In [1]:

```
data = ([[ 0.9526, -0.246 , -0.8856], [ 0.5639, 0.2379, 0.9104]])
data
```

Out[1]:

```
[[0.9526, -0.246, -0.8856], [0.5639, 0.2379, 0.9104]]
```

In [2]:

```
import numpy as np
data = np.array([[ 0.9526, -0.246 , -0.8856], [ 0.5639, 0.2379, 0.9104]])
data
```

Out[2]:

```
array([[ 0.9526, -0.246 , -0.8856],
       [ 0.5639,  0.2379,  0.9104]])
```

In [3]:

```
data * 10
```

Out[3]:

```
array([[ 9.526, -2.46 , -8.856],
       [ 5.639,  2.379,  9.104]])
```

In [4]:

```
data.shape
```

Out[4]:

```
(2, 3)
```

In [5]:

```
data.dtype
```

Out[5]:

```
dtype('float64')
```

In [9]:

```
data.ndim
data2 = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
data2.shape
```

Out[9]:

```
(2, 4)
```

In [10]:

```
data2.dtype
```

Out[10]:

```
dtype('int32')
```

In [11]:

```
np.zeros(10)
```

Out[11]:

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [12]:

```
np.zeros((3, 6))
```

Out[12]:

```
array([[0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.]])
```

In [13]:

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

Out[13]:

```
array([[[0., 0.],
        [0., 0.],
        [0., 0.]],

       [[0., 0.],
        [0., 0.],
        [0., 0.]]])
```

In [14]:

```
np.empty((2, 3, 4))
```

Out[14]:

```
array([[[6.23042070e-307, 4.67296746e-307, 1.69121096e-306,
         4.22788479e-307],
        [7.56599807e-307, 8.90104239e-307, 1.24610383e-306,
         1.69118108e-306],
        [8.06632139e-308, 1.20160711e-306, 1.69119330e-306,
         1.37962320e-306]],

       [[6.89812281e-307, 1.24611674e-306, 6.23060065e-307,
         6.89813978e-307],
        [8.90104239e-307, 6.23055651e-307, 8.90104239e-307,
         1.69119602e-306],
        [9.34607074e-307, 1.33511562e-306, 1.11260483e-306,
         8.34451079e-308]]])
```

In [15]:

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

Out[15]:

```
array([[9.526, 2.46 , 8.856],
       [5.639, 2.379, 9.104]])
```

In [16]:

```
np.arange(15)
```

Out[16]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [ ]:

```
## arange: built-in Python range function
```

In [17]:

```
np.arange(10)
```

Out[17]:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [22]:

```
np.ones(5)
```

Out[22]:

```
array([1., 1., 1., 1., 1.])
```

In [23]:

```
np.ones([2,3])
```

Out[23]:

```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

In [25]:

```
np.eye(5)
```

Out[25]:

```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

In [26]:

```
np.identity(4)
```

Out[26]:

```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

In [29]:

```
d1 = np.array([ 3, -1, -2, 0, 12, 10], dtype='int32')
d1.dtype
```

Out[29]:

```
dtype('int32')
```

In [30]:

```
d2 = np.eye(5, k=-1, dtype = float)
d2
```

Out[30]:

```
array([[0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.]])
```

Slicing

d = np.array([2,4,5,7,8,9,12]) d

In [31]:

```
d = np.array([2,4,5,7,8,9,12])
d
```

Out[31]:

```
array([ 2,  4,  5,  7,  8,  9, 12])
```

In [32]:

```
d[:3]
```

Out[32]:

```
array([2, 4, 5])
```

In [33]:

```
d[3:6]
```

Out[33]:

```
array([7, 8, 9])
```

In [34]:

```
d[3] = 20
d
```

Out[34]:

```
array([ 2,  4,  5, 20,  8,  9, 12])
```

Two Dimension

In [35]:

```
d4 = np.array([[1,3,5,7], [2,4,6,8]], float)
d4
```

Out[35]:

```
array([[1., 3., 5., 7.],
       [2., 4., 6., 8.]])
```

In [36]:

```
d4[2,4]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-36-8312d210b330> in <module>
----> 1 d4[2,4]

IndexError: index 2 is out of bounds for axis 0 with size 2
```

In [37]:

```
d4[1,4]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-37-c44ef2e4d9b3> in <module>
----> 1 d4[1,4]

IndexError: index 4 is out of bounds for axis 1 with size 4
```

In [38]:

```
d4[1,3]
```

Out[38]:

```
8.0
```

In [39]:

```
d4[:1,:1]
```

Out[39]:

```
array([[1.]])
```

In [40]:

```
d4
```

Out[40]:

```
array([[1., 3., 5., 7.],
       [2., 4., 6., 8.]])
```

In [41]:

```
d4[:1,:3]
```

Out[41]:

```
array([[1., 3., 5.]])
```

In [42]:

```
d4[0,:3]
```

Out[42]:

```
array([1., 3., 5.])
```

In [43]:

```
d4[0:,:3]
```

Out[43]:

```
array([[1., 3., 5.],
       [2., 4., 6.]])
```

In [44]:

```
d4[0:,3]
```

Out[44]:

```
array([7., 8.])
```

In [45]:

```
d4[:,2]
```

Out[45]:

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

In [46]:

```
d4[-1:, -2:]
```

Out[46]:

```
array([[6., 8.]])
```

"in" statement

In [47]:

```
4 in d4
```

Out[47]:

```
True
```

In [48]:

```
13 in d4
```

Out[48]:

```
False
```

In [49]:

```
d5 = np.array(range(20), float)
```

In [50]:

```
d5.dtype
```

Out[50]:

```
dtype('float64')
```

In [51]:

```
d5
```

Out[51]:

```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11., 12.,
       13., 14., 15., 16., 17., 18., 19.])
```

In [52]:

```
d5 = d5.reshape((4,5))
d5
```

Out[52]:

```
array([[ 0.,  1.,  2.,  3.,  4.],
       [ 5.,  6.,  7.,  8.,  9.],
       [10., 11., 12., 13., 14.],
       [15., 16., 17., 18., 19.]])
```

In [53]:

```
d5 = d5.reshape((5,5))
d5
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-53-6d61686e07f4> in <module>
----> 1 d5 = d5.reshape((5,5))
      2 d5

ValueError: cannot reshape array of size 20 into shape (5,5)
```

In [54]:

```
a = np.array([1,2,3,4], float)
b = a.tolist()
b
```

Out[54]:

```
[1.0, 2.0, 3.0, 4.0]
```

In [55]:

```
c = list(a)
c
```

Out[55]:

```
[1.0, 2.0, 3.0, 4.0]
```

In [56]:

```
print(c)
```

```
[1.0, 2.0, 3.0, 4.0]
```

In [57]:

```
d = a.fill(0)
d
```

In [58]:

```
a
```

Out[58]:

```
array([0., 0., 0., 0.])
```

In [60]:

```
d = np.array(range(20), float).reshape(5,4)
d
```

Out[60]:

```
array([[ 0.,  1.,  2.,  3.],
       [ 4.,  5.,  6.,  7.],
       [ 8.,  9., 10., 11.],
       [12., 13., 14., 15.],
       [16., 17., 18., 19.]])
```

In [61]:

```
d.flatten
```

Out[61]:

```
<function ndarray.flatten>
```

In [62]:

```
d.flatten()
```

Out[62]:

```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11., 12.,
       13., 14., 15., 16., 17., 18., 19.])
```

Concatenation

In [63]:

```
a = np.array([3,5,7], float)
b = np.array([1,11], float)
c = np.array([5,8,9], float)

np.concatenate((a,b,c))
```

Out[63]:

```
array([ 3.,  5.,  7.,  1., 11.,  5.,  8.,  9.])
```

In [64]:

```
x = np.array([[1,2,3], [4,7,9]], float)
y = np.array([[10,12,13], [14,17,19]], float)

np.concatenate((x,y))
```

Out[64]:

```
array([[ 1.,  2.,  3.],
       [ 4.,  7.,  9.],
       [10., 12., 13.],
       [14., 17., 19.]])
```

In [65]:

```
x + y
```

Out[65]:

```
array([[11., 14., 16.],
       [18., 24., 28.]])
```

In [66]:

```
np.concatenate((x,y), axis = 0)
```

Out[66]:

```
array([[ 1.,  2.,  3.],
       [ 4.,  7.,  9.],
       [10., 12., 13.],
       [14., 17., 19.]])
```

In [67]:

```
np.concatenate((x,y), axis = 1)
```

Out[67]:

```
array([[ 1.,  2.,  3., 10., 12., 13.],
       [ 4.,  7.,  9., 14., 17., 19.]])
```

slicing and broadcasting

In [73]:

```
arr = np.arange(10)
a_slice = arr[5:8]
a_slice[2] = 200

arr
```

Out[73]:

```
array([  0,   1,   2,   3,   4,   5,   6, 200,   8,   9])
```

In [71]:

```
a = arr[5:8].copy()
a[2] = 300

arr
```

Out[71]:

```
array([  0,   1,   2,   3,   4,   5,   6, 200,   8,   9])
```

In [72]:

```
a
```

Out[72]:

```
array([  5,   6, 300])
```

In [75]:

```
arr2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
arr2d[2] # picks row 2
```

Out[75]:

```
array([7, 8, 9])
```

In [76]:

```
arr2d[0][2] #picks row 0 col 3
```

Out[76]:

```
3
```

In [84]:

```
np.array(range(15), dtype=float).reshape((3,5))
```

Out[84]:

```
array([[ 0.,  1.,  2.,  3.,  4.],
       [ 5.,  6.,  7.,  8.,  9.],
       [10., 11., 12., 13., 14.]])
```

In [85]:

```
np.identity(5, float)
```

Out[85]:

```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

In [88]:

```
np.arange(6, dtype=float)
```

Out[88]:

```
array([0., 1., 2., 3., 4., 5.])
```

In [98]:

```
np.arange(16, dtype=int).reshape((4,4))
```

Out[98]:

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [119]:

```
dr = np.arange(16, dtype=int).reshape((4,4)).transpose()
dr
```

Out[119]:

```
array([[ 0,  4,  8, 12],
       [ 1,  5,  9, 13],
       [ 2,  6, 10, 14],
       [ 3,  7, 11, 15]])
```

In [103]:

```
dd = np.ones((5,4))
db = np.array([2,2,2,2])

dd + db
```

Out[103]:

```
array([[3., 3., 3., 3.],
       [3., 3., 3., 3.],
       [3., 3., 3., 3.],
       [3., 3., 3., 3.],
       [3., 3., 3., 3.]])
```

In [112]:

```
d6 = np.array([6.7, 2.3, 5.4, 1.192], float)
np.floor(d6)
```

Out[112]:

```
array([6., 2., 5., 1.])
```

In [113]:

```
np.ceil(d6)
```

Out[113]:

```
array([7., 3., 6., 2.])
```

In [114]:

```
np.rint(d6)
```

Out[114]:

```
array([7., 2., 5., 1.])
```

In [115]:

```
d7 = np.e
d7
```

Out[115]:

```
2.718281828459045
```

In [116]:

```
d8 = np.pi
d8
```

Out[116]:

```
3.141592653589793
```

In [118]:

```
for d in d6:
    print(d)
```

```
6.7
2.3
5.4
1.192
```

In [127]:

```
drr = np.array([[1,3],[3,4],[2,8]], float)
for (r,c) in drr:
    print(r,c)
```

```
1.0 3.0
3.0 4.0
2.0 8.0
```

Boolean and Random Array

In [18]:

```
import numpy as np
names = np.array(['Bob', 'Joe', 'Will', 'Bobi', 'Will', 'Joe', 'Joe'])
names
```

Out[18]:

```
array(['Bob', 'Joe', 'Will', 'Bobi', 'Will', 'Joe', 'Joe'], dtype='<U4')
```

In [ ]:

```

```

In [23]:

```
data = np.random.randn(7, 4)
data
```

Out[23]:

```
array([[-1.02474911,  1.68205614,  1.1495862 ,  0.31907058],
       [ 0.10168978, -0.51381455, -1.82596259, -0.03000468],
       [-0.18496147, -0.38196325,  0.22636415, -1.87676432],
       [ 0.0359543 ,  0.58142579,  0.53873212, -0.33530603],
       [ 2.00392195,  0.21599078, -0.79637594,  0.3798102 ],
       [ 0.29630905,  1.4128005 , -0.96989295, -0.40543093],
       [ 1.35828522,  1.93744017,  0.22942224, -0.80666504]])
```

In [10]:

```
names == 'Bob'
```

Out[10]:

```
array([ True, False, False,  True, False, False, False,  True])
```

In [24]:

```
data[names == 'Bob', 2:]
```

Out[24]:

```
array([[1.1495862 , 0.31907058]])
```

In [33]:

```
data[names != 'Bob']
```

Out[33]:

```
array([[ 0.10168978, -0.51381455, -1.82596259, -0.03000468],
       [-0.18496147, -0.38196325,  0.22636415, -1.87676432],
       [ 0.0359543 ,  0.58142579,  0.53873212, -0.33530603],
       [ 2.00392195,  0.21599078, -0.79637594,  0.3798102 ],
       [ 0.29630905,  1.4128005 , -0.96989295, -0.40543093],
       [ 1.35828522,  1.93744017,  0.22942224, -0.80666504]])
```

use boolean arithmetic operators like & (and) and | (or):

In [37]:

```
mask = (names == 'Bob') | (names == 'Joe')
mask
```

Out[37]:

```
array([ True,  True, False, False, False,  True,  True])
```

In [65]:

```
data = np.random.randn(7, 4)
```

In [40]:

```
data[mask]
```

Out[40]:

```
array([[-1.02474911,  1.68205614,  1.1495862 ,  0.31907058],
       [ 0.10168978, -0.51381455, -1.82596259, -0.03000468],
       [ 0.29630905,  1.4128005 , -0.96989295, -0.40543093],
       [ 1.35828522,  1.93744017,  0.22942224, -0.80666504]])
```

In [60]:

```
import numpy as np
names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])

names == 'Joe'
```

Out[60]:

```
array([False,  True, False, False, False,  True,  True])
```

In [62]:

```
data[names == 'Joe'] = 7
data
```

Out[62]:

```
array([[7., 7., 7., 7.],
       [7., 7., 7., 7.],
       [7., 7., 7., 7.],
       [7., 7., 7., 7.],
       [7., 7., 7., 7.],
       [7., 7., 7., 7.],
       [7., 7., 7., 7.]])
```

In [66]:

```
data[data < 0] = 0
```

In [ ]:

```
```