# Numpy中的矩阵运算

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

In [6]:
```
a = np.array([0, 2, 4])
a
```

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

In [7]:
```
a * 2
```

Out[7]:
```
array([0, 4, 8])
```

In [8]:
```
a = [0, 2, 4]
a * 2
```

Out[8]:
```
[0, 2, 4, 0, 2, 4]
```

In [9]:
```
import array
array.array('i', [0, 2, 4]*2)
```

Out[9]:
```
array('i', [0, 2, 4, 0, 2, 4])
```

In [10]:
```
[i**2 for i in a]
```

Out[10]:
```
[0, 4, 16]
```

In [11]:
```
res = []
for i in a:
    res.append(i**2)
res
```

Out[11]:
```
[0, 4, 16]
```

In [12]:
```
data = range(10**6)
data
```

Out[12]:
```
range(0, 1000000)
```

In [13]:
```
%%time
res = []
for i in data:
    res.append(i**2)
res
```

```
CPU times: user 500 ms, sys: 20.3 ms, total: 520 ms
Wall time: 522 ms
```

In [15]:
```
%time res = [i**2 for i in data]
```

```
CPU times: user 377 ms, sys: 29.5 ms, total: 406 ms
Wall time: 418 ms
```

In [18]:
```
arr = np.array(data)
```

In [20]:
```
%time res = arr * 2
```

```
CPU times: user 3.04 ms, sys: 3.75 ms, total: 6.79 ms
Wall time: 4.33 ms
```

numpy 中矩阵的加减乘除都是举证对应位置上的元素之间进行加减乘除

In [21]:
```
a = np.array([0, 2, 4])
a
```

Out[21]:
```
array([0, 2, 4])
```

In [22]:
```
a + 2
```

Out[22]:
```
array([2, 4, 6])
```

In [23]:
```
a - 2
```

Out[23]:
```
array([-2,  0,  2])
```

In [24]:
```
a / 2
```

Out[24]:
```
array([0., 1., 2.])
```

In [25]:
```
x = np.arange(1, 16).reshape((3, 5))
x
```

Out[25]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15]])
```

In [26]:
```
x + 1
```

Out[26]:
```
array([[ 2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11],
       [12, 13, 14, 15, 16]])
```

In [27]:
```
x - 1
```

Out[27]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

In [28]:
```
x * 2        # numpy中直接使用乘号，是矩阵对应位置元素相乘
```

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

In [29]:
```
x / 2
```

Out[29]:
```
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 [30]:
```
x // 2      # 双斜杠代表整除
```

Out[30]:
```
array([[0, 1, 1, 2, 2],
       [3, 3, 4, 4, 5],
       [5, 6, 6, 7, 7]])
```

In [31]:
```
x ** 2
```

Out[31]:
```
array([[  1,   4,   9,  16,  25],
       [ 36,  49,  64,  81, 100],
       [121, 144, 169, 196, 225]])
```

In [32]:
```
x % 2
```

Out[32]:
```
array([[1, 0, 1, 0, 1],
       [0, 1, 0, 1, 0],
       [1, 0, 1, 0, 1]])
```

In [33]:
```
1 / x
```

Out[33]:
```
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 [34]:
```
np.abs(x-2)
```

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

In [35]:
```
x - 2
```

Out[35]:
```
array([[-1,  0,  1,  2,  3],
       [ 4,  5,  6,  7,  8],
       [ 9, 10, 11, 12, 13]])
```

In [36]:
```
np.sin(x)
```

Out[36]:
```
array([[ 0.84147098,  0.90929743,  0.14112001, -0.7568025 , -0.958
92427],
       [-0.2794155 ,  0.6569866 ,  0.98935825,  0.41211849, -0.544
02111],
       [-0.99999021, -0.53657292,  0.42016704,  0.99060736,  0.650
28784]])
```

In [37]:
```
np.cos(x)
```

Out[37]:
```
array([[ 0.54030231, -0.41614684, -0.9899925 , -0.65364362,  0.283
66219],
       [ 0.96017029,  0.75390225, -0.14550003, -0.91113026, -0.839
07153],
       [ 0.0044257 ,  0.84385396,  0.90744678,  0.13673722, -0.759
68791]])
```

In [38]:
```
np.tan(x)
```

Out[38]:
```
array([[ 1.55740772e+00, -2.18503986e+00, -1.42546543e-01,
         1.15782128e+00, -3.38051501e+00],
       [-2.91006191e-01,  8.71447983e-01, -6.79971146e+00,
        -4.52315659e-01,  6.48360827e-01],
       [-2.25950846e+02, -6.35859929e-01,  4.63021133e-01,
         7.24460662e+00, -8.55993401e-01]])
```

In [39]:
```
np.exp(x)        # e的x次方
```

Out[39]:
```
array([[2.71828183e+00, 7.38905610e+00, 2.00855369e+01, 5.45981500
e+01,
        1.48413159e+02],
       [4.03428793e+02, 1.09663316e+03, 2.98095799e+03, 8.10308393
e+03,
        2.20264658e+04],
       [5.98741417e+04, 1.62754791e+05, 4.42413392e+05, 1.20260428
e+06,
        3.26901737e+06]])
```

In [40]:
```
from math import e
e ** 2
```

Out[40]:
```
7.3890560989306495
```

In [41]:
```
np.power(3, x)       # 3的x次方
```

Out[41]:
```
array([[      3,       9,      27,      81,     243],
       [    729,    2187,    6561,   19683,   59049],
       [ 177147,  531441, 1594323, 4782969, 14348907]])
```

In [42]:
```
3 ** x
```

Out[42]:
```
array([[      3,       9,      27,      81,     243],
       [    729,    2187,    6561,   19683,   59049],
       [ 177147,  531441, 1594323, 4782969, 14348907]])
```

In [43]:
```
np.log(x)
```

Out[43]:
```
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 [44]:
```
x
```

Out[44]:
```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15]])
```

In [45]:
```
e ** 0.69314718
```

Out[45]:
```
1.9999999988801092
```

In [46]:
```
np.log2(x)
```

Out[46]:
```
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 [47]:
```
np.log10(x)
```

Out[47]:
```
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
]])
```

# 矩阵中间的运算

In [48]:
```
A = np.arange(4).reshape(2, -1)
```

In [49]:
```
A
```

Out[49]:
```
array([[0, 1],
       [2, 3]])
```

In [50]:
```
B = np.full((2, 2), 10)
B
```

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

In [51]:
```
A + B
```

Out[51]:
```
array([[10, 11],
       [12, 13]])
```

In [52]:
```
A - B
```

Out[52]:
```
array([[-10,  -9],
       [ -8,  -7]])
```

In [53]:
```
A * B           # 直接相乘是两个矩阵对应元素相乘
```

Out[53]:
```
array([[ 0, 10],
       [20, 30]])
```

In [54]:
```
A / B
```

Out[54]:
```
array([[0. , 0.1],
       [0.2, 0.3]])
```

In [55]:
```
B = np.array([10, 20, 30, 40]).reshape(2, -1)
```

In [56]:
```
B
```

Out[56]:
```
array([[10, 20],
       [30, 40]])
```

In [57]:
```
A.dot(B)
```

Out[57]:
```
array([[ 30,  40],
       [110, 160]])
```

# 矩阵的逆

In [58]:
```
A
```

Out[58]:
```
array([[0, 1],
       [2, 3]])
```

In [59]:
```
invA = np.linalg.inv(A)       # 求矩阵的逆
```

In [60]:
```
invA
```

Out[60]:
```
array([[-1.5,  0.5],
       [ 1. ,  0. ]])
```

In [61]:
```
A.dot(invA)
```

Out[61]:
```
array([[1., 0.],
       [0., 1.]])
```

In [62]:
```
invA.dot(A)
```

Out[62]:
```
array([[1., 0.],
       [0., 1.]])
```

并不是所有矩阵都有逆，只有方阵才有。对于不是方阵的，可以求其伪逆

In [64]:
```
A = np.arange(6).reshape(2, -1)
```

In [65]:
```
A
```

Out[65]:
```
array([[0, 1, 2],
       [3, 4, 5]])
```

In [66]:
```
np.linalg.inv(A)
```

```
---------------------------------------------------------------------
---------
LinAlgError                               Traceback (most recent c
all last)
<ipython-input-66-ae645f97e1f8> in <module>()
----> 1 np.linalg.inv(A)

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/si
te-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

/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/si
te-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 ar
ray must be square')
    212
    213 def _assertFinite(*arrays):

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

In [67]:
```
pinvA = np.linalg.pinv(A)    # 求矩阵的伪逆
pinvA
```

Out[67]:
```
array([[-0.77777778,  0.27777778],
       [-0.11111111,  0.11111111],
       [ 0.55555556, -0.05555556]])
```

In [68]:
```
A.dot(pinvA)
```

Out[68]:
```
array([[ 1.00000000e+00, -1.11022302e-16],
       [ 2.22044605e-15,  1.00000000e+00]])
```

# 矩阵的转置

In [69]:
```
A
```

Out[69]:
```
array([[0, 1, 2],
       [3, 4, 5]])
```

In [70]:
```
A.T
```

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