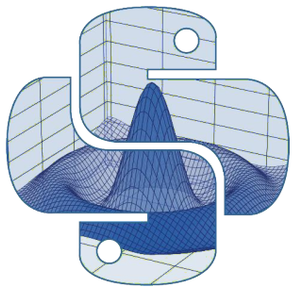
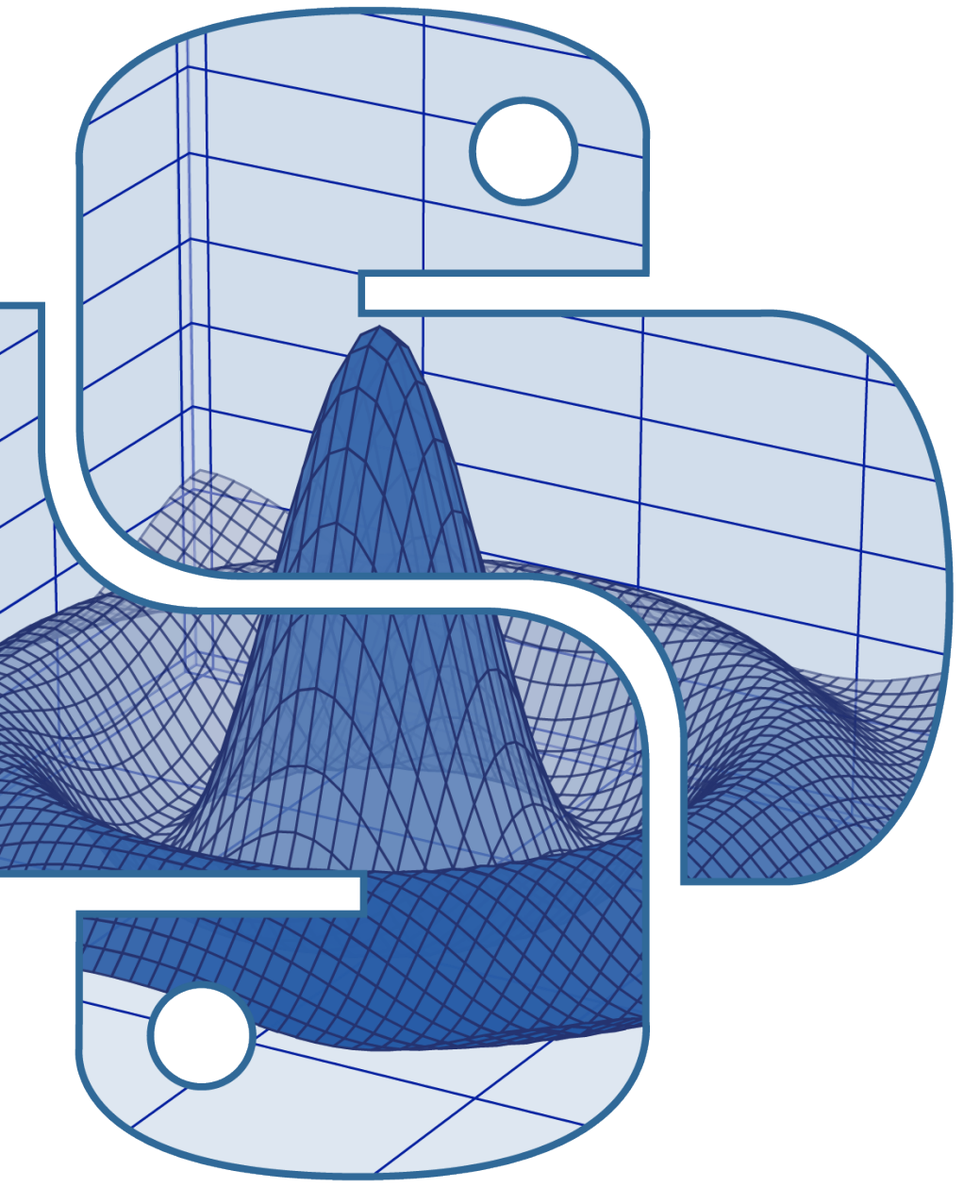


课题组培训—2020

Python Basics, Numpy

JinyuanHu (胡晋媛)

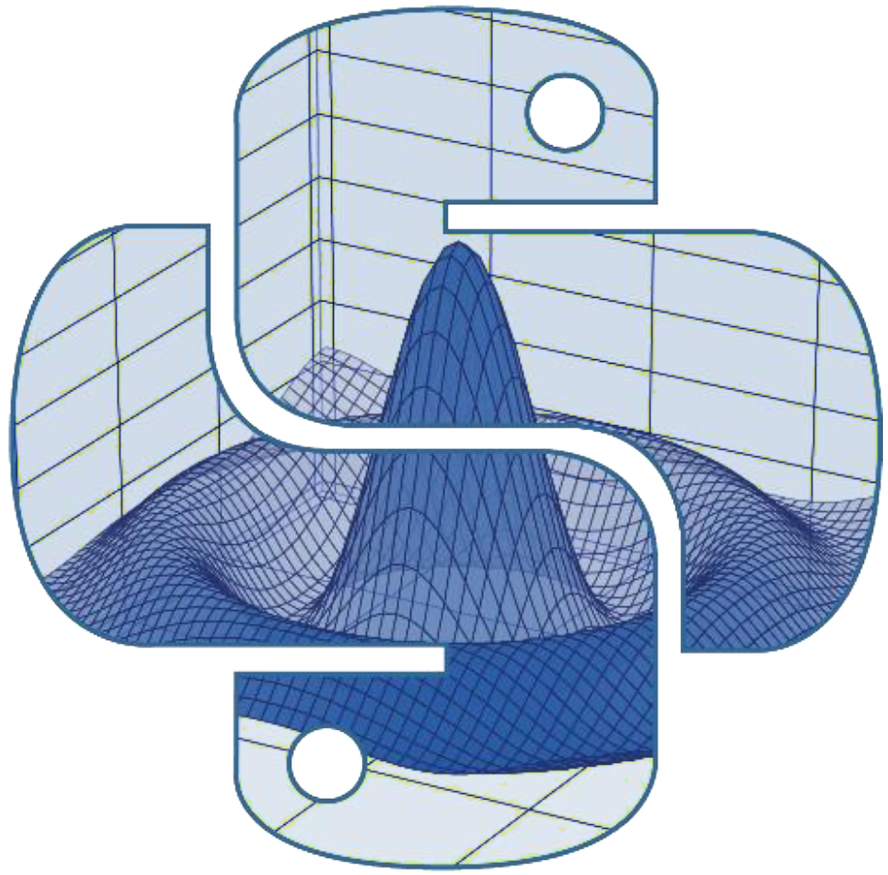
2020.09.05



Python Basics

目录

- 简介与安装
- 基础语法
- 函数
- 模块
- 应用

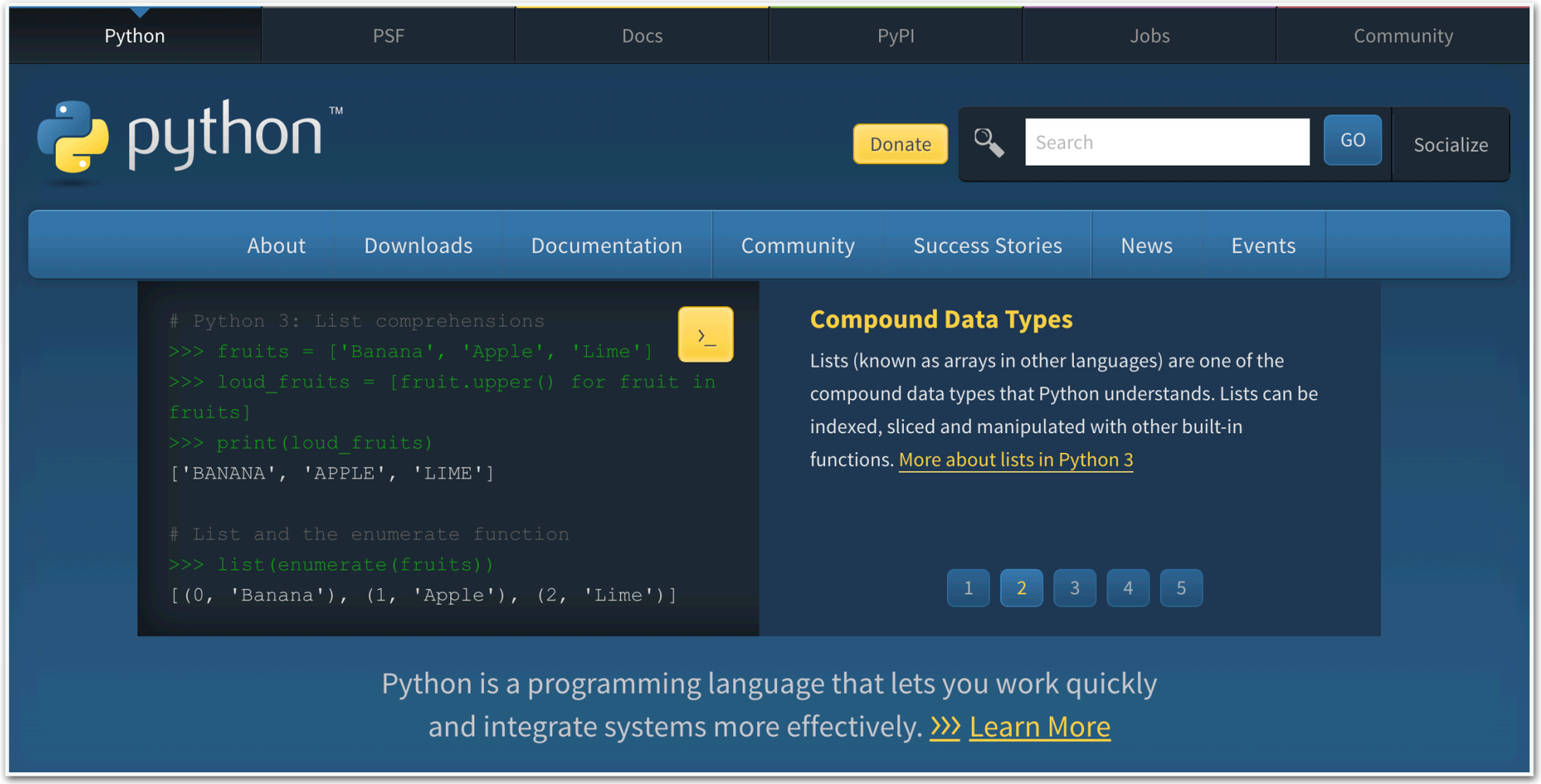


Python 简介

一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。

1. 易学习，易阅读，易维护
2. 可移植，可扩展，可嵌入
3. 互动模式，标准库，数据库

https://www.python.org



Python 环境搭建

ANACONDA. https://www.anaconda.com/products/individual

开源的Python包管理器

Anaconda Installers

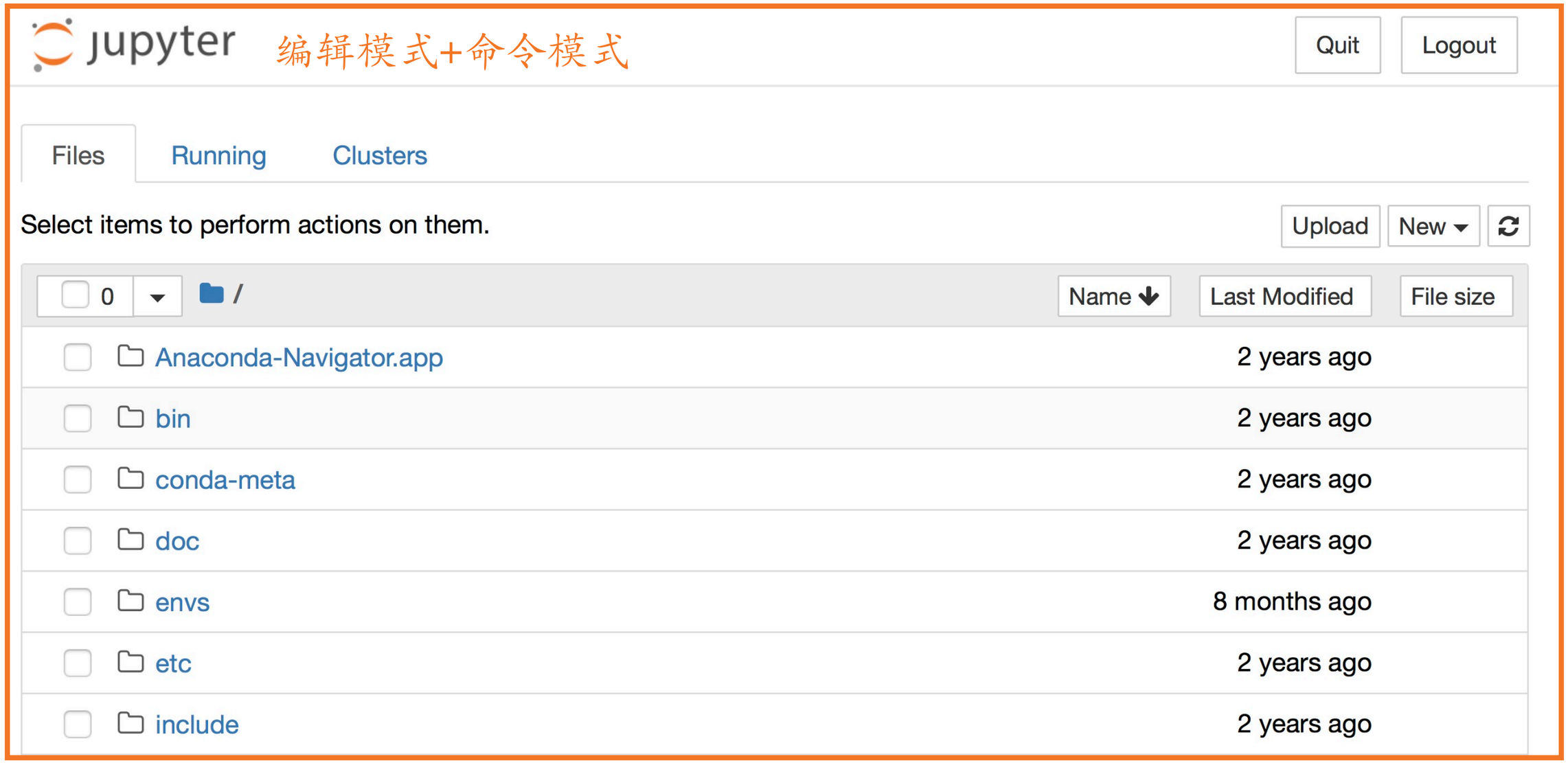
Windows	MacOS	Linux
Python 3.7	Python 3.7	Python 3.7
64-Bit Graphical Installer (466 MB)	64-Bit Graphical Installer (442 MB)	64-Bit (x86) Installer (522 MB)
32-Bit Graphical Installer (423 MB)	64-Bit Command Line Installer (430 MB)	64-Bit (Power8 and Power9) Installer (276 MB)
Python 2.7	Python 2.7	Python 2.7
64-Bit Graphical Installer (413 MB)	64-Bit Graphical Installer (637 MB)	64-Bit (x86) Installer (477 MB)
32-Bit Graphical Installer (356 MB)	64-Bit Command Line Installer (409 MB)	64-Bit (Power8 and Power9) Installer (295 MB)

Python 代码编辑器

jupyter https://jupyter.org

```
$ pip install jupyter
```

```
$ jupyter notebook
```



Python + Terminal

```
#!/Users/jinyuanhu/anaconda3/bin/python
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.gridspec as gridspec

data0 = np.loadtxt("TDOS.dat")

x, y1, y2 = data0[:,0], data0[:,1], data0[:,2]
#Parameter
```

- 设置解释器: #!/User/bin/env python
- 保存: filename.py
- 可执行文件: chmod u+x filename.py
- 运行脚本: python filename.py or ./filename.py

Python + Linux

```
#!/bin/bash
mkdir ./out ./dos
for i in $(seq 1 1 4)
do
awk '{print $1,-1*$2}' k${i}-B.out > 1.out && cat k${i}-A.out 1.out > k${i}.dat
done
rm 1.out
/share/apps/miniconda3/bin/python3 << EOF
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.gridspec as gridspec

data0 = np.loadtxt("TDOS.dat")

a, y1, y2 = data0[:,0], data0[:,1], data0[:,2]
#Parameter
.....
EOF
```

- 保存: script.sh

- 内置环境配置:
  python environment << EOF
  ***
  EOF

- 运行脚本: sh script.sh or ./script.sh

Python 基础语法

标识符

- 字母，下划线，数字
- 区分大小写

keyword

```
# keyword 不能用作标识符
```

```
import keyword
```

```
keyword.kwlist
```

print输出

```
print('Hello, Python!')
```

```
Hello, Python!
```

注释

```
# 输出
print('Hello, Python!')
```

```
Hello, Python!
```

python使用缩进来表示代码块，无需{}

import与from...import

```
# 导入整个模块
```

```
import numpy
```

```
# 从某模块导入某函数
```

```
from numpy import pi
```

```
# 从某模块导入多个函数
```

```
from numpy import pi,e
e
```

```
2.718281828459045
```

```
# 从某模块导入所有函数
```

```
from numpy import *
```

行与缩进

```
if True:
    print ("True")
else:
    print ("False")
```

```
True
```

Python 标准数据类型

- Number（数字） — int, float, bool, complex
- String (字符串) — ‘python’ or “python”
- List (列表) — [‘python’，2020]
- Tuple (元组) — (‘python’，2020)
- Set (集合) — { ‘python’，2020 }
- Dictionary (字典) — { ‘code’: ‘python’, ‘time’: ‘2020’ }

不可变数据：***Number，String，Tuple***

可变数据：***List，Set，Dictionary***

Python 条件控制

If 语句

```
is_black = True
is_white = False

if is_black:
    print("The color is black")
elif is_white:
    print("The color is white")
else:
    print("Don't know the color")
```

```
The color is black
```

for 循环

```
for letter in "Physical Chemistry":
    print(letter, end=" ")
```

```
P h y s i c a l   C h e m i s t r y
```

break: 跳出 for 和 while 的循环体。

continue: 跳过当前循环块剩余语句，继续下一轮循环。

while 循环

```
i = 1
while i <= 13:
    print(i, end=" ")
    i += 1
else:
    print("\nDone with loop")
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13
Done with loop
```

break + continue

```
num = 5
while num > 0:
    num -= 1
    if num == 1:
        continue
    print(num, end=" ")
print('Done with loop')
```

```
4 3 2 0 Done with loop
```

```
num = 5
while num > 0:
    num -= 1
    if num == 1:
        break
    print(num, end=" ")
print('Done with loop')
```

```
4 3 2 Done with loop
```

Python 函数

定义函数

```
def say_hi(name):
    print(f'Hello {name} !')

say_hi("Xiamen")
```

```
Hello Xiamen !
```

调用函数

```
def cube(num):
    num*num*num

print(cube(3))
```

```
None
```

```
def cube(num):
    return num*num*num

print(cube(3))
```

```
27
```

```
# 从其它文件中调用某一个函数
```

```
import useful_tools

print(useful_tools.roll_dice(10))
```

```
8
```

运算符函数

https://docs.python.org/3/library/operator.html

内置函数

https://docs.python.org/3/library/functions.html

数学函数

https://docs.python.org/3/library/math.html

数理统计函数

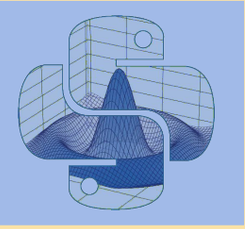
https://docs.python.org/3/library/statistics.html

迭代函数

https://docs.python.org/3/library/itertools.html

Python 模块

https://docs.python.org/3/py-modindex.html



从.py调用某一模块

```
import cube

print(cube.cube(3))
```

```
27
```

调用内置函数

```
from math import *

print(floor(3.7))
print(ceil(3.1))
print(sqrt(36))
```

```
3
4
6.0
```

安装第三方模块

```
pip --version
```

```
pip 20.0.2 from /Users/jinyuanhu/anaconda3/lib/python3.7/site-packages/pip (python 3.7)
Note: you may need to restart the kernel to use updated packages.
```

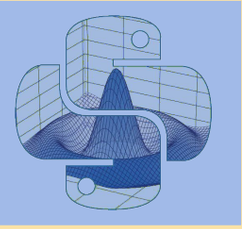
```
pip install python-docx
```

```
pip uninstall python-docx
```

```
import docx
```

⚠️自己创建模块时注意命名，不能与Python自带的模块名称冲突

Python 异常处理



```
number = int(input("Enter a number: "))
print(number)
```

```
Enter a number: python

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-100-714ac031a43d> in <module>
----> 1 number = int(input("Enter a number: "))
      2 print(number)

ValueError: invalid literal for int() with base 10: 'python'
```

```
value = 10/0
```

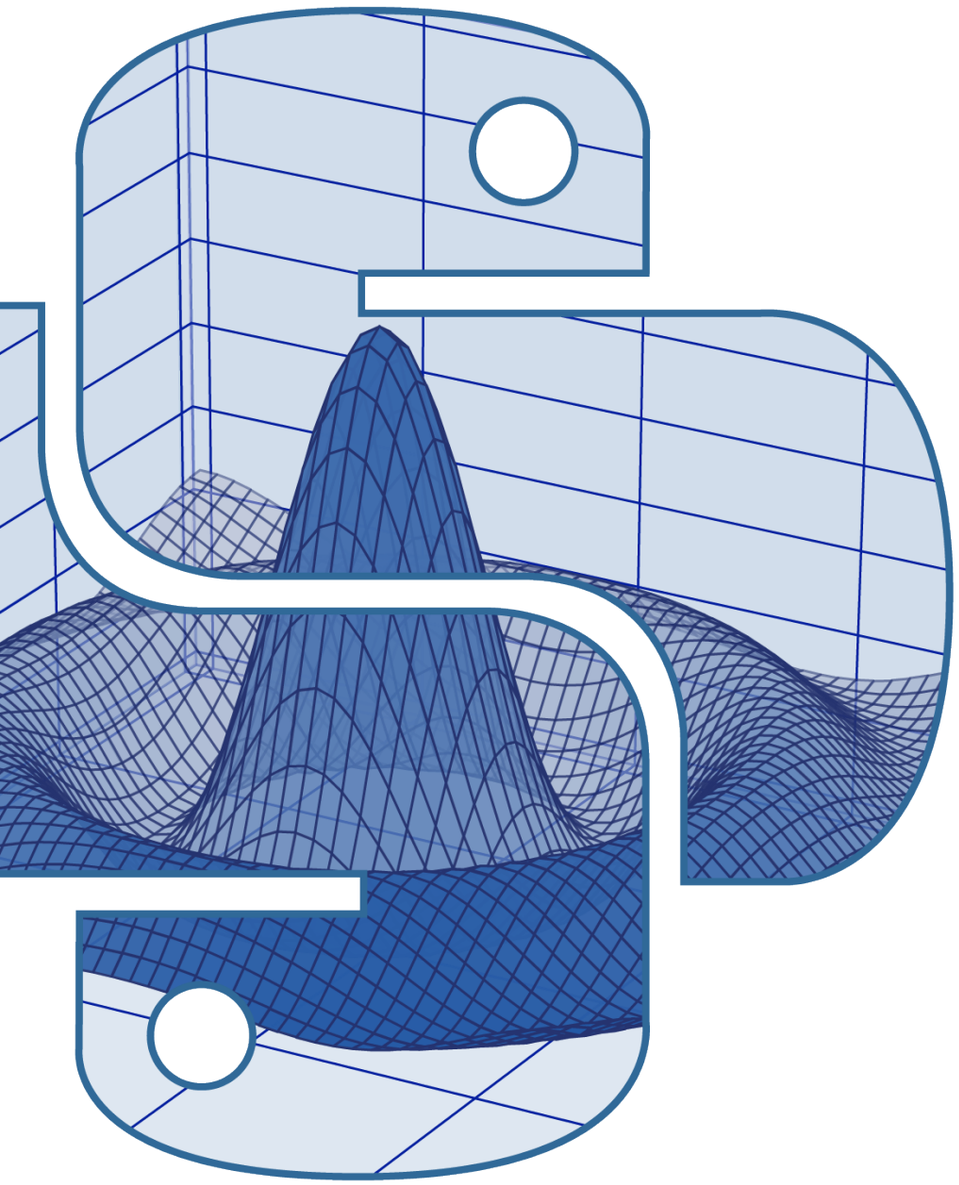
```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-104-669cf63e4443> in <module>
----> 1 value = 10/0

ZeroDivisionError: division by zero
```

```
try:
    #value = 10/0
    number = int(input("Enter a number: "))
    print(number)
except ZeroDivisionError as err:
    print(err)
except ValueError:
    print("Invalid Input")
```

```
Enter a number: Python
Invalid Input
```

try:

except:



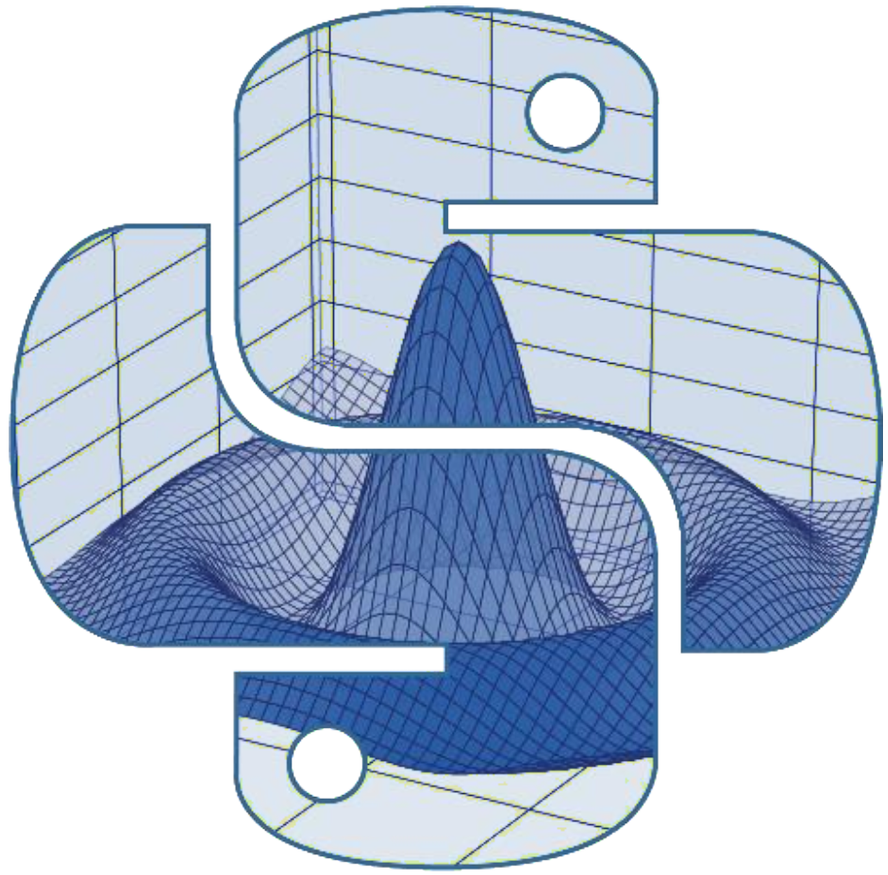
Numpy

The Standard Numerical Library
for Python

目录

NumPy

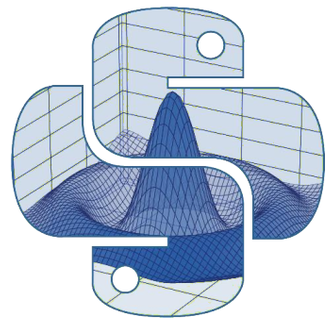
- 简介与安装
- ND Array
- 内置函数
- 应用实例



Numpy 简介与安装

NumPy

NumPy(Numerical Python) 是 Python 语言的一个扩展程序库，是一个运行速度非常快的数据库，主要用于数组计算。



- 强大的N维度数组对象 ndarray
- 整合C/C++/Fortran 代码的工具

NumPy + SciPy (Scientific Python) + Matplotlib（绘图库）

可以通过Pip 或 Anaconda 安装 Numpy

```
$ conda install numpy
```

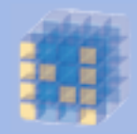
```
$ pip install numpy
```

```
$ python -m pip list | grep numpy
```

SciPy.org

matplotlib

相关网址



NumPy

- Python 官网: https://www.python.org
- NumPy 官网: http://www.numpy.org/
- NumPy 源代码： https://github.com/numpy/numpy
- SciPy 官网： https://www.scipy.org/
- SciPy 源代码： https://github.com/scipy/scipy
- Matplotlib 官网： https://matplotlib.org/
- Matplotlib 源代码： https://github.com/matplotlib/matplotlib

N 维数组 —(Array)

NumPy

将Numpy引入Jupyter

```
import numpy as np
```

创建简单数组

```
a = np.array([1,2,3])
print(a)
```

```
[1 2 3]
```

检查类型

```
type(a)
```

```
numpy.ndarray
```

检查元素类型

```
a.dtype
```

```
dtype('int64')
```

查看数组维度

```
a.ndim
```

```
1
```

数组形状

```
# 属性，返回一个元组
a.shape
```

```
(3,)
```

数组尺寸

```
# 每个元素的长度(字节)
a.itemsize
```

```
8
```

```
# 数组中的元素个数
a.size
```

```
3
```

内存占用 (字节)

```
# 数据部分占用的内存
a.nbytes
```

```
24
```

数组算符

NumPy

简单数学运算

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

```
a + b
```

```
array([3, 5, 7, 9])
```

```
a * b
```

```
array([ 2,  6, 12, 20])
```

```
a ** b
```

```
array([   1,    8,   81, 1024])
```

Numpy 中定义的constants：

pi = 3.14159265359

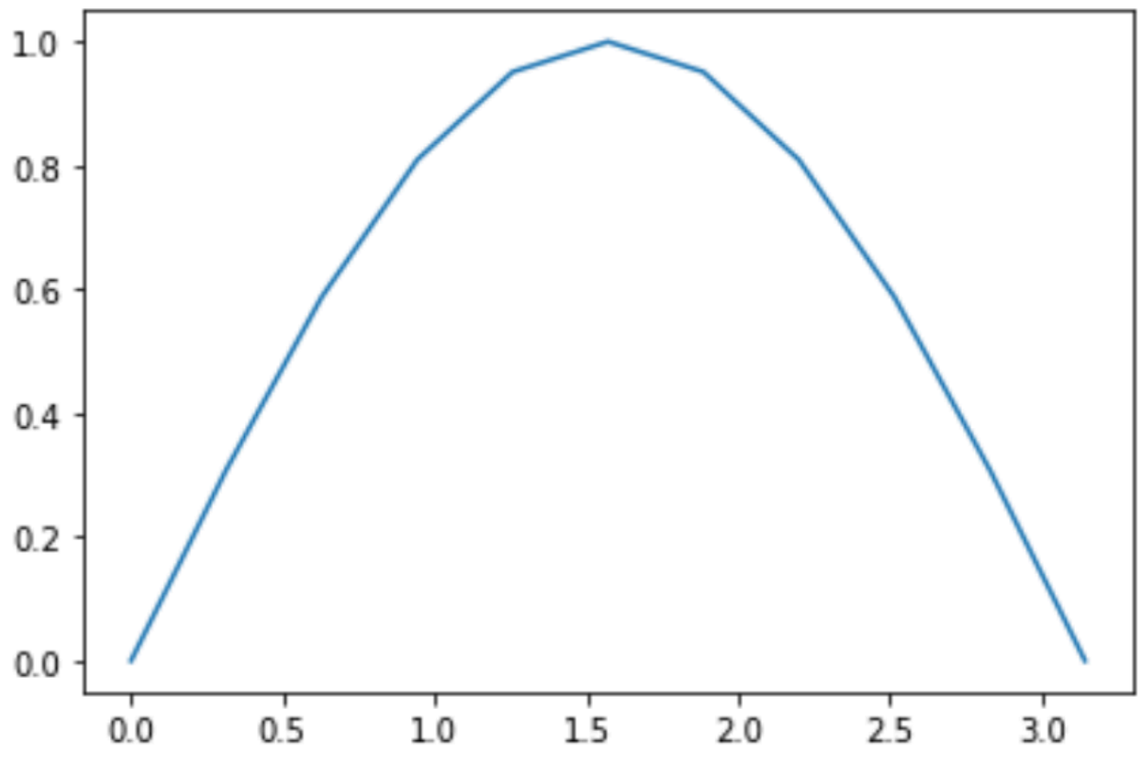
e = 2.71828182846

数学函数

```
# 创建(0~10)的数组
a = np.arange(11)
print(a)
```

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

```
# 数学函数
x = a*np.pi/10
y = np.sin(x)
```



元素修改

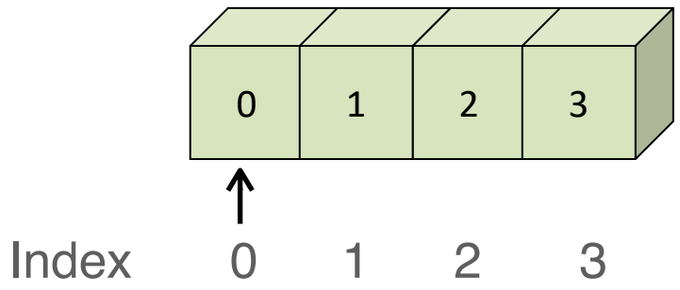
NumPy

数组索引

```
a = np.array([0,1,2,3])
```

```
a[0]
```

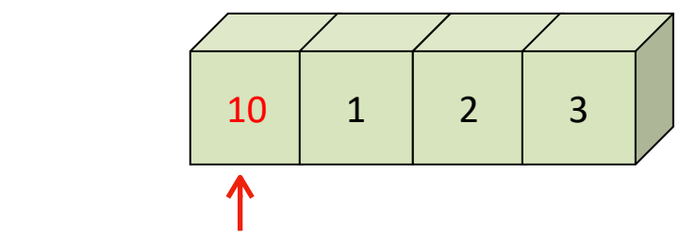
```
0
```




```
a[0]=10
```

```
a
```

```
array([10,  1,  2,  3])
```



注意数组类型

```
a.dtype
```

```
dtype('int64')
```

```
a[0] = 10.8
print(a)
```

```
[10   1   2   3]
```

```
# 人为传输浮点类型进去，浮点数会被截断
# 利用fill函数，作用相同
```

```
a.fill(-4.8)
print(a)
```

```
[-4.8 -4.8 -4.8 -4.8]
```

```
a = np.array([0,1,2,3], dtype='float64')
print(a)
print(a.dtype)
```

```
[0. 1. 2. 3.]
float64
```

```
a[0] = 10.8
print(a)
```

```
[10.8  1.   2.   3. ]
```

多维数组

NumPy

二维数组

```
a = np.array([[0,1,2,3], [10,11,12,13]])
print(a)
```

```
[[ 0  1  2  3]
 [10 11 12 13]]
```

形状 (Rows, Columns)

```
a.shape
```

```
(2, 4)
```

尺寸

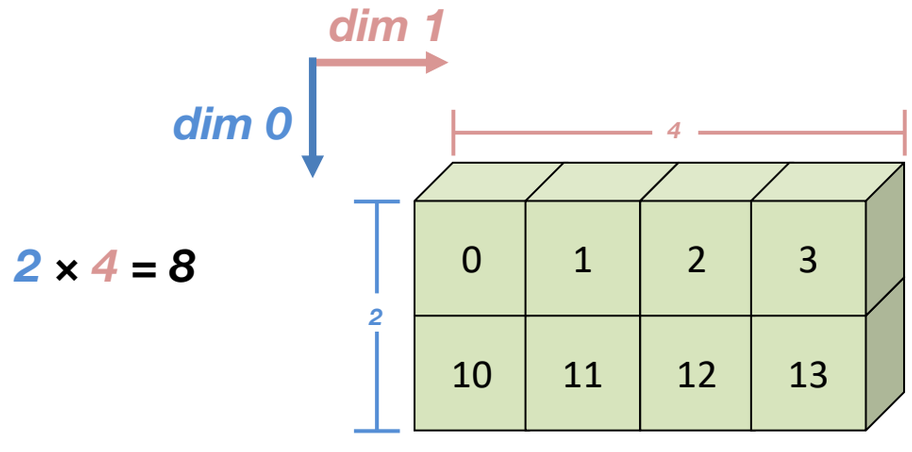
```
a.size
```

```
8
```

维度

```
a.ndim
```

```
2
```



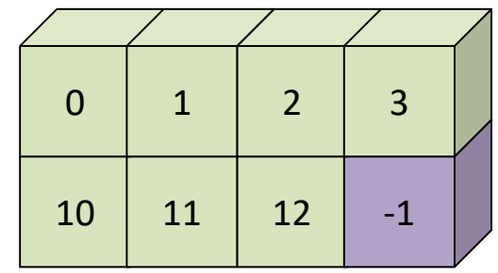
获取数组元素

```
a[1,3]
```

```
13
```

```
a[1,3] = -1
print(a)
```

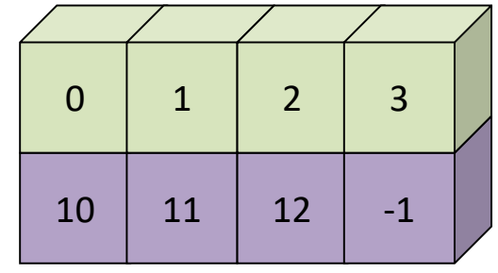
```
[[ 0  1  2  3]
 [10 11 12 -1]]
```



使用单个索引选取第二行

```
a[1]
```

```
array([10, 11, 12, -1])
```



切片

NumPy

简单数组

```
a = np.array([10,11,12,13,14])
print(a)
```

```
[10 11 12 13 14]
```

```
a[1:3]
```

```
array([11, 12])
```

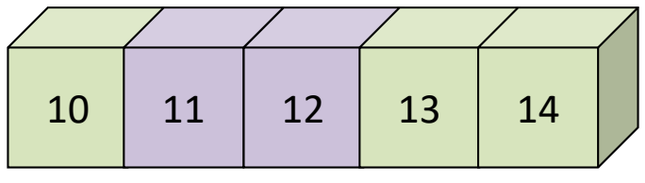
```
a[1:-2]
```

```
array([11, 12])
```

```
a[-4:3]
```

```
array([11, 12])
```

负指数也可以使用



省略式索引

省略的边界被认为是列表的开始（或结束）

```
a[:3]
```

```
array([10, 11, 12])
```

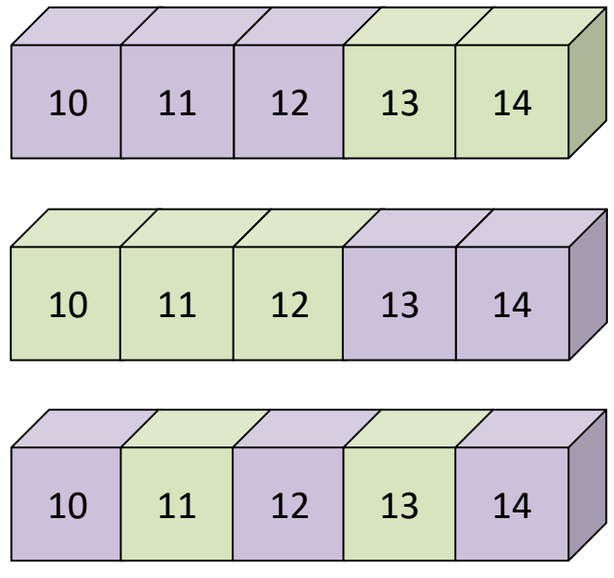
```
a[-2:]
```

```
array([13, 14])
```

步长为2进行索引

```
a[::2]
```

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



数组切片

NumPy

二维数组

```
a = np.arange(36).reshape(6,6)
```

```
orange = a[0, 3:5]
print(orange)
```

```
[3 4]
```

```
pink = a[:, 2]
print(pink)
```

```
[ 2  8 14 20 26 32]
```

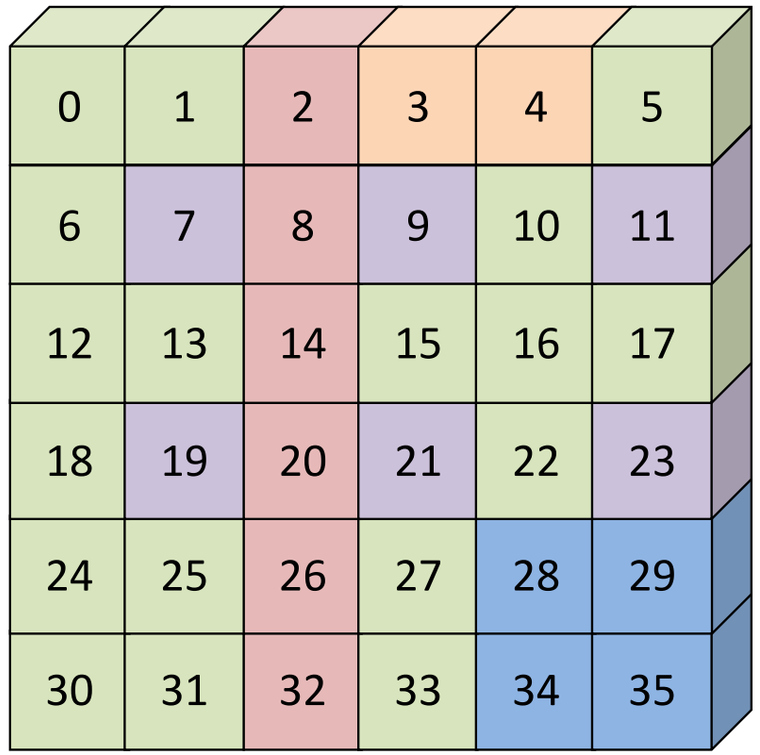
```
blue = a[4:, 4:]
print(blue)
```

```
[[28 29]
 [34 35]]
```

```
purple = a[1:-1:2, 1::2]
print(purple)
```

```
[[ 7  9 11]
 [19 21 23]]
```

numpy在大多数情况下尽量不创建一个副本，它不复制数据，只是指向内存中的同一个位置，不需要任何新的内存分配




```
pink.flags
```

```
  C_CONTIGUOUS : False
  F_CONTIGUOUS : False
  OWNDATA : False
  WRITEABLE : True
  ALIGNED : True
  WRITEBACKIFCOPY : False
  UPDATEIFCOPY : False
```

pink是a数组的一个切片，不拥有数据本身，数据属于别的变量

高级索引

NumPy

按位置索引

```
a = np.arange(0, 100, 10)
print(a)
```

```
[ 0 10 20 30 40 50 60 70 80 90]
```

```
indices = [1, 2, -3]
```

```
y = a[indices]
print(y)
```

```
[10 20 70]
```

用布尔值建立索引

```
# 手动构建一个布尔值数组
```

```
mask = np.array([0, 1, 1, 0, 0, 0, 0, 1, 0, 0],
                dtype = bool)
```

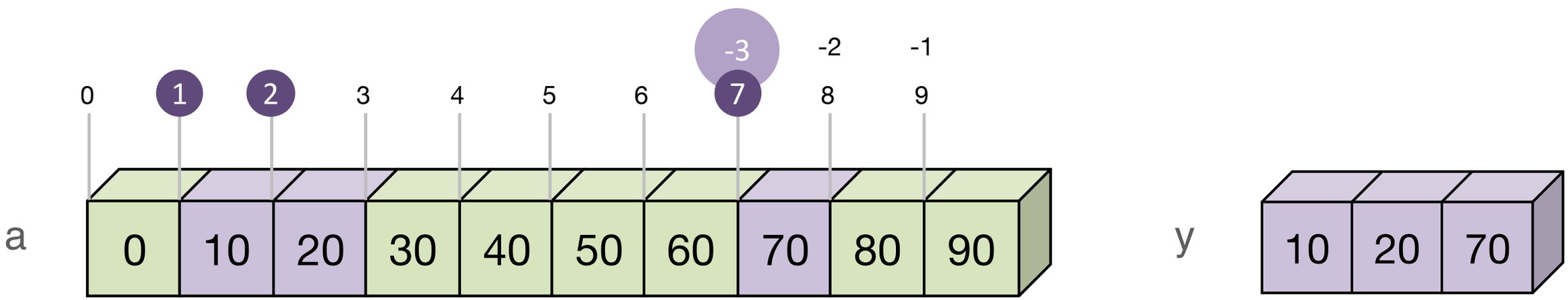
```
y = a[mask]
print(y)
```

```
[10 20 70]
```

```
mask2 = a < 30
```

```
z = a[mask2]
print(z)
```

```
[ 0 10 20]
```



位运算符

NumPy

& (and)　| (or)　~ (not)　^ (xor) 位运算符

```
a = np.arange(10)
print(a)
```

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

```
con1 = (a > 3) & (a < 8)
print(con1)
```

```
[False False False False  True  True  True  True
False False]
```

```
print(a[con1])
```

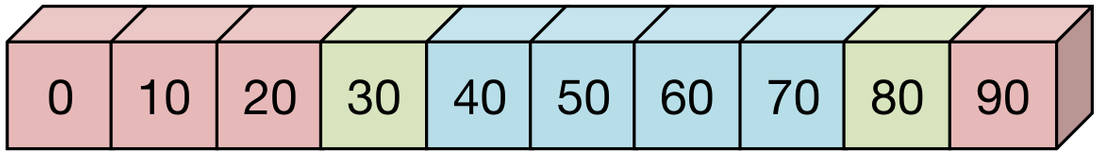
```
[4 5 6 7]
```

```
con2 = (a < 3) | (a > 8)
print(con2)
```

```
[ True  True  True False False False False False
False  True]
```

```
print(a[con2])
```

```
[0 1 2 9]
```



~ 对数组中整数进行位取反运算

```
a = np.binary_repr(13, width=8)
print(a)
print('**********')
b = ~ (np.array([13], dtype = np.uint8))
print(b)
print('**********')
c = np.binary_repr(242, width=8)
print(c)
```

```
00001101
**********
[242]
**********
11110010
```

^ 同为假，异为真

```
0 ^ 0
```

```
0
```

```
0 ^ 3
```

```
3
```

2D—高级索引

按位置索引

```
a = np.arange(36).reshape(6,6)
```

```
orange = a[[0,1,2,3,4], [1,2,3,4,5]]
print(orange)
```

```
[ 1  8 15 22 29]
```

```
blue = a[:3, [0,4,5]]
print(blue)
```

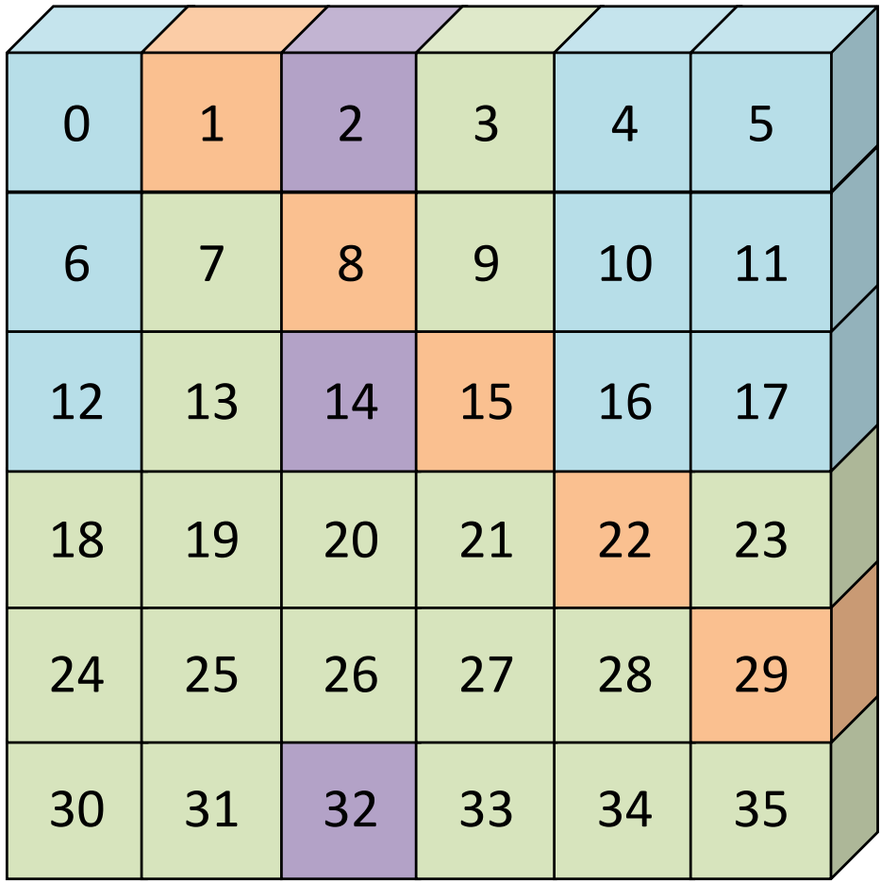
```
[[ 0  4  5]
 [ 6 10 11]
 [12 16 17]]
```

```
mask = np.array([1, 0, 1, 0, 0, 1],
                dtype = bool)
```

```
purple = a[mask, 2]
print(purple)
```

```
[ 2 14 32]
```

与切片不同，高级索引创建副本而不是原始数组中的视图



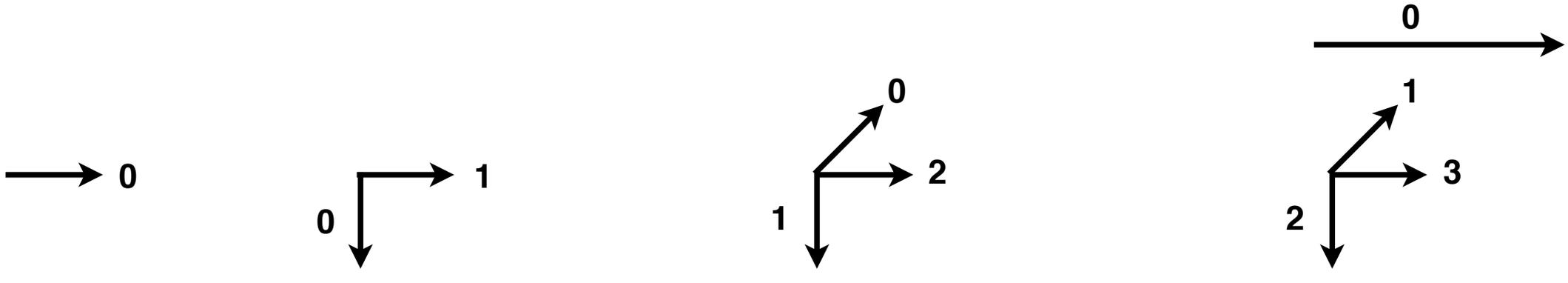

```
purple.flags
```

```
  C_CONTIGUOUS : True
  F_CONTIGUOUS : True
  OWNDATA : True
  WRITEABLE : True
  ALIGNED : True
  WRITEBACKIFCOPY : False
  UPDATEIFCOPY : False
```

多维数组

NumPy

可视化多维数组



数组构造实例

NumPy

浮点数组

```
a = np.array([0, 1.03, 2, 3])
print(a)
```

```
[0.   1.03 2.   3.  ]
```

```
a.dtype
```

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

```
a.nbytes
```

```
32
```

降低精度

```
a = np.array([0, 1.03, 2, 3], dtype='float32')
```

```
a.dtype
```

```
dtype('float32')
```

```
a.nbytes
```

```
16
```

无符号整数字节

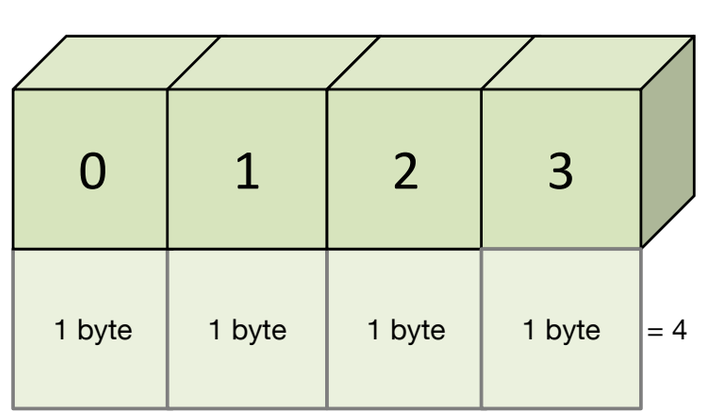
```
a = np.array([0, 1, 2, 3], dtype='uint8')
```

```
a.dtype
```

```
dtype('uint8')
```

```
a.nbytes
```

```
4
```




```
Base 2                  Base 10

00000000    ->   0 = 0*2**0 + 0*2**1 + ... + 0*2**7
00000001    ->   1 = 1*2**0 + 0*2**1 + ... + 0*2**7
00000010    ->   2 = 0*2**0 + 1*2**1 + ... + 0*2**7
...
11111111    -> 255 = 1*2**0 + 1*2**1 + ... + 1*2**7
```

数组构造函数

NumPy

arange

numpy.arange([start,] stop[, step,], dtype=None)

```
np.arange??
```

```
np.arange(4)
```

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

```
from numpy import pi
```

```
np.arange(0, 2*pi, pi/4)
```

```
array([0.        , 0.78539816, 1.57079633, 2.356194
49, 3.14159265,
       3.92699082, 4.71238898, 5.49778714])
```

```
np.arange(8)
```

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

```
np.arange(8).reshape(2,2,2)
```

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

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

ones, zeros

numpy.ones(shape, dtype=None)

numpy.zeros(shape, dtype=None)

shape是指定数组维数的数字或序列。
未指定dtype默认为float64

```
np.ones((2,3), dtype='float32')
```

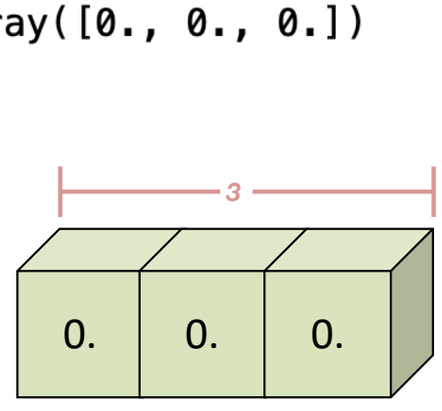
```
array([[1., 1., 1.],
       [1., 1., 1.]], dtype=float32)
```

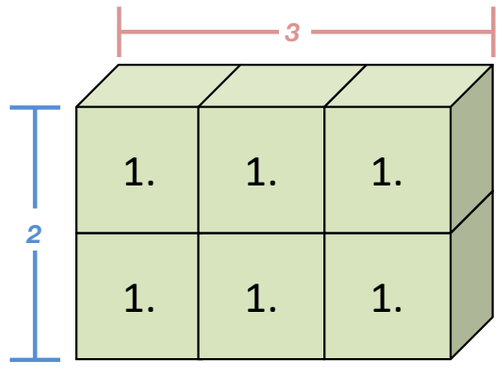
```
np.zeros(3)
```

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

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

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





数组构造函数

NumPy

Identity

numpy.identity(n, dtype=None)

单位矩阵，dtype默认为float64

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

```
print(a)
```

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

```
a.dtype
```

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

```
b = np.identity(4, dtype='int')
```

```
print(b)
```

```
[[1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]
 [0 0 0 1]]
```

```
b.dtype
```

```
dtype('int64')
```

empty, fill

```
np.empty(4)
```

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

```
a = np.arange(6, dtype = 'float').reshape(2, 3)
print(a)
```

```
[[0. 1. 2.]
 [3. 4. 5.]]
```

```
output = np.empty_like(a)
print(output)
```

```
[[0. 1. 2.]
 [3. 4. 5.]]
```

empty做的是申请一部分内存空间

```
a.fill(5.0)  # a[:] = 5.0
print(a)
```

```
[[5. 5. 5.]
 [5. 5. 5.]]
```

```
a.fill(np.nan)
print(a)
```

```
[[nan nan nan]
 [nan nan nan]]
```

nan用来表缺省

数组构造函数

NumPy

linspace

```
a = np.linspace(0,1,5)
```

```
print(a)
```

```
[0.   0.25 0.5  0.75 1.  ]
```

```
a.dtype
```

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

logspace

用来创建等比数列

```
a = np.logspace??
```

```
a = np.logspace(0,2,num=3)
print(a)
```

```
[  1.  10. 100.]
```

```
a.dtype
```

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

从文件中读取数据

TDOS.txt

```
 #Energy          TDOS-UP          TDOS-DOWN
-23.02835         0.00000          0.00000
-23.01135         0.00000          0.00000
-22.99435         0.00000          0.00000
-22.97735         0.00000          0.00000
-22.96035         0.00000          0.00000
......
```

```
np.savetxt??
```

```
data = np.loadtxt('TDOS.txt')
```

```
x, y1, y2 = data[:,0], data[:,1], data[:,2]
```

```
np.savetxt('./energy.txt', x, fmt='%0.6f' )
```

```
data = np.genfromtxt('./TDOS.txt')
```

```
%load pdos.py
```

```
%run pdos.py
```

Numpy 计算规则

- “广播规则”形状匹配
- 运算符和 UFuncs，逐元素运算
- 统计矩默认应用于整个数组
- 数组中存在缺省时，进行错误传递

Numpy — Universal Function

numpy.sum( )

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

```
[[1 2 3]
 [4 5 6]]
```

```
np.sum(a)
```

```
21
```

```
np.sum(a, axis=0)
```

```
array([5, 7, 9])
```

```
np.sum(a, axis=1)
```

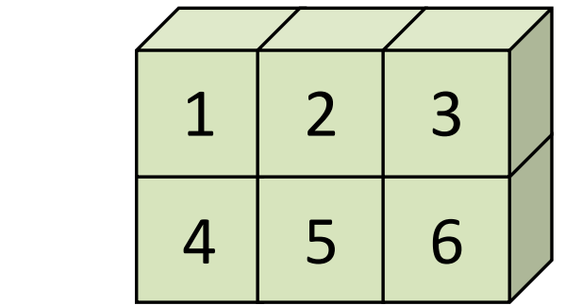
```
array([ 6, 15])
```

```
a.sum(axis=0)
```

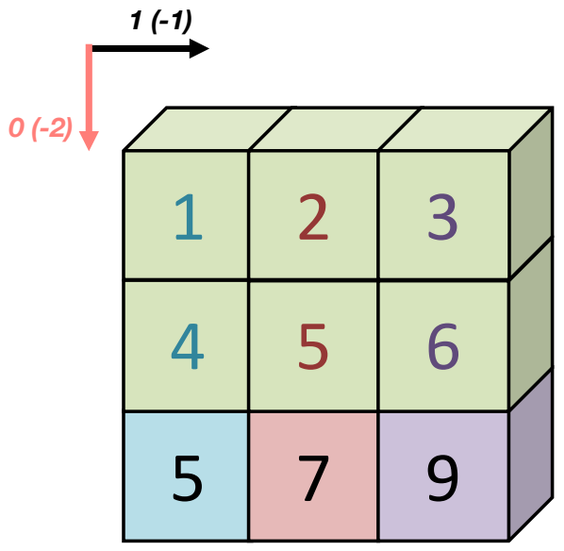
```
array([5, 7, 9])
```

区分函数【np.sum(a)】与方法【a.sum( )】

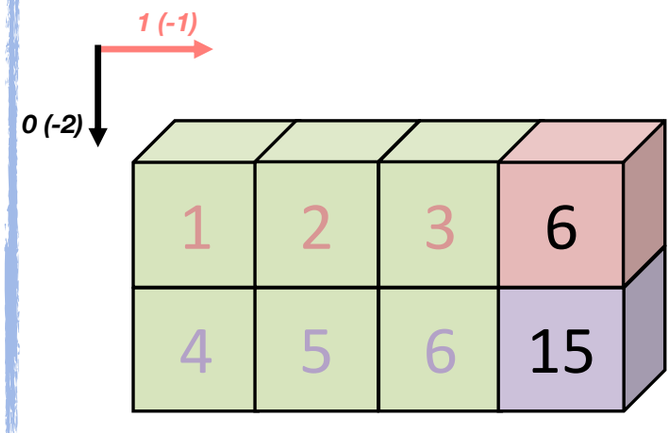
一次只能对一个数组求和



np.sum(a)



np.sum(a, axis=0)



np.sum(a, axis=1)

Numpy — Universal Function

NumPy

numpy.reshape( )

```
a = np.arange(8)
print(a)
```

```
[0 1 2 3 4 5 6 7]
```

```
b = np.reshape(a,(2,4))
print(b)
```

```
[[0 1 2 3]
 [4 5 6 7]]
```

```
b = a.reshape(2,4)
print(b)
```

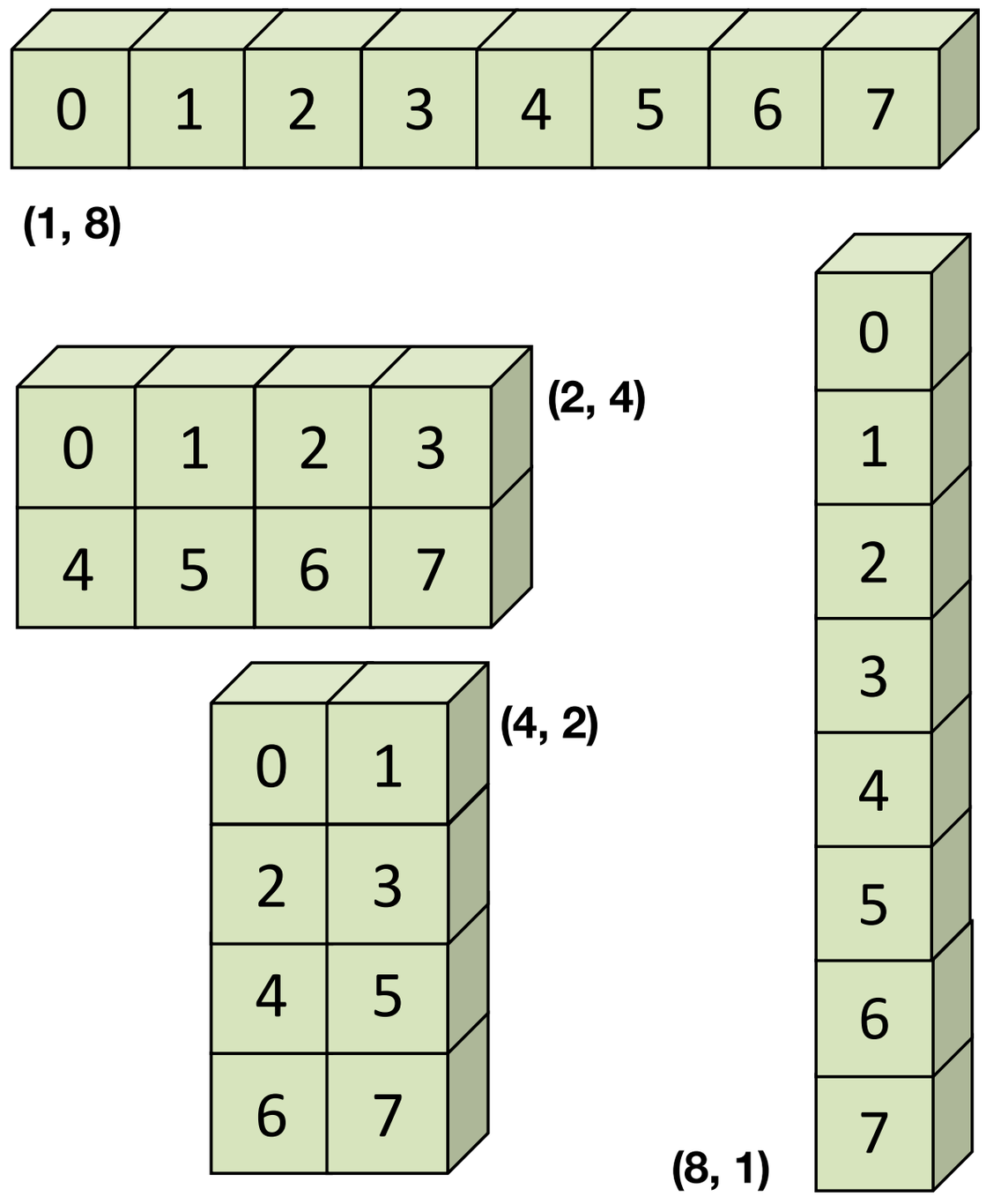
```
[[0 1 2 3]
 [4 5 6 7]]
```

```
c = a.reshape(1,-1)
print(c)
```

```
[[0 1 2 3 4 5 6 7]]
```

```
d = a.reshape(-1,1)
print(d)
```

```
[[0]
 [1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]]
```



Numpy — Universal Function

numpy.ravel( )

```
a = np.arange(8).reshape(2,4)
print(a)
```

```
[[0 1 2 3]
 [4 5 6 7]]
```

```
b = np.ravel(a)
print(b, b.shape)
```

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

```
b[1] = 100
print(b)
```

```
[  0 100   2   3   4   5   6   7]
```

```
print(a)
```

```
[[  0 100   2   3]
 [  4   5   6   7]]
```

numpy.flatten( )

```
a = np.arange(8).reshape(2,4)
print(a)
```

```
[[0 1 2 3]
 [4 5 6 7]]
```

```
b = a.flatten()
print(b, b.shape)
```

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

```
b[1] = 100
print(b)
```

```
[  0 100   2   3   4   5   6   7]
```

```
print(a)
```

```
[[0 1 2 3]
 [4 5 6 7]]
```

两者功能一致，区别在于返回拷贝（copy）还是返回视图（view）

numpy.flatten( )返回一份copy，不影响原始矩阵；

numpy.ravel( )返回view，影响原始矩阵。

Numpy — Universal Function

NumPy

numpy.stack( ), hstack( ), vstack( )

```
a = np.array([1,2,3])
b = np.array([4,5,6])
c = np.array([7,8,9])
print(a, b, c)
```

```
[1 2 3] [4 5 6] [7 8 9]
```

```
d = np.stack((a,b,c), axis=0)
print(d)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
e = np.stack((a,b,c), axis=1)
print(e)
```

```
[[1 4 7]
 [2 5 8]
 [3 6 9]]
```

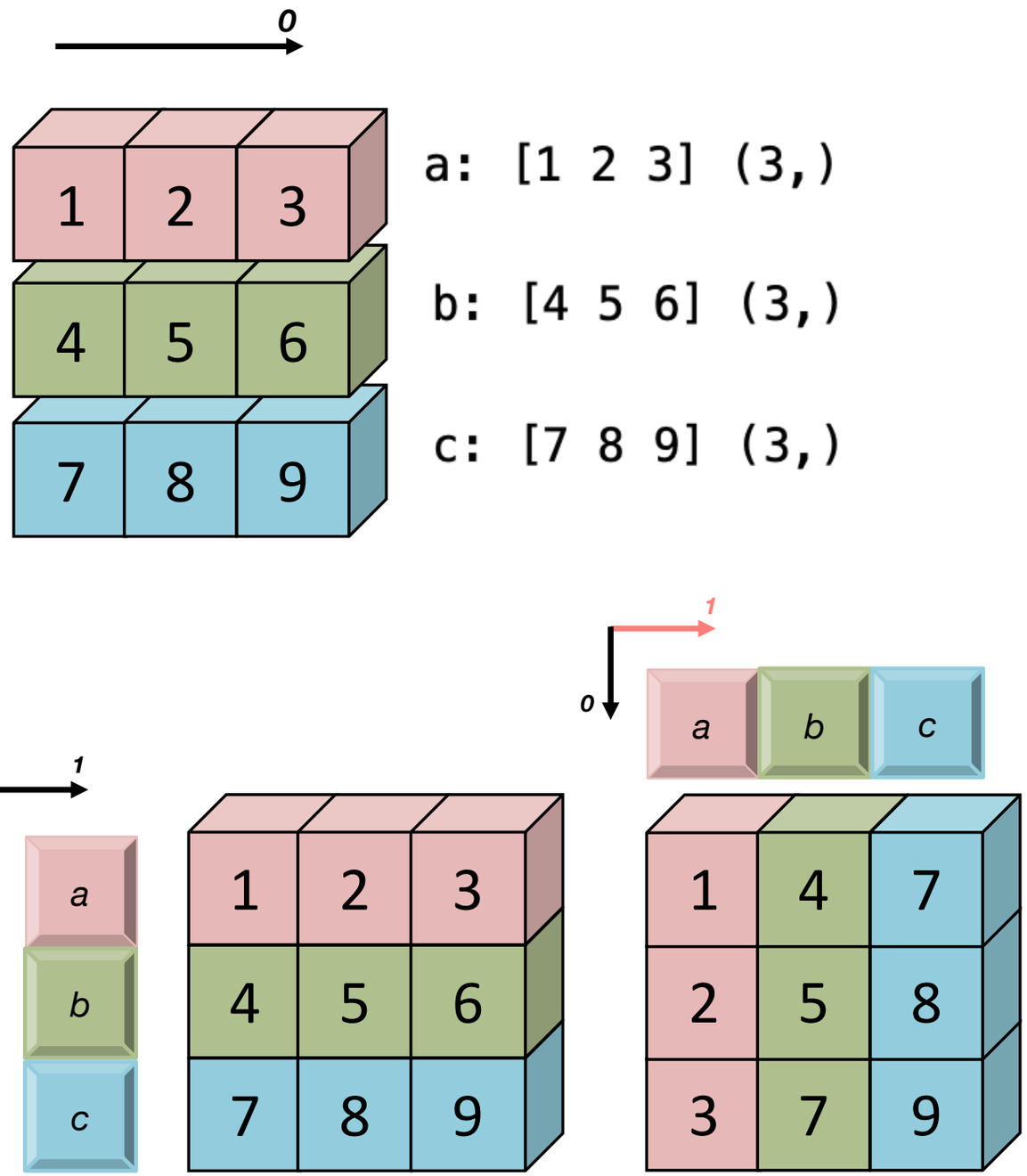
```
f = np.hstack((a,b,c))
print(f)
```

```
[1 2 3 4 5 6 7 8 9]
```

```
g = np.vstack((a,b,c))
print(g)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

* 将一系列的数组沿着一个新的轴进行堆叠



Numpy — Universal Function

numpy.mean( )

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

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
b = np.mean(a)
print(b)
```

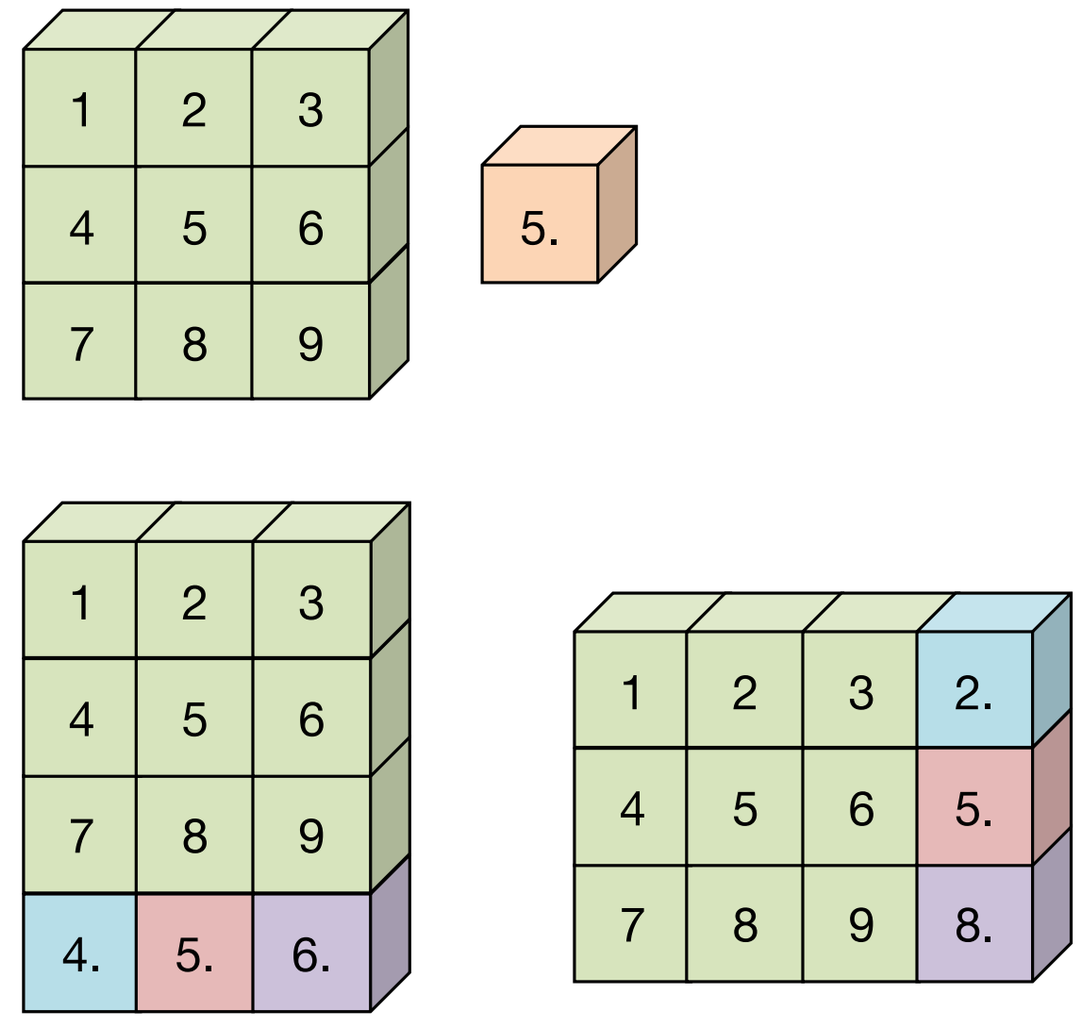
```
5.0
```

```
c = np.mean(a, axis=0)
print(c)
```

```
[4. 5. 6.]
```

```
d = np.mean(a, axis=1)
print(d)
```

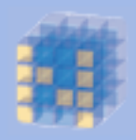
```
[2. 5. 8.]
```



axis=0，输出矩阵1行，求列平均

axis=1，输出矩阵1列，求行平均

Numpy — Universal Function



NumPy

$$s^2 = \frac{(x_1-x)^2+(x_2-x)^2+\ldots\ldots+(x_n-x)^2}{n}$$

$$s = \frac{(x_1-x)^2+(x_2-x)^2+\ldots\ldots+(x_n-x)^2}{n-1}$$

$$\sigma = \frac{(x_1-x)^2+(x_2-x)^2+\ldots\ldots+(x_n-x)^2}{n}$$

numpy.var ( )

```
np.var??
```

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

```
[[1 2 3]
 [4 5 6]]
```

```
a_var = np.var(a)
print(a_var)
```

```
2.9166666666666665
```

numpy.std ( )

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

```
[[1 2 3]
 [4 5 6]]
```

```
# 总体的标准差，维度为N
```

```
a_std = np.std(a)
print(a_std)
```

```
1.707825127659933
```

```
# 样本的标准差，维度为N-1
```

```
a_std_1 = np.std(a, ddof=1)
print(a_std_1)
```

```
1.8708286933869707
```

Array Broadcasting

NumPy

广播(Broadcast)是 numpy 对不同形状(shape)的数组进行数值计算的方式

```
a = np.array([[0,0,0],[10,10,10],[20,20,20],[30,30,30]])
b = np.array([0,1,2])
c = a + b
print(c)
```

```
[[ 0  1  2]
 [10 11 12]
 [20 21 22]
 [30 31 32]]
```

```
a = np.array([[0],[10],[20],[30]])
b = np.array([0,1,2])
c = a + b
print(c)
```

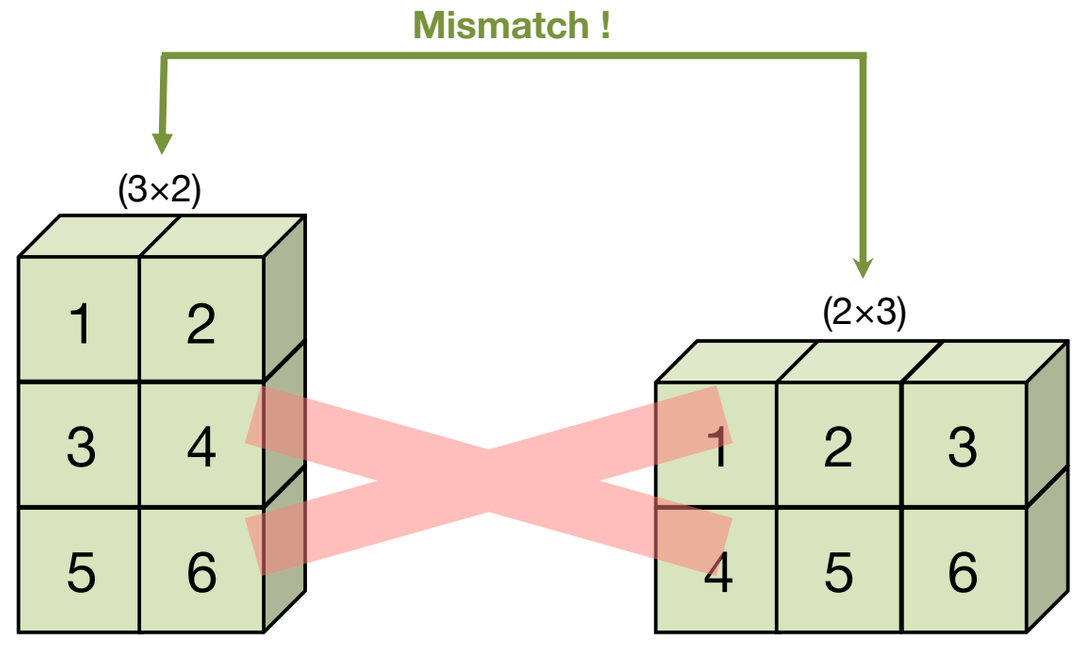
```
[[ 0  1  2]
 [10 11 12]
 [20 21 22]
 [30 31 32]]
```

```
a = np.array([[0],[10],[20],[30]])
b = np.array([0,1,2])

aa = np.tile(a,(1,3))
bb = np.tile(b, (4,1))

c = aa+bb
print(c)
```

```
[[ 0  1  2]
 [10 11 12]
 [20 21 22]
 [30 31 32]]
```

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

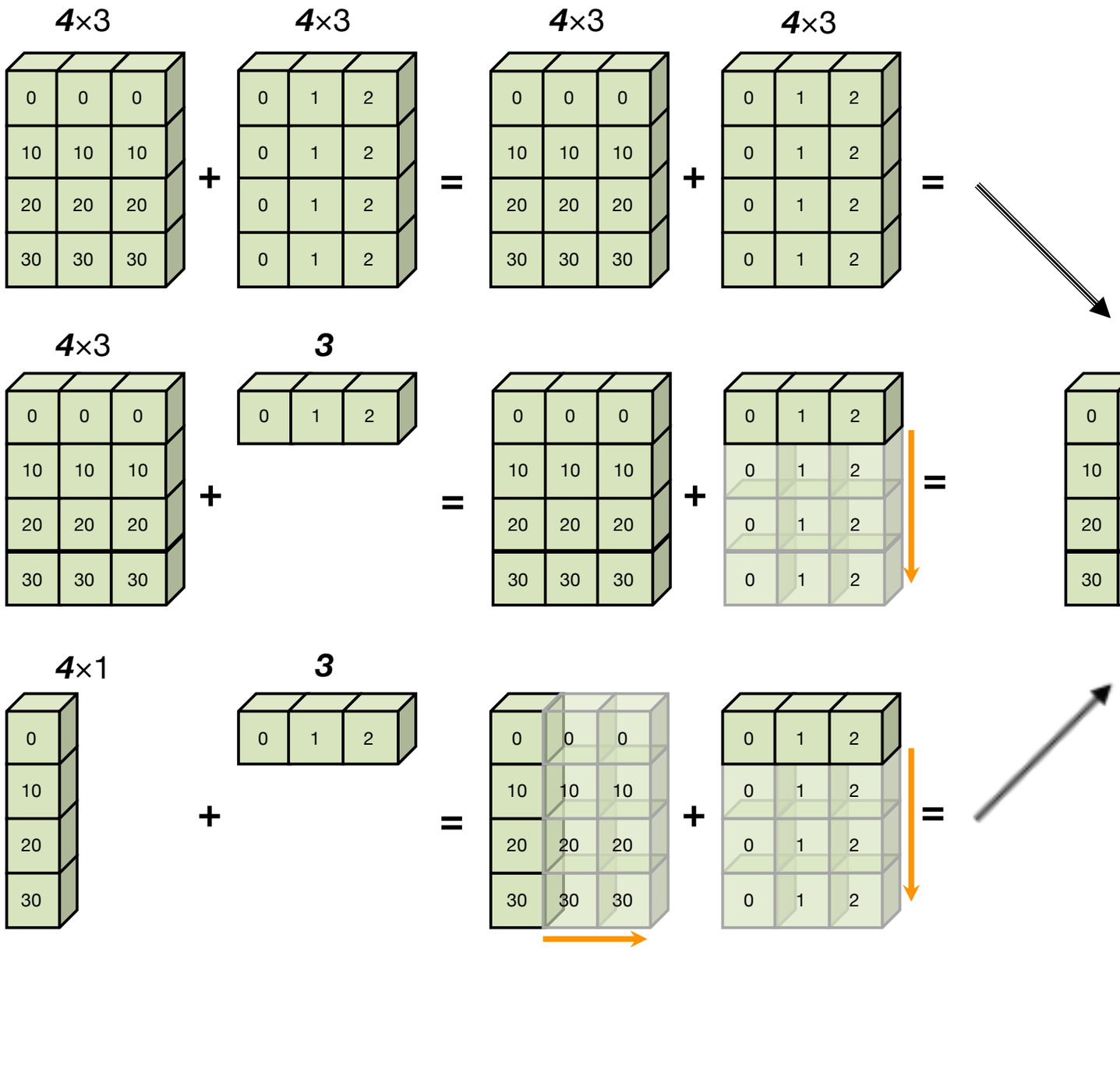
```
---------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-358-b17e002e6945> in <module>
      1 a = np.array([[1, 2], [3, 4], [5, 6]])
      2 b = np.array([[1, 2, 3], [4, 5, 6]])
----> 3 a + b

ValueError: operands could not be broadcast together with shapes (3,2) (2,3)
```



Array Broadcasting

NumPy

广播的规则

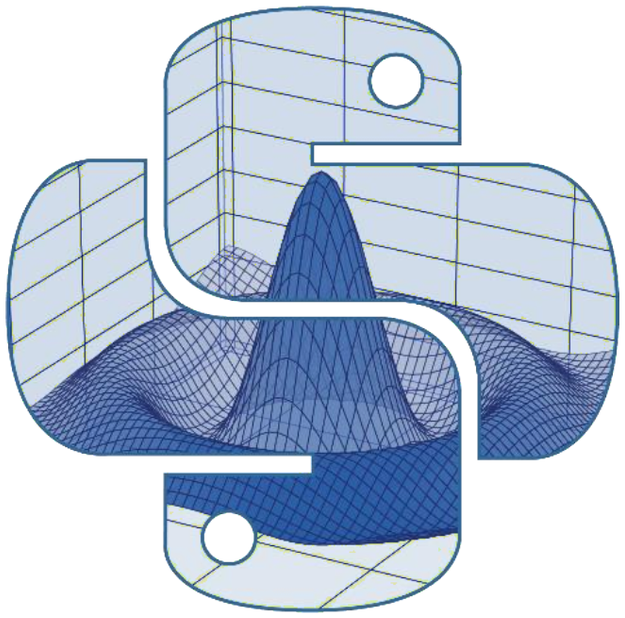


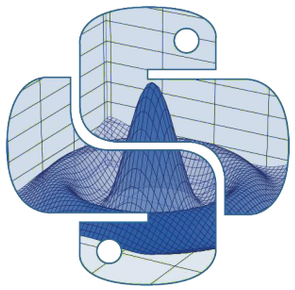
- 两个数组维度数不同,小维度数组的形状满足右侧对齐。
- 两个数组无匹配维度,沿着维度为1的维度扩展以匹配另一数组形状。
- 如果两个数组无匹配维度且无任一维度等于1,引起异常。

应用实例

NumPy

1. pdos文件数据读取
2. np.histogram( ) —pdos
3. np.mean( ) —MD
4. np.std( ) —机器学习误差分布
5. 自定义函数调用





课题组培训—2020

THANK YOU !