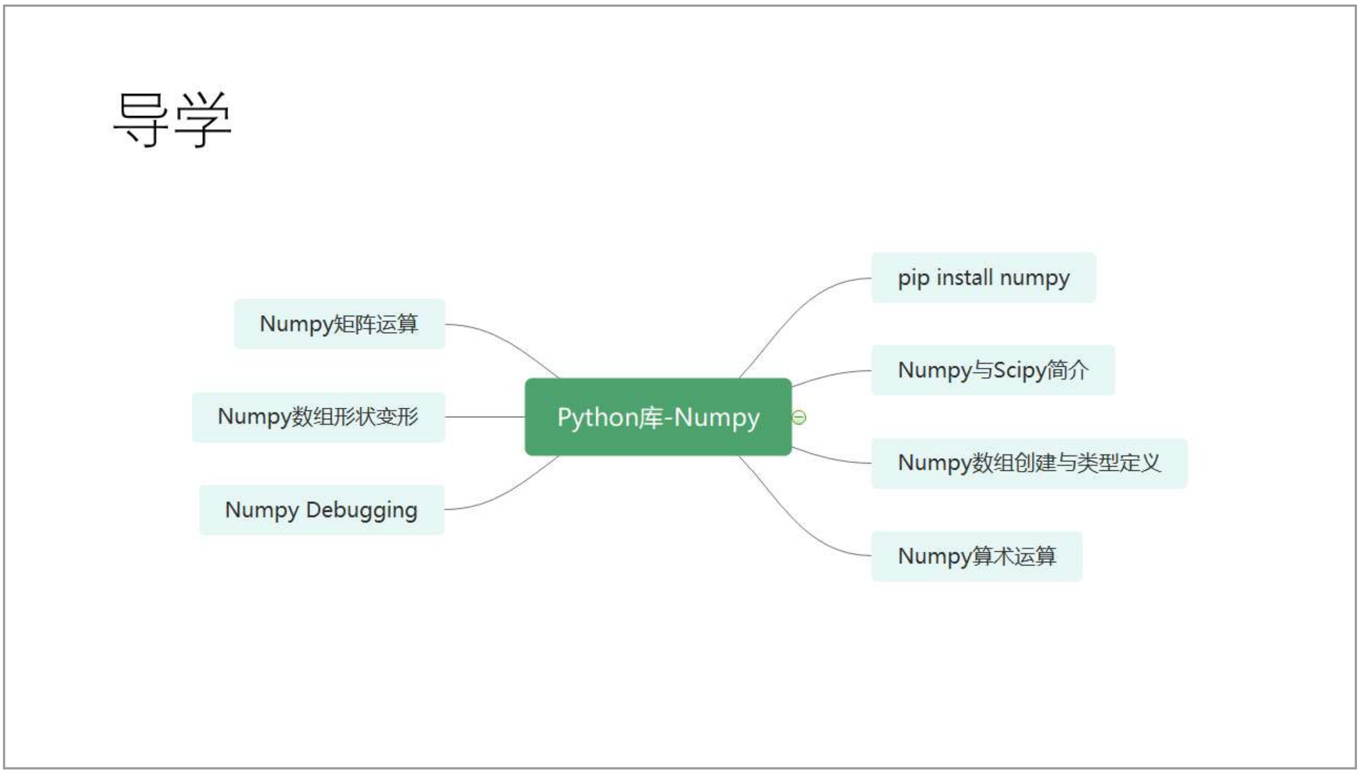
Python库

Numpy, Scipy

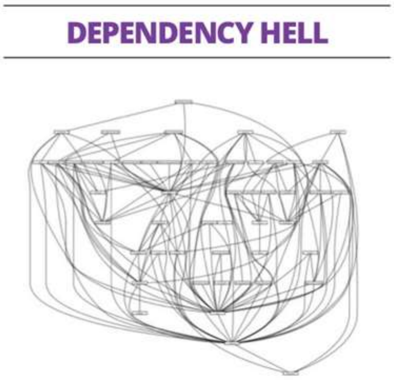
清华大学iCenter



导学

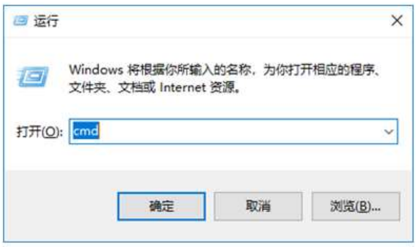
pip

- pip 是python的包管理工具(package manager)
- pip —version 查看pip版本和安装位置
- **pip install/uninstall 安装/删除包**
- pip list 列出所有的安装的包



安装使用Numpy, Scipy

- Windows的命令行(CMD)打开方式
  - 方法一：按下Win + R 键，弹出运行窗口，输入“cmd”后点击确定。
  - 方法二：在电脑左下角的搜索框搜索“cmd”或“命令提示符”，点击检索结果“命令提示符”。
  - 方法三：打开“开始”，点击“运行”，弹出运行窗口，输入“cmd”后点击确定。
  - 输入 **pip install numpy**
  - 输入 **pip install scipy**
- MacOS的终端（Terminal）打开方式
  - 搜索termianl应用（自带）
  - 输入 **pip install numpy**
  - 输入 **pip install scipy**



Windows 将根据你所输入的名称，为你打开相应的程序、文件夹、文档或 Internet 资源。

打开(O): cmd

Numpy & Scipy

- Numpy – package for vector and matrix manipulation
- https://numpy.org/

NumPy

- Scipy – package for scientific and technical computing
- https://www.scipy.org/

SciPy.org

创建数组和数据类型定义

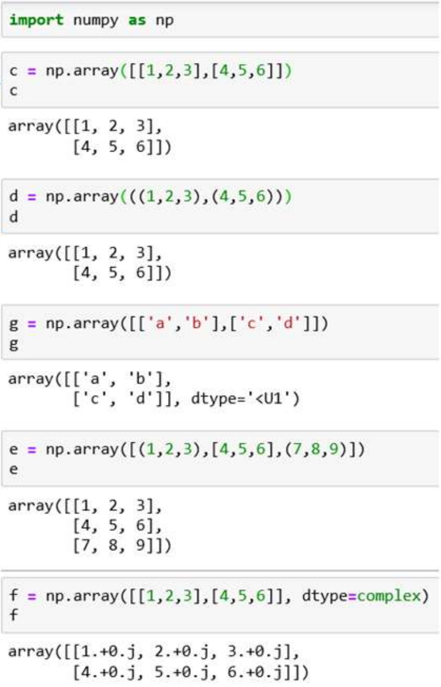
使用array()函数创建数组

array()函数接受tuples和tuples的序列

Numpy arrays 包含一组data types, 不限于integers

array()函数接收tuples 和list的混合

array()函数的dtype option 设置数据类型



```
import numpy as np
```

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

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

```
d = np.array(((1,2,3),(4,5,6)))
d
```

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

```
g = np.array([['a','b'],['c','d']])
g
```

```
array([['a', 'b'],
       ['c', 'd']], dtype='<U1')
```

```
e = np.array([(1,2,3),[4,5,6],(7,8,9)])
e
```

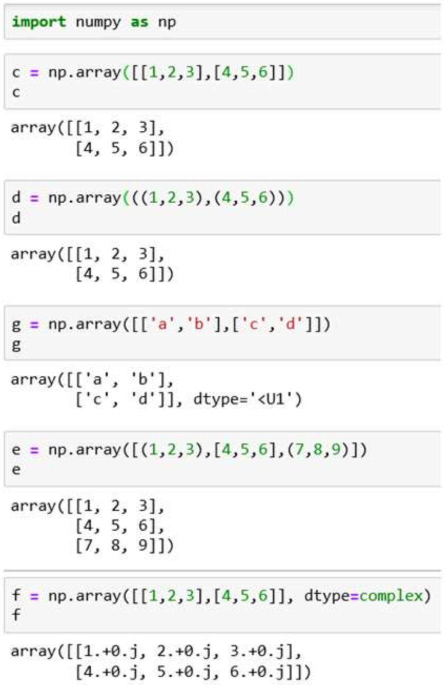
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
f = np.array([[1,2,3],[4,5,6]], dtype=complex)
f
```

```
array([[1.+0.j, 2.+0.j, 3.+0.j],
       [4.+0.j, 5.+0.j, 6.+0.j]])
```

想一想，练一练

- 创建数组



```
import numpy as np
```

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

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

```
d = np.array(((1,2,3),(4,5,6)))
d
```

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

```
g = np.array([['a','b'],['c','d']])
g
```

```
array([['a', 'b'],
       ['c', 'd']], dtype='<U1')
```

```
e = np.array([(1,2,3),[4,5,6],(7,8,9)])
e
```

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

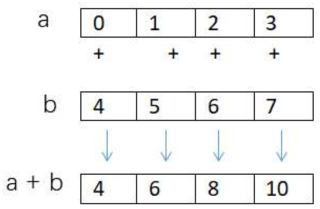
```
f = np.array([[1,2,3],[4,5,6]], dtype=complex)
f
```

```
array([[1.+0.j, 2.+0.j, 3.+0.j],
       [4.+0.j, 5.+0.j, 6.+0.j]])
```

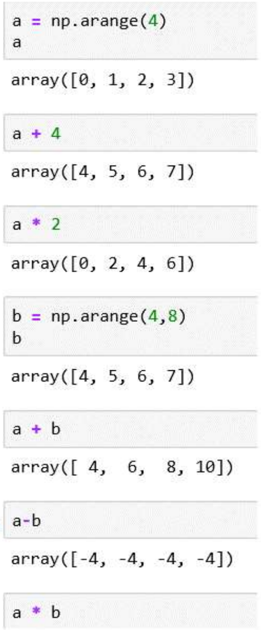
算术运算

数组与标量（scalar）的算术运算

Element-wise operation: 算符operators仅作用于对应的元素（corresponding elements)



a	0	1	2	3
	+	+	+	+
b	4	5	6	7
	↓	↓	↓	↓
a + b	4	6	8	10



```
a = np.arange(4)
a
```

```
array([0, 1, 2, 3])
```

```
a + 4
```

```
array([4, 5, 6, 7])
```

```
a * 2
```

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

```
b = np.arange(4,8)
b
```

```
array([4, 5, 6, 7])
```

```
a + b
```

```
array([ 4,  6,  8, 10])
```

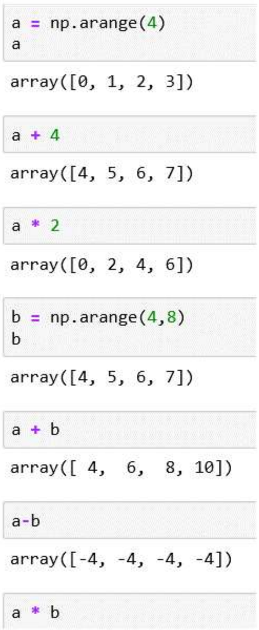
```
a-b
```

```
array([-4, -4, -4, -4])
```

```
a * b
```

想一想，练一练

- 进行算术运算



```
a = np.arange(4)
a
```

```
array([0, 1, 2, 3])
```

```
a + 4
```

```
array([4, 5, 6, 7])
```

```
a * 2
```

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

```
b = np.arange(4,8)
b
```

```
array([4, 5, 6, 7])
```

```
a + b
```

```
array([ 4,  6,  8, 10])
```

```
a-b
```

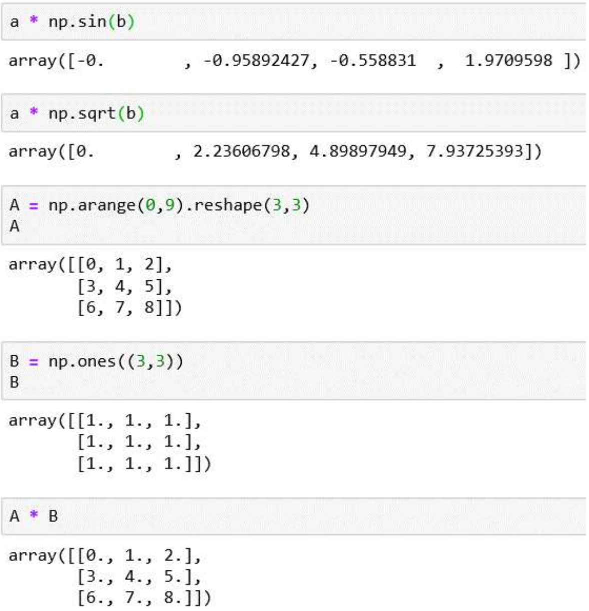
```
array([-4, -4, -4, -4])
```

```
a * b
```

函数算术运算符

数组a乘以数组b的sine函数或平方根函数（square root)

Element-wise multidimensional operation



```
a * np.sin(b)
```

```
array([-0.        , -0.95892427, -0.558831  ,  1.9709598 ])
```

```
a * np.sqrt(b)
```

```
array([0.        , 2.23606798, 4.89897949, 7.93725393])
```

```
A = np.arange(0,9).reshape(3,3)
A
```

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

```
B = np.ones((3,3))
B
```

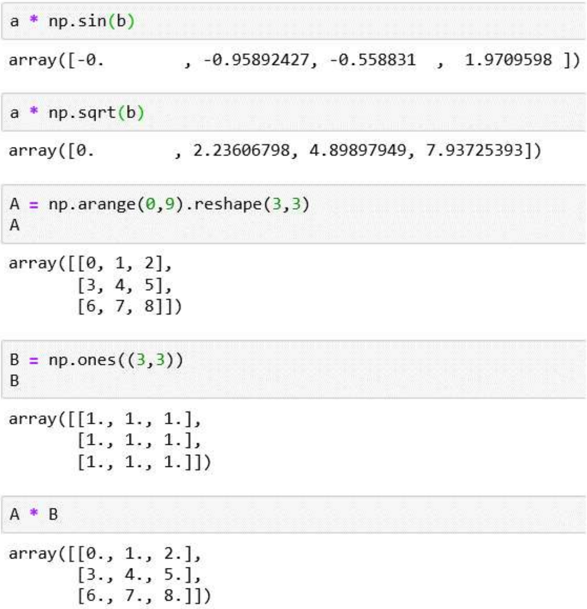
```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

```
A * B
```

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

想一想，练一练

- 函数运算



```
a * np.sin(b)
```

```
array([-0.        , -0.95892427, -0.558831  ,  1.9709598 ])
```

```
a * np.sqrt(b)
```

```
array([0.        , 2.23606798, 4.89897949, 7.93725393])
```

```
A = np.arange(0,9).reshape(3,3)
A
```

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

```
B = np.ones((3,3))
B
```

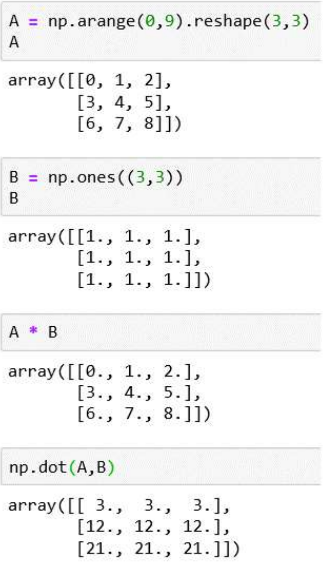
```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

```
A * B
```

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

矩阵乘积

- * operator as a matrix product when it is applied to two matrices.
- This operation is element-wise
- 矩阵代数相乘使用NumPy的dot()函数
- This operation is not element-wise



```
A = np.arange(0,9).reshape(3,3)
A
```

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

```
B = np.ones((3,3))
B
```

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

```
A * B
```

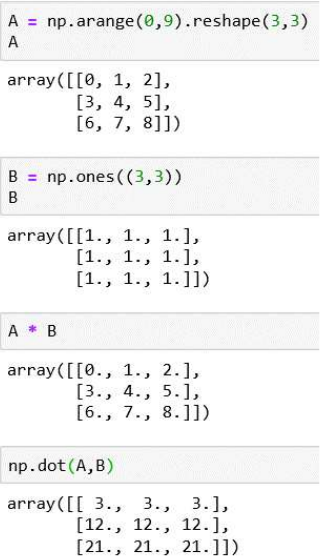
```
array([[0., 1., 2.],
       [3., 4., 5.],
       [6., 7., 8.]])
```

```
np.dot(A,B)
```

```
array([[ 3.,  3.,  3.],
       [12., 12., 12.],
       [21., 21., 21.]])
```

想一想，练一练

- 矩阵运算



```
A = np.arange(0,9).reshape(3,3)
A
```

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

```
B = np.ones((3,3))
B
```

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

```
A * B
```

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

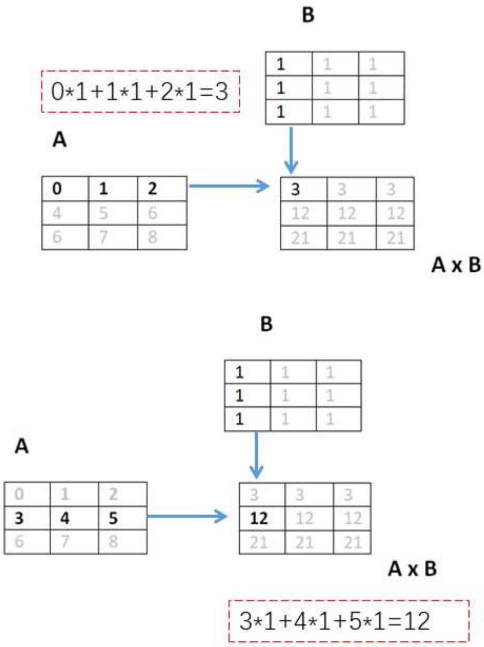
```
np.dot(A,B)
```

```
array([[ 3.,  3.,  3.],
       [12., 12., 12.],
       [21., 21., 21.]])
```

矩阵乘积

- NumPy使用点乘 `dot()` 函数.

- This operation is not element-wise

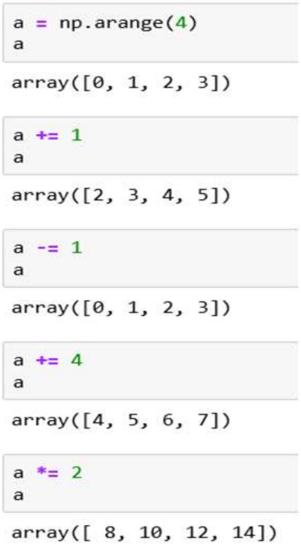


0*1+1*1+2*1=3

3*1+4*1+5*1=12

增减算符Operators

- Python中没有++ 或 - -
- Python中使用 +=
- Python中使用 −=



```
a = np.arange(4)
a
```

```
array([0, 1, 2, 3])
```

```
a += 1
a
```

```
array([2, 3, 4, 5])
```

```
a -= 1
a
```

```
array([0, 1, 2, 3])
```

```
a += 4
a
```

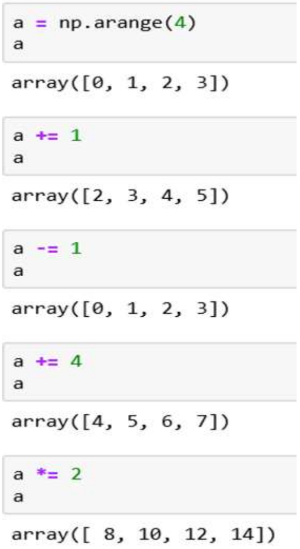
```
array([4, 5, 6, 7])
```

```
a *= 2
a
```

```
array([ 8, 10, 12, 14])
```

想一想，练一练

- 矩阵增减运算



```
a = np.arange(4)
a
```

```
array([0, 1, 2, 3])
```

```
a += 1
a
```

```
array([2, 3, 4, 5])
```

```
a -= 1
a
```

```
array([0, 1, 2, 3])
```

```
a += 4
a
```

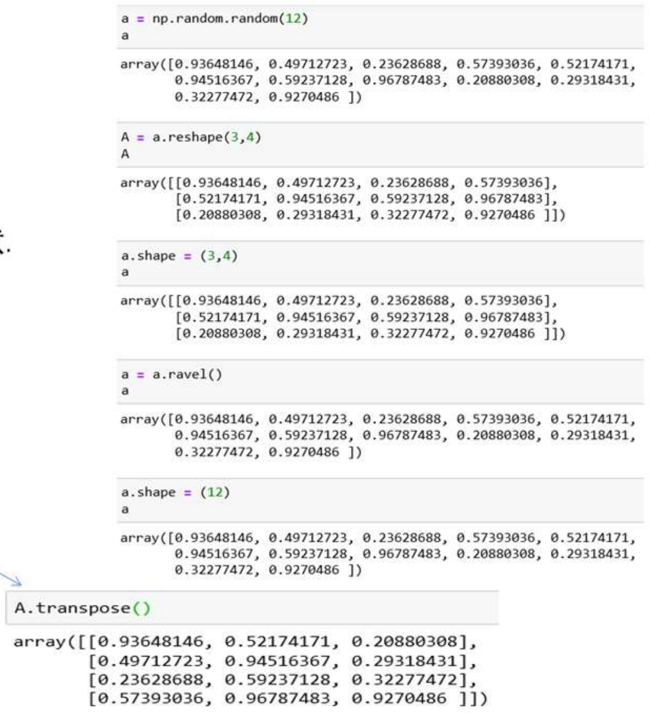
```
array([4, 5, 6, 7])
```

```
a *= 2
a
```

```
array([ 8, 10, 12, 14])
```

数组变形

- Shape manipulation
- reshape() 函数转换数组的形状.
  返回新的数据对象.
- ravel()
- transpose()



```
a = np.random.random(12)
a
```

```
array([0.93648146, 0.49712723, 0.23628688, 0.57393036, 0.52174171,
       0.94516367, 0.59237128, 0.96787483, 0.20880308, 0.29318431,
       0.32277472, 0.9270486 ])
```

```
A = a.reshape(3,4)
A
```

```
array([[0.93648146, 0.49712723, 0.23628688, 0.57393036],
       [0.52174171, 0.94516367, 0.59237128, 0.96787483],
       [0.20880308, 0.29318431, 0.32277472, 0.9270486 ]])
```

```
a.shape = (3,4)
a
```

```
array([[0.93648146, 0.49712723, 0.23628688, 0.57393036],
       [0.52174171, 0.94516367, 0.59237128, 0.96787483],
       [0.20880308, 0.29318431, 0.32277472, 0.9270486 ]])
```

```
a = a.ravel()
a
```

```
array([0.93648146, 0.49712723, 0.23628688, 0.57393036, 0.52174171,
       0.94516367, 0.59237128, 0.96787483, 0.20880308, 0.29318431,
       0.32277472, 0.9270486 ])
```

```
a.shape = (12)
a
```

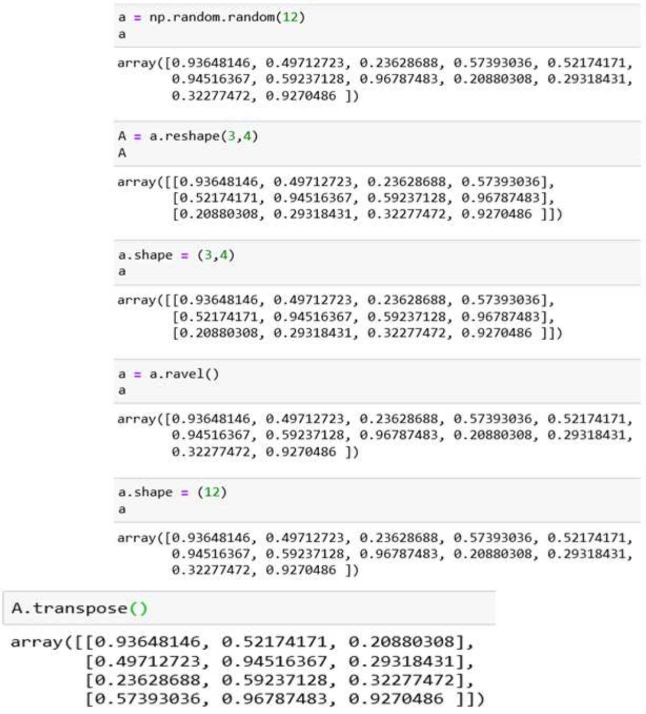
```
array([0.93648146, 0.49712723, 0.23628688, 0.57393036, 0.52174171,
       0.94516367, 0.59237128, 0.96787483, 0.20880308, 0.29318431,
       0.32277472, 0.9270486 ])
```

```
A.transpose()
```

```
array([[0.93648146, 0.52174171, 0.20880308],
       [0.49712723, 0.94516367, 0.29318431],
       [0.23628688, 0.59237128, 0.32277472],
       [0.57393036, 0.96787483, 0.9270486 ]])
```

想一想，练一练

- 数组变形reshape
- ravel
- transpose



```
a = np.random.random(12)
a
```

```
array([0.93648146, 0.49712723, 0.23628688, 0.57393036, 0.52174171,
       0.94516367, 0.59237128, 0.96787483, 0.20880308, 0.29318431,
       0.32277472, 0.9270486 ])
```

```
A = a.reshape(3,4)
A
```

```
array([[0.93648146, 0.49712723, 0.23628688, 0.57393036],
       [0.52174171, 0.94516367, 0.59237128, 0.96787483],
       [0.20880308, 0.29318431, 0.32277472, 0.9270486 ]])
```

```
a.shape = (3,4)
a
```

```
array([[0.93648146, 0.49712723, 0.23628688, 0.57393036],
       [0.52174171, 0.94516367, 0.59237128, 0.96787483],
       [0.20880308, 0.29318431, 0.32277472, 0.9270486 ]])
```

```
a = a.ravel()
a
```

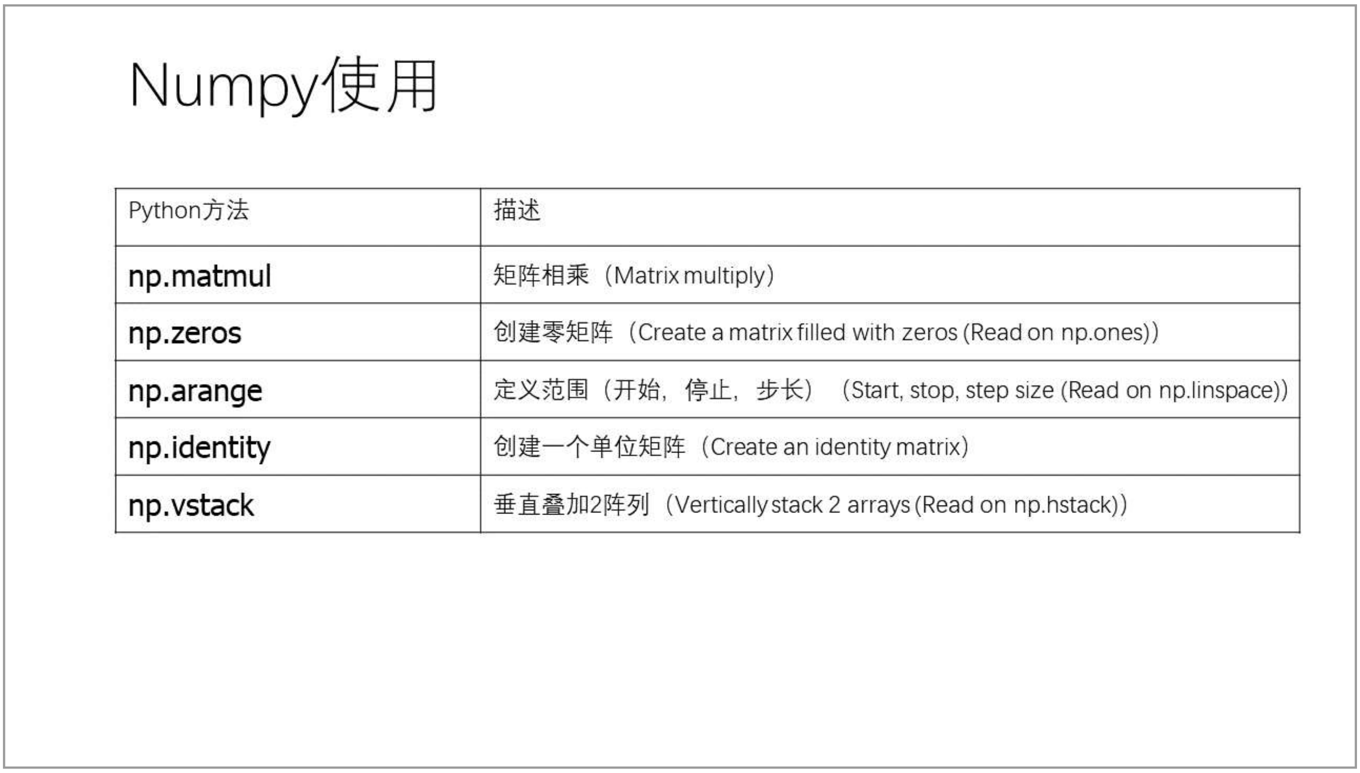
```
array([0.93648146, 0.49712723, 0.23628688, 0.57393036, 0.52174171,
       0.94516367, 0.59237128, 0.96787483, 0.20880308, 0.29318431,
       0.32277472, 0.9270486 ])
```

```
a.shape = (12)
a
```

```
array([0.93648146, 0.49712723, 0.23628688, 0.57393036, 0.52174171,
       0.94516367, 0.59237128, 0.96787483, 0.20880308, 0.29318431,
       0.32277472, 0.9270486 ])
```

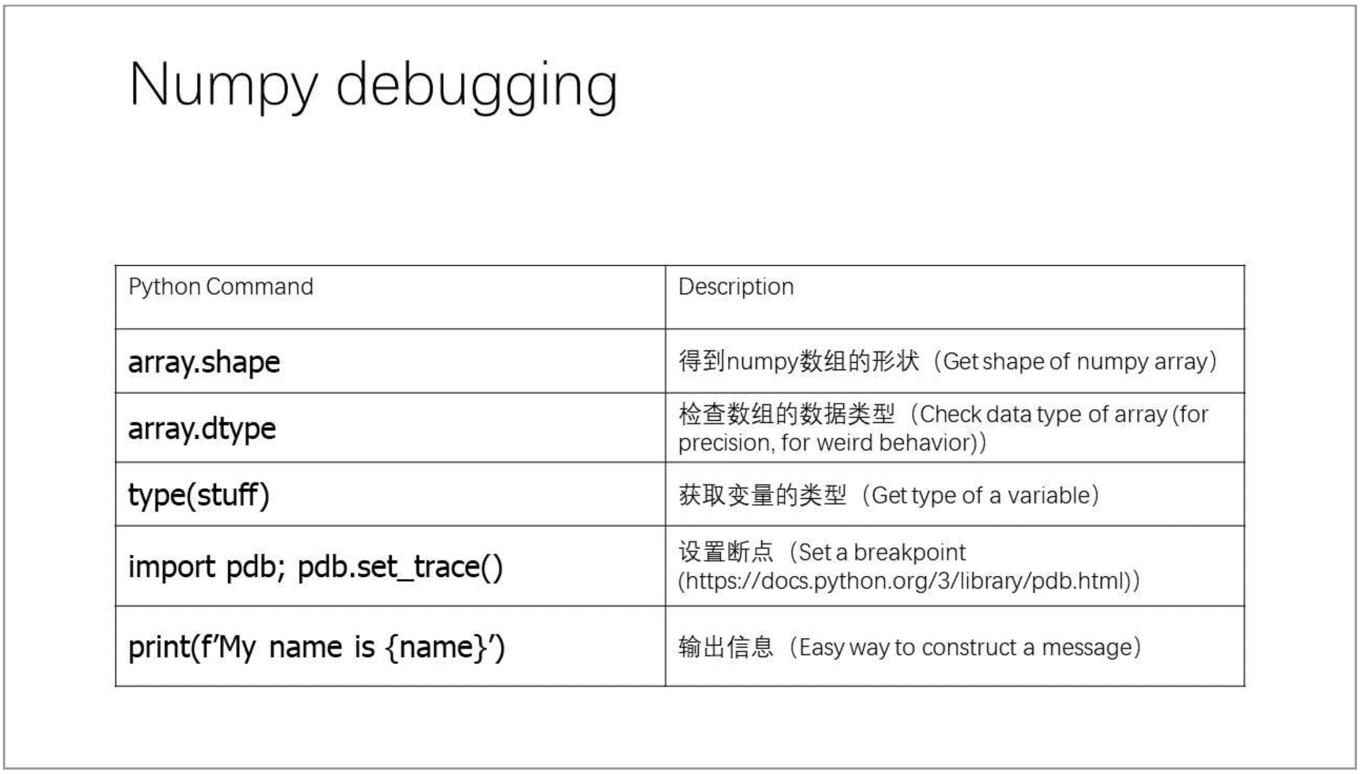
```
A.transpose()
```

```
array([[0.93648146, 0.52174171, 0.20880308],
       [0.49712723, 0.94516367, 0.29318431],
       [0.23628688, 0.59237128, 0.32277472],
       [0.57393036, 0.96787483, 0.9270486 ]])
```



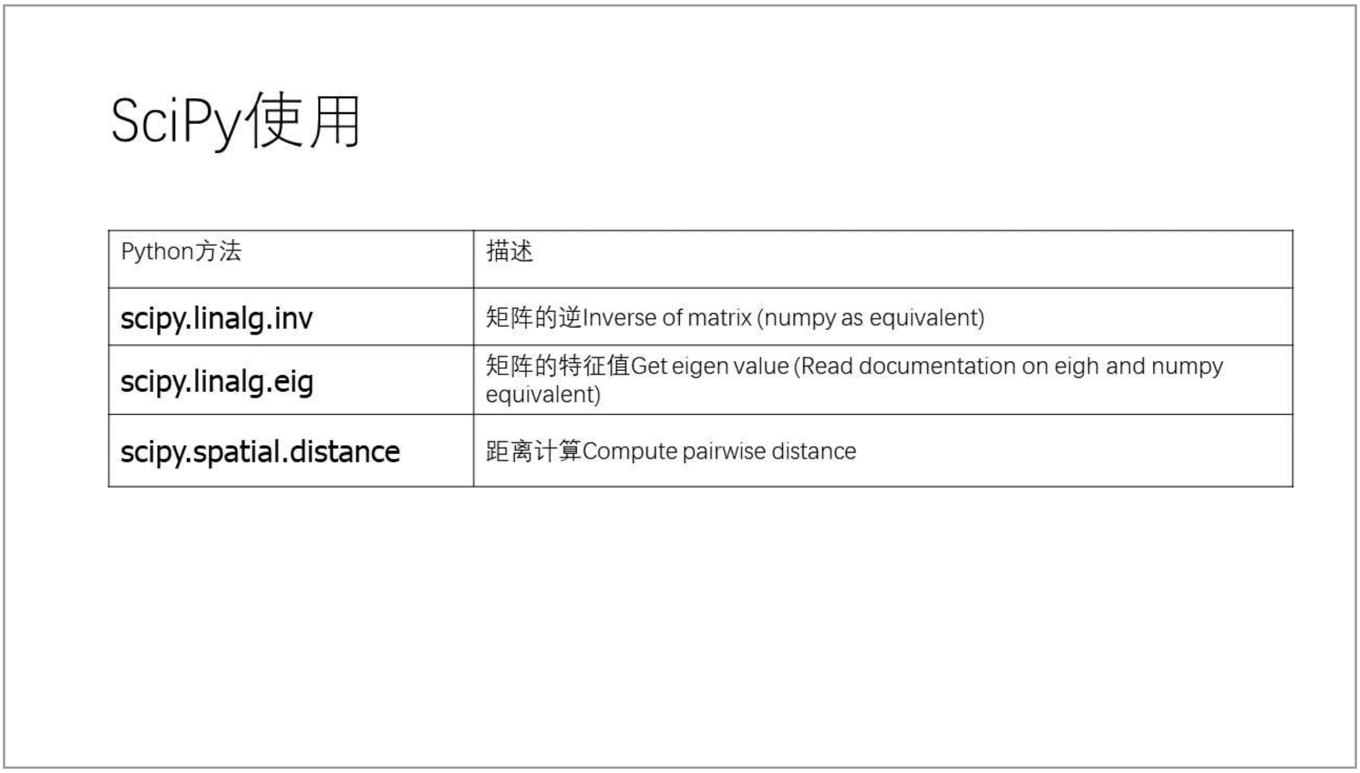
Numpy使用

Python方法	描述
np.matmul	矩阵相乘（Matrix multiply）
np.zeros	创建零矩阵（Create a matrix filled with zeros (Read on np.ones)）
np.arange	定义范围（开始，停止，步长）（Start, stop, step size (Read on np.linspace)）
np.identity	创建一个单位矩阵（Create an identity matrix）
np.vstack	垂直叠加2阵列（Vertically stack 2 arrays (Read on np.hstack)）



Numpy debugging

Python Command	Description
array.shape	得到numpy数组的形状（Get shape of numpy array）
array.dtype	检查数组的数据类型（Check data type of array (for precision, for weird behavior)）
type(stuff)	获取变量的类型（Get type of a variable）
import pdb; pdb.set_trace()	设置断点（Set a breakpoint (https://docs.python.org/3/library/pdb.html)）
print(f’My name is {name}’)	输出信息（Easy way to construct a message）



SciPy使用

Python方法	描述
scipy.linalg.inv	矩阵的逆Inverse of matrix (numpy as equivalent)
scipy.linalg.eig	矩阵的特征值Get eigen value (Read documentation on eigh and numpy equivalent)
scipy.spatial.distance	距离计算Compute pairwise distance

谢谢指正！