Numpy

In [1]:
```
1 import numpy as np
```

In [2]:
```
1 x = np.arange(10)
2 x
```

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

In [3]:
```
1 X = np.arange(15).reshape(3,5)
2 X
```

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

基本属性

In [4]:
```
1 x.ndim
```

Out[4]:
```
1
```

In [5]:
```
1 X.ndim
```

Out[5]:
```
2
```

In [6]:
```
1 x.shape
```

Out[6]:
```
(10,)
```

In [7]:
```
1 X.shape
```

Out[7]:
```
(3, 5)
```

In [8]:
```
1 x.size
```

Out[8]:
```
10
```

In [9]:
```
1 X.size
```

Out[9]:
```
15
```

numpy.array的数据访问

In [10]:
```
1 x
```

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

In [11]:
```
1 x[0]
```

Out[11]:
```
0
```

In [12]:
```
1 x[-1]
```

Out[12]:
```
9
```

In [13]:
```
X
```

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

In [14]:
```
X[0][0]
```

Out[14]:
```
0
```

In [15]:
```
X[2,2]
```

Out[15]:
```
12
```

In [16]:
```
x[0:5]
```

Out[16]:
```
array([0, 1, 2, 3, 4])
```

In [17]:
```
x[:5]
```

Out[17]:
```
array([0, 1, 2, 3, 4])
```

In [18]:
```
x[5:]
```

Out[18]:
```
array([5, 6, 7, 8, 9])
```

In [19]:
```
x[::2]
```

Out[19]:
```
array([0, 2, 4, 6, 8])
```

In [20]:
```
x[::-1]
```

Out[20]:
```
array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])
```

In [21]:
```
X
```

Out[21]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

In [22]:
```
X[:2,:3]
```

Out[22]:
```
array([[0, 1, 2],
       [5, 6, 7]])
```

In [23]:
```
X[:2][:3]
```

Out[23]:
```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

In [24]:
```
X[:2]
```

Out[24]:
```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

In [25]:
```
X[:2,::2]
```

Out[25]:
```
array([[0, 2, 4],
       [5, 7, 9]])
```

In [26]:
```
X[0]
```

Out[26]:
```
array([0, 1, 2, 3, 4])
```

In [27]:
```
X[0,:]
```

Out[27]:
```
array([0, 1, 2, 3, 4])
```

In [28]:
```
X[0,:].ndim
```

Out[28]:
```
1
```

In [29]:
```
X[:,0]
```

Out[29]:
```
array([ 0,  5, 10])
```

In [30]:
```
X[:,0].ndim
```

Out[30]:
```
1
```

In [31]:
```
subX = X[:2,:3]
subX
```

Out[31]:
```
array([[0, 1, 2],
       [5, 6, 7]])
```

In [32]:
```
subX[0,0] = 100
subX
```

Out[32]:
```
array([[100,   1,   2],
       [  5,   6,   7]])
```

In [33]:
```
X
```

Out[33]:
```
array([[100,   1,   2,   3,   4],
       [  5,   6,   7,   8,   9],
       [ 10,  11,  12,  13,  14]])
```

In [34]:
```
X[0,0] = 0
X
```

Out[34]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

In [35]:
```
subX
```

Out[35]:
```
array([[0, 1, 2],
       [5, 6, 7]])
```

In [36]:
```
subX = X[:2,:3].copy()
subX
```

Out[36]:
```
array([[0, 1, 2],
       [5, 6, 7]])
```

In [37]:
```
subX[0,0] = 100
subX
```

Out[37]:
```
array([[100,   1,   2],
       [  5,   6,   7]])
```

In [38]:
```
1 X
```

Out[38]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

Reshape

In [39]:
```
1 x.shape
```

Out[39]:
```
(10,)
```

In [40]:
```
1 x.ndim
```

Out[40]:
```
1
```

In [41]:
```
1 x.reshape(2,5)
```

Out[41]:
```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

In [42]:
```
1 x
```

Out[42]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [43]:
```
1 A = x.reshape(2,5)
2 A
```

Out[43]:
```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

In [44]:
```
1 x
```

Out[44]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [45]:
```
1 B = x.reshape(1,10)
2 B
```

Out[45]:
```
array([[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]])
```

In [46]:
```
1 B.shape
```

Out[46]:
```
(1, 10)
```

In [47]:
```
1 B.ndim
```

Out[47]:
```
2
```

In [48]:
```
1 x.shape
```

Out[48]:
```
(10,)
```

In [49]:
```
x.reshape(10,-1)
```

Out[49]:
```
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5],
       [6],
       [7],
       [8],
       [9]])
```

In [50]:
```
x.reshape(-1,10)
```

Out[50]:
```
array([[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]])
```

In [51]:
```
x.reshape(2,-1)
```

Out[51]:
```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

合并操作

In [52]:
```
x = np.array([1,2,3])
y = np.array([3,2,1])
```

In [53]:
```
x
```

Out[53]:
```
array([1, 2, 3])
```

In [54]:
```
y
```

Out[54]:
```
array([3, 2, 1])
```

In [55]:
```
np.concatenate([x,y])
```

Out[55]:
```
array([1, 2, 3, 3, 2, 1])
```

In [56]:
```
z = np.array([666,666,666])
```

In [57]:
```
np.concatenate([x , y ,z])
```

Out[57]:
```
array([  1,   2,   3,   3,   2,   1, 666, 666, 666])
```

In [58]:
```
A = np.array([[1, 2 ,3],
              [4, 5, 6]])
```

In [59]:
```
np.concatenate([A, A])
```

Out[59]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [1, 2, 3],
       [4, 5, 6]])
```

In [60]:
```
np.concatenate([A , A],axis=1)
```

Out[60]:
```
array([[1, 2, 3, 1, 2, 3],
       [4, 5, 6, 4, 5, 6]])
```

In [61]:
```
np.concatenate([A, z])
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-61-932a3f3533a2> in <module>()
----> 1 np.concatenate([A, z])

ValueError: all the input arrays must have same number of dimensions
```

In [62]:
```
np.concatenate([A, z.reshape(1,-1)])
```

Out[62]:
```
array([[  1,   2,   3],
       [  4,   5,   6],
       [666, 666, 666]])
```

In [63]:
```
A
```

Out[63]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [64]:
```
A2 = np.concatenate([A, z.reshape(1,-1)])
```

In [65]:
```
A2
```

Out[65]:
```
array([[  1,   2,   3],
       [  4,   5,   6],
       [666, 666, 666]])
```

In [66]:
```
np.vstack([A, z])
```

Out[66]:
```
array([[  1,   2,   3],
       [  4,   5,   6],
       [666, 666, 666]])
```

In [67]:
```
B = np.full((2,2), 100)
```

In [68]:
```
B
```

Out[68]:
```
array([[100, 100],
       [100, 100]])
```

In [69]:
```
np.hstack([A, B])
```

Out[69]:
```
array([[  1,   2,   3, 100, 100],
       [  4,   5,   6, 100, 100]])
```

分割

In [70]:
```
x = np.arange(10)
x
```

Out[70]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [71]:
```
x1, x2, x3 = np.split(x,[3, 7])
```

In [72]:
```
x
```

Out[72]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [73]:
```
x1
```

Out[73]:
```
array([0, 1, 2])
```

In [74]:
```
x2
```

Out[74]:
```
array([3, 4, 5, 6])
```

In [75]:
```
x3
```

Out[75]:
```
array([7, 8, 9])
```

In [76]:
```
x1, x2 = np.split(x, [5])
```

In [77]:
```
x1
```

Out[77]:
```
array([0, 1, 2, 3, 4])
```

In [78]:
```
x2
```

Out[78]:
```
array([5, 6, 7, 8, 9])
```

In [79]:
```
A = np.arange(16).reshape((4,4))
A
```

Out[79]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [80]:
```
A1, A2 = np.split(A,[2])
```

In [81]:
```
A1
```

Out[81]:
```
array([[0, 1, 2, 3],
       [4, 5, 6, 7]])
```

In [82]:
```
A2
```

Out[82]:
```
array([[ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [83]:
```
A1, A2 = np.split(A,[2], axis = 1)
```

In [84]:
```
A1
```

Out[84]:
```
array([[ 0,  1],
       [ 4,  5],
       [ 8,  9],
       [12, 13]])
```

In [85]:
```
A2
```

Out[85]:
```
array([[ 2,  3],
       [ 6,  7],
       [10, 11],
       [14, 15]])
```

In [86]:
```
upper, lower = np.vsplit(A, [2])
```

In [87]:
```
upper
```

Out[87]:
```
array([[0, 1, 2, 3],
       [4, 5, 6, 7]])
```

In [88]:
```
lower
```

Out[88]:
```
array([[ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [89]:
```
left, right = np.hsplit(A,[2])
```

In [90]:
```
left
```

Out[90]:
```
array([[ 0,  1],
       [ 4,  5],
       [ 8,  9],
       [12, 13]])
```

In [91]:
```
right
```

Out[91]:
```
array([[ 2,  3],
       [ 6,  7],
       [10, 11],
       [14, 15]])
```

In [92]:
```
data = np.arange(16).reshape((4,4))
data
```

Out[92]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [93]:
```
X, y = np.hsplit(data,[-1])
```

In [94]:
```
X
```

Out[94]:
```
array([[ 0,  1,  2],
       [ 4,  5,  6],
       [ 8,  9, 10],
       [12, 13, 14]])
```

In [95]:
```
y
```

Out[95]:
```
array([[ 3],
       [ 7],
       [11],
       [15]])
```

In [96]:
```
y[:,0]
```

Out[96]:
```
array([ 3,  7, 11, 15])
```

```

```

```

```

```

```

```
```

numpy.array中的运算

给定一个向量，让向量中每一个数乘以2

a = (0, 1, 2) a*2 = (0, 2, 4)

In [97]:
```
n = 10
L = [i for i in range(n)]
```

In [98]:
```
2*L
```

Out[98]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [99]:
```
A = []
for e in L:
    A.append(2*e)
A
```

Out[99]: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

In [100]:
```
n = 1000000
L = [i for i in range(n)]
```

In [101]:
```
%%time
A = []
for e in L:
    A.append(2*e)
```

```
Wall time: 280 ms
```

In [102]:
```
%%time
A = [2*e for e in L]
```

```
Wall time: 147 ms
```

In [103]:
```
L = np.arange(n)
```

In [104]:
```
%%time
A = np.array(2*e for e in L)
```

```
Wall time: 16 ms
```

In [105]:
```
%%time
A = 2 * L
```

```
Wall time: 3 ms
```

In [106]:
```
A
```

Out[106]: array([0,   2,   4, ..., 1999994, 1999996, 1999998])

In [107]:
```
n = 10
L = np.arange(n)
2 * L
```

Out[107]: array([0, 2, 4, 6, 8, 10, 12, 14, 16, 18])

```
1
```

Universal Function

In [108]:
```
X = np.arange(1,16).reshape((3, 5))
X
```

Out[108]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15]])
```

In [109]:
```
X + 1
```

Out[109]:
```
array([[ 2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11],
       [12, 13, 14, 15, 16]])
```

In [110]:
```
X - 1
```

Out[110]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

In [111]:
```
X * 2
```

Out[111]:
```
array([[ 2,  4,  6,  8, 10],
       [12, 14, 16, 18, 20],
       [22, 24, 26, 28, 30]])
```

In [112]:
```
X / 2
```

Out[112]:
```
array([[0.5, 1. , 1.5, 2. , 2.5],
       [3. , 3.5, 4. , 4.5, 5. ],
       [5.5, 6. , 6.5, 7. , 7.5]])
```

In [113]:
```
X // 2
```

Out[113]:
```
array([[0, 1, 1, 2, 2],
       [3, 3, 4, 4, 5],
       [5, 6, 6, 7, 7]], dtype=int32)
```

In [114]:
```
X ** 2
```

Out[114]:
```
array([[  1,   4,   9,  16,  25],
       [ 36,  49,  64,  81, 100],
       [121, 144, 169, 196, 225]], dtype=int32)
```

In [115]:
```
X % 2
```

Out[115]:
```
array([[1, 0, 1, 0, 1],
       [0, 1, 0, 1, 0],
       [1, 0, 1, 0, 1]], dtype=int32)
```

In [116]:
```
1 / X
```

Out[116]:
```
array([[1.        , 0.5       , 0.33333333, 0.25      , 0.2       ],
       [0.16666667, 0.14285714, 0.125     , 0.11111111, 0.1       ],
       [0.09090909, 0.08333333, 0.07692308, 0.07142857, 0.06666667]])
```

In [117]:
```
np.abs(X)
```

Out[117]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15]])
```

In [118]:
```
np.sin(X)
```

Out[118]:
```
array([[ 0.84147098,  0.90929743,  0.14112001, -0.7568025 , -0.95892427],
       [-0.2794155 ,  0.6569866 ,  0.98935825,  0.41211849, -0.54402111],
       [-0.99999021, -0.53657292,  0.42016704,  0.99060736,  0.65028784]])
```

In [119]:
```
np.exp(X)
```

Out[119]:
```
array([[2.71828183e+00, 7.38905610e+00, 2.00855369e+01, 5.45981500e+01,
        1.48413159e+02],
       [4.03428793e+02, 1.09663316e+03, 2.98095799e+03, 8.10308393e+03,
        2.20264658e+04],
       [5.98741417e+04, 1.62754791e+05, 4.42413392e+05, 1.20260428e+06,
        3.26901737e+06]])
```

In [120]:
```
np.power(3, X)
```

Out[120]:
```
array([[      3,       9,      27,      81,     243],
       [    729,    2187,    6561,   19683,   59049],
       [ 177147,  531441, 1594323, 4782969, 14348907]], dtype=int32)
```

In [121]:
```
3 ** X
```

Out[121]:
```
array([[      3,       9,      27,      81,     243],
       [    729,    2187,    6561,   19683,   59049],
       [ 177147,  531441, 1594323, 4782969, 14348907]], dtype=int32)
```

In [122]:
```
np.log(X)
```

Out[122]:
```
array([[0.        , 0.69314718, 1.09861229, 1.38629436, 1.60943791],
       [1.79175947, 1.94591015, 2.07944154, 2.19722458, 2.30258509],
       [2.39789527, 2.48490665, 2.56494936, 2.63905733, 2.7080502 ]])
```

In [123]:
```
np.log2(X)
```

Out[123]:
```
array([[0.        , 1.        , 1.5849625 , 2.        , 2.32192809],
       [2.5849625 , 2.80735492, 3.        , 3.169925  , 3.32192809],
       [3.45943162, 3.5849625 , 3.70043972, 3.80735492, 3.9068906 ]])
```

In [124]:
```
np.log10(X)
```

Out[124]:
```
array([[0.        , 0.30103   , 0.47712125, 0.60205999, 0.69897   ],
       [0.77815125, 0.84509804, 0.90308999, 0.95424251, 1.        ],
       [1.04139269, 1.07918125, 1.11394335, 1.14612804, 1.17609126]])
```

矩阵运算

In [125]:
```
A = np.arange(4).reshape(2, 2)
A
```

Out[125]:
```
array([[0, 1],
       [2, 3]])
```

In [126]:
```
B = np.full((2,2), 10)
B
```

Out[126]:
```
array([[10, 10],
       [10, 10]])
```

In [127]:
```
A + B
```

Out[127]:
```
array([[10, 11],
       [12, 13]])
```

In [128]:
```
A - B
```

Out[128]:
```
array([[-10,  -9],
       [ -8,  -7]])
```

In [129]:
```
A * B
```

Out[129]:
```
array([[ 0, 10],
       [20, 30]])
```

In [130]:
```
A / B
```

Out[130]:
```
array([[0. , 0.1],
       [0.2, 0.3]])
```

In [131]:
```
A.dot(B)
```

Out[131]:
```
array([[10, 10],
       [50, 50]])
```

In [132]:
```
A
```

Out[132]:
```
array([[0, 1],
       [2, 3]])
```

In [133]:
```
A.T
```

Out[133]:
```
array([[0, 2],
       [1, 3]])
```

In [134]:
```
c = np.full((3, 3), 666)
```

向量和矩阵的运算

In [135]:
```
v = np.array([1, 2])
```

In [136]:
```
A
```

Out[136]:
```
array([[0, 1],
       [2, 3]])
```

In [137]:
```
v + A
```

Out[137]:
```
array([[1, 3],
       [3, 5]])
```

In [138]:
```
1 np.vstack([v] * A.shape[0])
```

Out[138]:
```
array([[1, 2],
       [1, 2]])
```

In [139]:
```
1 np.vstack([v] * A.shape[0]) + A
```

Out[139]:
```
array([[1, 3],
       [3, 5]])
```

In [140]:
```
1 np.tile(v, (2, 1))
```

Out[140]:
```
array([[1, 2],
       [1, 2]])
```

In [141]:
```
1 v
```

Out[141]:
```
array([1, 2])
```

In [142]:
```
1 A
```

Out[142]:
```
array([[0, 1],
       [2, 3]])
```

In [143]:
```
1 v * A
```

Out[143]:
```
array([[0, 2],
       [2, 6]])
```

In [144]:
```
1 v.dot(A)
```

Out[144]:
```
array([4, 7])
```

In [145]:
```
1 A.dot(v)
```

Out[145]:
```
array([2, 8])
```

矩阵的逆

In [146]:
```
1 A
```

Out[146]:
```
array([[0, 1],
       [2, 3]])
```

In [147]:
```
1 np.linalg.inv(A)
```

Out[147]:
```
array([[-1.5,  0.5],
       [ 1. ,  0. ]])
```

In [148]:
```
1 invA= np.linalg.inv(A)
```

In [149]:
```
1 A.dot(invA)
```

Out[149]:
```
array([[1., 0.],
       [0., 1.]])
```

In [150]:
```
X
```

Out[150]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15]])
```

In [151]:
```
np.linalg.inv(X)
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
<ipython-input-151-47889a8f1529> in <module>()
----> 1 np.linalg.inv(X)

~\Anaconda3\lib\site-packages\numpy\linalg\linalg.py in inv(a)
    521     a, wrap = _makearray(a)
    522     _assertRankAtLeast2(a)
--> 523     _assertNdSquareness(a)
    524     t, result_t = _commonType(a)
    525

~\Anaconda3\lib\site-packages\numpy\linalg\linalg.py in _assertNdSquareness(*arrays)
    209     for a in arrays:
    210         if max(a.shape[-2:]) != min(a.shape[-2:]):
--> 211             raise LinAlgError('Last 2 dimensions of the array must be square')
    212
    213 def _assertFinite(*arrays):

LinAlgError: Last 2 dimensions of the array must be square
```

In [152]:
```
pinvX = np.linalg.pinv(X)
```

In [153]:
```
pinvX
```

Out[153]:
```
array([[-2.46666667e-01, -6.66666667e-02,  1.13333333e-01],
       [-1.33333333e-01, -3.33333333e-02,  6.66666667e-02],
       [-2.00000000e-02, -2.51534904e-17,  2.00000000e-02],
       [ 9.33333333e-02,  3.33333333e-02, -2.66666667e-02],
       [ 2.06666667e-01,  6.66666667e-02, -7.33333333e-02]])
```

In [154]:
```
pinvX.shape
```

Out[154]:
```
(5, 3)
```

In [155]:
```
X.dot(pinvX)
```

Out[155]:
```
array([[ 0.83333333,  0.33333333, -0.16666667],
       [ 0.33333333,  0.33333333,  0.33333333],
       [-0.16666667,  0.33333333,  0.83333333]])
```

聚合操作

In [156]:
```
L = np.random.random(100)
```

In [157]:
```
L
```

Out[157]:
```
array([0.09203433, 0.54696119, 0.68144857, 0.6177403 , 0.32622209,
       0.98727473, 0.84754394, 0.730217  , 0.73095486, 0.79383974,
       0.59540056, 0.91111319, 0.64391333, 0.49294928, 0.99923273,
       0.6172888 , 0.74453739, 0.16470727, 0.0564202 , 0.928241  ,
       0.36476476, 0.92474047, 0.52480633, 0.6062084 , 0.46831873,
       0.63282681, 0.95792083, 0.98016491, 0.28628449, 0.96363442,
       0.82905292, 0.16401964, 0.67412641, 0.46999459, 0.34927009,
       0.86494291, 0.43170649, 0.92955335, 0.59648474, 0.90477494,
       0.01417333, 0.07921472, 0.09694792, 0.68575408, 0.83508618,
       0.36244954, 0.94410437, 0.80756523, 0.76726758, 0.78034448,
       0.49102979, 0.34753356, 0.79688188, 0.87695891, 0.44119478,
       0.22013165, 0.45423317, 0.27824791, 0.38502875, 0.49035642,
       0.94540193, 0.28835998, 0.14390196, 0.016507  , 0.00552772,
       0.80532682, 0.78245718, 0.99352088, 0.81085732, 0.18394585,
       0.77407104, 0.99282351, 0.60401437, 0.60011158, 0.23395581,
       0.01979987, 0.03512999, 0.92344429, 0.00896518, 0.15870639,
       0.33706136, 0.31379032, 0.79742427, 0.53642556, 0.66472886,
       0.21655404, 0.04708589, 0.00245684, 0.9367032 , 0.77236017,
       0.46814999, 0.25220664, 0.90050106, 0.92599604, 0.66613144,
       0.43453405, 0.96656757, 0.78310829, 0.24586073, 0.76987304])
```

In [158]:
```
sum(L)
```

Out[158]:
```
55.950489062170085
```

In [159]:
```
np.sum(L)
```

Out[159]:
```
55.95048906217009
```

In [160]:
```
big_array = np.random.rand(1000000)
%timeit sum(big_array)
%timeit np.sum(big_array)
```

```
193 ms ± 31 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
1.68 ms ± 200 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [161]:
```
np.min(big_array)
```

Out[161]:
```
1.5050130142135743e-08
```

In [162]:
```
np.max(big_array)
```

Out[162]:
```
0.9999998247249198
```

In [163]:
```
big_array.min()
```

Out[163]:
```
1.5050130142135743e-08
```

In [164]:
```
big_array.sum()
```

Out[164]:
```
500028.4207628005
```

In [165]:
```
X = np.arange(16).reshape(4,-1)
X
```

Out[165]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [166]:
```
np.sum(X)
```

Out[166]: 120

In [167]:
```
np.sum(X ,axis = 1)
```

Out[167]: array([6, 22, 38, 54])

In [168]:
```
np.sum(X ,axis = 0)
```

Out[168]: array([24, 28, 32, 36])

In [169]:
```
np.prod(X)
```

Out[169]: 0

In [170]:
```
np.prod(X + 1)
```

Out[170]: 2004189184

In [171]:
```
np.mean(X)
```

Out[171]: 7.5

In [172]:
```
np.median(X)
```

Out[172]: 7.5

In [173]:
```
v = np.array([1, 1, 2, 2, 10])
np.mean(v)
```

Out[173]: 3.2

In [174]:
```
np.median(v)
```

Out[174]: 2.0

In [175]:
```
np.percentile(big_array, q = 50)
```

Out[175]: 0.4998067870371987

In [176]:
```
np.median(big_array)
```

Out[176]: 0.4998067870371987

In [177]:
```
np.percentile(big_array, q = 100)
```

Out[177]: 0.9999998247249198

In [178]:
```
np.max(big_array)
```

Out[178]: 0.9999998247249198

In [179]:
```
for percent in [0, 25, 50, 75, 100]:
    print(np.percentile(big_array, q = percent))
```

```
1.5050130142135743e-08
0.25007377138553794
0.4998067870371987
0.7501708013374507
0.9999998247249198
```

In [180]:
```
np.var(big_array)
```

Out[180]: 0.08331298513457178

In [181]:
```
np.std(big_array)
```

Out[181]: 0.2886398883289899

In [182]:
```
x = np.random.normal(0, 1, size = 1000000)
```

In [183]:
```
np.mean(x)
```

Out[183]: -0.0010722261764876749

In [184]:
```
np.std(x)
```

Out[184]: 1.0013011722424001

索引

In [185]:
```
np.min(x)
```

Out[185]: -4.72131365565871

In [186]:
```
np.argmin(x)
```

Out[186]: 985142

In [187]:
```
x[720143]
```

Out[187]: -0.1257829961086523

In [188]:
```
np.argmax(x)
```

Out[188]: 894706

In [189]:
```
x[693938]
```

Out[189]: -0.8901520446861358

In [190]:
```
np.max(x)
```

Out[190]: 4.490983940870949

排序和使用索引

In [191]:
```
x = np.arange(16)
x
```

Out[191]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15])

In [192]:
```
np.random.shuffle(x)
x
```

Out[192]: array([13, 11, 9, 10, 14, 8, 12, 3, 2, 0, 1, 5, 15, 7, 4, 6])

In [193]:
```
np.sort(x)
```

Out[193]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

In [194]:
```
x
```

Out[194]:
```
array([13, 11,  9, 10, 14,  8, 12,  3,  2,  0,  1,  5, 15,  7,  4,  6])
```

In [195]:
```
x.sort()
```

In [196]:
```
x
```

Out[196]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

In [197]:
```
X = np.random.randint(10, size = (4, 4))
X
```

Out[197]:
```
array([[4, 7, 1, 0],
       [7, 4, 3, 8],
       [1, 4, 0, 2],
       [2, 3, 3, 7]])
```

In [198]:
```
np.sort(X)
```

Out[198]:
```
array([[0, 1, 4, 7],
       [3, 4, 7, 8],
       [0, 1, 2, 4],
       [2, 3, 3, 7]])
```

In [199]:
```
np.sort(X, axis = 1)
```

Out[199]:
```
array([[0, 1, 4, 7],
       [3, 4, 7, 8],
       [0, 1, 2, 4],
       [2, 3, 3, 7]])
```

In [200]:
```
np.sort(X, axis = 0)
```

Out[200]:
```
array([[1, 3, 0, 0],
       [2, 4, 1, 2],
       [4, 4, 3, 7],
       [7, 7, 3, 8]])
```

In [201]:
```
x = np.arange(16)
x
```

Out[201]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

In [202]:
```
np.random.shuffle(x)
```

In [203]:
```
x
```

Out[203]:
```
array([11,  1,  3,  5, 13,  7,  8,  9, 10,  6, 15,  0, 12,  2, 14,  4])
```

In [204]:
```
np.argsort(x)
```

Out[204]:
```
array([11,  1, 13,  2, 15,  3,  9,  5,  6,  7,  8,  0, 12,  4, 14, 10],
      dtype=int64)
```

In [205]:
```
1 np.partition(x, 3)
```

Out[205]:
```
array([ 1,  0,  2,  3,  4,  5,  6,  8,  7,  9, 15, 10, 12, 13, 14, 11])
```

In [206]:
```
1 np.argpartition(x, 3)
```

Out[206]:
```
array([ 1, 11, 13,  2, 15,  3,  9,  6,  5,  7, 10,  8, 12,  4, 14,  0],
      dtype=int64)
```

In [207]:
```
1 X
```

Out[207]:
```
array([[4, 7, 1, 0],
       [7, 4, 3, 8],
       [1, 4, 0, 2],
       [2, 3, 3, 7]])
```

In [208]:
```
1 np.argsort(X, axis = 1)
```

Out[208]:
```
array([[3, 2, 0, 1],
       [2, 1, 0, 3],
       [2, 0, 3, 1],
       [0, 1, 2, 3]], dtype=int64)
```

In [209]:
```
1 np.argpartition(X , 2, axis = 1)
```

Out[209]:
```
array([[3, 2, 0, 1],
       [2, 1, 0, 3],
       [2, 0, 3, 1],
       [0, 1, 2, 3]], dtype=int64)
```

Fancy Indexing

In [210]:
```
1 x = np.arange(16)
2 x
```

Out[210]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

In [211]:
```
1 x[3]
```

Out[211]:
```
3
```

In [212]:
```
1 x[3:9]
```

Out[212]:
```
array([3, 4, 5, 6, 7, 8])
```

In [213]:
```
1 x[3:9:2]
```

Out[213]:
```
array([3, 5, 7])
```

In [214]:
```
1 [x[3],x[5],x[8]]
```

Out[214]:
```
[3, 5, 8]
```

In [215]:
```
1 ind = [3, 5, 8]
```

In [216]:
```
1 x[ind]
```

Out[216]:
```
array([3, 5, 8])
```

In [217]:
```
ind = np.array([[0, 2],
                [1, 3]])
x[ind]
```

Out[217]:
```
array([[0, 2],
       [1, 3]])
```

In [218]:
```
X = x.reshape(4, -1)
X
```

Out[218]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [219]:
```
row = np.array([0, 1 ,2])
col = np.array([1, 2 ,3])
X[row, col]
```

Out[219]:
```
array([ 1,  6, 11])
```

In [220]:
```
X[0, col]
```

Out[220]:
```
array([1, 2, 3])
```

In [221]:
```
X[:2, col]
```

Out[221]:
```
array([[1, 2, 3],
       [5, 6, 7]])
```

In [222]:
```
col = [True, False, True, True]
```

In [223]:
```
X[1:3, col]
```

Out[223]:
```
array([[ 4,  6,  7],
       [ 8, 10, 11]])
```

numpy.array的比较

In [224]:
```
x
```

Out[224]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

In [225]:
```
x < 3
```

Out[225]:
```
array([ True,  True,  True, False, False, False, False, False, False,
       False, False, False, False, False, False, False])
```

In [226]:
```
x <= 3
```

Out[226]:
```
array([ True,  True,  True,  True, False, False, False, False, False,
       False, False, False, False, False, False, False])
```

In [227]:
```
x == 3
```

Out[227]:
```
array([False, False, False,  True, False, False, False, False, False,
       False, False, False, False, False, False, False])
```

In [228]:
```
x != 3
```

Out[228]:
```
array([ True,  True,  True, False,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True])
```

In [229]:
```
2 * x == 24- 4 * x
```

Out[229]:
```
array([False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False])
```

In [230]:
```
X
```

Out[230]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [231]:
```
X < 6
```

Out[231]:
```
array([[ True,  True,  True,  True],
       [ True,  True, False, False],
       [False, False, False, False],
       [False, False, False, False]])
```

In [232]:
```
x
```

Out[232]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

In [233]:
```
np.sum(x <= 3)
```

Out[233]:
```
4
```

In [234]:
```
np.count_nonzero(x <= 3)
```

Out[234]:
```
4
```

In [235]:
```
np.any(x == 0)
```

Out[235]:
```
True
```

In [236]:
```
np.any(x < 1)
```

Out[236]:
```
True
```

In [237]:
```
np.all(x >= 0)
```

Out[237]:
```
True
```

In [238]:
```
np.all(x > 0)
```

Out[238]:
```
False
```

In [239]:
```
X
```

Out[239]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [240]:
```
np.sum(X % 2 == 0)
```

Out[240]:
```
8
```

In [241]:
```
1 np.sum(X % 2 == 0,axis = 1)
```

Out[241]:
```
array([2, 2, 2, 2])
```

In [242]:
```
1 np.sum(X % 2 == 0,axis = 0)
```

Out[242]:
```
array([4, 0, 4, 0])
```

In [243]:
```
1 np.all(X > 0,axis = 1)
```

Out[243]:
```
array([False,  True,  True,  True])
```

In [244]:
```
1 x
```

Out[244]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15])
```

In [245]:
```
1 np.sum((x > 3) & (x < 10))
```

Out[245]:
```
6
```

In [246]:
```
1 np.sum((x > 3) && (x < 10))
```

```
  File "<ipython-input-246-d834f65999a2>", line 1
    np.sum((x > 3) && (x < 10))
                   ^
SyntaxError: invalid syntax
```

In [247]:
```
1 np.sum((x % 2 == 0) | (x > 10))
```

Out[247]:
```
11
```

In [248]:
```
1 np.sum(~(x == 0))
```

Out[248]:
```
15
```

In [249]:
```
1 x[x < 5]
```

Out[249]:
```
array([0, 1, 2, 3, 4])
```

In [250]:
```
1 x[x % 2 == 0]
```

Out[250]:
```
array([ 0,  2,  4,  6,  8, 10, 12, 14])
```

In [251]:
```
1 X
```

Out[251]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

In [252]:
```
1 X[X[:,3] % 3 == 0,:]
```

Out[252]:
```
array([[ 0,  1,  2,  3],
       [12, 13, 14, 15]])
```

In [ ]:
```
1
```