pass` .

- `Exception` (all) (`xml`).

```
try:
    self.version = "Expat %d.%d.%d" % expat.version_info
except AttributeError:
    pass # unknown
```

: `pandas` . `KeyboardInterrupt` `SystemExit` (`HardwareIsOnFireError` catch . ?).

```
try:
    os.unlink(filename_larry)
except:
    pass
```

`except Error:` `except OSError:` `except Error:` . `except ...: pass` 10 % `except ...: pass` catch .

- `scipy` (: `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
```

- , . , `pass` , " ." `pebl` :

```
class ParsingError(Exception):
    """Error encountered while parsing an ill-formed datafile."""
    pass
```

- `Ellipsis` `pass` " / / if-block / ... " ... (: -3) " " . ,

```
def update_agent(agent):
    ...
```

```
def update_agent(agent):
    pass
```

```
def time_step(agents):
    for agent in agents:
        update_agent(agent)
```

`update_agent` . (`raise NotImplementedError` . " `raise NotImplementedError` " " , . ")

Examples

```
try:
    metadata = metadata['properties']
except KeyError:
    pass
```

Exception

```
class CompileError(Exception):
    pass
```

: https://riptutorial.com/ko/python/topic/6891/-

153: (Regex)

re .

. , 'amount\D+\d+' amount . amount=100 , amount is 3 , amount is equal to: 33 .

-
- re.match (pattern, string, flag = 0) # Out :
- re.search (pattern, string, flag = 0) # Out :
- re.findall (pattern, string, flag = 0) # Out : string []
- re.finditer (pattern, string, flag = 0) # Out : re.findall iterator .
- re (, , , = 0) # : ( )
-
- precompiled_pattern = re.compile (, = 0)
- precompiled_pattern.match (string) # Out :
- precompiled_pattern.search (string) # Out :
- precompiled_pattern.findall (string) # Out :
- precompiled_pattern.sub (/ / ,) # 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'
```

pattern r .

. , \ " " (\n) " " . , (\t), (\ ), (\r) . .

r"\n"

`\ n` 2 . . `\d` . (`r"\d"`) (`"\\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 paranthesized 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+22 11 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.!']
```

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'
```

'10' ID . `\10` 'ID 1 0' . `\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}"` r . "".

`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 () .

(az, AZ 0-9) .

```
import re

def is_allowed(string):
    characherRegex = re.compile(r'[^a-zA-Z0-9.]')
    string = characherRegex.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.search re .

```
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.I	.
re.DOTALL , re.S	. .
re.MULTILINE , re.M	^ $
re.DEBUG	.

docs .

:

(?iLmsux) ('i', 'L', 'm', 's', 'u', 'x')

. : re.I ( ), re.L ( ), re.M ( ), re.S ( ), re.U ( ) re.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).
```

"apple" , regex *backtracking control* . .

```
forget_this | or this | and this as well | (but keep this)
```

Apple .

```
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
```

"" .

.

- , , `(*SKIP)` " true-assertion" . `(*FAIL)` .
- `(*SKIP)` `(*SKIP)` `(*SKIP)` . `(*SKIP)` .

(Regex) : https://riptutorial.com/ko/python/topic/632/---regex-

154: ,

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)]
```

max, min.

`max min` .

```
min([])
```

ValueError : min () arg .

Python 3 `default` `default` .

```
max([], default=42)
# Output: 42
max([], default=0)
# Output: 0
```

:

`sorted` `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` , `dict` `OrderedDict` .

```
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 . .

(iterable) sorted .

```
min([2, 7, 5])
# Output: 2
sorted([2, 7, 5])[0]
# Output: 2
```

sorted 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 , max sorted . __lt__ , __gt__ , __ge__ , __le__ , __ne__ __eq__ 6 .

```
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
```

.

:

```
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__` `__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__ TypeError .

```
del IntegerContainer.__gt__   # The IntegerContainer no longer implements "greater then"

res = min(alist)
```

TypeError : : IntegerContainer () <IntegerContainer ()

---

functools.total_ordering .total_ordering total_ordering , __eq__ , __ne__ __lt__ , __le__ , __ge__ __gt__ __lt__ . __le__ .

```
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__ . , .

iterable N N

iterable heapq nlargest nsmallest .

```
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 , max sorted key . .

1000 :

```
import heapq
with open(filename) as f:
    longest_lines = heapq.nlargest(1000, f, key=len)
```

f nlargest . . nlargest nlargest () len . len .

1000 .

```
longest_lines = sorted(f, key=len)[1000:]
```

.

, : https://riptutorial.com/ko/python/topic/252/----

155: CLI

hg svn . .

.

hg svn .

```
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
```

:

- .
- .

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 messge
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/ko/python/topic/7701/---cli--

156:

if, elif else Python (True False). Python , .

- <expression> if <conditional> else <expression> #

Examples

if, elif else

`if` , `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
```

`elif` `else if` .

(" ")

. .

- (C, Ruby, Java) , Python "" ( ) .
- ( ) " " .

```
n = 5

"Greater than 2" if n > 2 else "Smaller than or equal to 2"
# Out: 'Greater than 2'
```

. , .

Tenary .

```
n = 5
"Hello" if n > 10 else "Goodbye" if n > 5 else "Good day"
```

.

If

```
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!
```

Else

```
if condition:
    body
else:
    body
```

else False .

```
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!
```

`True` `False` `True` `False`. if-else Python .

`and` `True` . `False` .

```
>>> 1 and 2
2

>>> 1 and 0
0
```

```
>>> 1 and "Hello World"
"Hello World"

>>> "" and "Pancakes"
""
```

---

or True (True).

```
>>> 1 or 2
1

>>> None or 1
1

>>> 0 or []
[]
```

. . :

```
>>> def print_me():
        print('I am here!')
>>> 0 and print_me()
0
```

, print_me . 0 False (False). print_me .

.

2 . - if (a) and (b > 2) . a 0 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` `False` .

-
-
- `0` `0` (: `0L` , `0.0` , `0j`
- : `''` , `""` , `()` , `[]`
- : `{}`
- `__bool__` `__len__` 0 `False`

`True` .

: . , `if foo()` `if foo() is None` `if foo() is None`

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
x == y	0
x> y	1

3. `cmp_to_key(func)` `functools` 3.

Python comprehension .

,

```
[value_false, value_true][<conditional-test>]
```

:

```
>> n = 16
>> print [10, 20][n <= 15]
10
```

`n<=15 False` (0). :

```
[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

inbuilt `__cmp__` 3 : 0, 1, -1, cmp (x, y) . 0 : 1 : x> y -1 : x <y

(, 0), (, 1) (, -1) . :

```
[value_equals, value_greater, value_less][<conditional-test>]
```

, . :

```
[lambda: value_false, lambda: value_true][<test>]()
```

`()` / . .

:

```
count = [lambda:0, lambda:N+1][count==N]()
```

None . aDate aDate .

is None is None .

```
if aDate is None:
    aDate=datetime.date.today()
```

(is None == None is None .)

not None True not None . .

```
if not aDate:
    aDate=datetime.date.today()
```

Pythonic . .

```
aDate=aDate or datetime.date.today()
```

. aDate not None not None , . is None datetime.date.today() aDate .

: https://riptutorial.com/ko/python/topic/1111/

157:

- reduce (, iterable [, initializer])

	iterable ( ). ( )
	iterable . ( )
	. (,)

reduce . .

- *addable* ( ) sum()) :

```
sum([1,2,3])                                    # = 6
```

- str.join :

```
''.join(['Hello', ',', ' World'])               # = 'Hello, World'
```

- next reduce reduce :

```
# 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
```

reduce iterable , .

```
def add(s1, s2):
    return s1 + s2

asequence = [1, 2, 3]

reduce(add, asequence)  # equivalent to: add(add(1,2),3)
# Out: 6
```

add . operator .

```
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` iterable .

```
cumprod = reduce(multiply, asequence, 5)
# Out: 5 * 1 = 5
#      5 * 2 = 10
#      10 * 3 = 30
print(cumprod)
# Out: 30
```

`initializer` `reduce` .

```
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
```

/

`reduce` `iterable` completly iterated `any()` `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/ko/python/topic/328/

158:

- value1 ** value2
- pow (1, 2 [, 3])
- value1 .__ pow __ (value2 [, value3])
- value2 .__ rpow __ (value1)
- operator.pow (1, 2)
- .__ pow __ (value1, value2)
- math.pow (1, 2)
- math.sqrt (1)
- math.exp (1)
- cmath.exp (1)
- math.expm1 (1)

Examples

: math.sqrt () cmath.sqrt

`math` `math.sqrt()` `math.sqrt()` `float`) `float` .

```
import math

math.sqrt(9)                # 3.0
math.sqrt(11.11)            # 3.3331666624997918
math.sqrt(Decimal('6.25'))  # 2.5
```

`math.sqrt()` `complex` `ValueError` .

```
math.sqrt(-10)
```

ValueError :

`math.sqrt(x)` `math.pow(x, 0.5)` `x ** 0.5` . `cmath` + bi `math` . .`sqrt()` :

```
import cmath

cmath.sqrt(4)  # 2+0j
cmath.sqrt(-4) # 2j
```

`j` ? `j` -1 . a + bi a + bj . `a` 2 `2+0j` . `b` 0. `b` `2j` 2 . `2j` `0 + 2j` .

: ** pow ()

`pow` -function `**` .

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

(2.x) . ** . .

- : int , : int < 0 :

```
2 ** -3
# Out: 0.125 (result is a float)
```

- Python 3.x .
- Python 2.2.0 ValueError .
- : int < 0 float < 0 , : 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 math.pow() math.pow() . pow() -function ** float .

```
import math
math.pow(2, 2)    # 4.0
math.pow(-2., 2)  # 4.0
```

:

```
math.pow(2, 2+0j)
```

TypeError : complex float .

```
math.pow(-2, 0.5)
```

ValueError :

: math.exp () cmath.exp ()

`math` `cmath` Euler : e builtin `pow()` -function `**` -operator `math.exp()` :

```
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` `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.expm1 ()

`math` `math.exp(x)` `cmath.exp(x)` x `math.e ** x - 1` `expm1()` .

```
import math

print(math.e ** 1e-3 - 1)  # 0.0010005001667083846
print(math.exp(1e-3) - 1)  # 0.0010005001667083846
print(math.expm1(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.expm1(1e-15))   # 1.0000000000000007e-15
#                            ^--------------------
```

. 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.expm1(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__ __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__() 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
```

math float float-computation .

```
import math

math.pow(Integer(2), 0.5) # 1.4142135623730951
```

```
# Prints: Using __float__
```

cmath complex complex float .

```
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__()` -method `math` `cmath` :

```
del Integer.__float__  # Deleting __complex__ method

math.sqrt(Integer(2))  # also cmath.exp(Integer(2))
```

TypeError : float .

: pow () 3

`pow()` 3 `pow(a, b, c)` $A^B C$:

```
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 .

, 3 `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]
```

: n

`math.sqrt` `(**)` n .

. 3 1/3 .

```
>>> x = 3
>>> y = x ** 3
>>> y
27
>>> z = y ** (1.0 / 3)
>>> z
3.0
>>> z == x
True
```

, n .

```
x = 2 ** 100
cube = x ** 3
root = cube ** (1.0 / 3)
```

OverflowError : long int float .

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/ko/python/topic/347/-

159:

Examples

, , .

```
#!/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/ko/python/topic/5370/--

160: (abc)

Examples

ABC

.

.

(stub) 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 ?

(ABC) .

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()` `punchline()`) .

```
 >>> sketch = ArgumentClinic()
 >>> sketch.punchline()
NotImplementedError
```

ArgumentClinic . punchline () .

ABC (Abstract Base Class) . .

```
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/ko/python/topic/5442/----abc-

161:

(Stack Overflow) . "ImportError: No module named '??????' , SyntaxError: invalid syntax NameError: name '???' is not defined NameError: name '???' is not defined .

Examples

IndentationErrors (SyntaxErrors)

Python (: FORTRAN, Makefiles, Whitespace ( )) . . .

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 / SyntaxError : .

.

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()
```

: `if` , `else` , `except` , `class` , `method definition` / true `pass` ( ) (`except` :) :

```
def foo():
    pass
```

IndentationError :

```
def foo():
    if ok:
      return "Two != Four != Tab"
        return "i dont care i do whatever i want"
```

. Python `PEP8` .

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 : [operand] : '???' '???'

.

: + . , 'set1' 'tuple1' (+ ing) set . :

```
set1, tuple1 = {1,2}, (3,4)
a = set1 + tuple1
```

(: int string) + .

```
b = 400 + 'foo'
```

.

```
c = ["a","b"] - [1,2]
```

int float .

```
d = 1 + 1.0
```

TypeError : '???' / .

, IndexError 0 (__iterator__ __iter__ __getitem__).).

bar 1 0 . :

```
foo = 1
bar = foo[0]
```

`for x amount[0]` , INT :

```
amount = 10
for x in amount: print(x)
```

TypeError : '???' .

( )

```
foo = "notAFunction"
foo()
```

NameError : '???' .

, ( ). , . `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)
```

LEGB :

LEGB . .

```
Local → Enclosed → Global → Built-in.
```

- **L** ocal : .
- **E** nclosing : .
- **G** lobal : .
- **B** uilt-in : .

:

```
for i in range(4):
    d = i * 2
print(d)
```

`d` . `for` :

```
def noaccess():
    for i in range(4):
        d = i * 2
noaccess()
print(d)
```

```
NameError: name 'd' is not defined
```

AssertError

`assert` . :

```
assert condition
```

:

```
assert condition, message
```

.

```
if __debug__:
    if not condition: raise AssertionError(message)
```

.

: True, False (-O). . .

KeyboardInterrupt

(`Ctrl` + `C` `del`) .

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)
```

`for x ZeroDivisionError . .`

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
```

/ .

3 print .

Python 3.x 3.0

```
 >>> print "hello world"
  File "<stdin>", line 1
    print "hello world"
                      ^
```

```
SyntaxError: invalid syntax
```

`print` `print()` .

```
print("hello world")  # Note this is valid for both Py2 & Py3
```

: https://riptutorial.com/ko/python/topic/9300/--

162: ("with")

, . . .

- "context_manager"("") ("context_manager"("")?) * :

PEP 343 . `try ... finally` . .

PEP with `__enter__()` `__exit__()` .

`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` `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__() __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
```

triple exc_type , exc_value , traceback (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')
```

contextmanager

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__ yield (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") : https://riptutorial.com/ko/python/topic/928/-----with--

163:

`collections` `dict`, `list`, `tuple` `set` `set`. (: `MutableSet` `ItemsView`) .

collections .

1. UserDict
2.
3. UserString

(wrapper) . , *UserDict* *dict* . . . . . .

Examples

.

dict . .

```
import collections
counts = collections.Counter([1,2,3])
```

counts . `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'>, {})
```

default_factory defaultdict .

`defaultdict` `str` , `int` , `list` `dict` default_factory . .

```
>>> str()
''
>>> int()
0
>>> list
[]
```

defaultdict .

```
>>> state_capitals['Alaska']
''
>>> state_capitals
defaultdict(<class 'str'>, {'Alaska': ''})
```

`int` :

```
>>> fruit_counts = defaultdict(int)
>>> fruit_counts['apple'] += 2  # No errors should occur
>>> fruit_counts
default_dict(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

., .

:

```
>>> 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.')
```

.

```
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

iterable (append ()) `deque` . iterable , `deque` .

Deques ("deck" "double-ended queue"). Deques O (1) `deque` thread , .

pop (0) insert (0, v) O (n) .

2.4 .

`maxlen` `None` deques . , `deque` . `deque` . deques Unix tail . .

2.6 : maxlen .

```
>>> from collections import deque
>>> d = deque('ghi')                 # make a new deque with three items
>>> for elem in d:                   # iterate over the deque's elements
...     print elem.upper()
G
H
I

>>> d.append('j')                    # add a new entry to the right side
>>> d.appendleft('f')                # add a new entry to the left side
>>> d                                # show the representation of the deque
deque(['f', 'g', 'h', 'i', 'j'])

>>> d.pop()                          # return and remove the rightmost item
'j'
>>> d.popleft()                      # return and remove the leftmost item
'f'
>>> list(d)                          # list the contents of the deque
['g', 'h', 'i']
>>> d[0]                             # peek at leftmost item
'g'
>>> d[-1]                            # peek at rightmost item
'i'

>>> list(reversed(d))                # list the contents of a deque in reverse
['i', 'h', 'g']
>>> 'h' in d                         # search the deque
True
>>> d.extend('jkl')                  # add multiple elements at once
>>> d
deque(['g', 'h', 'i', 'j', 'k', 'l'])
>>> d.rotate(1)                      # right rotation
>>> d
deque(['l', 'g', 'h', 'i', 'j', 'k'])
>>> d.rotate(-1)                     # left rotation
>>> d
deque(['g', 'h', 'i', 'j', 'k', 'l'])
```

```
>>> deque(reversed(d))               # make a new deque in reverse order
deque(['l', 'k', 'j', 'i', 'h', 'g'])
>>> d.clear()                        # empty the deque
>>> d.pop()                          # cannot pop from an empty deque
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in -toplevel-
    d.pop()
IndexError: pop from an empty deque

>>> d.extendleft('abc')              # extendleft() reverses the input order
>>> d
deque(['c', 'b', 'a'])
```

: https://docs.python.org/2/library/collections.html

.ChainMap

ChainMap **3.3** .

maps , ChainMap ChainMap . dicts .

ChainMap . Django Context . . update() .

ChainMap . . Django Flask POST GET . 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/ko/python/topic/498/-

164: ,

, Python . . . () ( ).

Examples

	ns	
( )	`__main__` ns	
	`__main__` ns	
	ns	
	ns	
`exec`	ns	
`eval()`	ns	
`execfile()`	ns	
`input()`	ns	

, : https://riptutorial.com/ko/python/topic/10741/-------

165: /

. IDEA MODE CTR Python / . :

RSA . SHA-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 : . linux (alt + ctrl + t) CMD (shift + +) . python setup.py install (Windows OS Python Environment)

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_INET, socket.SOCK_STREAM" accept () . "socket.AF_INET, socket.SOCK_DGRAM" setblocking (value) .

:

- (CLIENT) . . : Crypto import Random Crypto.PublicKey import RSA. .

```
random_generator = Random.new().read
        key = RSA.generate(1024,random_generator)
```

```
        public = key.publickey().exportKey()
```

random_generator " **Crypto import Random** " . " **Crypto.PublicKey import RSA** " . **RSA** 1024 . Public .

- (CLIENT) SHA-1 . SHA-1 "import hashlib" . .

```
  hash_object = hashlib.sha1(public)
  hex_digest = hash_object.hexdigest()
```

hash_object hex_digest . hex_digest public Server . . . RSA `server_public_key = RSA.importKey(getpbk)` . getpbk .

- (SERVER) . "os" 16 "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.sha1(encrypto)
 en_digest = en_object.hexdigest()
```

en_digest .

- (SERVER) .

```
 #encrypting session key and public key
 E = server_public_key.encrypt(encrypto,16)
```

.

- () ( ) . ( ) eval () . . .

```
 en = eval(msg)
 decrypt = key.decrypt(en)
 # hashing sha1
 en_object = hashlib.sha1(decrypt) en_digest = en_object.hexdigest()
```

SHA-1 .

:

IDEA MODE_CTR KEY . IDEA.MODE_CTR .

- () IDEA 16 . MODE_CTR . IDEA 40 . . [value : value] . . 0 16 " [: 16]" . [1:17] [16 :] . 3 IDEA.new () IDEA . KEY, IDEA (IDEA.MODE_CTR) counter = . = . counter = . , KEY . = Counter.Util . Counter.Util . .

```
  ideaEncrypt = IDEA.new(key, IDEA.MODE_CTR, counter=lambda : key)
```

IDEA "ideaEncrypt" .

```
eMsg = ideaEncrypt.encrypt(whole)
#converting the encrypted message to HEXADECIMAL to readable eMsg =
eMsg.encode("hex").upper()
```

eMsg . HEXADECIMAL upper () . .

- **()**

. . 16 . 16 . .

```
decoded = newmess.decode("hex")
ideaDecrypt = IDEA.new(key, IDEA.MODE_CTR, counter=lambda: key)
dMsg = ideaDecrypt.decrypt(decoded)
```

.

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.sha1(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 sha1
    en_object = hashlib.sha1(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.sha1(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_recv = threading.Thread(target=recv,args=("------Recieving Message------
",en_digest))
        thread_send.start()
        thread_recv.start()

        thread_send.join()
        thread_recv.join()
        time.sleep(0.5)
    time.sleep(60)
    server.close()
```

/ : https://riptutorial.com/ko/python/topic/8710/----------

166: : __str__ __repr__

(: "Ace of Spaces") `__str__` `__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()` ( ). `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) ace_of_spades Card . ,.

. type id . (16) print . [1]

" " . .

```
string_of_card = str(ace_of_spades)
print(string_of_card)
```

Card .

Card Card Card .

( ) str(ace_of_spades) , , Card .

---

(1)

. __str__ "dunder"( ) "magic".

, __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__ . , .

("return" . str(ace_of_spades) , print . __str__ .

__str__ self .

```
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)
```

16 .

```
[<__main__.Card instance at 0x00000000026F95C8>,
 <__main__.Card instance at 0x000000000273F4C8>,
 <__main__.Card instance at 0x0000000002732E08>]
```

? Card , ?

(2)

, . __str__ .

__repr__ , , "16 " . [2]

? print 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__ __str__ (S) __repr__ (R) .

__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)
```

■■■

str() repr() .

. snuck. __repr__ __str__ str() ( ) Python __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__ . , (, ID) .

__repr__ __str__ . ( ).

, eval-round-trip __repr__ ()

```
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__ __repr__ : https://riptutorial.com/ko/python/topic/4845/--------str------repr---

167:

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
```

. False None .

```
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 . " FAQ" . : :

```
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 3 Python 2 ***Python*** .

.

. 0 . "INDENT" . "INDENT" (`IndentationError`).

, ( ) . "DEDENT" . "DEDENT" .

( ) " " " " .

0 "DEDENT" .

:

```
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>
```

"" "" .

: https://riptutorial.com/ko/python/topic/2597/-

168:

. (value = 1, 2, 3) .

- (1, a, "hello") # a
- () #
- (1,) # 1 . (1) .
- 1, 2, 3 # 3 (1, 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)
```

.

list tuple . , . :

```
>>> 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 hashabe
```

set A dict .

```
{ (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** .

.

```
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` .

one-value .

```
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,)
```

3 * ( ).

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")
```

(iterable) :

```
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 .

```
min(tuple1)
Out: 'a'

min(tuple2)
Out: '1'
```

---

tuple 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/ko/python/topic/927/

169:

(Pygame) , . 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 (count)
- pygame.mixer.get_num_channels ()
- pygame.mixer.set_reserved ()
- pygame.mixer.find_channel (force)
- pygame.mixer.get_busy ()

	.
	(`False` `True`) `find_channel()` ( ) `True` ( ) `False`

Examples

`pip` :

```
pip install pygame
```

`conda` :

```
conda install -c tlatorre pygame=1.9.2
```

: http://www.pygame.org/download.shtml

.

http://www.pygame.org/ .

`pygame.mixer` `pygame` . `mixer` 15 .

`pygame.init()` `pygame` `pygame` `pygame.mixer` .

. . . :

```
pygame.mixer.init(frequency=22050, size=-16, channels=2, buffer=4096)
```

pygame.mixer.get_init() . pygame.mixer.get_init() True , False False . 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.set_reserved(count) . . .

pygame.mixer.find_channel(force) . True False . force False None . force true , .

: https://riptutorial.com/ko/python/topic/8761/-

170:

PyAudio I / O PortAudio Python . PyAudio Python . PyAudio .

1.pyPortAudio / fastaudio : PortAudio v18 API .

2.tkSnack : Tcl / Tk Python .

: stream_callback ( ) . stream_callback .
1, .print
2. ( ) .
3. PortAudio paAbort .
: Stream.read () Stream.write () .
PortAudio .
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()
```

PyAudio ( ) (2) . PyAudio . `callback(<input_data>, <frame_count>, <time_info>, <status_flag>)` `frame_count` / .

pyaudio.Stream.start_stream () (4) . **pyaudio.paComplete** .

(:).

I / O

"" "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.PyAudio ()** (1) PyAudio . portaudio .

pyaudio.PyAudio.open () (2) . **pyaudio.Stream** .

pyaudio.Stream.write () **pyaudio.Stream.read ()** . ()

" *blocking mode* " **pyaudio.Stream.write ()** **pyaudio.Stream.read ()** / / . " " ( )

/ , **pyaudio.Stream.close ()** pyaudio.Stream.stop_stream () . (4)

, **pyaudio.PyAudio.terminate ()** (5) portaudio .

: https://riptutorial.com/ko/python/topic/10627/-

171: HTTP

Examples

HTTP

2.x 2.3

```
python -m SimpleHTTPServer 9000
```

Python 3.x 3.0

```
python -m http.server 9000
```

9000 .

8000 .

-m sys.path .py .

localhost Python .

```
import sys
import BaseHTTPServer
from SimpleHTTPServer import SimpleHTTPRequestHandler

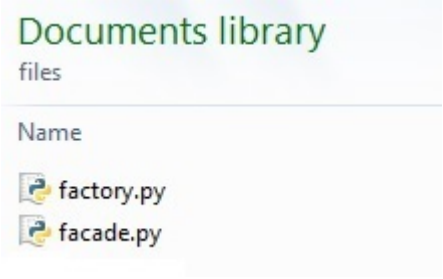
HandlerClass = SimpleHTTPRequestHandler
ServerClass  = BaseHTTPServer.HTTPServer
Protocol     = "HTTP/1.0"

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()
```

.



.

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` .

`SocketServer TCPServer` TCP . (, IP) .

`SimpleHTTPServer SimpleHTTPRequestHandler` .

.

HTTP :

2.x 2.3

-m SimpleHTTPServer 8000

Python 3.x 3.0

-m http.server 8000

'-m' '.py' 'sys.path' .

localhost : 8000 .

SimpleHTTPServer API

`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) .

, Python `9000` `SimpleHTTPServer` . 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 ..

BaseHTTPRequestHandler GET, POST, PUT

```
# 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
```

HTTP : https://riptutorial.com/ko/python/topic/4247/-http-

172: - virtualenv

("virtualenv") Python . env . " A 2.xxx B 2.xxx " .

"virtualenv" libs bins .

Examples

pip / (apt-get) virtualenv :

```
pip install virtualenv
```

```
apt-get install python-virtualenv
```

: sudo .

```
$ cd test_proj
```

:

```
$ virtualenv test_proj
```

.

```
$ source test_project/bin/activate
```

virtualenv "" .

```
$ deactivate
```

Virtualenv

virtualenv bin easy_install virtualenv site-packages . .

```
$ source test_project/bin/activate
$ pip install flask
```

virtualenv site-packages sudo . .

virtualenv

lsvirtualenv : .

cdvirtualenv : .

cdsitepackages : site-packages .

lssitepackages : site-packages .

- virtualenv : https://riptutorial.com/ko/python/topic/9782/-----virtualenv

173:

()

Examples

-

:

```
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 sth / . sth .

HTTP

(http javascript) Python SimpleHTTPServer . Path . .

`python 2` :

```
$ python -m SimpleHTTPServer <portnumber>
```

`python 3` :

```
$ python3 -m http.server <portnumber>
```

`8000` . .

0.0.0.0 8000 HTTP ...

http://hostipaddress:8000/ .

hostipaddress 192.168.xx IP .

ctrl+c. ctrl+c.

TCP

socketserver TCP . .

```
from sockerserver 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'
```

socketserver TCP . . ThreadingTCPServer .

```
from socketserver import ThreadingTCPServer
...
if __name__ == '__main__':
    server = ThreadingTCPServer(('', 5000), EchoHandler)
    server.serve_forever()
```

UDP

UDP socketserver .

:

```
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 .

.

```
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))
```

: https://riptutorial.com/ko/python/topic/1309/-

174:

. . .

Examples

. int, float, complex long.

```
int_num = 10    #int value
float_num = 10.2    #float value
complex_num = 3.14j    #complex value
long_num = 1234567L    #long value
```

. . ., .

```
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'})
```

: https://riptutorial.com/ko/python/topic/9366/--

175:

`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()
```

CPython , GIL , **G L** nterpreter lobal .

Python .

David Beazley (YouTube)

```
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()
```

. VM `GIL` .

`Process.start` `args` `target` . `Process.join` `p1` `p2` .

python plateform . :

- Windows `spawn` .
- 3.3 `fork` .

POSIX fork .

- Unix `forkserver` `multiprocessing.set_start_method` `fork` , `forkserver` `spawn` . `forkserver` `spawn` .

POSIX :

, execve async-signal-safe .
( )

`MainThread` . :

- `MainThread` `Lock` . `fork` .

, `multiprocessing` , (: numOS / accelerated on macOS).

( ) . 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
```

try : except : . (`queue.Empty==True` `queue.Empty==False`) . 'if' Iftry .

```
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 expencive.
        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
```

: https://riptutorial.com/ko/python/topic/3357/-

176:

SSE (Server Sent Events) ( ) , " ". . SSE SSE , . . SSE / .

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()
```

: https://riptutorial.com/ko/python/topic/9100/---

177:

Examples

(?

.

```
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)
```

dis dis.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__)
```

inspect . . .

random randint .

```
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
```

inspect dill.source.getsource dill.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
```

c Python (Mercurial https://www.python.org/downloads/source/) .

```
print(inspect.getsource(sorted)) # raises a TypeError
type(sorted) # <class 'builtin_function_or_method'>
```

: https://riptutorial.com/ko/python/topic/4351/-----

178:

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 (5) KeyboardInterrupt . .
2. except . get_result 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)
```

. .

.

, value None :

```
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")
```

. assert AssertionError catch .

. :

```
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) *dict.get(key, default)* ,

: https://riptutorial.com/ko/python/topic/4700/--

179:

HTTP POST Python

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)
```

(: http https). .

```
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)
```

.

.

json .

```
from requests import post

payload = {'key1' : 'value1', 'key2' : 'value2'}

foo = post('http://httpbin.org/post', json=payload)
```

.read() .

```
from requests import post

files = {'file' : open('data.txt', 'rb')}

foo = post('http://http.org/post', files=files)
```

Filename, content_type .

```
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 Requests .

```
from requests import post
from requests.auth import HTTPDigestAuth

foo = post('http://natas0.natas.labs.overthewire.org', auth=HTTPDigestAuth('natas0',
'natas0'))
```

. .

HTTP . AuthBase .

```
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)
```

HTTPBasicAuth `requests[socks]` .

```
proxies = {
'http': 'socks5://user:pass@host:port',
'https': 'socks5://user:pass@host:port'
}
```

```
foo = requests.post('http://httpbin.org/post', proxies=proxies)
```

: https://riptutorial.com/ko/python/topic/10021/--

180:

- pickle.dump (obj, file, protocol = None, *, fix_imports = True)
- pickle.load (file, *, fix_imports = True, encoding = "ASCII", errors = "strict")

obj	obj
	, pickler , `0` -ASCII, `1` .
	file read () write () `wb` . `rb`

Examples

, , , .

pickle .

`wb` pickle `rb` .

. ,

```
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. , , bytearrays
4. pickable , ,
5.

(def)
6.
7.
8. **dict getstate** ()

```
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'
```

: https://riptutorial.com/ko/python/topic/7810/-

181: (pyserial)

- ser.read (= 1)
- ser.readline ()
- ser.write ()

(: GNU / Linux / dev / ttyUSB0) Windows COM3.
baudrate : int : 9600 : 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200

pyserial documentation .

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
```

.

(pyserial) : https://riptutorial.com/ko/python/topic/5744/----pyserial-

182:

pypa setup-tools `setup.py` .

Examples

`setup.py` .

.

```
+-- package_name
|       |
|       +-- __init__.py
|
+-- setup.py
```

`__init__.py def foo(): return 100` `def 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 . 1 .

```
$ python setup.py register -r pypitest
```

:

```
$ twine upload dist/* -r pypitest
```

.

. testpypi PyPI `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
```

. PyPI API . . testpypi .

PyPI

`twine` .

```
$ pip install twine
```

PyPI .

```
$ python setup.py register -r pypi
$ twine upload dist/*
```

! .

.

. PyPi reStructuredText .

Readme

`README.rst` . PyPi .

`setup.cfg` `setup.cfg` .

```
[metadata]
description-file = README.rst
```

Markdown PyPi .

OpenSource `LICENSE.txt` .

TL; DR .

`__main__.py` .

`__main__.py package_name` . .

```
python -m package_name
```

`__main__.py` `__main__.py` .

python : `package_name.__main__;` `package_name.__main__;` 'package_name' .

: https://riptutorial.com/ko/python/topic/1381/--

183:

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

OpenPyXL `xlsx/xlsm/xltx/xltm` .

:

```
import openpyxl as opx
#To change an existing wookbook 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.sheets

```
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 rows Cell , Cell.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')
```

xlsxwriter Excel

```
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'})

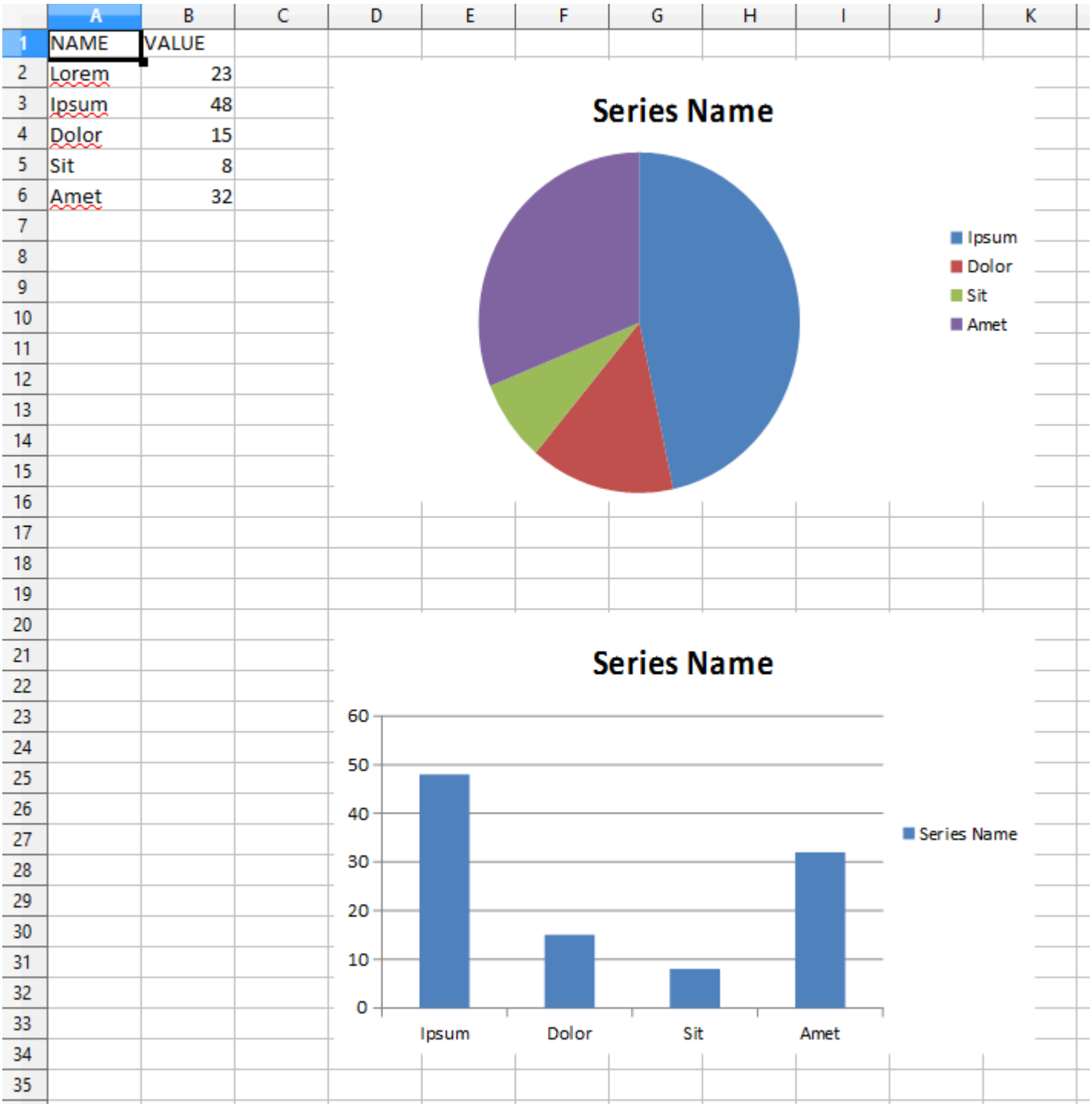
# add series to pie chart
pie_chart.add_series({
    'name': 'Series Name',
    'categories': '=Sheet1!$A$3:$A$%s' % row_,
    'values': '=Sheet1!$B$3:$B$%s' % row_,
    'marker': {'type': 'circle'}
})
# insert pie chart
worksheet.insert_chart('D2', pie_chart)

# 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()
```

:



xlrd Excel

Python xlrd Microsoft Excel (tm) .

:-

```
pip install xlrd
```

pypi setup.py .

https://pypi.python.org/pypi/xlrd

Excel : - xlrd open_workbook () Excel .

```
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
```

.

```
sheets = book.sheet_names()
cur_sheet = book.sheet_by_name(sheets[0])
```

xlsxwriter Excel

```
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()
```

: https://riptutorial.com/ko/python/topic/2986/-

184: (Hashable)

Examples

Mutable Immutable

. .

Immutables

. .

, , , , , , .

`id` . . id . . *(, , ...)*

```
>>> a = 1
>>> id(a)
140128142243264
>>> a += 2
>>> a
3
>>> id(a)
140128142243328
```

, 1 3 ... . , .

```
>>> 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 += ","
```

, - . .

,, .

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 a second ...

```
>>> id(c) == id(b)
True
```

.c b .c b.

. ?

```
>>> ll = [ [] ]*4 # Create a list of 4 lists to contain our results
>>> ll
[[], [], [], []]
>>> ll[0].append(23) # Add result 23 to first list
>>> ll
[[23], [23], [23], [23]]
>>> # Oops...
```

.

mutability . . . .

```
>>> 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 . , lin . a . , lin a . lin , a . a b .

.

```
>>> 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)
```

, tin . a . , tin a a . tin .

```
>>> def yoda(prologue, sentence):
    sentence.reverse()
    prologue += " ".join(sentence)
    return prologue

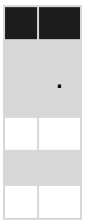
>>> focused = ["You must", "stay focused"]
>>> saying = "Yoda said: "
>>> yoda_sentence = yoda(saying, focused)
```

: *reverse* .

? ? ? , saying ? focused ? ?

(Hashable) : https://riptutorial.com/ko/python/topic/9182/-----hashable---

185:



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
```

: https://riptutorial.com/ko/python/topic/5709/---

186:

. . , , .

Examples

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
```

Filter . True .

```
arr=[1,2,3,4,5,6]
[i for i in filter(lambda x:x>4,arr)]    # outputs[5,6]
```

: https://riptutorial.com/ko/python/topic/9552/--

187:

Curses Python ( ) . TUI .

C 'ncurses' .

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 () .

curses `wrapper(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)
```

curses `stdscr` , WindowObject `stdscr` `func` . `func` `wrapper` .

: https://riptutorial.com/ko/python/topic/5851/--

188:

Examples

Matplotlib

Matplotlib Python .

matplotlib SO Docs .

Matplotlib .

- , matplotlib MATLAB `pyplot` `pyplot` .
- , matplotlib (, ,) . .

`pyplot` `pyplot` .

```
import matplotlib.pyplot as plt

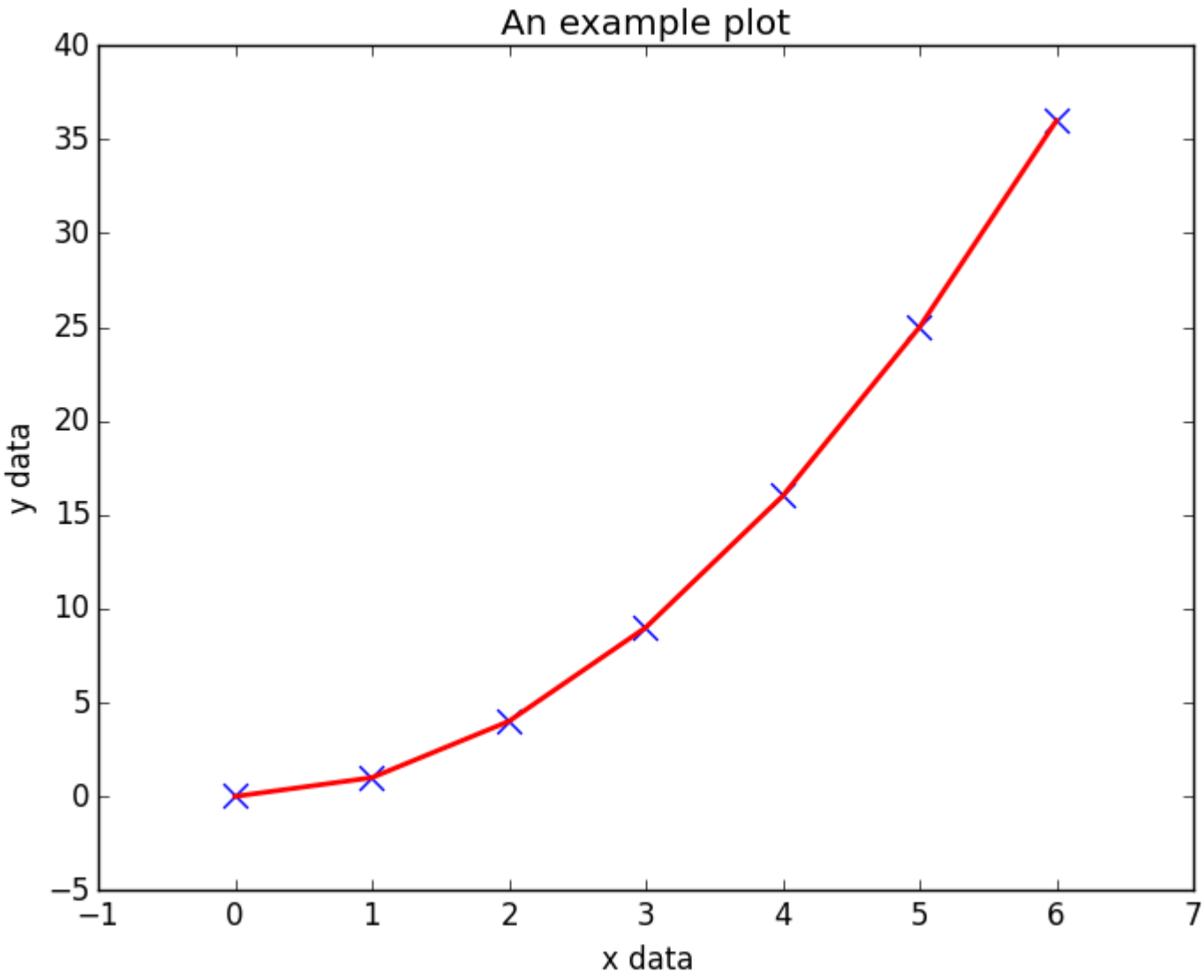
# Generate some data for plotting.
x = [0, 1, 2, 3, 4, 5, 6]
y = [i**2 for i in x]

# 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()
```



`plt.show()` `matplotlib.pyplot` , `plt.show(block=True)` `plt.show(block=True)` `plt.show(block=True)`

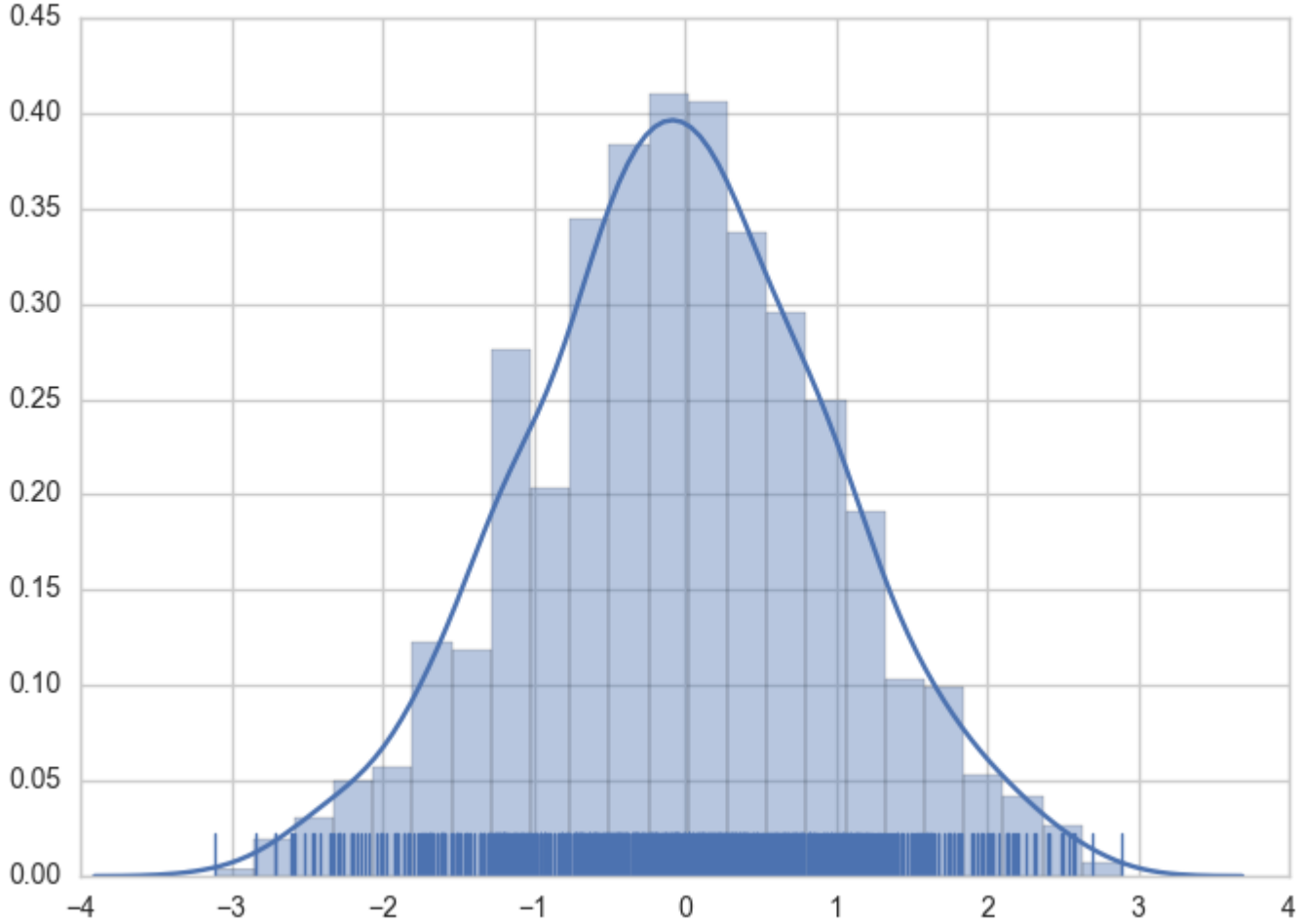
Seaborn Matplotlib . , . Seaborn API .

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)

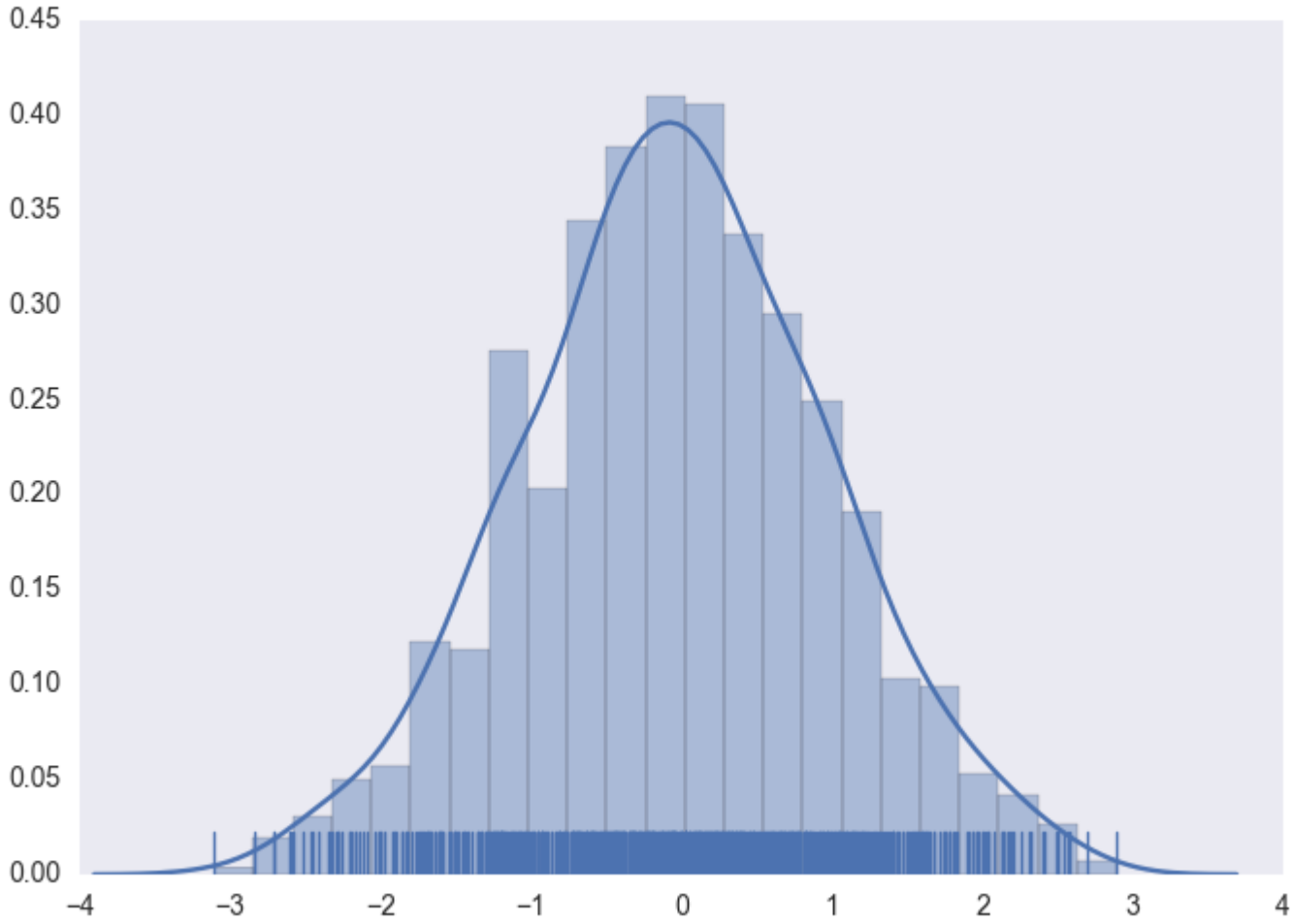
# Plot a histogram with both a rugplot and kde graph superimposed
sns.distplot(data, kde=True, rug=True)
```



.

```
# 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)
```

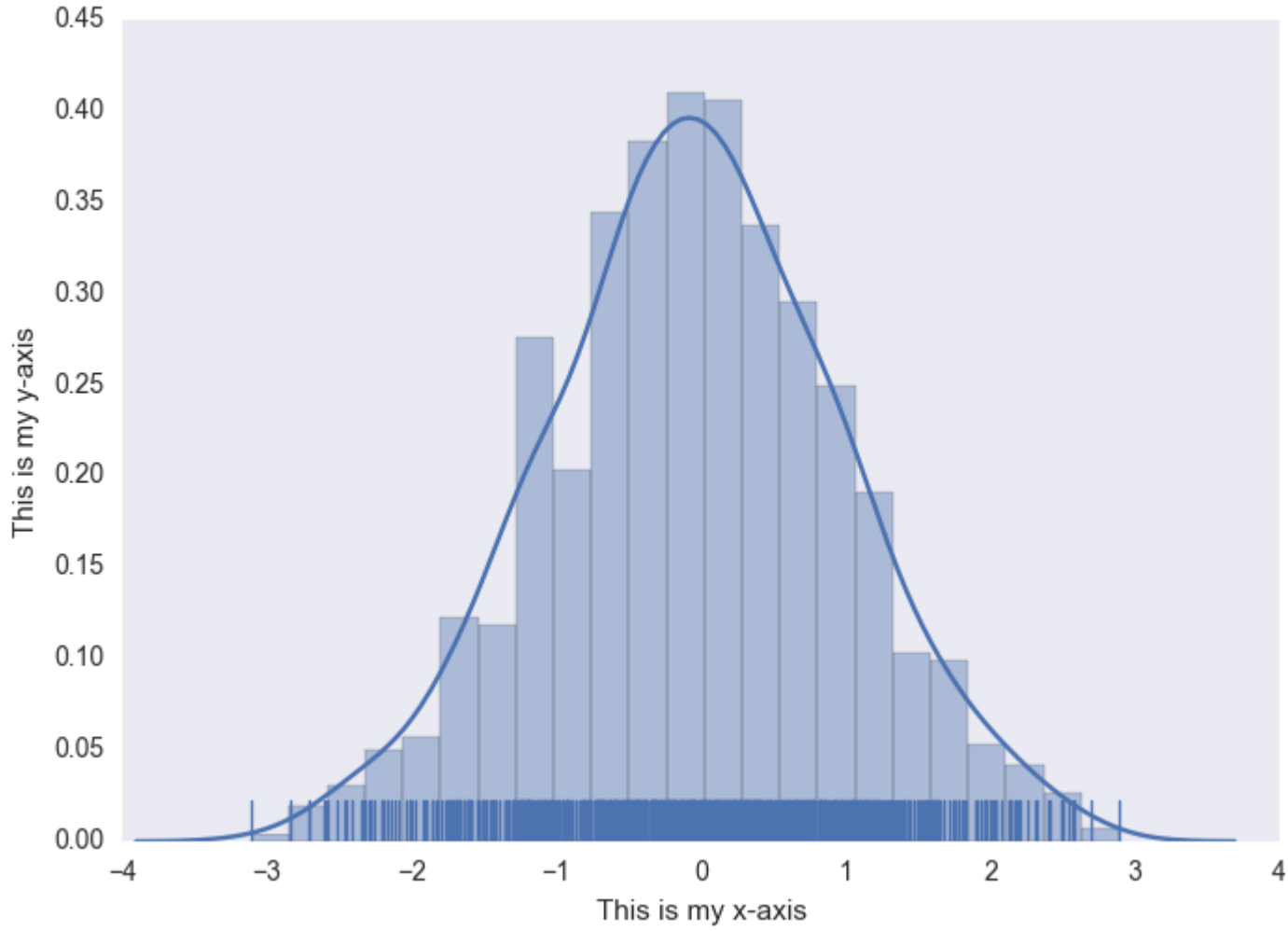


, matplotlib Seaborn . .

```
# 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 3D . VTK . **MayaVI** VTK 3 . . Matplotlib **VTK** .

MayaVI Python 2.7x ! Python 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

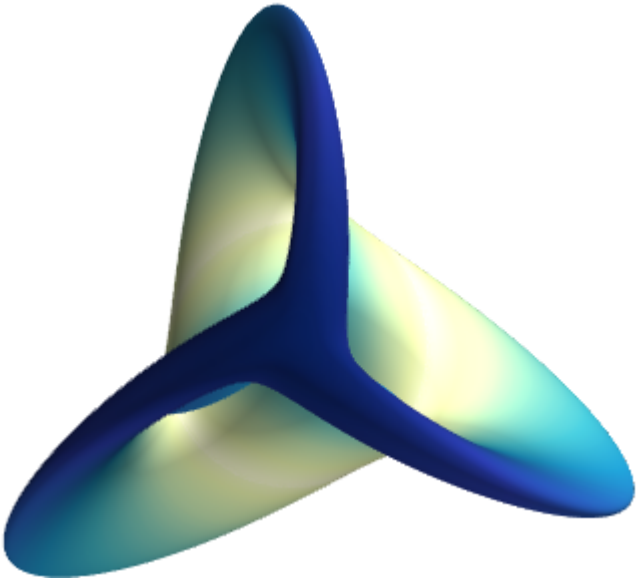
mlab.figure(fgcolor=(0, 0, 0), bgcolor=(1, 1, 1))
u, v = mgrid[- 0.035:pi:0.01, - 0.035:pi:0.01]

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 Python** , **R** , **JavaScript** , **Julia MATLAB** . . .

. . , .

. .

250 . , . .

.

:

```
import plotly.graph_objs as go
import plotly as ply

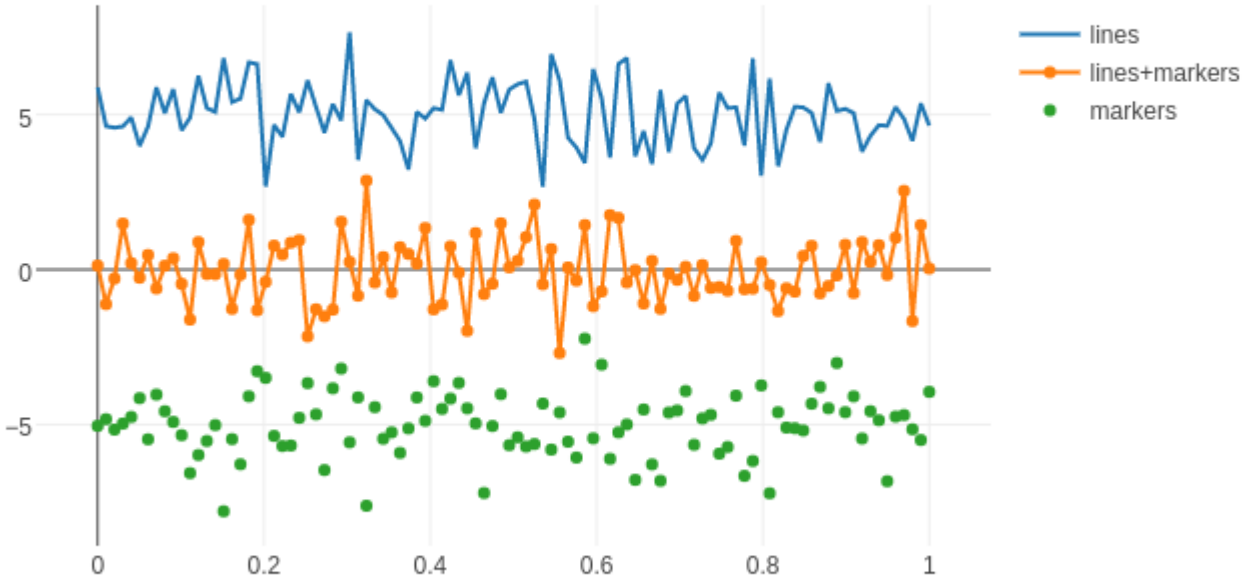
# Create random data with numpy
import numpy as np

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')
```



: https://riptutorial.com/ko/python/topic/2388/--

189:

" . . .

( )

requests

HTTP .

requests-cache

`requests` ; . . , (? ...?) .

scrapy

. `requests` .

selenium

Selenium WebDriver Python . `requests` HTTP . `requests` , JavaScript .

HTML

BeautifulSoup

(Python HTML , `html5lib` , `lxml` `lxml.html`) HTML XML .

lxml

HTML XML . CSS XPath HTML .

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()
```

. `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,
        }
```

spider `projectName\spiders` . - `projectName\spiders\stackoverflow_spider.py` .

. , ( ).

```
scrapy crawl stackoverflow
```

Scrapy

Scrapy (`"Scrapy/VERSION (+http://scrapy.org)"`) . **settings.py** .

```
#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

. . Selenium .

```
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 votes\n-----------\n" % (question_title, question_excerpt,
question_vote)
```

. , , , , JavaScript .

urllib.request

urllib.request .

```
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')
```

: https://riptutorial.com/ko/python/topic/1792/--

190:

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
```

$ $. **Substitute** (mapping, keywords) .

. price qty . . .

"$" . .

```
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))
```

.

: https://riptutorial.com/ko/python/topic/6029/-

191: I / O

, . Python , , . Python .

- file_object = open ([, access_mode] [,])

open() encoding . .

linux utf-8 , MAC Windows .

.

```
import sys
sys.getdefaultencoding()
```

.

.

```
with open('somefile.txt', 'r', encoding='UTF-8') as f:
    for line in f:
        print(line)
```

Examples

mode . .

- 'r' .. . .
- 'w' . .
- 'a' a'- . . .
- 'rb' . 2 r . .
- 'r+' - . r w .
- 'rb+' - . 2 r+ .
- 'wb' . 2 w .
- 'w+' - . r+ . .

- `'wb+'` - . `w+` .
- `'ab'` . 2 `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 +	w	w +		a +
	✔	✔	✘	✔	✘	✔
	✘	✔	✔	✔	✔	✔
.	✘	✘	✔	✔	✔	✔
	✘	✘	✔	✔	✘	✘

---

3 `exclusive creation` .

- `'x'` `FileExistsError` `FileExistsError` .
- `'xb'` . `x` .
- `'x+'` - . `w+` . `FileExistsError` .
- `'xb+'` - . `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
```

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)
```

for readline () .

, 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 / O with with . .

```
with open('myfile.txt') as in_file:
    content = in_file.read()
```

```
print(content)
```

, `with` `close` :

```
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` .

1 2 3 4

.

```
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` .

`encoding` `open` .

```
with open('my_file.txt', 'w', encoding='utf-8') as f:
    f.write('utf-8 text')
```

print . 2 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
```

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
```

, .

```
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 `try` .

```
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
```

. .

- `os` :

```
import os
os.path.isfile('/path/to/some/file.txt')
```

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
```

0 13 20 . 1 0

`next()` .

`readline()` `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
```

. .

```
import os
def is_empty_file(fpath):
    return os.path.isfile(fpath) and os.path.getsize(fpath) > 0
```

`bool` .

I / O : https://riptutorial.com/ko/python/topic/267/---i---o

192:

tarball, zip gzip Python tarfile, zipfile gzip . tarfile tarball `TarFile.extractall(path=".", members=None)` . Python zipfile ZIP `ZipFile.extractall([path[, members[, pwd]]])` . gzip GzipFile .

Examples

Python ZipFile.extractall () ZIP .

```
file_unzip = 'filename.zip'
unzip = zipfile.ZipFile(file_unzip, 'r')
unzip.extractall()
unzip.close()
```

TarFile.extractall () tarball

```
file_untar = 'filename.tar.gz'
untar = tarfile.TarFile(file_untar)
untar.extractall()
untar.close()
```

: https://riptutorial.com/ko/python/topic/9505/--

193: : .

Examples

▪

n , m ,

```
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
```

.

.

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 tranform each group using the logic defined above
orders_df['number_of_orders_per_cient'] = (               # Put the results into a new column
that is called 'number_of_orders_per_cient'
                     orders_df                            # Take the original dataframe
                    .groupby(['customer_id'])['order_id'] # Create a seperate group for each
customer_id & select the order_id
                    .transform(count_number_of_orders))   # Apply the function to each group
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
                        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
```

: . : https://riptutorial.com/ko/python/topic/10947/----------

194: ,

, .

Examples

,

```
# 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)
```

, : https://riptutorial.com/ko/python/topic/8854/----------

195:

. . . .

Examples

(Global Interpreter Lock) . , CPU .

.

```
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
```

`process` `process` 2 4 2 .

.

```
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/ko/python/topic/4110/-

196:

Examples

IPython %%% timeit % timeit

concatanation :

```
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 = 10000000)
10.997665435877963
>>> timeit.timeit('["a"]*100', number = 10000000)
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
html=s.get("https://en.wikipedia.org/").text
     8        50      5306958 106139.2     10.0      sum([pow(ord(x),3.1) for x in
list(html)])
```

.

cProfile ( )

cProfile . timeit .

.

- `ncalls` :
- `tottime` : ( )
- `percall` : . tottime ncalls
- `cumtime` : ( ) . .
- `percall` : cumtime
- `filename:lineno(function)` : .

cProfiler Command Line .

```
$ python -m cProfile main.py
```

:

```
$ python -m cProfile -s time main.py
```

: https://riptutorial.com/ko/python/topic/3818/-

197:

(Flask) Pintrest, Twilio Linkedin Python . Flask .

- @ 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()
```

. **localhost** `127.0.0.1` . **5000** . URL `localhost:5000` `127.0.0.1:5000` ( ). .

`app.run()` **host** , **port** **debug** . `127.0.0.1` `0.0.0.0` URL IP . 5000 `80` 80 . ( ) True . Flask .

```
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=80, debug=True)
```

URL

Flask URL . URL . Flask `www.example.com` `www.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")
```

```
    return "You went to www.example.com/guido-van-rossum"
```

/ users / URL . Flask `@app.route()` `@app.route()` URL `@app.route()` .

```
@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

HTTP **GET** **POST** . Flask HTTP URL . URL . POST ( ) URL GET . DELETE PUT 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"
```

`request` .

```
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"
```

POST `request` .

```
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"]
```

return HTML `render_template()` .

```
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` . , , .

Jinja

Meteor.js Flask . . 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` . flask .

```
from flask import request
```

URL

`request.method` `request.form` `request.args` URL / .

```
@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 `request` .

```
@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
```

URL .

```
@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/ko/python/topic/8682/

198:

Examples

, mixin . mixin .

. Mixins . .

mixin , mixin . .

```
class Mixin1(object):
    def test(self):
        print "Mixin1"

class Mixin2(object):
    def test(self):
        print "Mixin2"

class MyClass(Mixin1, Mixin2):
    pass
```

MyClass test ,

```
>>> obj = MyClass()
>>> obj.test()
Mixin1
```

Order 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__` . . .

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/ko/python/topic/4724/---

199:

Pyglet . . [pyglet.org] [1] . [1] : http://pyglet.org

Examples

Hello World in Pyglet

```
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

:

2 :

```
pip install pyglet
```

3 :

```
pip3 install pyglet
```

Pyglet

```
sound = pyglet.media.load(sound.wav)
sound.play()
```

OpenGL Pyglet

```
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()
```

Pyglet OpenGL

```
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_POINTS` `GL_LINE_LOOP` `GL_LINE_LOOP` .

: https://riptutorial.com/ko/python/topic/8208/

200:

- pickle.dump (, ,) #
- pickle.load (file) #
- pickle.dumps (object, protocol) # .
- pickle.loads (buffer) #

	( )
	bytes

picklable.

- `None` , `True` `False`
- ( )
- ( )
- `tuple` s, `list` s, `set` s, picklable `dict`
-
-
  - `__dict__` `__getstate__()` `__getstate__()` ( ).

Python .

pickle

. (:) .

Examples

`pickle` . . .

`dump()` `load()` .

serialize

```
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)
```

---

deserialize

```
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)
```

---

dumps loads dump load .

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

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

Pickled Data

. .

__getstate__ . picklable .

oposite __setstate__ : __getstate__ __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__` , `__setstate__` `__setstate__` , `__getstate__` `__getstate__` . : `{'important_data': self.important_data}` .

! `a2 pickle.loads A.__init__` `A.__setstate__` `__init__` .

: https://riptutorial.com/ko/python/topic/2606/--

201:

- (,)
- itertools.ifilter (,)
- future_builtins.filter (,)
- itertools.ifilterfalse (,)
- 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` ID .

```
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` (python 3.x) `ifilter` (python 2.x) `or` 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 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
```

filter False .

```
# 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/ko/python/topic/201/

202:

- subprocess.call (args, *, stdin = , stdout = , stderr = , shell = , timeout =)
- subprocess.Popen (args, bufsize = -1, = , stdin = , stdout = , stderr = , preexec_fn = , close_fds = , shell = , cwd = , env = , universal_newlines = , startupinfo = , = 0, restore_signals = True, start_new_session = , pass_fds = ())

args	- 'ls' , ['ls', '-la']
shell	? POSIX /bin/sh .
cwd	.

Examples

subprocess.call . . . .

```
subprocess.call([r'C:\path\to\app.exe', 'arg1', '--flag', 'arg'])
```

, 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
```

---

stdin stdout . .

```
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.
```

stdin stdout communicate() .

:

```
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)
```

1 read 1 . . 0 EOF read ( ).

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/ko/python/topic/1393/--

203:

Python "return" .

Examples

return

, var 1

```
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/ko/python/topic/10883/---

204:

hashlib . FIPS SHA1, SHA224, SHA256, SHA384 SHA512 .

Examples

MD5

. RSA MD5 (RFC 1321) FIPS SHA1, SHA224, SHA256, SHA384 SHA512 (FIPS 180-2) .

. . : `sha1()` SHA1 .

```
hash.sha1()
```

`md5()` , `sha1()` , `sha224()` , `sha256()` , `sha384()` `sha512()` .

`update()` . `digest()` `hexdigest()` .

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

arg . . m.update (a); m.update (b) m.update (a + b) .

```
hash.digest()
```

update () . null ASCII digest_size .

```
hash.hexdigest()
```

digest () 16 double . - .

.

```
>>> import hashlib
>>> m = hashlib.md5()
>>> m.update("Nobody inspects")
>>> m.update(" the spammish repetition")
>>> m.digest()
'\xbbd\x9c\x83\xdd\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()` .

OpenSSL `new()` :

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

: https://riptutorial.com/ko/python/topic/8980/

205:

Examples

C Hello World

C (`hello.c`) `hello` `greet()` .

```
#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

C .

`PyObject_AsFileDescriptor` :

```
PyObject *fobj;
int fd = PyObject_AsFileDescriptor(fobj);
if (fd < 0){
    return NULL;
}
```

`PyFile_FromFd` `PyFile_FromFd` .

```
int fd; /* Existing file descriptor */
PyObject *fobj = PyFile_FromFd(fd, "filename","r",-1,NULL,NULL,NULL,1);
```

C ++ C

C ++ Boost *C* .

C ++

hello.cpp C ++ :

```
#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)
      ;
}
```

. 12.04 python 3.4 gcc . Boost . :

```
sudo apt-get install gcc libboost-dev libpython3.4-dev
```

.so :

```
gcc -shared -o hello.so -fPIC -I/usr/include/python3.4 hello.cpp -lboost_python-py34 -
lboost_system -l:libpython3.4m.so
```

example.py :

```
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/ko/python/topic/557/--

206:

Examples

, heapq nlargest   ,

```
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]
```

nlargest nsmallest ( ). age people .

```
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]
```

: https://riptutorial.com/ko/python/topic/7489/

S. No		Contributors
1	Python	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, deeenes, 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, glS, GoatsWearHats, GThamizh, H. Pauwelyn, hardmooth, hayalci, hichris123, Ian, IanAuld, icesin, Igor Raush, Ilyas Mimouni, itsthejoker, 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, Mattew Whitt, mdegis, Mechanic, Media, mertyildiran, 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, noɥʎd ʌzɐɹƆ, numbermaniac, orvi, Panda, Patrick Haugh, Pavan Nath, Peter Masiar, PSN, PsyKzz, pylang, pzp, Qchmqs, 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, Sнаđoɯƒa, 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, vasili111, Vin, W.Wong, weewooquestionaire,

		Will, wintermute, Yogendra Sharma, Zach Janicki, Zags
2	"pip" : PyPI Package Manager	Zydnar
3	* args ** kwargs	cjds, Eric Zhang, ericmarkmartin, Geeklhem, J F, Jeff Hutchins , Jim Fasarakis Hilliard, JuanPablo, kdopen, loading..., Marlon Abeykoon, Mattew Whitt, Pasha, pcurry, PsyKzz, Scott Mermelstein, user2314737, Valentin Lorentz, Veedrac
4	__name__	Annonymous, BusyAnt, Christian Ternus, jonrsharpe, Lutz Prechelt, Steven Elliott
5	`exec``eval`	Antti Haapala, Ilja Everilä
6	2to3	Alessandro Trinca Tornidor, Dartmouth, Firix, Kevin Brown, Naga2Raja, Stephen Leppik
7	AMQPStorm RabbitMQ	eandersson
8	ArcPy	Midavalo, PolyGeo, Zhanping Shi
9	Asyncio	2Cubed, Alessandro Trinca Tornidor, Cimbali, hiro protagonist, obust, pylang, RamenChef, Seth M. Larson, Simplans, Stephen Leppik, Udi
10	base64	Thomas Gerot
11	C #	Julij Jegorov
12	ChemPy -	Biswa_9937
13	configparser	Chinmay Hegde, Dunatotatos
14	CSV	Adam Matan, Franck Dernoncourt, Martin Valgur, mnoronha, ravigadila, Setu
15	Deque	Anthony Pham, BusyAnt, matsjoyce, ravigadila, Simplans, Thomas Ahle, user2314737
16	dis	muddyfish, user2314737
17	Functools	Alessandro Trinca Tornidor, enrico.bacis, flamenco, RamenChef, Shrey Gupta, Simplans, Stephen Leppik, StuxCrystal
18	groupby ()	Parousia, Thomas Gerot
19	GZip	orvi

20	HTML	alecxe, talhasch
21	Itertools	ADITYA, Alessandro Trinca Tornidor, Andy Hayden, balki, bpachev, Ffisegydd, jackskis, Julien Spronck, Kevin Brown, machine yearning, nlsdfnbch, pylang, RahulHP, RamenChef, Simplans, Stephen Leppik, Symmitchry, Wickramaranga, wnnmaw
22	JSON	Indradhanush Gupta, Leo, Martijn Pieters, pzp, theheadofabroom, Underyx, Wolfgang
23	kivy - NUI	dhimanta
24	Matplotlib	Arun, user2314737
25	os.path	Claudiu, Fábio Perez, girish946, Jmills, Szabolcs Dombi, VJ Magar
26	pip : PyPI	Andy, Arpit Solanki, Community, InitializeSahib, JNat, Mahdi, Majid, Matt Giltaji, Nathaniel Ford, Rápli András, SerialDev, Simplans, Steve Barnes, StuxCrystal, tlo
27	PostgreSQL	Alessandro Trinca Tornidor, RamenChef, Stephen Leppik, user2027202827
28	py.test	Andy, Claudiu, Ffisegydd, Kinifwyne, Matt Giltaji
29	Py2Neo Neo4j Cypher	Wingston Sharon
30	pyautogui	Damien, Rednivrug
31	PyInstaller -	ChaoticTwist, Eric, mnoronha
32	Python 2 Python 3	671620616, Abhishek Kumar, Akshit Soota, Alex Gaynor, Allan Burleson, 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, deeenes, 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'L'l, Mahdi, Martijn Pieters, Martin Thoma, masnun, Matt, Matt Dodge, Matt Rowland, Mattew Whitt, Max Feng, mgwilliams, Michael Recachinas, mkj, mnoronha, Moinuddin Quadri, muddyfish, Nathaniel Ford, niemmi, niyasc, noɥʎdʎzɐɔ, 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
33	Python Lex-Yacc	CLDSEED
34	Python	ADITYA, Antonio, Elodin, Neil A., Vinzee
35	Python Raspberry PI IoT	dhimanta
36	Python SQL Server	metmirr
37	Python Windows	Simon Hibbs
38	setup.py	Adam Brenecki, amblina, JNat, ravigadila, strpeter, user2027202827, Y0da
39	Sqlite3	Chinmay Hegde, Simplans
40	sys	blubberdiblub
41	tempfile NamedTemporaryFile	Alessandro Trinca Tornidor, amblina, Kevin Brown, Stephen Leppik
42	urllib	Amitay Stern, ravigadila, sth, Will
43	virtualenvwrapper	Sirajus Salayhin
44	Windows virtualenvwrapper	Sirajus Salayhin
45	WSGI ( )	David Heyman, Kevin Brown, Preston, techydesigner

46	XML	4444, Brad Larson, Chinmay Hegde, Francisco Guimaraes, greuze, heyhey2k, Rob Murray
47	ZIP	Chinmay Hegde, ghostarbeiter, Jeffrey Lin, SuperBiasedMan
48		Adrian17, Artem Kolontay, ArtOfCode, Bhargav, brennan, Dair, Daniil Ryzhkov, Darkade, Darth Shadow, edwinksl, Fernando, ghostarbeiter, ha_1694, Hans Then, Iancnorden, J F, Majid, Marco Pashkov, Matt Giltaji, Mattew 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
49		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
50		Luca Van Oort, Stephen Leppik
51		Esteis, Marlon Abeykoon, mnoronha, PYPL
52		Andy Hayden, MSeifert, Peter Mølgaard Pallesen, pylang
53		rassar
54		amblina, Braiam, Claudiu, cledoux, Elazar, Gerard Roche, krato, loading..., Marco Pashkov, Or Duan, Pasha, RamenChef , rfkortekaas, Simplans, Thomas Gerot, Topperfalkon, zmo, zondo
55	, : Python JavaScript	user2683246
56		xiaoyi
57	(GIL)	Scott Mermelstein
58		Adriano, Akshat Mahajan, AlexV, Andy, Andy Hayden, Anthony Pham, Arkady, B8vrede, Benjamin Hodgson, btel, CamelBackNotation, Camsbury, Chandan Purohit, ChaoticTwist, Charlie H, Chris Larson, Community, D. Alveno, danidee, DawnPaladin, Delgan, duan, duckman_1991, elegent , Elodin, Emma, EsmaeelE, Ffisegydd, Gal Dreiman, ghostarbeiter, Hurkyl, J F, James, Jeffrey Lin, JGreenwell, Jim Fasarakis Hilliard, jkitchen, Jossie Calderon, Justin, Kevin Brown, L3viathan, Lee Netherton, Martijn Pieters, Martin

		Thurau, Matt Giltaji, Mike - SMT, Mike Driscoll, MSeifert, muddyfish, Murphy4, nd., noɥʎdʎzɐɔ, Pasha, pylang, pzp, Rahul Nair, Severiano Jaramillo Quintanar, Simplans, Slayther, Steve Barnes, Steven Maude, SuperBiasedMan, textshell, then0rTh, Thomas Gerot, user2314737, user3333708, user405 , Utsav T, vaultah, Veedrac, Will, Will, zxxz, λuser
59		Doraemon, GoatsWearHats, J F, JNat, Marco Pashkov, Mark Miller, Martijn Pieters, Nathaniel Ford, Nicolás, pcurry, pzp, SashaZd, SuperBiasedMan, Vilmar
60		bbayles, cizixs, Nemo, pylang, SuperBiasedMan
61		davidism, J F, zmo, Валерий Павлов
62		Dartmouth, rlee827, Thomas Gerot, TidB
63		surfthecity
64		Ajean, alecxe, Andy, Antti Haapala, BusyAnt, Conrad.Dean, Elazar, ghostarbeiter, J F, Jeffrey Lin, jonrsharpe, Kevin Brown , Nicole White, nlsdfnbch, Ohad Eytan, Paul, paulmorriss, proprius, RahulHP, RamenChef, sagism, Simplans, Sirajus Salayhin, Suku, Will
65		Alon Alexander, Nander Speerstra, unutbu, Vinzee, Will
66		boboquack, Buzz, rrao
67		Benedict Bunting, DeepSpace, depperm, Simplans, skrrgwasme, Vinzee
68		Alireza Savand, Ami Tavory, antimatter15, Arpit Solanki, bijancn, Claudiu, Dartmouth, engineercoding, Ffisegydd, J F, JGreenwell, jmunsch, joel3000, Kevin Brown, Kinifwyne, Mario Corchero, Matt Giltaji, Mattew Whitt, mgilson, muddyfish, pylang, strpeter
69		Prem Narain
70		hashcode55, StuxCrystal
71		Devesh Saini, Infinity, rfkortekaas
72		Alessandro Trinca Tornidor, Antonio, bee-sting, cLDSEED, D. Alveno, John Y, LostAvatar, mbsingh, Michel Touw, qwertyuip9 , RamenChef, rrawat, Stephen Leppik, Stephen Nyamweya, sumitroy, user2314737, valeas, zweiterlinde
73		Aldo, B8vrede, joel3000, Sardathrion, Sardorbek Imomaliev,

		Vlad Bezden
74		Charul, denvaar, djaszczurowski
75		Will, XonAether
76		Adriano, Alex L, alfonso.kim, Alleo, Anthony Pham, Antti Haapala, Chris Hunt, Christian Ternus, Darth Kotik, DeepSpace, Delgan, DhiaTN, ebo, Elazar, Eric Finn, Felix D., Ffisegydd, Gal Dreiman, Generic Snake, ghostarbeiter, GoatsWearHats, Guy, Inbar Rose, intboolstring, J F, James, Jeffrey Lin, JGreenwell, Jim Fasarakis Hilliard, jrast, Karl Knechtel, machine yearning, Mahdi, manetsus, Martijn Pieters, Math, Mathias711, MSeifert, pnhgiol, rajah9, Rishabh Gupta, Ryan, sarvajeetsuman, sevenforce, SiggyF, Simplans, skrrgwasme, SuperBiasedMan, textshell, The_Curry_Man, Thomas Gerot, Tom, Tony Suffolk 66, user1349663, user2314737, Vinzee, Will
77	( )	J F, sth, zmo
78		orvi
79		APerson, cfi, Igor Raush, Jon Ericson, Karl Knechtel, Marco Pashkov, MSeifert, noɥʎdʎzɐɔ, Parousia, Simplans, SuperBiasedMan, tlama, user2314737
80		Alu, cʟᴅsᴇᴇᴅ, juggernaut, Kevin Brown, Kristof, mattgathu, Nabeel Ahmed, nlsdfnbch, Rahul, Rahul Nair, Riccardo Petraglia, Thomas Gerot, Will, Yogendra Sharma
81		DeepSpace, James
82		2Cubed, Amir Rachum, Antoine Pinsard, Camsbury, Community, driax, Igor Raush, InitializeSahib, Marco Pashkov, Martijn Pieters, Mattew Whitt, OozeMeister, Pasha, Paulo Scardine, RamenChef, Rob Bednark, Simplans, sisanared, zvone
83		Adriano, Alexander, Anthony Pham, Ares, Barry, blueenvelope, Bosoneando, BusyAnt, Çağatay Uslu, caped114, Chandan Purohit, ChaoticTwist, cizixs, Daniel Porteous, Darth Kotik, deeenes, Delgan, Elazar, Ellis, Emma, evuez, exhuma, Ffisegydd, Flickerlight, Gal Dreiman, ganesh gadila, ghostarbeiter, Igor Raush, intboolstring, J F, j3485, jalanb, James, James Elderfield, jani, jimsug, jkdev, JNat, jonrsharpe, KartikKannapur, Kevin Brown, Lafexlos, LDP, Leo Thumma, Luke Taylor, lukewrites, lxer, Majid, Mechanic, MrP01, MSeifert, muddyfish, n12312, noɥʎdʎzɐɔ, Oz Bar-Shalom, Pasha,

		Pavan Nath, poke, RamenChef, ravigadila, ronrest, Serenity, Severiano Jaramillo Quintanar, Shawn Mehan, Simplans, sirin, solarc, SuperBiasedMan, textshell, The_Cthulhu_Kid, user2314737, user6457549, Utsav T, Valentin Lorentz, vaultah , Will, wythagoras, Xavier Combelle
84		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, wnnmaw
85		DeepSpace, Simplans, tjohnson
86		3442, 4444, acdr, Ahsanul Haque, Akshay Anand, Akshit Soota, Alleo, Amir Rachum, André Laszlo, Andy Hayden, Ankit Kumar Singh, Antoine Bolvy, APerson, Ashwinee K Jha, B8vrede, bfontaine, Brian Cline, Brien, Casebash, Celeo, cfi, ChaoticTwist, Charles, Charlie H, Chong Tang, Community, Conrad.Dean, Dair, Daniel Stradowski, Darth Shadow, Dartmouth, David Heyman, Delgan, Dima Tisnek, eenblam, Elazar, Emma, enrico.bacis, EOL, ericdwang, ericmarkmartin, Esteis, Faiz Halde, Felk, Fermi paradox, Florian Bender, Franck Dernoncourt, Fred Barclay, freidrichen, G M, Gal Dreiman, garg10may, ghostarbeiter, GingerHead, griswolf, Hannele, Harry, Hurkyl, IanAuld, iankit, Infinity, intboolstring, J F, J0HN, James, JamesS, Jamie Rees, jedwards, Jeff Langemeier, JGreenwell, JHS, jjwatt, JKillian, JNat, joel3000, John Slegers, Jon, jonrsharpe, Josh Caswell, JRodDynamite, Julian, justhalf, Kamyar Ghasemlou, kdopen, Kevin Brown, KIDJourney, Kwarrtz, Lafexlos, lapis, Lee Netherton, Liteye, Locane, Lyndsy Simon, machine yearning, Mahdi, Marc, Markus Meskanen, Martijn Pieters, Matt, Matt Giltaji, Matt S, Mattew Whitt, Maximillian Laumeister, mbrig, Mirec Miskuf, Mitch Talmadge, Morgan Thrapp, MSeifert, muddyfish, n8henrie, Nathan Arthur, nehemiah, noɥʎdʎzɐɹƆ, Or East, Ortomala Lokni, pabouk, Panda, Pasha, pktangyue, Preston, Pro Q, pylang, R Nar, Rahul Nair, rap-2-h, Riccardo Petraglia, rll, Rob Fagen, rrao, Ryan Hilbert, Ryan Smith, ryanyuyu, Samuel McKay, sarvajeetsuman, Sayakiss, Sebastian Kreft, Shoe, SHOWMEWHATYOUGOT, Simplans, Slayther, Slickytail, solidcell, StuxCrystal, sudo bangbang, Sunny Patel,

		SuperBiasedMan, syb0rg, Symmitchry, The_Curry_Man, theheadofabroom, Thomas Gerot, Tim McNamara, Tom Barron, user2314737, user2357112, Utsav T, Valentin Lorentz, Veedrac, viveksyngh, vog, W.P. McNeill, Will, Will, Wladimir Palant, Wolf, XCoder Real, yurib, Yury Fedorov, Zags, Zaz
87		3442, Akshit Soota, André Laszlo, Andy Hayden, Annonymous , Ari, Bhargav, Chris Mueller, Darth Shadow, Dartmouth, Delgan, enrico.bacis, Franck Dernoncourt, garg10may, intboolstring, Jeff Langemeier, Josh Caswell, JRodDynamite, justhalf, kdopen, Ken T, Kevin Brown, kiliantics, longyue0521, Martijn Pieters, Mattew Whitt, Moinuddin Quadri, MSeifert, muddyfish, noɥʎdʎzɐɔ, pktangyue, Pyth0nicPenguin, Rahul Nair, Riccardo Petraglia, SashaZd, shrishinde, Simplans, Slayther, sudo bangbang, theheadofabroom, then0rTh, Tim McNamara, Udi, Valentin Lorentz, Veedrac, Zags
88		zenlc2000
89	( )	Greg, JakeD
90		Alex Gaynor, Andrzej Pronobis, Anthony Pham, Community, David Robinson, Delgan, giucal, Jim Fasarakis Hilliard, michaelrbock, MSeifert, Nobilis, ppperry, RamenChef, Simplans, SuperBiasedMan
91	CSV	Hriddhi Dey, Thomas Crowley
92		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, wnnmaw
93		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_tvv, ragesz,

		SashaZd, Serenity, serv-inc, Simplans, Slayther, Sometowngeek, SuperBiasedMan, Thomas Gerot, tobias_k, Tony Suffolk 66, UloPe, user2314737, user312016, Vin, zondo
94		Doc, Rahul Nair, SashaZd
95		4444, Conrad.Dean, demonplus, Ilia Barahovski, Pythonista
96		2Cubed, Ahsanul Haque, Akshat Mahajan, Andy Hayden, Arthur Dent, ArtOfCode, Augustin, Barry, Chankey Pathak, Claudiu, CodenameLambda, Community, deeenes, Delgan, Devesh Saini, Elazar, ericmarkmartin, Ernir, ForceBru, Igor Raush, Ilia Barahovski, J0HN, 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
97		Andy, Pavan Nath, RamenChef, Vin
98		Gal Dreiman, Jörn Hees, sxnwlfkk
99		Razik
100	(int, float, str, tuple frozensets)	Alessandro Trinca Tornidor, FazeL, Ganesh K, RamenChef, Stephen Leppik
101		Anthony Pham, davidism, Elazar, Esteis, Mike Driscoll, SuperBiasedMan, user2314737, zvone
102		Akshat Mahajan, Dair, Franck Dernoncourt, J F, Mahdi, nlsdfnbch, Ryan Smith, Vinzee, Xavier Combelle
103		adeora, ArtOfCode, BSL-5, Kevin Brown, matsjoyce, SuperBiasedMan, Thomas Gerot, Wladimir Palant, wrwrwr
104		Adeel Ansari, Bosoneando, bpachev
105		FrankBr
106		boboquack, Brett Cannon, Dair, Ffisegydd, John Zwinck, Severiano Jaramillo Quintanar, Steven Maude
107		Alessandro Trinca Tornidor, JGreenwell, metahost, Pigman168, RamenChef, Stephen Leppik
108	Python	Jacques de Hooge, Squidward

109		Anthony Pham, Ares, Elazar, J F, MSeifert, Shawn Mehan, SuperBiasedMan, Will, Xavier Combelle
110		Abhishek Jain, boboquack, Charles, Gal Dreiman, intboolstring , JakeD, JNat, Kevin Brown, Matías Brignone, nemesisfixx, poke, R Colmenares, Shawn Mehan, Simplans, Thomas Gerot, tmr232, Tony Suffolk 66, viveksyngh
111		Alessandro Trinca Tornidor, Beall619, mnoronha, RamenChef, Stephen Leppik, Sun Qingyao
112	/	naren
113		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, noɥʞɑʎzɐɹƆ, ronrest, Roy Iacob, Shawn Mehan, Simplans, SuperBiasedMan, TehTris, Valentin Lorentz, viveksyngh, Xavier Combelle
114		Biswa_9937
115		A. Ciclet, RamenChef, user2314737
116		Andrzej Pronobis, Andy Hayden, Bahrom, Cimbali, Cody Piersall, Conrad.Dean, Elazar, evuez, J F, James, Or East, pylang, RahulHP, RamenChef, Simplans, user2314737
117		David Cullen, Dev, MattCorr, nlsdfnbch, Rob H, StuxCrystal, textshell, Thomas Gerot, Will
118		Alessandro Trinca Tornidor, Darth Shadow, DhiaTN, J F, Jacques de Hooge, Leo, Martijn Pieters, mnoronha, Priya, RamenChef, Stephen Leppik
119		Elazar, SashaZd, SuperBiasedMan
120		Dan Sanderson, Igor Raush, MSeifert
121		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, lmiguelvargasf, Mailerdaimon, 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
122		Anthony Pham, ArtOfCode, asmeurer, Christofer Ohlsson, Ellis , fredley, ghostarbeiter, Igor Raush, intboolstring, J F, James Elderfield, JGreenwell, MSeifert, niyasc, RahulHP, rajah9, Simplans, StardustGogeta, SuperBiasedMan, yurib
123		Benjamin Hodgson, Elazar, Faiz Halde, J F, Lee Netherton, loading..., Mister Mister
124		Aaron Hall, Akshat Mahajan, Anthony Pham, Antti Haapala, Byte Commander, dermen, Elazar, Ellis, ericmarkmartin, Fermi paradox, Ffisegydd, japborst, Jim Fasarakis Hilliard, jonrsharpe , Justin, kramer65, Lafexlos, LDP, Morgan Thrapp, muddyfish, nico, OrangeTux, pcurry, Pythonista, Selcuk, Serenity, Tejas Jadhav, tobias_k, Vlad Shcherbina, Will
125		ADITYA, boboquack, Chromium, cjds, depperm, Hannes Karppila, JGreenwell, Jonatan, kdopen, OliPro007, orvi, SashaZd, Snađoшƒa, textshell, Thomas Ahle, user2314737
126		bogdanciobanu, Claudiu, Conrad.Dean, Elazar, FazeL, J F, James Elderfield, lukess, muddyfish, Sam Whited, SiggyF, Stephen Leppik, SuperBiasedMan, Xavier Combelle
127		Prem Narain
128		Nemo
129		Andy, Elazar, evuez, Martijn Pieters, techydesigner
130		Adrian Antunez, Alessandro Trinca Tornidor, Alfe, Andy, Benjamin Hodgson, Brian Rodriguez, BusyAnt, Claudiu, driax, Elazar, flazzarini, ghostarbeiter, Ilia Barahovski, J F, Marco Pashkov, muddyfish, noɥʎdʌzɐɹO, Paul Weaver, Rahul Nair, RamenChef, Shawn Mehan, Shiven, Shkelqim Memolla, Simplans, Slickytail, Stephen Leppik, Sudip Bhandari, SuperBiasedMan, user2314737
131		blueberryfields, Comrade SparklePony, frankyjuang, jmunsch, orvi, qwertyuip9, Stephen Leppik, Thomas Gerot
132		Andy Hayden, Darth Shadow, ericmarkmartin, Ffisegydd, Igor Raush, Jonas S, jonrsharpe, L3viathan, Majid, RamenChef, Simplans, Valentin Lorentz

133		Andy, Christian Ternus, JelmerS, JL Peyret, mnoronha, Vinzee
134		MSeifert
135		HoverHell, JGreenwell, MathSquared, SashaZd, Shreyash S Sarnayak
136		Thomas Gerot
137		2Cubed, Stephen Leppik, Tyler Gubala
138		wim
139		Claudiu, KeyWeeUsr
140		Or East
141		alecxe, Annonymous, Antti Haapala, Elazar, Jim Fasarakis Hilliard, Jonatan, RamenChef, Seth M. Larson, Simplans, Stephen Leppik
142		Ani Menon, FunkySayu, MattCorr, SuperBiasedMan, TuringTux
143		Eleftheria, evuez, mnoronha
144		Alleo, amblina, Antoine Bolvy, Bonifacio2, Ffisegydd, Guy, Igor Raush, Jonatan, Martec, MSeifert, MUSR, pzp, RahulHP, Reut Sharabani, SashaZd, Sayed M Ahamad, SuperBiasedMan, theheadofabroom, user2314737, yurib
145		Beall619, Frustrated, Justin, Leon Z., lukewrites, SuperBiasedMan, Valentin Lorentz
146	( )	Aaron Christiansen, David, Elazar, Peter Shinners, ppperry
147		abukaj, ADITYA, Alec, Alessandro Trinca Tornidor, Alex, Antoine Bolvy, Baaing Cow, Bhargav Rao, Billy, bixel, Charles, Cheney, Christophe Roussy, Dartmouth, DeepSpace, DhiaTN, Dilettant, fox, Fred Barclay, Gerard Roche, greatwolf, hiro protagonist, Jeffrey Lin, JGreenwell, Jim Fasarakis Hilliard, Lafexlos, maazza, Malt, Mark, matsjoyce, Matt Dodge, MervS, MSeifert, ncmathsadist, omgimanerd, Patrick Haugh, pylang, RamenChef, Reut Sharabani, Rob Bednark, rrao, SashaZd, Shihab Shahriar, Simplans, SuperBiasedMan, Tim D, Tom Dunbavan, tyteen4a03, user2314737, Will Vousden, Wombatz
148		code_geek, orvi
149		Alessandro Trinca Tornidor, ChaoticTwist, Community, Dair,

		doratheexplorer0911, Emolga, greut, iankit, JGreenwell, jonrsharpe, kefkius, Kevin Brown, Mattew 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
150		Bastian, japborst, JGreenwell, Jossie Calderon, mbomb007, SashaZd, Tyler Crompton
151		Gal Dreiman, Iancnorden, Wayne Werner
152		Anaphory
153	(Regex)	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	,	Antti Haapala, APerson, GoatsWearHats, Mirec Miskuf, MSeifert, RamenChef, Simplans, Valentin Lorentz
155	CLI	Alessandro Trinca Tornidor, anatoly techtonik, Darth Shadow
156		Andy Hayden, BusyAnt, Chris Larson, deepakkt, Delgan, Elazar, evuez, Ffisegydd, Geeklhem, Hannes Karppila, James, Kevin Brown, krato, Max Feng, noɥʎdʎzɐɔ, rajah9, rrao, SashaZd, Simplans, Slayther, Soumendra Kumar Sahoo, Thomas Gerot, Trimax, Valentin Lorentz, Vinzee, wwii, xgord, Zack
157		APerson, Igor Raush, Martijn Pieters, MSeifert
158		Anthony Pham, intboolstring, jtbandes, Luke Taylor, MSeifert, Pasha, supersam654
159		Teepeemm
160	(abc)	Akshat Mahajan, Alessandro Trinca Tornidor, JGreenwell, Kevin Brown, Mattew Whitt, mkrieger1, SashaZd, Stephen Leppik
161		Juan T, TemporalWolf
162	("with")	Abhijeet Kasurde, Alessandro Trinca Tornidor, Andy Hayden, Antoine Bolvy, carrdelling, Conrad.Dean, Dartmouth, David

		Marx, DeepSpace, Elazar, Kevin Brown, magu_, Majid, Martijn Pieters, Matthew, nlsdfnbch, Pasha, Peter Brittain, petrs, Shuo, Simplans, SuperBiasedMan, The_Cthulhu_Kid, Thomas Gerot, tyteen4a03, user312016, Valentin Lorentz, vaultah, λuser
163		asmeurer, Community, Elazar, jmunsch, kon psych, Marco Pashkov, MSeifert, RamenChef, Shawn Mehan, Simplans, Steven Maude, Symmitchry, void, XCoder Real
164	,	Jeremy, Mohammed Salman
165	/	Mohammad Julfikar
166	: __str__ __repr__	Alessandro Trinca Tornidor, jedwards, JelmerS, RamenChef, Stephen Leppik
167		Alessandro Trinca Tornidor, depperm, J F, JGreenwell, Matt Giltaji, Pasha, RamenChef, Stephen Leppik
168		Anthony Pham, Antoine Bolvy, BusyAnt, Community, Elazar, James, Jim Fasarakis Hilliard, Joab Mendes, Majid, Md.Sifatul Islam, Mechanic, mezzode, nlsdfnbch, noɥʎdʎzɐɹƆ, Selcuk, Simplans, textshell, tobias_k, Tony Suffolk 66, user2314737
169		Anthony Pham, Aryaman Arora, Pavan Nath
170		Biswa_9937
171	HTTP	Arpit Solanki, J F, jmunsch, Justin Chadwell, Mark, MervS, orvi , quantummind, Raghav, RamenChef, Sachin Kalkur, Simplans , techydesigner
172	- virtualenv	Vikash Kumar Jain
173		atayenel, ChaoticTwist, David, Geeklhem, mattgathu, mnoronha, thsecmaniac
174		Gavin, lorenzofeliz, Pike D., Rednivrug
175		David Heyman, Faiz Halde, Iván Rodríguez Torres, J F, Thomas Moreau, Tyler Gubala
176		Nick Humrich
177		muddyfish, StuxCrystal, user2314737
178		Alessandro Trinca Tornidor, Annonymous, eenblam, Mahmoud Hashemi, RamenChef, Stephen Leppik
179		Ken Y-N, RandomHash

180		RamenChef, user2728397
181	(pyserial)	Alessandro Trinca Tornidor, Ani Menon, girish946, mnoronha, Saranjith, user2314737
182		Claudiu, KeyWeeUsr, Marco Pashkov, pylang, SuperBiasedMan, Thtu
183		bee-sting, Chinmay Hegde, GiantsLoveDeathMetal, hackvan, Majid, talhasch, user2314737, Will
184	(Hashable)	Cilyan
185		mnoronha, Shijo
186		Imran Bughio, mvis89, Rednivrug
187		4444, Guy, kollery, Vinzee
188		Aquib Javed Khan, Arun, ChaoticTwist, cledoux, Ffisegydd, ifma
189		alecxe, Amitay Stern, jmunsch, mrtuovinen, Ni., RamenChef, Saiful Azad, Saqib Shamsi, Simplans, Steven Maude, sth, sytech, talhasch, Thomas Gerot
190		4444, Alessandro Trinca Tornidor, Fred Barclay, RamenChef, Ricardo, Stephen Leppik
191	I / O	Ajean, Anthony Pham, avb, Benjamin Hodgson, Bharel, Charles, crhodes, David Cullen, Dov, Esteis, ilse2005, isvforall, jfsturtz, Justin, Kevin Brown, mattgathu, MSeifert, nlsdfnbch, Ozair Kafray, PYPL, pzp, RamenChef, Ronen Ness, rrao, Serenity, Simplans, SuperBiasedMan, Tasdik Rahman, Thomas Gerot, Umibozu, user2314737, Will, WombatPM, xgord
192		andrew
193	: .	Dee
194	,	Mark Miller
195		Claudiu, Thomas Gerot
196		J F, keiv.fly, SashaZd
197		Stephen Leppik, Thomas Gerot
198		2Cubed, proprefenetre, pylang, rrao, Simon Hibbs, Simplans

199		Comrade SparklePony, Stephen Leppik
200		J F, Majid, Or East, RahulHP, rfkortekaas, zvone
201		APerson, cfi, J Atkin, MSeifert, rajah9, SuperBiasedMan
202		Adam Matan, Andrew Schade, Brendan Abel, jfs, jmunsch, Riccardo Petraglia
203		naren
204		Mark Omo, xiaoyi
205		Dartmouth, J F, mattgathu, Nathan Osman, techydesigner, ygram
206		ettanany