

 免费电子书

学习

numpy

Free unaffiliated eBook created from
Stack Overflow contributors.

#numpy

.....	1
1: numpy	2
.....	2
.....	2
Examples.....	2
Mac.....	2
Windows.....	3
Linux.....	3
.....	4
RackspaceJupyter.....	4
2: numpy.cross	5
.....	5
.....	5
Examples.....	5
.....	5
.....	5
.....	6
3: numpy.dot	7
.....	7
.....	7
.....	7
Examples.....	7
.....	7
.....	7
out.....	8
.....	8
4:	11
.....	11
Examples.....	11
numpy.savenumpy.load.....	11
5: ndarray	12
.....	

Examples.....	12
.....	12

6: 13

Examples.....	13
.....	13

7: 14

.....	14
.....	14
Examples.....	14
.....	14
.....	15
.....	16
.....	17
.....	18
.....	18
.....	19

..... 20

CSV.....	20
Numpy ndarray.....	21

8: IOnumpy 23

Examples.....	23
numpy.....	23
.....	23
CSVASCII.....	23
CSV.....	24

9: 25

.....	25
Examples.....	25
.....	25
.....	25
.....	25

.....	25
.....	26
10:	27
.....	27
Examples.....	27
np.polyfit.....	27
np.linalg.lstsq.....	27
11: np.linalg	29
.....	29
Examples.....	29
np.solve.....	29
np.linalg.lstsq.....	29
12:	31
Examples.....	31
.....	31
.....	31
.....	32

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

It is an unofficial and free numpy 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 numpy.

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

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

1: numpy

NumPy “numb pie” “numb pea” Python。

1.3.0	2009-03-20
1.4.0	2010-07-21
1.5.0	2010-11-18
1.6.0	2011-05-15
1.6.1	2011-07-24
1.6.2	2012-05-20
1.7.0	2013212
1.7.1	201347
1.7.2	2013-12-31
1.8.0	20131110
1.8.1	2014326
1.8.2	201489
1.9.0	201497
1.9.1	2014112
1.9.2	201531
1.10.0	2015107
1.10.1	20151012
1.10.2	20151214
1.10.4 *	201617
1.11.0	2016529

Examples

Mac

MacNumPypip

```
pip install numpy
```

Conda

CondaWindowsMacLinux

1. Conda Anaconda numpy Miniconda Conda Python Anaconda Miniconda Conda
2. Miniconda `conda install numpy`

Windows

[pypi](#) [Numpy](#) [pip](#) [Windows](#) [Windows](#)

[Christopher Gohkle](#) Python 64 Python 3.5

1. `numpy-1.11.1+mkl-cp35-cp35m-win_amd64.whl`
2. Windows cmd powershell
3. `pip install C:\path_to_download\numpy-1.11.1+mkl-cp35-cp35m-win_amd64.whl`

[Winpython](#) [Anaconda Python distribution](#) [numpy](#)

[conda](#)

1. [conda](#)
2. Windows
3. `conda install numpy`

Linux

[NumPy](#) [Linux](#) [Linux](#)

[Linux](#) [Python 2.x](#) [Python 3.x](#) [NumPy](#) [Ubuntu](#) [Debian](#) [APT](#) [numpy](#)

```
sudo apt-get install python-numpy
sudo apt-get install python3-numpy
```

[zypper](#) [Suse](#) [yum](#) [Fedora](#)

[numpy](#) [Python](#) [Python 2](#) [pip](#) [Python 3](#) [pip3](#)

```
pip install numpy # install numpy for Python 2
pip3 install numpy # install numpy for Python 3
```

[pip](#) [Linux](#) [Python 2](#) [Python 3](#)

```
sudo apt-get install python-pip # pip for Python 2
sudo apt-get install python3-pip # pip for Python 3
```

pip for Python 2 pip3 for Python 3 pip Python. numpy fortran.

numpy Python virtualenv. Ubuntu virtualenv

```
sudo apt-get install virtualenv
```

Python 2 Python 3 virtualenv pip numpy

```
virtualenv venv # create virtualenv named venv for Python 2
virtualenv venv -p python3 # create virtualenv named venv for Python 3
source venv/bin/activate # activate virtualenv named venv
pip install numpy # use pip for Python 2 and Python 3; do not use pip3 for Python3
```

numpy.

```
import numpy as np
```

np numpy. "np" "numpy".

```
x = np.array([1,2,3,4])
```

Rackspace Jupyter

Jupyter. python numpy. Jupyter numpy Rackspace tmpnb.org.

◦ Jupyter Cal Poly San Luis Obispo. Rackspace ◦

tmpnb.org numpy

1. tmpnb.org
2. Welcome to Python.ipynb
3. >> Python 2
4. >> Python 3

numpy <https://riptutorial.com/zh-CN/numpy/topic/823/numpy>

2: numpy.cross

- `numpy.cross(a, b)` *abb*
- `numpy.cross(a, b, axisa=-1)` *aab asta*
- `numpy.cross(a, b, axisa=-1, axisb=-1, axisc=-1)` *abaxisc*
- `numpy.cross(a, b, axis=None)` *bAB*

AB	<code>ab</code> ²³ ◦ ◦ <code>a'b'</code> <code>cross(a,b)</code> <code>ab</code> ◦ <code>b a</code> <code>cross(a,b)</code> <code>ab</code> ◦ <code>ab</code> ◦ <code>cross(a,b)</code> <code>cross(a[0],b[0]),</code> <code>cross(a[1], b[1]), ...</code>
axisa / B	<code>a</code> ◦ <code>axisa</code> <code>cross()</code> <code>a</code> ◦ <code>axisb</code> <code>cross()</code> ; <code>axisc</code> <code>cross()</code> ◦ <code>axisc</code>
	<code>axisa axisb axisc</code> all ◦ <code>axisaxis</code>

Examples

Numpy `cross` ◦ `[1, 0, 0]` `[0, 1, 0]` `[1, 0, 0]` `[0, 0, 1]` ◦ Numpy

```
>>> a = np.array([1, 0, 0])
>>> b = np.array([0, 1, 0])
>>> np.cross(a, b)
array([0, 0, 1])
```

-
- Numpy ◦ `c[c1, c2]` Numpy `[c1, c2, 0]` ◦

```
>>> c = np.array([0, 2])
>>> np.cross(a, c)
array([0, 0, 2])
```

Numpy `ndarray` `dot` `cross`

```
>>> a.cross(b)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'numpy.ndarray' object has no attribute 'cross'
```

3-2-

```
>>> a=np.array([[1,0,0],[0,1,0],[0,0,1]])
>>> b=np.array([1,0,0])
>>> np.cross(a,b)
array([[ 0,  0,  0],
       [ 0,  0, -1],
       [ 0,  1,  0]])
```

`[np.cross(a[0],b[0]), np.cross(a[1],b[1]), np.cross(a[2],b[2])]`

`b3-2-a` ◦ `""` ◦

```
>>> b=np.array([[0,0,1],[1,0,0],[0,1,0]])
>>> np.cross(a,b)
array([[ 0, -1,  0],
       [ 0,  0, -1],
       [-1,  0,  0]])
```

`array([np.cross(a[0],b[0]), np.cross(a[1],b[1]), np.cross(a[2],b[2])])`

`numpy` `a[0,:]` `a[1,:]` `a[2,:]` ◦ `Numpy.cross` `axis` ◦

```
>>> a=np.array([[1,1,1],[0,1,0],[1,0,-1]])
>>> b=np.array([0,0,1])
>>> np.cross(a,b)
array([[ 1, -1,  0],
       [ 1,  0,  0],
       [ 0, -1,  0]])
>>> np.cross(a,b,axis=0)
array([[ 0, -1,  0],
       [ 1, -1,  0],
       [ 0, -1,  0]])
>>> np.cross(a,b,axis=1)
array([[ 1, -1,  0],
       [ 1,  0,  0],
       [ 0, -1,  0]])
```

`axis=1` (`np.cross([1,1,1],b)`, `np.cross([0,1,0],b)`, `np.cross([1,0,-1],b)`) ◦ `axis` ◦ `axis=0`
(`np.cross([1,0,1],b)`, `np.cross([1,1,0],b)`, `np.cross([1,0,-1],b)`) ◦

`axis` `b2` ◦

`axis` `axis` `numpy` ◦ `axis` `numpy` `ab` ◦ `ab`

```
>>> np.cross(a,b,1)
array([[ 1, -1,  0],
       [ 1,  0,  0],
       [ 0, -1,  0]])
>>> np.cross(a,b,1,axis=0)
array([[ 1,  1,  0],
       [-1,  0, -1],
       [ 0,  0,  0]])
>>> np.cross(a,b,1,axis=1)
array([[ 1, -1,  0],
       [ 1,  0,  0],
       [ 0, -1,  0]])
```

`axis=1` `axis` ◦ `axis` ◦ `axis=0` ◦

`axis` `axis` `axis` ◦ `axis` ◦ `axis` `axis` `axis` ◦

[numpy.cross](https://riptutorial.com/zh-CN/numpy/topic/6166/numpy-cross) <https://riptutorial.com/zh-CN/numpy/topic/6166/numpy-cross>

3: numpy.dot

- `numpy.dotabout = None`

	numpy
b	numpy
	numpy

numpy.dot

ab₁-D; out₀

Examples

- `numpy.ndarray`

```
>>> import numpy as np
>>> A = np.ones((4,4))
>>> A
array([[ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.]])
>>> B = np.ones((4,2))
>>> B
array([[ 1.,  1.],
       [ 1.,  1.],
       [ 1.,  1.],
       [ 1.,  1.]])
>>> A.dot(B)
array([[ 4.,  4.],
       [ 4.,  4.],
       [ 4.,  4.],
       [ 4.,  4.]])
```

numpy₀

```
>>> np.dot(A,B)
array([[ 4.,  4.],
       [ 4.,  4.],
       [ 4.,  4.],
       [ 4.,  4.]])
```

numpy₀

```
>>> v = np.array([1,2])
>>> w = np.array([1,2])
```

```
>>> v.dot(w)
5
>>> np.dot(w,v)
5
>>> np.dot(v,w)
5
```

out

numpy dotout = None. . .

```
>>> I = np.eye(2)
>>> I
array([[ 1.,  0.],
       [ 0.,  1.]])
>>> result = np.zeros((2,2))
>>> result
array([[ 0.,  0.],
       [ 0.,  0.]])
>>> np.dot(I, I, out=result)
array([[ 1.,  0.],
       [ 0.,  1.]])
>>> result
array([[ 1.,  0.],
       [ 0.,  1.]])
```

dtypeint.

```
>>> np.dot(I, I, out=result)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: output array is not acceptable (must have the right type, nr dimensions, and be a C-Array)
```

Fortran.

```
>>> result = np.zeros((2,2), order='F')
>>> np.dot(I, I, out=result)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: output array is not acceptable (must have the right type, nr dimensions, and be a C-Array)
```

numpy.dot

```
>>> a
array([[ 1.,  2.],
       [ 3.,  1.]])
>>> b
array([[ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.],
       [ 9., 10., 11., 12., 13., 14., 15., 16.]])
>>> np.dot(a, b)
array([[ 19., 22., 25., 28., 31., 34., 37., 40.],
       [ 12., 16., 20., 24., 28., 32., 36., 40.]])
```

```

>>> b
array([[ 1.,  9.],
       [ 2., 10.],
       [ 3., 11.],
       [ 4., 12.],
       [ 5., 13.],
       [ 6., 14.],
       [ 7., 15.],
       [ 8., 16.]])
>>> np.dot(a, b.T).T
array([[ 19., 12.],
       [ 22., 16.],
       [ 25., 20.],
       [ 28., 24.],
       [ 31., 28.],
       [ 34., 32.],
       [ 37., 36.],
       [ 40., 40.]])

```

einsum

```
>>> np.einsum('...ij,...j', a, b)
```

o

```

>>> a
array([[[ 0,  1],
        [ 2,  3]],
       [[ 4,  5],
        [ 6,  7]],
       [[ 8,  9],
        [10, 11]],
       [[12, 13],
        [14, 15]],
       [[16, 17],
        [18, 19]],
       [[20, 21],
        [22, 23]],
       [[24, 25],
        [26, 27]],
       [[28, 29],
        [30, 31]]])
>>> np.einsum('...ij,...j', a, b)
array([[ 9., 29.],
       [ 58., 82.],
       [123., 151.],
       [204., 236.],
       [301., 337.],
       [414., 454.],
       [543., 587.],
       [688., 736.]])

```

numpy.dot

```

>>> np.diag(np.dot(b,b.T))
array([ 82., 104., 130., 160., 194., 232., 274., 320.])

```

```
>>> (b * b).sum(axis=-1)
array([ 82., 104., 130., 160., 194., 232., 274., 320.])
```

einsum

```
>>> np.einsum('...j,...j', b, b)
array([ 82., 104., 130., 160., 194., 232., 274., 320.])
```

[numpy.dot](https://riptutorial.com/zh-CN/numpy/topic/3198/numpy-dot) <https://riptutorial.com/zh-CN/numpy/topic/3198/numpy-dot>

4:

Numpy。

Examples

`numpy.save``numpy.load`

`np.save``np.load``numpy`

```
import numpy as np

a = np.random.randint(10, size=(3,3))
np.save('arr', a)

a2 = np.load('arr.npy')
print a2
```

<https://riptutorial.com/zh-CN/numpy/topic/10891/>

5: ndarray

- `def __array_prepare__(self, out_arr: ndarray, context: Tuple[ufunc, Tuple, int] = None) -> ndarray: # called on the way into a ufunc`
- `def __array_wrap__(self, out_arr: ndarray, context: Tuple[ufunc, Tuple, int] = None) -> ndarray: # called on the way out of a ufunc`
- `__array_priority__`: int # used to determine which argument to invoke the above methods on when a ufunc is called
- `def __array_finalize__(self, obj: ndarray): # called whenever a new instance of this class comes into existence, even if this happens by routes other than __new__`

Examples

```
class MySubClass(np.ndarray):
    def __new__(cls, input_array, info=None):
        obj = np.asarray(input_array).view(cls)
        obj.info = info
        return obj

    def __array_finalize__(self, obj):
        # handles MySubClass(...)
        if obj is None:
            pass

        # handles my_subclass[...] or my_subclass.view(MySubClass) or ufunc output
        elif isinstance(obj, MySubClass):
            self.info = obj.info

        # handles my_arr.view(MySubClass)
        else:
            self.info = None

    def __array_prepare__(self, out_arr, context=None):
        # called before a ufunc runs
        if context is not None:
            func, args, which_return_val = context

        return super().__array_prepare__(out_arr, context)

    def __array_wrap__(self, out_arr, context=None):
        # called after a ufunc runs
        if context is not None:
            func, args, which_return_val = context

        return super().__array_wrap__(out_arr, context)
```

context func np.add np.add argstuple which_return_val which_return_val

[ndarray https://riptutorial.com/zh-CN/numpy/topic/6431/ndarray](https://riptutorial.com/zh-CN/numpy/topic/6431/ndarray)

6:

Examples

dtype=bool dtype=bool 0 None False True

```
import numpy as np

bool_arr = np.array([1, 0.5, 0, None, 'a', '', True, False], dtype=bool)
print(bool_arr)
# output: [ True  True False False  True False  True False]
```

numpy

```
arr_1 = np.random.randn(3, 3)
arr_2 = np.random.randn(3, 3)

bool_arr = arr_1 < 0.5
print(bool_arr.dtype)
# output: bool

bool_arr = arr_1 < arr_2
print(bool_arr.dtype)
# output: bool
```

<https://riptutorial.com/zh-CN/numpy/topic/6072/>

7:

Ndarrays numpy. Python.

o

Examples

```
np.empty((2,3))
```

o o

```
np.array([0,1,2,3])  
# Out: array([0, 1, 2, 3])
```

```
np.arange(4)  
# Out: array([0, 1, 2, 3])
```

```
np.zeros((3,2))  
# Out:  
# array([[ 0.,  0.],  
#        [ 0.,  0.],  
#        [ 0.,  0.]])
```

```
np.ones((3,2))  
# Out:  
# array([[ 1.,  1.],  
#        [ 1.,  1.],  
#        [ 1.,  1.]])
```

```
np.linspace(0,1,21)  
# Out:  
# array([ 0. ,  0.05,  0.1 ,  0.15,  0.2 ,  0.25,  0.3 ,  0.35,  0.4 ,  
#        0.45,  0.5 ,  0.55,  0.6 ,  0.65,  0.7 ,  0.75,  0.8 ,  0.85,  
#        0.9 ,  0.95,  1.  ])
```

```
np.logspace(-2,2,5)  
# Out:  
# array([ 1.00000000e-02,  1.00000000e-01,  1.00000000e+00,  
#        1.00000000e+01,  1.00000000e+02])
```

```
np.fromfunction(lambda i: i**2, (5,))  
# Out:  
# array([ 0.,  1.,  4.,  9., 16.])  
np.fromfunction(lambda i,j: i**2, (3,3))  
# Out:  
# array([[ 0.,  0.,  0.],  
#        [ 1.,  1.,  1.],  
#        [ 4.,  4.,  4.]])
```

```
x = np.arange(4)
x
#Out: array([0, 1, 2, 3])
```

```
x+10
#Out: array([10, 11, 12, 13])
```

```
x*2
#Out: array([0, 2, 4, 6])
```

```
x+x
#Out: array([0, 2, 4, 6])
```

```
x*x
#Out: array([0, 1, 4, 9])
```

```
x.dot(x)
#Out: 14
```

Python 3.5 @

```
x = np.diag(np.arange(4))
print(x)
'''
  Out: array([[0, 0, 0, 0],
             [0, 1, 0, 0],
             [0, 0, 2, 0],
             [0, 0, 0, 3]])
'''
print(x@x)
print(x)
'''
  Out: array([[0, 0, 0, 0],
             [0, 1, 0, 0],
             [0, 0, 4, 0],
             [0, 0, 0, 9]])
'''
```

o o o

```
#np.append(array, values_to_append, axis=None)
x = np.array([0,1,2,3,4])
np.append(x, [5,6,7,8,9])
# Out: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
x
# Out: array([0, 1, 2, 3, 4])
y = np.append(x, [5,6,7,8,9])
y
# Out: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

hstack o o

vstack o o

```
# np.hstack(tup), np.vstack(tup)
x = np.array([0,0,0])
y = np.array([1,1,1])
z = np.array([2,2,2])
np.hstack(x,y,z)
# Out: array([0, 0, 0, 1, 1, 1, 2, 2, 2])
np.vstack(x,y,z)
# Out: array([[0, 0, 0],
#            [1, 1, 1],
#            [2, 2, 2]])
```

i:j:ki j k° python0

```
x = np.arange(10)
x[0]
# Out: 0

x[0:4]
# Out: array([0, 1, 2, 3])

x[0:4:2]
# Out: array([0, 2])
```

° -1

```
x[-1]
# Out: 9
x[-1:0:-1]
# Out: array([9, 8, 7, 6, 5, 4, 3, 2, 1])
```

°

```
x = np.arange(16).reshape((4,4))
x
# Out:
#      array([[ 0,  1,  2,  3],
#            [ 4,  5,  6,  7],
#            [ 8,  9, 10, 11],
#            [12, 13, 14, 15]])

x[1,1]
# Out: 5

x[0:3,0]
# Out: array([0, 4, 8])

x[0:3, 0:3]
# Out:
#      array([[ 0,  1,  2],
#            [ 4,  5,  6],
#            [ 8,  9, 10]])

x[0:3:2, 0:3:2]
# Out:
#      array([[ 0,  2],
#            [ 8, 10]])
```

```
arr = np.arange(10).reshape(2, 5)
```

`.transpose`

```
arr.transpose()
# Out:
#      array([[0, 5],
#            [1, 6],
#            [2, 7],
#            [3, 8],
#            [4, 9]])
```

`.T`

```
arr.T
# Out:
#      array([[0, 5],
#            [1, 6],
#            [2, 7],
#            [3, 8],
#            [4, 9]])
```

`np.transpose`

```
np.transpose(arr)
# Out:
#      array([[0, 5],
#            [1, 6],
#            [2, 7],
#            [3, 8],
#            [4, 9]])
```

◦ `array.shape`

```
a = np.arange(12).reshape((3,2,2))
a.transpose() # equivalent to a.transpose(2,1,0)
# Out:
#      array([[[ 0,  4,  8],
#             [ 2,  6, 10]],
#            [[ 1,  5,  9],
#             [ 3,  7, 11]])
```

```
a.transpose(2,0,1)
# Out:
#      array([[[ 0,  2],
#             [ 4,  6],
#             [ 8, 10]],
#            [[ 1,  3],
#             [ 5,  7],
#             [ 9, 11]])
```

```
a = np.arange(24).reshape((2,3,4)) # shape (2,3,4)
a.transpose(2,0,1).shape
```

```
# Out:  
# (4, 2, 3)
```

```
arr = np.arange(7)  
print(arr)  
# Out: array([0, 1, 2, 3, 4, 5, 6])
```

```
arr > 4  
# Out: array([False, False, False, False, False,  True,  True], dtype=bool)
```

4

```
arr[arr>4]  
# Out: array([5, 6])
```

```
# Two related arrays of same length, i.e. parallel arrays  
idxs = np.arange(10)  
sqrs = idxs**2  
  
# Retrieve elements from one array using a condition on the other  
my_sqrs = sqrs[idxs % 2 == 0]  
print(my_sqrs)  
# Out: array([0, 4, 16, 36, 64])
```

numpy.reshape numpy.ndarray.reshape

```
print(np.arange(10).reshape((2, 5)))  
# [[0 1 2 3 4]  
#  [5 6 7 8 9]]
```

```
a = np.arange(12)  
a.reshape((3, 4))  
print(a)  
# [ 0  1  2  3  4  5  6  7  8  9 10 11]
```

shapendarray

```
a = np.arange(12)  
a.shape = (3, 4)  
print(a)  
# [[ 0  1  2  3]  
#  [ 4  5  6  7]  
#  [ 8  9 10 11]]
```

ndarrayshape°

shape-1 ° numpy

```
a = np.arange(12)  
print(a.reshape((3, -1)))  
# [[ 0  1  2  3]
```

```
# [ 4  5  6  7]
# [ 8  9 10 11]]
```

```
a = np.arange(12)
print(a.reshape((3, 2, -1)))

# [[[ 0  1]
#    [ 2  3]]
#
#    [[ 4  5]
#     [ 6  7]]
#
#    [[ 8  9]
#     [10 11]]]
```

a.reshape((3, -1, -1))a.reshape((3, -1, -1)) ValueError ◦

Numpy ◦ ◦

```
# Create two arrays of the same size
a = np.arange(6).reshape(2, 3)
b = np.ones(6).reshape(2, 3)

a
# array([0, 1, 2],
#        [3, 4, 5])
b
# array([1, 1, 1],
#        [1, 1, 1])

# a + b: a and b are added elementwise
a + b
# array([1, 2, 3],
#        [4, 5, 6])
```

Numpy ◦ “”◦

```
# Create arrays of shapes (1, 5) and (13, 1) respectively
a = np.arange(5).reshape(1, 5)
a
# array([[0, 1, 2, 3, 4]])
b = np.arange(4).reshape(4, 1)
b
# array([0],
#        [1],
#        [2],
#        [3])

# When multiplying a * b, slices with the same dimensions are multiplied
# elementwise. In the case of a * b, the one and only row of a is multiplied
# with each scalar down the one and only column of b.
a*b
# array([[ 0,  0,  0,  0,  0],
#        [ 0,  1,  2,  3,  4],
#        [ 0,  2,  4,  6,  8],
#        [ 0,  3,  6,  9, 12]])
```

2D3D。

```
# Create arrays of shapes (2, 2, 3) and (2, 3) respectively
a = np.arange(12).reshape(2, 2, 3)
a
# array([[[ 0  1  2]
#         [ 3  4  5]]
#        [[ 6  7  8]
#         [ 9 10 11]]])
b = np.arange(6).reshape(2, 3)
# array([[0, 1, 2],
#        [3, 4, 5]])

# Executing a*b broadcasts b to each (2, 3) slice of a,
# multiplying elementwise.
a*b
# array([[[ 0,  1,  4],
#         [ 9, 16, 25]],
#        [[ 0,  7, 16],
#         [27, 40, 55]]])

# Executing b*a gives the same result, i.e. the smaller
# array is broadcast over the other.
```

-
- 1 ◦ ◦

```
(7, 5, 3) # compatible because dimensions are the same
(7, 5, 3)

(7, 5, 3) # compatible because second dimension is 1
(7, 1, 3)

(7, 5, 3, 5) # compatible because exceeding dimensions are not compared
(3, 5)

(3, 4, 5) # incompatible
(5, 5)

(3, 4, 5) # compatible
(1, 5)
```

-

CSV

```
filePath = "file.csv"
data = np.genfromtxt(filePath)
```

```
data = np.genfromtxt(filePath, dtype='float', delimiter=';', skip_header=1, usecols=(0,1,3) )
```


Numpy ndarray

`numpy.ndarray` n° ndarray

-
- n

```
x = np.arange(15)
# array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
x.shape
# (15,)
```

```
x = np.asarray([[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [10, 11, 12, 13, 14]])
x
# array([[ 0,  1,  2,  3,  4],
#        [ 5,  6,  7,  8,  9],
#        [10, 11, 12, 13, 14]])
x.shape
# (3, 5)
```

```
np.arange(12).reshape([2,3,2])
```

```
x = np.empty([2, 2])
# array([[ 0.,  0.],
#        [ 0.,  0.]])
```

data-type float

```
x = np.empty([2, 2])
# array([[ 0.,  0.],
#        [ 0.,  0.]])

x.dtype
# dtype('float64')
```

numpy

```
x = np.asarray([[1, 2], [3, 4]])
x.dtype
# dtype('int32')
```

numpy ndarray

```
x[1, 1] = 1.5 # assign a float value
x[1, 1]
# 1
# value has been casted to int
x[1, 1] = 'z' # value cannot be casted, resulting in a ValueError
```

◦

```
x = np.asarray([[1, 2], [3, 4]])
# array([[1, 2],
#        [3, 4]])
y = np.asarray([[5, 6]])
# array([[5, 6]])
```

2x21x2。

```
# x + y
array([[ 6,  8],
       [ 8, 10]])
```

y“ ”

```
array([[5, 6],
       [5, 6]])
```

x°

- ndarray [Nndarray](#)
- ndarray 。

<https://riptutorial.com/zh-CN/numpy/topic/1296/>

8: IO numpy

Examples

numpy

```
x = np.random.random([100,100])
x.tofile('/path/to/dir/saved_binary.npy')
y = fromfile('/path/to/dir/saved_binary.npy')
z = y.reshape(100,100)
all(x==z)
# Output:
# True
```

np.loadtxt CSV

```
# File:
# # Col_1 Col_2
# 1, 1
# 2, 4
# 3, 9
np.loadtxt('/path/to/dir/csvlike.txt', delimiter=',', comments='#')
# Output:
# array([[ 1.,  1.],
#        [ 2.,  4.],
#        [ 3.,  9.]])
```

np.fromregex

```
np.fromregex('/path/to/dir/csvlike.txt', r'(\d+),\s(\d+)', np.int64)
# Output:
# array([[1, 1],
#        [2, 4],
#        [3, 9]])
```

CSV ASCII

np.loadtxt np.savetxt ASCII

```
import numpy as np
x = np.random.random([100,100])
np.savetxt("filename.txt", x)
```

```
np.savetxt("filename.txt", x, delimiter=", " ,
           newline="\n", comments="$ ", fmt="%1.2f",
           header="commented example text")
```

```
$ commented example text
0.30, 0.61, 0.34, 0.13, 0.52, 0.62, 0.35, 0.87, 0.48, [...]
```

CSV

`fromfile` - ◦ `tofile`◦

`genfromtxt` - ◦ `skip_header`◦

`loadtxt` - ◦ ◦

`genfromtxtloadtxt` ◦ `genfromtxt`◦

`myfile.csv`

```
#descriptive text line to skip
1.0, 2, 3
4, 5.5, 6

import numpy as np
np.genfromtxt('path/to/myfile.csv',delimiter=',',skiprows=1)
```

```
array([[ 1. ,  2. ,  3. ],
       [ 4. ,  5.5,  6. ]])
```

```
1  2.0000  buckle_my_shoe
3  4.0000  margery_door

import numpy as np
np.genfromtxt('filename', dtype= None)

array([(1, 2.0, 'buckle_my_shoe'), (3, 4.0, 'margery_door')],
      dtype=[('f0', '<i4'), ('f1', '<f8'), ('f2', '|S14')])
```

`dtype` `dtype=None`◦

`file1 2 3 4 5 6 7 8 9 10 11 22 13 14 15 16 17 18 19 20 21 22 23 24`

```
result=np.fromfile(path_to_file,dtype=float,sep="\t",count=-1)
```

IO`numpy` <https://riptutorial.com/zh-CN/numpy/topic/4973/ionumpy>

9:

NumPy random

o

Examples

```
# Generates 5 random numbers from a uniform distribution [0, 1)
np.random.rand(5)
# Out: array([ 0.4071833 ,  0.069167  ,  0.69742877,  0.45354268,  0.7220556 ])
```

random.seed

```
np.random.seed(0)
np.random.rand(5)
# Out: array([ 0.5488135 ,  0.71518937,  0.60276338,  0.54488318,  0.4236548 ])
```

```
prng = np.random.RandomState(0)
prng.rand(5)
# Out: array([ 0.5488135 ,  0.71518937,  0.60276338,  0.54488318,  0.4236548 ])
```

```
# Creates a 5x5 random integer array ranging from 10 (inclusive) to 20 (inclusive)
np.random.randint(10, 20, (5, 5))
```

```
'''
Out: array([[12, 14, 17, 16, 18],
           [18, 11, 16, 17, 17],
           [18, 11, 15, 19, 18],
           [19, 14, 13, 10, 13],
           [15, 10, 12, 13, 18]])
'''
```

```
letters = list('abcde')
```

-

```
np.random.choice(letters, 3)
'''
Out: array(['e', 'e', 'd'],
          dtype='<U1')
'''
```

```
np.random.choice(letters, 3, replace=False)
'''
Out: array(['a', 'c', 'd'],
          dtype='<U1')
'''
```

```
# Choses 'a' with 40% chance, 'b' with 30% and the remaining ones with 10% each
np.random.choice(letters, size=10, p=[0.4, 0.3, 0.1, 0.1, 0.1])

'''
Out: array(['a', 'b', 'e', 'b', 'a', 'b', 'b', 'c', 'a', 'b'],
          dtype='<U1')
'''
```

```
# Generate 5 random numbers from a standard normal distribution
# (mean = 0, standard deviation = 1)
np.random.randn(5)
# Out: array([-0.84423086,  0.70564081, -0.39878617, -0.82719653, -0.4157447 ])

# This result can also be achieved with the more general np.random.normal
np.random.normal(0, 1, 5)
# Out: array([-0.84423086,  0.70564081, -0.39878617, -0.82719653, -0.4157447 ])

# Specify the distribution's parameters
# Generate 5 random numbers drawn from a normal distribution with mean=70, std=10
np.random.normal(70, 10, 5)
# Out: array([ 72.06498837,  65.43118674,  59.40024236,  76.14957316,  84.29660766])
```

numpy.random
numpy.random.poisson
numpy.random.binomial
numpy.random.logistic

```
np.random.poisson(2.5, 5) # 5 numbers, lambda=5
# Out: array([0, 2, 4, 3, 5])

np.random.binomial(4, 0.3, 5) # 5 numbers, n=4, p=0.3
# Out: array([1, 0, 2, 1, 0])

np.random.logistic(2.3, 1.2, 5) # 5 numbers, location=2.3, scale=1.2
# Out: array([ 1.23471936,  2.28598718, -0.81045893,  2.2474899 ,  4.15836878])
```

<https://riptutorial.com/zh-CN/numpy/topic/2060/>

10:

◦

Examples

np.polyfit

\$ $fx = mx + c$ \$◦

```
npoints = 20
slope = 2
offset = 3
x = np.arange(npoints)
y = slope * x + offset + np.random.normal(size=npoints)
p = np.polyfit(x,y,1) # Last argument is degree of polynomial
```

```
import matplotlib.pyplot as plt
f = np.poly1d(p) # So we can call f(x)
fig = plt.figure()
ax = fig.add_subplot(111)
plt.plot(x, y, 'bo', label="Data")
plt.plot(x,f(x), 'b-',label="Polyfit")
plt.show()
```

<https://docs.scipy.org/doc/numpy/reference/generated/numpy.polyfit.html>numpy◦

np.linalg.lstsq

polyfit

```
npoints = 20
slope = 2
offset = 3
x = np.arange(npoints)
y = slope * x + offset + np.random.normal(size=npoints)
```

| $cA b$ | $** 2A b = c$

```
import matplotlib.pyplot as plt # So we can plot the resulting fit
A = np.vstack([x,np.ones(npoints)]).T
m, c = np.linalg.lstsq(A, y)[0] # Don't care about residuals right now
fig = plt.figure()
ax = fig.add_subplot(111)
plt.plot(x, y, 'bo', label="Data")
plt.plot(x, m*x+c, 'r--',label="Least Squares")
plt.show()
```

<https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.lstsq.html>numpy◦

11: np.linalg

1.8 np.linalg.solve: A 3x3

```
np.random.seed(123)
A = np.random.rand(2,3,3)
b = np.random.rand(2,3)
x = np.linalg.solve(A, b)

print np.dot(A[0,:,:], x[0,:])
# array([ 0.53155137,  0.53182759,  0.63440096])

print b[0,:]
# array([ 0.53155137,  0.53182759,  0.63440096])
```

npa : (... , M, M) array_like

Examples

np.solve

```
x0 + 2 * x1 + x2 = 4
           x1 + x2 = 3
x0 +           x2 = 5
```

A * x = b

```
A = np.array([[1, 2, 1],
              [0, 1, 1],
              [1, 0, 1]])
b = np.array([4, 3, 5])
```

np.linalg.solve

```
x = np.linalg.solve(A, b)
# Out: x = array([ 1.5, -0.5,  3.5])
```

A/ Alinalg.solveLinAlgError: Singular matrix

```
A = np.array([[1, 2, 1],
              [2, 4, 2], # Note that this row 2 * the first row
              [1, 0, 1]])
b = np.array([4,8,5])
```

np.linalg.lstsq

np.linalg.lstsq

.

```
x0 + 2 * x1 + x2 = 4
x0 + x1 + 2 * x2 = 3
2 * x0 + x1 + x2 = 5
x0 + x1 + x2 = 4
```

A * x = b

```
A = np.array([[1, 2, 1],
              [1, 1, 2],
              [2, 1, 1],
              [1, 1, 1]])
b = np.array([4, 3, 5, 4])
```

np.linalg.lstsq

```
x, residuals, rank, s = np.linalg.lstsq(A,b)
```

x residuals rankA sA ◦ b lstsqb

```
A = np.array([[1, 2, 1],
              [1, 1, 2],
              [2, 1, 1],
              [1, 1, 1]])
b = np.array([[4, 3, 5, 4], [1, 2, 3, 4]]).T # transpose to align dimensions
x, residuals, rank, s = np.linalg.lstsq(A,b)
print x # columns of x are solutions corresponding to columns of b
#[[ 2.05263158  1.63157895]
# [ 1.05263158 -0.36842105]
# [ 0.05263158  0.63157895]]
print residuals # also one for each column in b
#[ 0.84210526  5.26315789]
```

ranksA ◦

[np.linalg https://riptutorial.com/zh-CN/numpy/topic/3753/np-linalg](https://riptutorial.com/zh-CN/numpy/topic/3753/np-linalg)

12:

Examples

`numpy.where` `true` condition `numpy.nonzero`

```
import numpy as np

a = np.arange(20).reshape(2,10)
# a = array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
#           [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]])

# Generate boolean array indicating which values in a are both greater than 7 and less than 13
condition = np.bitwise_and(a>7, a<13)
# condition = array([[False, False, False, False, False, False, False, False, True, True],
#                   [ True,  True,  True, False, False, False, False, False, False, False]],
#                  dtype=bool)

# Get the indices of a where the condition is True
ind = np.where(condition)
# ind = (array([0, 0, 1, 1, 1]), array([8, 9, 0, 1, 2]))

keep = a[ind]
# keep = [ 8  9 10 11 12]
```

`extract` `condition` `array` `true`

```
# np.extract(condition, array)
keep = np.extract(condition, a)
# keep = [ 8  9 10 11 12]
```

`xywhere` `True` `False`

```
# Set elements of a which are NOT greater than 7 and less than 13 to zero, np.where(condition,
x, y)
a = np.where(condition, a, a*0)
print(a)
# Out: array([[ 0,  0,  0,  0,  0,  0,  0,  0,  8,  9],
#            [10, 11, 12,  0,  0,  0,  0,  0,  0,  0]])
```

o

```
a = np.random.normal(size=10)
print(a)
# [-1.19423121  1.10481873  0.26332982 -0.53300387 -0.04809928  1.77107775
#  1.16741359  0.17699948 -0.06342169 -1.74213078]
b = a[a>0]
print(b)
#[ 1.10481873  0.26332982  1.77107775  1.16741359  0.17699948]
```

<https://riptutorial.com/zh-CN/numpy/topic/6187/>

S. No		Contributors
1	numpy	Andras Deak , Chris Mueller , Code-Ninja , Community , Dux , edwinksl , grovduck , Hammer , hashcode55 , Joshua Cook , karel , Kersten , Mpaul , prasastoadi , rlee827 , sebix , user2314737 , Yassie
2	numpy.cross	bob.sacramento , Mad Physicist
3	numpy.dot	bpachev , paddyg , Shubham Dang
4		obachtos
5	ndarray	Eric
6		Chris Mueller
7		Andras Deak , ayhan , B Samed , B8vrede , Benjamin , DataSwede , Dux , Gwen , Hamlet , Hammer , KARANJ , Keith L , pixatlazaki , Ryan , Sean Easter , Sparkler , The Hagen , TPVasconcelos , user2314737
8	lOnumpy	Alex , atomh33ls , Sparkler
9		amin , ayhan , B8vrede , Dux , Fermi paradox , user2314737
10		Alex
11	np.linalg	Daniel , DataSwede , Fermi paradox , Mahdi , Sean Easter
12		Alex , farleytpm