数据计算

数据分析的基本流程

分析的问题是什么？
你想得出哪些结论？

问题定义

数据获取

数据预处理

数据分析

数据可视化及报告撰写

网络爬虫

互联网数据
内部数据
外部数据

统计量的描述和展示

数据清洗，集成，变换

探究内部关系，进行分析预测

数据处理及分析常用库

- NumPy：使用Python进行科学计算的基础软件包
- Pandas：基于NumPy的分析结构化数据的工具集，是数据挖掘和数据分析的核心支持库，也提供数据预处理功能

内容

- NumPy
- Pandas

NumPy

Numerical+Python：Python数组计算、矩阵计算和科学计算的核心库

- C语言实现
- 高性能的数组对象
- 用于数组和矩阵操作的大量函数和方法
- 线性代数、傅里叶变换、随机数生成、图形操作等功能
- 可整合C/C++/Fortran代码

numpy.org

NumPy 中文网

www.numpy.org.cn

```
import numpy as np
```

NumPy

- 数组的概念
- 数组的创建
- 数组的基本操作
- 矩阵的基本操作
- 常用统计分析函数

数组的相关概念

Description

NumPy is the fundamental package for array computing with Python.

Version

1.23.3

Author

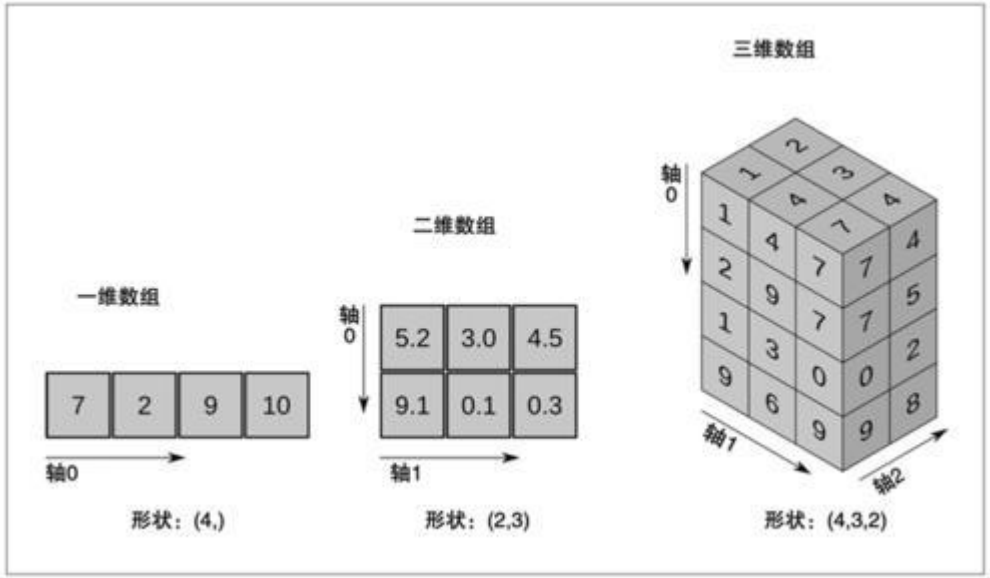
Travis E. Oliphant et al.

https://www.numpy.org

数组的概念：数组分类

数组按维度分类

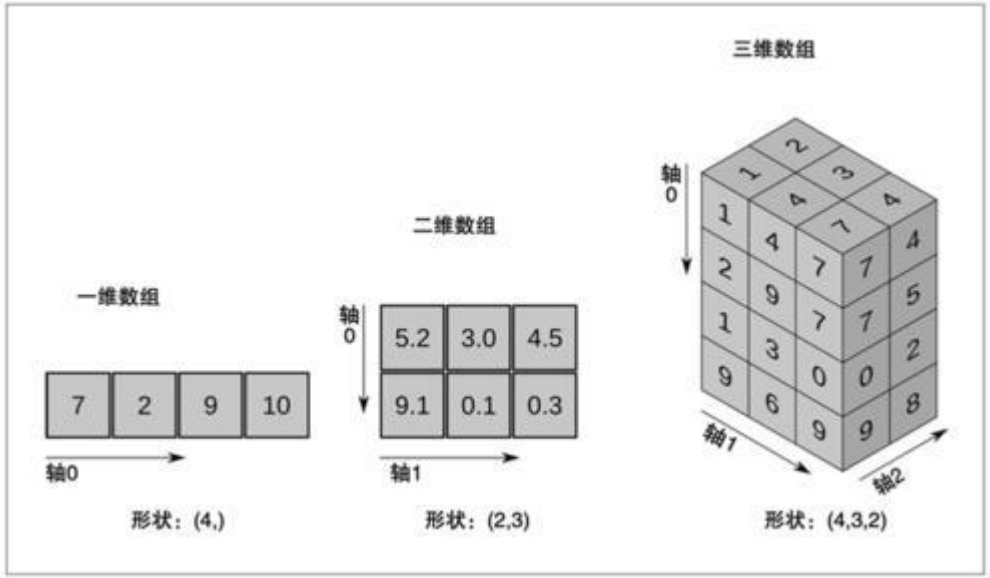
- 一维数组：类似于Python列表
- 二维数组：以一维数组作为数组元素，包括行和列，类似于表格形式，又称矩阵
- 多维数组：维数大于等于三的数组结构。三维数组是最常见的多维数组



数组的概念：数组的轴

数组的轴（axis）

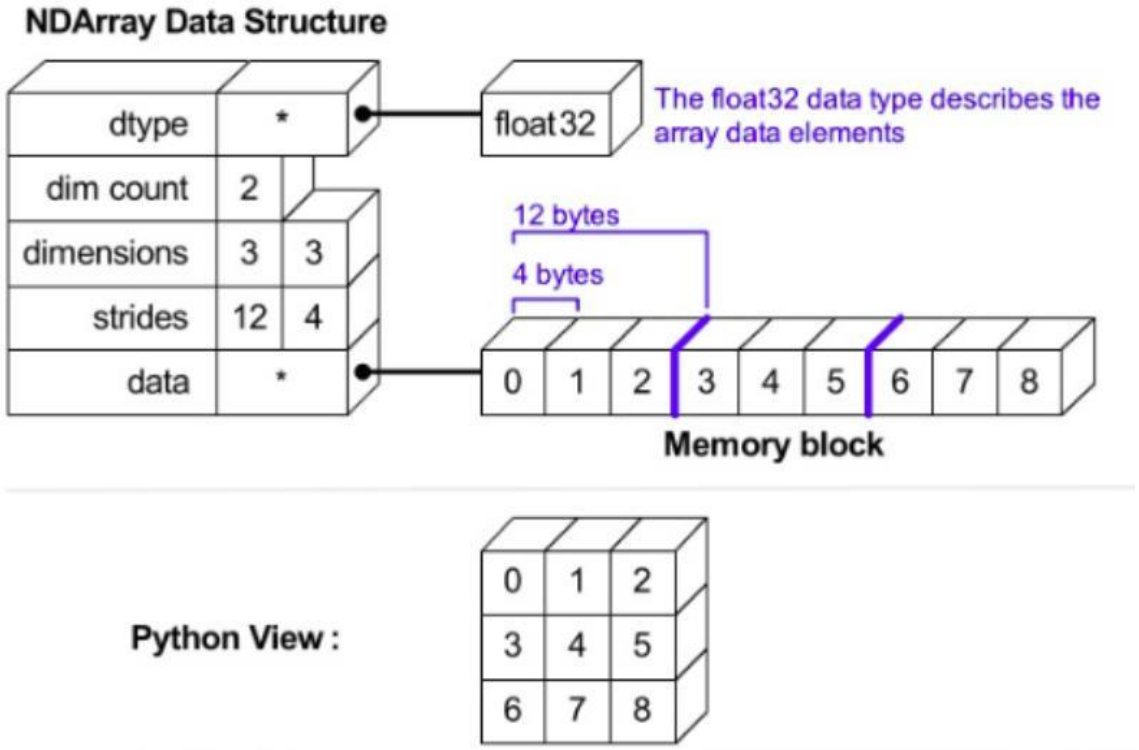
- 数组的”维”称为轴（axis）
- 指定某个axis，就是沿着这个axis做相关操作



In Numpy dimensions are called *axes*. The number of axes is *rank*.

NumPy中的数组：ndarray

- 用于存放同类型元素的多维数组
- 每个元素在内存中都有相同大小的存储区域
- 以0为开始下标进行元素的索引



数组元素的数据类型

数据类型	描述
bool_	存储一个字节的布尔值（True，False）
int_	默认整数，相对于C的long，通常为int32
intc	相当于C的int，通常为int32
intp	用于索引的整数，相当于C的size_t，通常为int64
int8	字节（-128~127）
int16	16位整数（-32768~32767）
int32	32位整数（-2147483648~2147483647）
int64	64位整数（-9223372036854775808~9223372036854775807）
uint8	8位无符号整数（0~255）
uint16	16位无符号整数（0~65535）

数组元素的数据类型

数据类型	描述
uint32	32位无符号整数（0~4294967295）
uint64	64位无符号整数（0~18446744073709551615）
float_	float64的简写
float16	半精度浮点数，包括：1 个符号位，5 个指数位，10 个尾数位
float32	单精度浮点数，包括：1 个符号位，8 个指数位，23 个尾数位
float64	双精度浮点数，包括：1 个符号位，11 个指数位，52 个尾数位
complex_	complex128 类型的简写，即 128 位复数
complex64	复数，由两个 32 位浮点数表示（实数部分和虚数部分）
complex128	复数，由两个 64 位浮点数表示（实数部分和虚数部分）
datatime64	日期时间类型
timedelta64	两个时间之间的间隔

数据转换函数

每一种数据类型都有相应的数据转换函数

```
>>> import numpy as np
>>> np.int8(3.1415)
3
>>> np.float64(8)
8.0
>>> np.float16(True)
1.0
>>> np.bool_(3)
True
```

数组的常用属性

- a = np.array([0, 1, 2, 3, 4])
- b = np.array([(1.0, 2.0, 3.0), (4.0, 5.0, 6.0)])
- c = np.array([[1, 2, 3], [4, 5, 6]], dtype = complex)

属性名	描述	数组a	数组b	数组c
ndim	维数，等于秩	1	2	2
shape	每个维度上的长度	(5,)	(2, 3)	(2, 3)
size	数组元素的总个数	5	6	6
dtype	表示数组中元素类型的对象	int32	float64	complex128
itemsize	数组中每个元素的大小(以字节为单位)	4	8	16

数组的属性

```
>>> a = np.array([0,1,2,3,4])
```

```
a = {ndarray: (5,)} [0 1 2 3 4] ...View as Array
  min = {int32: ()} 0
  max = {int32: ()} 4
  shape = {tuple: 1} (5,)
  dtype = {dtype[int32]: ()} int32
  size = {int} 5
```

NumPy

- 数组的概念
- 数组的创建
- 数组的基本操作
- 矩阵的基本操作
- 常用统计分析函数

创建数组

创建数组有六种常规机制

1.从其它Python结构（例如，列表，元组）转换

2.numpy原生数组的创建（例如，arange、ones、zeros等）

3.复制、连接或更改已有数组

4.从磁盘读取数组，无论是标准格式还是自定义格式

5.通过使用字符串或缓冲区从原始字节创建数组

6.使用特殊库函数（例如，random）

创建数组：由其它结构创建

numpy.array(object, dtype=None, copy=True, order= ‘K’ ,
subok=False, ndmin=0, like=None)

- object：任何具有数组接口方法的对象
- dtype：数据类型
- copy: 若为True，则复制object对象；否则，只有当满足某些要求时，才会进行复制
- order：指定数组元素的内存布局，值可以为K、A、C、F
- subok：若为True，则传递子类；否则返回的数组将强制为基类数组
- ndmin：指定生成数组的最小维数
- like: 允许创建非NumPy数组的数组

创建数组：由其它结构创建

```
>>> n1 = np.array([1,2,3])
```

> n1 = {ndarray: (3,)} [1 2 3] ...View as Array

```
>>> n2 = np.array([[1,2.0], [0,0], (1+3j,3.)])
```

> n2 = {ndarray: (3, 2)} [[1.+0.j 2.+0.j], [0.+0.j 0.+0.j], [1.+3.j 3.+0.j]] ...View as Array

创建数组：由其它结构创建

```
>>> list1 = [1, 2, 3]
>>> n3 = np.array(list1, dtype = float_)
```

n3 = {ndarray: (3,)} [1. 2. 3.] ...View as Array

```
>>> n4 = np.array(list1, ndmin = 3)
```

n4 = {ndarray: (1, 1, 3)} [[[1 2 3]]] ...View as Array

```
>>> n4[0]
array([[1, 2, 3]])
>>> n4[0][0]
array([1, 2, 3])
>>> n4[0][0][0]
1
```

创建数组：原生数组创建

创建指定形状、数据类型且未初始化的数组

numpy.empty(shape, dtype=float, order= ‘C’ , like=None)

- order：元素在内存中存储的顺序，C：行优先；F：列优先

```
>>> n7 = np.empty([2,3],dtype=int)
```

	0	1	2
0	0	1072693248	0
1	1073741824	0	1074266112

创建数组：原生数组创建

创建指定维度、数据类型，以0填充的数组

numpy.zeros(shape, dtype=float, order= ‘C’ , like = None)

```
>>> n8 = np.zeros((3,3))
```

> n8 = {ndarray: (3, 3)} [[0. 0. 0.], [0. 0. 0.], [0. 0. 0.]]

	0	1	2
0	0.00000	0.00000	0.00000
1	0.00000	0.00000	0.00000
2	0.00000	0.00000	0.00000

创建数组：原生数组创建

创建指定维度、数据类型，以1填充的数组

numpy.ones(shape, dtype=float, order= ‘C’ , like=None)

```
>>> n9 = np.ones((4,2),dtype=int)
```

```
n9 = {ndarray: (4, 2)} [[1 1], [1 1], [1 1], [1 1]]
```

创建数组：原生数组创建

创建指定维度、数据类型，以指定值填充的数组

numpy.full(shape, fill_value, dtype=None, order= ‘C’ *, like=None)

```
>>> n10 = np.full((3,3),8)
```

```
> n10 = {ndarray: (3, 3)} [[8 8 8], [8 8 8], [8 8 8]]
```

创建数组：原生数组创建

通过数据范围创建数组

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

```
>>> n11 = np.arange(2, 3, 0.2)
```

n11 = {ndarray: (5,)} [2. 2.2 2.4 2.6 2.8]

创建数组：原生数组创建

创建等差数列

numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0)

- start：序列的起始值
- stop：序列的终止值
- num：要生成的等步长的样本数量，默认值为50
- endpoint：若为True，数列中包含stop参数的值；否则不包括
- retstep：若为True，则生成的数组中会显示间距，否则不显示

创建数组：原生数组创建

创建等差数列

```
>>> n12 = np.linspace(1,4,6) >>> n13 = np.linspace(1,4,6,endpoint=False)
```

n12 = {ndarray: (6,)} [1. 1.6 2.2 2.8 3.4 4.] n13 = {ndarray: (6,)} [1. 1.5 2. 2.5 3. 3.5]

创建数组：原生数组创建

创建等差数列

```
>>> n12 = np.linspace(1,4,6,retstep=True)
>>> n12
(array([1. , 1.6, 2.2, 2.8, 3.4, 4. ]), 0.6)
```

创建数组：原生数组创建

创建等比数列

numpy.logspace(start, stop, num=50, endpoint=True, base=10.0, dtype=None, axis=0)

- start：序列的起始值为base**start
- stop：序列的终止值为base**stop
- num：要生成的等比的样本数量，默认值为50
- endpoint：若为True，数列中包含stop参数的值；否则不包括
- base：序列范围的基数，默认值为10

创建数组：原生数组创建

创建等比数列

```
>>> n14 = np.logspace(0,9,10,base=2,dtype='int')
```

```
> n14 = {ndarray: (10,)} [  1   2   4   8  16  32  64 128 256 512]
```

创建数组：使用random模块生成随机数组

生成0~1之间的随机数组

numpy.random.rand(d0, d1, d2, ...dn)

- d0、d1~dn：表示数组的shape，可以为空

```
>>> n15 = np.random.rand(5)

>>> print(n15)
[0.02790447 0.43099083 0.76816699 0.41171074 0.18020472]
```

创建数组：使用random模块生成随机数组

生成一定范围内的随机整数数组

numpy.random.randint (low, high=None, size=None)

- low：低值（起始值），整数
- high: 高值（终止值），整数
- size：数组shape，整数或元组，表示一维或多维数组。默认为空，如果为空，则仅返回一个整数

创建数组：使用random模块生成随机数组

生成一定范围内的随机整数数组

```
>>> n19 = np.random.randint(1,3,10)
>>> print(n19)
[1 2 2 1 2 2 1 1 1 1]
>>> n20 = np.random.randint(5,10)
>>> print(n20)
9
```

创建数组：使用random模块生成随机数组

生成满足标准正态分布的随机数组

numpy.random.randn(d0, d1, d2, ...dn)

- d0、d1~dn：表示数组的shape，可以为空

```
>>> n17 = np.random.randn(5)
>>> print(n17)
[-0.42837384 -1.91183818  1.62630172 -2.19883829  1.38377623]
```

创建数组：使用random模块生成随机数组

生成满足正态分布的随机数组

numpy.random.normal(loc, scale, size)

- loc：正态分布的均值
- scale：正态分布的标准差
- size: 数组的shape

创建数组：使用random模块生成随机数组

生成满足正态分布的随机数组

```
>>> n22 = np.random.normal(0,0.1,10)
>>> print(n22)
[ 0.07774851 -0.25509784  0.17550369 -0.02185514 -0.05450839  0.02559789
 -0.17311362  0.1642502   0.1486492  -0.11539874]

>>> n23 = np.random.normal(1,0.3,(3,3))
>>> print(n23)
[[0.83502214 0.66895048 0.93064742]
 [0.74280158 0.53162327 1.02717876]
 [1.00990879 1.14287733 1.06037926]]
```

NumPy

- 数组的概念
- 数组的创建
- 数组的基本操作
- 矩阵的基本操作
- 常用统计分析函数

数组的基本操作：数组运算

NumPy数组不需要编写循环即可对数组数据执行批量运算，称为“矢量化”

- 算术运算和比较运算
- 广播运算
- 函数运算

数组运算：算术运算和比较运算

```
>>> n1 = np.array([[1, 2, 3], [4, 5, 6]])
>>> n2 = np.array([[10, 15, 20], [30, 40, 50]])
>>> print(n1 + n2)
[[11 17 23]
 [34 45 56]]
```

数组运算：算术运算和比较运算

```
>>> n1 = np.array([1, 2])
>>> n2 = np.array([3, 4])
>>> print(n1 - n2)
[-2 -2]
>>> print(n1 * n2)
[3 8]
>>> print(n1 / n2)
[0.33333333 0.5       ]
>>> print(n1 ** n2)
[ 1 16]
```

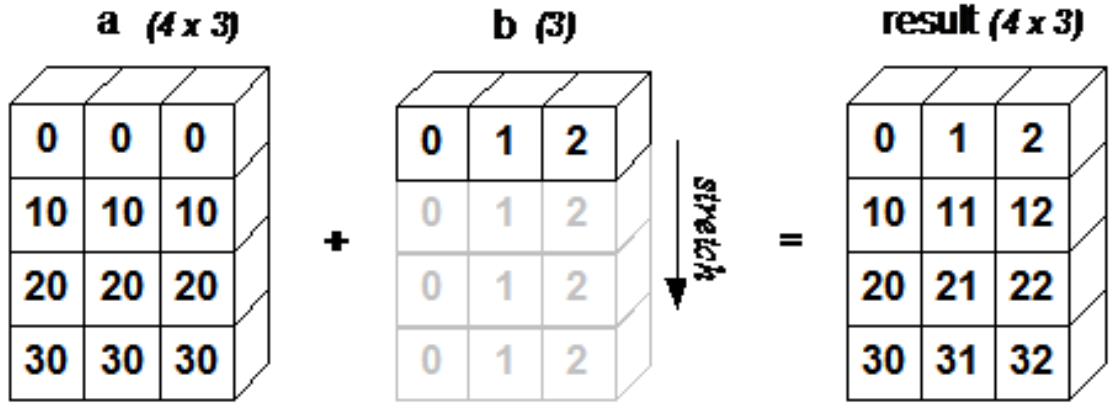
```
>>> print(n1 > n2)
[False False]
>>> print(n1 ⩽ n2)
[ True  True]
>>> print(n1 ≠ n2)
[ True  True]
>>> print(n1 == n2)
[False False]
```

数组运算：广播（Broadcast）

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

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

```
>>> print(a + b)
[[ 0  1  2]
 [10 11 12]
 [20 21 22]
 [30 31 32]]
```



数组运算：广播（Broadcast）

- 如果两个数组的后缘维度（从末尾开始算起的维度）的轴长度相等或其中一方的长度为1，则认为它们是广播兼容的
- 广播在缺失的维度和（或）轴长度为1的维度上进行

数组运算：函数运算

函数	描述
add()、subtract()、multiply()、divide（）	数组的加减乘除运算
abs()	数组中各元素取绝对值
sqrt()	数组中各元素的平方根
square()	数组中各元素的平方
log()、log10()、log2()	数组中各元素的自然对数和分别以10、2为底的对数
reciprocal()	数组中各元素的倒数
power()	第一个数组中的元素为底数，第二个数组中的元素为幂，进行幂运算
mod()	数组之间相应元素相除后的余数

数组运算：函数运算

函数	描述
around()	数组中各元素指定小数位的四舍五入数
ceil()、floor()	数组中各元素向上取整和向下取整
sin()、cos()、tan()	数组中各元素(角度)的正弦值、余弦值和正切值
modf()	数组中各元素的小数和整数部分分割为两个独立数组
exp()	数组中各元素的指数值
maximum()、fmax()	计算数组元素的最大值
minimum()、fmin()	计算数组元素的最小值
copysign()	将第二个数组中各个元素的符号赋给第一个数组中对应的元素

数组运算：函数运算

```
>>> n4 = np.random.randint(-2, 8 ,5)
>>> print(n4)
[ 4  3  3  0 -2]

>>> print(np.abs(n4))
[4 3 3 0 2]
>>> print(np.sqrt(n4))
<input>:1: RuntimeWarning: invalid value encountered in sqrt
[2.         1.73205081 1.73205081 0.                   nan]
>>> print(np.exp(n4))
[54.59815003 20.08553692 20.08553692  1.          0.13533528]
```

数组运算：函数运算

```
>>> n5 = np.array([[1,2,3], [4,5,6]])
>>> n6 = np.array([[7,8,9], [1,2,3]])
```

```
>>> print(np.add(n5,n6))
[[ 8 10 12]
 [ 5  7  9]]
>>> print(np.subtract(n5,n6))
[[-6 -6 -6]
 [ 3  3  3]]
>>> print(np.multiply(n5,n6))
[[ 7 16 27]
 [ 4 10 18]]
```

```
>>> print(np.divide(n5,n6))
[[0.14285714 0.25       0.33333333]
 [4.         2.5        2.        ]]
>>> print(np.power(n5,n6))
[[    1   256 19683]
 [    4    25   216]]
>>> print(np.mod(n5,n6))
[[1 2 3]
 [0 1 0]]
```

数组的基本操作：索引和切片

NumPy数组元素一般通过索引和切片访问和修改

- NumPy数组使用标准Python语法array[index]对数组进行索引
- NumPy数组的切片与Python列表的切片操作类似

  arr[start: stop: step]

数组的基本操作：索引和切片

```
>>> n8 = np.arange(1,10)
>>> print(n8)
[1 2 3 4 5 6 7 8 9]
```

```
>>> print(n8[4])
5
```

```
>>> print(n8[2:7:2])
[3 5 7]
```

```
>>> print(n8[3::4])
[4 8]
```

```
>>> print(n8[3:])
[4 5 6 7 8 9]
>>> print(n8[:5])
[1 2 3 4 5]
>>> print(n8[::3])
[1 4 7]
```

```
>>> print(n8[::-1])
[9 8 7 6 5 4 3 2 1]
>>> print(n8[:-3:-1])
[9 8]
>>> print(n8[-5::-2])
[5 3 1]
```

数组的基本操作：索引和切片

```
>>> n9 = np.array([[1,2,3],[4,5,6],[7,8,9]])
>>> print(n9)
[[1 2 3]
 [4 5 6]
 [7 8 9]]

>>> print(n9[1])
[4 5 6]
>>> print(n9[2,1])
8
>>> print(n9[-1])
[7 8 9]
```

```
>>> print(n9[:2,1:])
[[2 3]
 [5 6]]

>>> print(n9[1,:2])
[4 5]

>>> print(n9[:2,2])
 [3 6]
```

```
>>> print(n9[:,:1])
 [[1]
  [4]
  [7]]

>>> print(n9[:,0])
 [1 4 7]
```

数组的基本操作：索引和切片

```
>>> n8 = np.arange(1,10)
>>> print(n8)
[1 2 3 4 5 6 7 8 9]
```

```
>>> n9 = n8[2:7:2]
>>> print(n9)
[3 5 7]
```

```
>>> n9[0] = 0
>>> print(n8)
[1 2 0 4 5 6 7 8 9]
```

```
>>> n9 = [-1, -2, -3]
>>> print(n8)
[1 2 0 4 5 6 7 8 9]
```

```
>>> print(n9)
[-1, -2, -3]
```

数组的基本操作：布尔索引

```
>>> n9 = np.array([[1,2,3],[4,5,6],[7,8,9]])
>>> print(n9)
[[1 2 3]
 [4 5 6]
 [7 8 9]]

>>> mask = np.array([[True,False,False],[True,False,True],[False,False,True]])
>>> print(n9[mask])
[1 4 6 9]

 >>> print(n9[n9>5])
 [6 7 8 9]
```

数组的基本操作：花式索引

```
>>> n1
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19],
       [20, 21, 22, 23],
       [24, 25, 26, 27],
       [28, 29, 30, 31]])
```

```
>>> n1[[4,2,6]]
array([[16, 17, 18, 19],
       [ 8,  9, 10, 11],
       [24, 25, 26, 27]])

>>> n1[[-4,-2,-6]]
array([[16, 17, 18, 19],
       [24, 25, 26, 27],
       [ 8,  9, 10, 11]])

>>> n1[[1,5,7,2],[0,3,2,1]]
array([ 4, 23, 30,  9])
```

数组的基本操作：数组重塑

数组重塑指更改数组的形状（shape），即调整维度。数组重塑不改变数组中元素的数量和类型

- 通用重塑方法
- 数组展平
- 数组转置

数组重塑：通用重塑方法

arr.reshape(shape)

```
>>> n1 = np.arange(6)
>>> print(n1)
[0 1 2 3 4 5]

>>> n2 = n1.reshape(2,3)
>>> n3 = n1.reshape(3,2)
>>> n4 = n1.reshape(6,1)
```

```
>>> print(n2)
[[0 1 2]
 [3 4 5]]
>>> print(n3)
[[0 1]
 [2 3]
 [4 5]]
```

```
>>> print(n4)
[[0]
 [1]
 [2]
 [3]
 [4]
 [5]]
```

数组重塑：通用重塑方法

arr.reshape(shape)

```
>>> n1 = np.arange(6)
>>> print(n1)
[0 1 2 3 4 5]
```

```
>>> n2 = n1.reshape(2,3)
>>> n3 = n1.reshape(3,2)
>>> n4 = n1.reshape(6,1)
```

```
>>> n1[2] = 10
>>> print(n1)
[ 0  1 10  3  4  5]
```

```
>>> print(n2)
[[ 0  1 10]
 [ 3  4  5]]
>>> print(n3)
[[ 0  1]
 [10  3]
 [ 4  5]]
```

```
>>> print(n4)
[[ 0]
 [ 1]
 [10]
 [ 3]
 [ 4]
 [ 5]]
```

数组重塑：数组展平

arr.flatten()

```
>>> n4= np.array([[n+m*10 for n in range(3)]for m in range(2)])
>>> print(n4)
[[ 0  1  2]
 [10 11 12]]
>>> n5 = n4.flatten()
>>> print(n5)
[ 0  1  2 10 11 12]
```

数组重塑：数组转置

arr.T

```
>>> n6 = np.arange(24).reshape(4,6)
>>> print(n6)
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]
 [18 19 20 21 22 23]]
```

```
>>> print(n6.T)
[[ 0  6 12 18]
 [ 1  7 13 19]
 [ 2  8 14 20]
 [ 3  9 15 21]
 [ 4 10 16 22]
 [ 5 11 17 23]]
```

数组重塑：数组转置

arr.transpose()

```
>>> n7 = np.array([['A',100],['B',200],['C',300],['D',400],['E',500]])
 >>> n8 = n7.transpose()
```

数组重塑：数组转置

n7

	0	1
0	A	100
1	B	200
2	C	300
3	D	400
4	E	500

n8

	0	1	2	3	4
0	A	B	C	D	E
1	100	200	300	400	500

数组的基本操作：数组元素增加

numpy.hstack((arr1, arr2,... arrn))

```
>>> n8 = np.arange(6).reshape(2,3)
>>> n9 = np.arange(10,16).reshape(2,3)
>>> print(n8)
[[0 1 2]
 [3 4 5]]
>>> print(n9)
[[10 11 12]
 [13 14 15]]
```

```
>>> n10 = np.hstack((n8,n9))
>>> print(n10)
[[ 0  1  2 10 11 12]
 [ 3  4  5 13 14 15]]
```

```
>>> n11 = np.hstack((n8,n9,n10))
>>> print(n11)
[[ 0  1  2 10 11 12  0  1  2 10 11 12]
 [ 3  4  5 13 14 15  3  4  5 13 14 15]]
```

数组的基本操作：数组元素增加

numpy.vstack((arr1, arr2,... arrn))

```
>>> n8 = np.arange(6).reshape(2,3)
>>> n9 = np.arange(10,16).reshape(2,3)
>>> print(n8)
[[0 1 2]
 [3 4 5]]
>>> print(n9)
[[10 11 12]
 [13 14 15]]
```

```
>>> n12 = np.vstack((n8,n9))
>>> print(n12)
[[ 0  1  2]
 [ 3  4  5]
 [10 11 12]
 [13 14 15]]
```

数组的基本操作：数组元素删除

numpy.delete(arr, obj, axis=None))

```
>>> n13 = np.arange(16).reshape(4,4)
>>> print(n13)
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
```

```
>>> n14 = np.delete(n13, 2, axis=0)
>>> print(n14)
[[ 0  1  2  3]
 [ 4  5  6  7]
 [12 13 14 15]]
```

```
>>> n15 = np.delete(n13, 2, axis=1)
>>> print(n15)
[[ 0  1  3]
 [ 4  5  7]
 [ 8  9 11]
 [12 13 15]]
```

数组的基本操作：数组元素删除

numpy.delete(arr, obj, axis=None))

```
>>> n13 = np.arange(16).reshape(4,4)
>>> print(n13)
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
```

```
>>> n16 = np.delete(n13, (1, 3), axis=0)
>>> print(n16)
[[ 0  1  2  3]
 [ 8  9 10 11]]
```

```
>>> n17 = np.delete(n13, (1, 3))
>>> print(n17)
[ 0  2  4  5  6  7  8  9 10 11 12 13 14 15]
```

数组的基本操作：数组元素修改

```
>>> n8 = np.arange(6).reshape((2,3))

>>> print(n8)
[[0 1 2]
 [3 4 5]]

>>> n8[1]=[10,20,30]
>>> print(n8)
[[ 0  1  2]
 [10 20 30]]
```

```
>>> n8[0][0],n8[1][0] = 5,15
>>> print(n8)
[[ 5  1  2]
 [15 20 30]]
```

数组的基本操作：数组条件查询

numpy.where(condition, x, y)

```
>>> n15 = np.arange(9).reshape(3,3)
>>> print(n15)
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

```
>>> print(np.where(n15>5,1,-1))
[[-1 -1 -1]
 [-1 -1 -1]
 [ 1  1  1]]
```

NumPy

- 数组的概念
- 数组的创建
- 数组的基本操作
- 矩阵的基本操作
- 常用统计分析函数

NumPy中的数组（Array）和矩阵（Matrix）

- 矩阵即二维数组，是数组的分支
- Array和Matrix是NumPy中的两种不同的数据类型
- 在Array上可以执行的运算及函数，一般都可用于Matrix
- 在Array和Matrix上执行相同的数学运算时，可能得到不同的结果

矩阵创建

numpy.mat(data, dtype=None)

```
>>> m1 = np.mat([[1,2],[3,4]])
```

```
>>> print(m1)
[[1 2]
 [3 4]]
```

```
>>> m1
matrix([[1, 2],
        [3, 4]])
```

矩阵创建

numpy.mat(data, dtype=None)

```
>>> m1 = np.mat('1 2 3;4 5 6')
>>> print(m1)
[[1 2 3]
 [4 5 6]]
```

矩阵创建

numpy.mat(data, dtype=None)

```
>>> m2 = np.mat(np.zeros((3,3)))
>>> print(m2)
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

```
>>> m3 = np.mat(np.random.randint(1,8,size=(3,5)))
>>> print(m3)
[[1 7 7 4 2]
 [7 7 3 5 5]
 [2 7 6 1 3]]
```

矩阵运算：+，-，/

```
>>> m4 = np.mat(np.arange(6).reshape(2,3))
>>> m5 = np.mat(np.arange(10,16).reshape(2,3))
>>> print(m4)
[[0 1 2]
 [3 4 5]]
>>> print(m5)
[[10 11 12]
 [13 14 15]]
```

```
>>> print(m4+m5)
[[10 12 14]
 [16 18 20]]
```

```
>>> print(m4-m5)
[[-10 -10 -10]
 [-10 -10 -10]]
```

```
>>> print(m4/m5)
[[0.         0.09090909 0.16666667]
 [0.23076923 0.28571429 0.33333333]]
```

矩阵运算：矩阵相乘

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1s} \\ \cdots & \cdots & \cdots & \cdots \\ a_{i1} & a_{i2} & \cdots & a_{is} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{ms} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & \cdots & b_{1j} & \cdots & b_{1n} \\ b_{21} & \cdots & b_{2j} & \cdots & b_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ b_{s1} & \cdots & b_{sj} & \cdots & b_{sn} \end{bmatrix}$$

$$== \begin{bmatrix} c_{11} & \cdots & a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots\cdots + a_{is}b_{sj} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{i1} & & c_{ij} & & c_{in} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{m1} & \cdots & c_{mj} & \cdots & c_{mn} \end{bmatrix}$$

矩阵相乘使用点乘（点积），即对应位置——相乘后求和

矩阵运算：矩阵相乘

```
>>> print(m4)
[[0 1 2]
 [3 4 5]]
>>> print(m5)
[[10 11 12]
 [13 14 15]]
```

```
>>> m5 = m5.T
>>> print(m5)
[[10 13]
 [11 14]
 [12 15]]
```

```
>>> print(m4*m5)
[[ 35  44]
 [134 170]]
```

NumPy数组的点乘

numpy.dot(arr1, arr2)

```
>>> n1 = np.array([[1,2],[3,4]])
>>> n2 = np.array([[5,6],[7,8]])
```

```
>>> print(n1)
[[1 2]
 [3 4]]
>>> print(n2)
[[5 6]
 [7 8]]
```

```
>>> print(n1*n2)
[[ 5 12]
 [21 32]]
```

```
>>> print(np.dot(n1,n2))
[[19 22]
 [43 50]]
```

矩阵求逆

matrix.I

```
>>> m6 = np.mat('1 3 3;4 5 6;7 12 9')
>>> print(m6)
[[ 1  3  3]
 [ 4  5  6]
 [ 7 12  9]]
```

```
>>> print(m6.I)
[[-0.9         0.3         0.1       ]
 [ 0.2        -0.4         0.2       ]
 [ 0.43333333  0.3        -0.23333333]]
```

NumPy

- 数组的概念
- 数组的创建
- 数组的基本操作
- 矩阵的基本操作
- 常用统计分析函数

统计分析函数

函数	描述
sum()	对数组中的元素或某行某列的元素求和
cumsum()	数组元素累计求和
cumprod()	数组元素累计求积
mean()	数组元素求平均值
amin()、amax()	数组的最小值和最大值
average()	计算加权平均数
median()	数组元素的中位数（中值）
var()	方差
std()	标准差

统计分析函数

函数	描述
argmin()、argmax()	数组最小值和最大值的下标（默认情况下为一维下标值）
ptp()	计算最大值和最小值的差
numpy.unravel_index()	根据数组形状将一维下标转换为多维下标
numpy.percentile()	求百分位数的值

统计分析函数

```
>>> n7 = np.arange(12).reshape(3,4)
>>> n7
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])

>>> n7.sum()
66
>>> n7.sum(axis=0)
array([12, 15, 18, 21])
>>> n7.sum(axis=1)
array([ 6, 22, 38])
```

```
>>> n7.cumsum(axis=0)
array([[ 0,  1,  2,  3],
       [ 4,  6,  8, 10],
       [12, 15, 18, 21]], dtype=int32)

>>> n7.cumsum(axis=1)
array([[ 0,  1,  3,  6],
       [ 4,  9, 15, 22],
       [ 8, 17, 27, 38]], dtype=int32)
```

统计分析函数

```
>>> n7 = np.arange(3,15).reshape(3,4)
>>> print(n7)
[[ 3  4  5  6]
 [ 7  8  9 10]
 [11 12 13 14]]
```

```
>>> n7.argmin()
0
>>> n7.argmax()
11
```

```
>>> np.percentile(n7,50)
8.5
>>> np.percentile(n7,30)
6.3
```

```
>>> np.percentile(n7,50,axis=0)
array([ 7.,  8.,  9., 10.])
>>> np.percentile(n7,50,axis=1)
array([ 4.5,  8.5, 12.5])
```

统计分析函数

```
>>> np.unravel_index([22,41,37], (7,6))
(array([3, 6, 6], dtype=int64), array([4, 5, 1], dtype=int64))
>>> np.unravel_index([31,41,13], (7,6), order='F')
(array([3, 6, 6], dtype=int64), array([4, 5, 1], dtype=int64))
```

统计分析函数

```
>>> n7 = np.array([[1,2,3],[4,5,6],[9,10,11]])
```

```
>>> print(np.sum(n7))
51
>>> print(np.amax(n7),np.amin(n7))
11 1
>>> print(np.median(n7),np.mean(n7))
5.0 5.666666666666667
>>> print(np.std(n7),np.var(n7))
3.39934634239519 11.555555555555557
```

```
>>> print(np.amax(n7,axis=0))
[ 9 10 11]
>>> print(np.amin(n7,axis=1))
[1 4 9]
>>> print(np.median(n7,axis=0))
[4. 5. 6.]
>>> print(np.mean(n7,axis=1))
[ 2.  5. 10.]
```

统计分析函数

```
>>> score = np.array([92,78,95,86,100])
>>> credit = np.array([3,2,2,3,2])

>>> print(np.average(score,weights=credit))
90.0
```

排序函数

```
>>> n3 = np.array([[4,5,3],[6,2,8],[7,1,9]])
>>> print(n3)
[[4 5 3]
 [6 2 8]
 [7 1 9]]

>>> print(np.sort(n3))
[[3 4 5]
 [2 6 8]
 [1 7 9]]
```

```
>>> print(np.sort(n3,axis=0))
[[4 1 3]
 [6 2 8]
 [7 5 9]]
>>> print(np.sort(n3,axis=1))
[[3 4 5]
 [2 6 8]
 [1 7 9]]
```

推荐参阅

Lev Maximov, NumPy Illustrated: The Visual Guide to NumPy

https://www.yanxishe.com/TextTranslation/3198

内容

- NumPy
- Pandas

Pandas

Pandas：panel data + Python data analysis：是一款快速、强大、灵活、易用的开源数据分析和操作工具

- 基于Numpy，是Python的核心数据分析支持库
- 提供了快速、灵活、明确的数据结构
- 可以从各种文件格式如 CSV、JSON、SQL、Microsoft Excel 导入数据
- 可以对各种数据进行运算操作，并具有数据清洗和数据加工能力

import pandas as pd

https://pandas.pydata.org

Pandas能够处理的数据类型

- 与SQL或Excel表类似的数据
- 有序和无序（非固定频率）的时间序列数据
- 带行列标签的矩阵数据
- 任意其他形式的观测、统计数据集

Pandas主要数据结构

- Series类型
- DataFrame类型

Series类型：带有标签的一维同构数组

由一组数据及与之相关的数据索引（标签）组成

```
>>> print(s)
0    89
1    92
2    75
dtype: int64
```

Series对象：创建

pandas.Series(data=None, index=None, dtype=None, name=None, copy=False)

- data：数据，支持列表、字典、numpy数组、标量值
- index: 行标签（索引）
- dtype：数据类型
- name: 所创建Series的名字
- copy：是否创建输入数据的副本

Series对象创建：由列表创建

```
>>> s = pd.Series([89,92,75])
```

```
>>> print(s)
0    89
1    92
2    75
dtype: int64
```

Series对象创建：由列表创建

```
>>> s1 = pd.Series([89,92,75],index=['A','B','C'])
```

```
>>> print(s1)
A    89
B    92
C    75
dtype: int64
```

Series对象创建：由ndarray创建

```
>>> import numpy as np
>>> s2 = pd.Series(np.arange(3),index=['a','b','c'])
```

```
>>> print(s2)
a    0
b    1
c    2
dtype: int32
```

Series对象创建：由字典创建

```
>>> dic = {"A":1,"B":2,"C":3,"D":6}
>>> s3 = pd.Series(dic)
```

```
>>> print(s3)
A    1
B    2
C    3
D    6
dtype: int64
```

Series对象创建：由字典创建

```
>>> dic = {"A":1,"B":2,"C":3,"D":6}
>>> s3 = pd.Series(dic)
```

```
>>> print(s3)
A    1
B    2
C    3
D    6
dtype: int64
```

```
>>> index4 =['A','B','C','D','E']
>>> s4 = pd.Series(dic,index4)
```

```
>>> print(s4)
A    1.0
B    2.0
C    3.0
D    6.0
E    NaN
dtype: float64
```

```
>>> pd.isnull(s4)
A    False
B    False
C    False
D    False
E     True
dtype: bool
```

Series对象创建：由Series创建

```
>>> s3
A    1
B    2
C    3
D    6
dtype: int64
```

```
>>> s5 = s3.reindex(['D','C','E','A','B'])
>>> s5
D    6.0
C    3.0
E    NaN
A    1.0
B    2.0
dtype: float64
```

Series对象创建：由Series创建

```
>>> s3
A    1
B    2
C    3
D    6
dtype: int64
```

```
>>> s6 = s3.reindex(
... ['D','C','E','A','B','F'],fill_value=88)
>>> s6
D     6
C     3
E    88
A     1
B     2
F    88
dtype: int64
```

Series对象的索引可重复

```
>>> s4 = pd.Series(np.arange(4),index=[1,2,4,1])

>>> print(s4)
1    0
2    1
4    2
1    3
dtype: int32
```

Series对象的属性和方法

pandas.Series — pandas 1.5.0 documentation

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html

Series对象的常见运算和操作

- Python列表操作
- NumPy一维数组运算及操作
- Python字典操作
- 特有操作

Series对象查看

```
>>> s5.describe()
count    4.000000
mean     3.000000
std      2.160247
min      1.000000
25%      1.750000
50%      2.500000
75%      3.750000
max      6.000000
dtype: float64
```

```
>>> s5.values
array([ 6.,  3., nan,  1.,  2.])
>>> s5.index
Index(['D', 'C', 'E', 'A', 'B'], dtype='object')
```

Series对象的索引和切片：位置索引

```
>>> print(s3)
A    1
B    2
C    3
D    6
dtype: int64
```

```
>>> print(s3[0])
1
>>> print(s3[-1])
6
>>> print(s3[1:3])
B    2
C    3
dtype: int64
```

Series对象的索引和切片：标签索引

```
>>> print(s3)
A    1
B    2
C    3
D    6
dtype: int64
```

```
>>> print(s3['A'])
1
>>> print(s3['A':'C'])
A    1
B    2
C    3
dtype: int64
```

Series对象的索引和切片：标签索引

```
>>> s4 = pd.Series(np.arange(4),index=[1,2,4,1])
>>> print(s4)
1    0
2    1
4    2
1    3
dtype: int32
```

```
>>> print(s4[1])
1    0
1    3
dtype: int32
```

Series对象的索引和切片：标签索引

```
>>> s2 = pd.Series(np.arange(3),index=[5,6,7])
>>> print(s2)
5    0
6    1
7    2
dtype: int32

>>> print(s2[6])
1
```

```
>>> print(s2[5:7])
Series([], dtype: int32)
>>> print(s2[0:3])
5    0
6    1
7    2
dtype: int32
```

Series对象的索引和切片：点索引

```
>>> print(s1)
A    89
B    92
C    75
dtype: int64
```

```
>>> print(s1.A)
89
```

Series对象的算术运算

```
>>> s=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])
```

```
>>> s[1:] + s[:-1]
a    NaN
b    4.0
c    6.0
d    8.0
e    NaN
dtype: float64
```

s[1:]

索引	数据
b	2
c	3
d	4
e	100

+

s[:-1]

索引	数据
a	1
b	2
c	3
d	4

=

s[1:]+s[:-1]

索引	数据
a	NaN
b	4.0
c	6.0
d	8.0
e	NaN

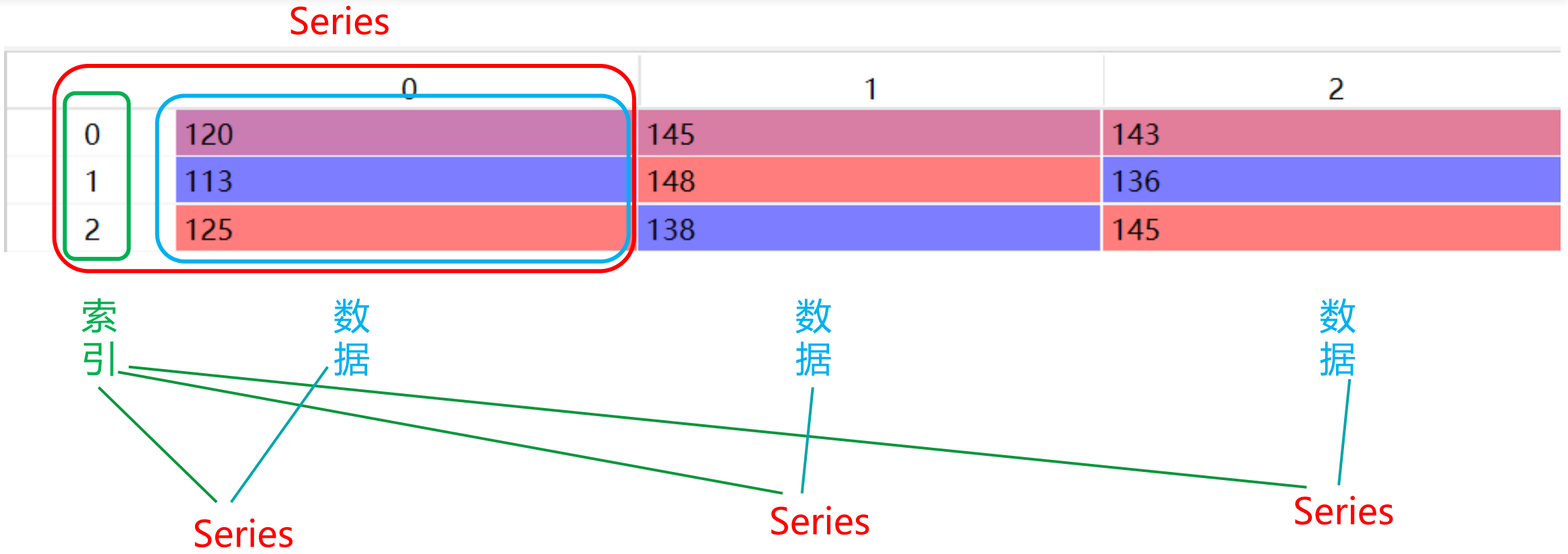
DataFrame类型：带有标签的二维异构表格

由二维数据及与之相关的行列索引（标签）组成

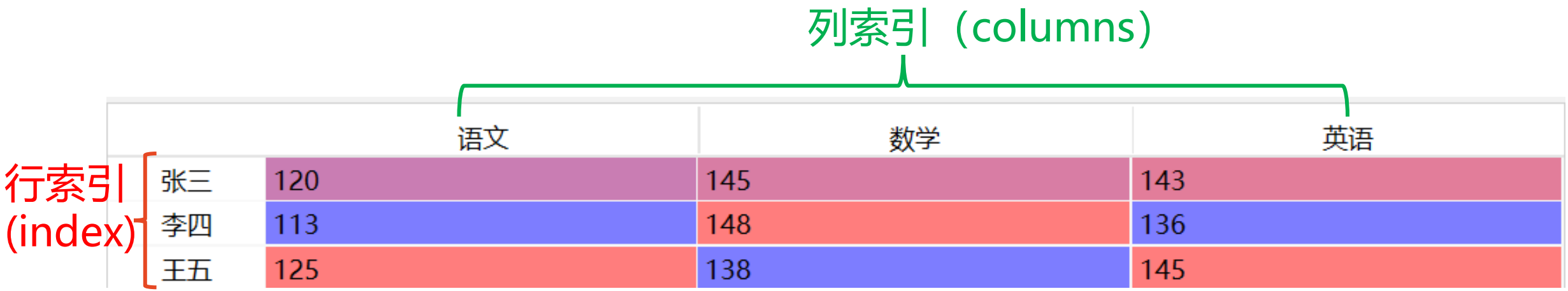
df = {DataFrame: (3, 3)} 0 1 2 [0 120 145 143] [1 113 148 136] [2 125 138 145]

```
>>> print(df)
     0    1    2
0  120  145  143
1  113  148  136
2  125  138  145
```

DataFrame类型



DataFrame类型




```
>>> print(df)
      语文    数学    英语
张三   120   145   143
李四   113   148   136
王五   125   138   145
```

```
>>> df.index = ['张三','李四','王五']
>>> df.columns = ['语文','数学','英语']
```

DataFrame对象：创建

pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)

- data：数据，支持列表、字典、numpy数组、Series对象
- index: 行标签（索引）
- columns：列标签（索引）
- dtype：数据类型
- copy：是否从输入拷贝数据

DataFrame对象创建：由二维数组创建

```
>>> data = [[120,145,143],[113,148,136],[125,138,145]]
>>> df = pd.DataFrame(data)
```

```
>>> print(df)
     0    1    2
0  120  145  143
1  113  148  136
2  125  138  145
```

DataFrame对象创建：由字典创建

```
>>> dict1={'语文':[120,113,125],'数学':[145,148,138],'英语':[143,136,145]}
>>> df1 = pd.DataFrame(dict1)
```

```
>>> print(df1)
    语文   数学   英语
0  120  145  143
1  113  148  136
2  125  138  145
```

DataFrame对象创建：由字典创建

```
>>> dict1 = {'语文':[120,113,125],'数学':[145,148,138],'英语':[143,136,145]}
... index2 = ['张三', '李四', '王五']
... columns2 = ['数学', '语文', '英语', '班级']
... df2 = pd.DataFrame(dict1, index=index2, columns=columns2)
>>> df2['班级'] = '高三五班'
```

```
>>> print(df2)
     数学  语文  英语    班级
张三  145  120  143  高三五班
李四  148  113  136  高三五班
王五  138  125  145  高三五班
```

DataFrame对象的属性和方法

pandas.DataFrame — pandas 1.5.0 documentation

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

DataFrame对象的常见运算和操作

- NumPy二维数组运算及操作
- Python字典操作
- 特有操作

DataFrame对象查看

```
>>> df2.describe()
              数学         语文         英语
count    3.000000    3.000000    3.000000
mean   143.666667  119.333333  141.333333
std      5.131601    6.027714    4.725816
min    138.000000  113.000000  136.000000
25%    141.500000  116.500000  139.500000
50%    145.000000  120.000000  143.000000
75%    146.500000  122.500000  144.000000
max    148.000000  125.000000  145.000000
```

```
>>> df2.values
array([[145, 120, 143, '高三五班'],
       [148, 113, 136, '高三五班'],
       [138, 125, 145, '高三五班']], dtype=object)
>>> df2.index
Index(['张三', '李四', '王五'], dtype='object')
>>> df2.columns
Index(['数学', '语文', '英语', '班级'], dtype='object')
```

DataFrame对象的索引结构

		0	1	2	3	4	5
		A	B	C	D	E	F
0	A						
1	B						
2	C						
3	D						
4	E						
5	F						

DataFrame数据抽取：抽取一行数据

```
>>> print(df1)
      语文   数学   英语      班级
张三   120   145   143   高三五班
李四   113   148   136   高三五班
王五   125   138   145   高三五班
```

```
>>> print(df1.loc['张三'])
语文          120
数学          145
英语          143
班级       高三五班
Name: 张三, dtype: object
```

```
>>> print(df1.iloc[1])
语文          113
数学          148
英语          136
班级       高三五班
Name: 李四, dtype: object
```

DataFrame数据抽取：抽取多行数据

```
>>> print(df1)
      语文   数学   英语      班级
张三   120   145   143  高三五班
李四   113   148   136  高三五班
王五   125   138   145  高三五班
```

```
>>> print(df1.loc[['张三','王五']])
      语文   数学   英语      班级
张三   120   145   143  高三五班
王五   125   138   145  高三五班
```

```
>>> print(df1.iloc[[0,2]])
      语文   数学   英语      班级
张三   120   145   143  高三五班
王五   125   138   145  高三五班
```

DataFrame数据抽取：抽取连续多行数据

```
>>> print(df1.loc['张三':'王五'])
      语文   数学   英语      班级
张三   120   145   143   高三五班
李四   113   148   136   高三五班
王五   125   138   145   高三五班
>>> print(df1.loc[:'李四':])
      语文   数学   英语      班级
张三   120   145   143   高三五班
李四   113   148   136   高三五班
```

```
>>> print(df1.iloc[0:2])
      语文   数学   英语      班级
张三   120   145   143   高三五班
李四   113   148   136   高三五班
>>> print(df1.iloc[1::])
      语文   数学   英语      班级
李四   113   148   136   高三五班
王五   125   138   145   高三五班
```

DataFrame数据抽取：抽取指定列数据

```
>>> print(df1)
      语文   数学   英语     班级
张三   120   145   143  高三五班
李四   113   148   136  高三五班
王五   125   138   145  高三五班
```

```
>>> print(df1[['语文','数学']])
      语文   数学
张三   120   145
李四   113   148
王五   125   138
```

DataFrame数据抽取：抽取指定列数据

```
>>> print(df1.loc[:,['语文','英语']])
      语文   英语
张三    120    143
李四    113    136
王五    125    145
```

```
>>> print(df1.iloc[:,[0,2]])
      语文   英语
张三    120    143
李四    113    136
王五    125    145
```

```
>>> print(df1.loc[:,'数学':])
      数学   英语      班级
张三    145    143   高三五班
李四    148    136   高三五班
王五    138    145   高三五班
```

```
>>> print(df1.iloc[:,:2])
      语文   数学
张三    120    145
李四    113    148
王五    125    138
```

DataFrame数据抽取：抽取指定行列数据

```
>>> print(df1.loc['李四','英语'])
136

>>> print(df1.iloc[1,2])
136

>>> print(df1.loc[['李四'],['英语']])
      英语
李四   136

>>> print(df1.iloc[[1],[2]])
      英语
李四   136
```

```
>>> print(df1.loc['李四':,['数学']])
      数学
李四   148
王五   138

>>> print(df1.iloc[1:,[2]])
      英语
李四   136
王五   145

>>> print(df1.iloc[:,2])
张三    143
李四    136
王五    145
Name: 英语, dtype: int64
```

DataFrame数据抽取：按指定条件抽取

```
>>> print(df1)
      语文   数学   英语      班级
张三   120   145   143   高三五班
李四   113   148   136   高三五班
王五   125   138   145   高三五班
```

```
>>> print(df1.loc[(df1['语文']>115) & (df1['数学']>140)])
      语文   数学   英语      班级
张三   120   145   143   高三五班
```

DataFrame索引设置：设置某列为索引

```
DataFrame.set_index(keys, drop=True, append=False, inplace=False,
verify_integrity=False)
```

- keys：设置为索引的列
- drop: 是否从DataFrame中删除已设置为索引的列
- append：是否将索引列增加到现有索引中
- inplace：是否替代原DataFrame
- verify_integrity：检查新索引是否存在重复项

DataFrame索引设置：设置某列为索引

```
df = pd.DataFrame({'month': [1, 4, 7, 10],
                   'year': [2012, 2014, 2013, 2014],
                   'sale': [55, 40, 84, 31]})
```

```
>>> df
   month  year  sale
0      1  2012    55
1      4  2014    40
2      7  2013    84
3     10  2014    31
```

```
>>> df.set_index('month')
       year  sale
month
1      2012    55
4      2014    40
7      2013    84
10     2014    31
```

```
>>> df.set_index(['year', 'month'])
                sale
year month
2012 1            55
2014 4            40
2013 7            84
2014 10           31
```

DataFrame数据修改：按列增加数据

```
pd.set_option('display.unicode.east_asian_width', True)
data = [[110,105,99],[105,88,115],[109,120,130],[112,115,140]]
name = ['张三','李四','王五','赵六']
columns = ['语文','数学','英语']
df = pd.DataFrame(data=data, index=name, columns=columns)
```

```
>>> df
     语文  数学  英语
张三   110   105    99
李四   105    88   115
王五   109   120   130
赵六   112   115   140
```

```
>>> df.loc[:,'物理'] = [88,79,60,50]
>>> df
     语文  数学  英语  物理
张三   110   105    99    88
李四   105    88   115    79
王五   109   120   130    60
赵六   112   115   140    50
```

```
>>> df.loc[:,'物理'] = 55
>>> df
     语文  数学  英语  物理
张三   110   105    99    55
李四   105    88   115    55
王五   109   120   130    55
赵六   112   115   140    55
```

DataFrame数据修改：按列增加数据

```
>>> df
      语文   数学   英语
张三   110   105    99
李四   105    88   115
王五   109   120   130
赵六   112   115   140
```

```
wl =[88,79,60,50]
df.insert(1,'物理',wl)
```

```
>>> df
      语文   物理   数学   英语
张三   110    88   105    99
李四   105    79    88   115
王五   109    60   120   130
赵六   112    50   115   140
```

DataFrame数据修改：按行增加数据

```
>>> df
      语文   数学   英语
张三   110   105    99
李四   105    88   115
王五   109   120   130
赵六   112   115   140
```

```
>>> df.loc['钱多多'] = [100,120,99]
>>> df
      语文   数学   英语
张三   110   105    99
李四   105    88   115
王五   109   120   130
赵六   112   115   140
钱多多  100   120    99
```

DataFrame数据修改：修改行数据

```
>>> df
      语文   数学   英语
张三   110   105    99
李四   105    88   115
王五   109   120   130
赵六   112   115   140
```

```
>>> df.loc['李四']=[120,115,109]
>>> df
      语文   数学   英语
张三   110   105    99
李四   120   115   109
王五   109   120   130
赵六   112   115   140
```

```
>>> df.loc['张三']=df.loc['张三']+10
>>> df
      语文   数学   英语
张三   120   115   109
李四   120   115   109
王五   109   120   130
赵六   112   115   140
```

DataFrame数据修改：修改列数据

```
>>> df
      语文   数学   英语
张三   110   105    99
李四   105    88   115
王五   109   120   130
赵六   112   115   140
```

```
>>> df.loc[:,'语文']=[115,108,112,118]
>>> df
      语文   数学   英语
张三   115   105    99
李四   108    88   115
王五   112   120   130
赵六   118   115   140
```

```
>>> df.loc[:,'数学']=df.loc[:,'数学']*1.1
>>> df
      语文    数学   英语
张三   115   115.5    99
李四   108    96.8   115
王五   112   132.0   130
赵六   118   126.5   140
```

DataFrame数据修改：删除数据

```
DataFrame.drop(labels=None, axis=0, index=None, columns=None, level=None,
inplace=False, errors='raise')
```

- labels：要删除数据的行标签或列标签
- axis: 执行删除的轴
- index：要删除数据的行标签
- columns：要删除数据的列标签
- level：对存在多级索引的数据，指明从哪级索引中进行删除
- inplace：是否替换原DataFrame
- error：错误处理

DataFrame数据修改：删除列数据

```
>>> df
      语文   数学   英语
张三   110   105    99
李四   105    88   115
王五   109   120   130
赵六   112   115   140
```

```
>>> df.drop(labels='数学',axis=1)
>>> df.drop(columns='数学')
>>> df.drop(['数学'],axis=1)
```

```
      语文   英语
张三   110    99
李四   105   115
王五   109   130
赵六   112   140
```

DataFrame数据修改：删除行数据

```
>>> df
      语文   数学   英语
张三    110    105     99
李四    105     88    115
王五    109    120    130
赵六    112    115    140
```

```
>>> df.drop(['王五'])
>>> df.drop(index='王五')
>>> df.drop(labels='王五',axis=0)
```

```
      语文   数学   英语
张三    110    105     99
李四    105     88    115
赵六    112    115    140
```

DataFrame数据修改：按条件删除行

```
>>> df
      语文   数学   英语
张三   110   105    99
李四   105    88   115
王五   109   120   130
赵六   112   115   140
```

```
>>> df.drop(index=df[df['数学'].isin([88,120])].index)
      语文   数学   英语
张三   110   105    99
赵六   112   115   140

>>> df.drop(index=df[df['英语']<120].index,inplace=True)
>>> df
      语文   数学   英语
王五   109   120   130
赵六   112   115   140
```

DataFrame对象排序：按值排序

DataFrame.sort_values(by, axis=0, ascending=True, inplace=False)

```
>>> print(df2)
      语文   数学   英语      班级
张三   120   145   143   高三五班
李四   113   148   136   高三五班
王五   125   138   145   高三五班

>>> df2.sort_values(by='语文')
      语文   数学   英语      班级
李四   113   148   136   高三五班
张三   120   145   143   高三五班
王五   125   138   145   高三五班
```

```
>>> df2.sort_values(by=['英语','数学'],ascending=False)
      语文   数学   英语      班级
王五   125   138   145   高三五班
张三   120   145   143   高三五班
李四   113   148   136   高三五班
>>> df2.sort_values(by=['数学','英语'],ascending=False)
      语文   数学   英语      班级
李四   113   148   136   高三五班
张三   120   145   143   高三五班
王五   125   138   145   高三五班
```

DataFrame对象排序：按值排序

DataFrame.sort_values(by, axis=0, ascending=True, inplace=False)

```
>>> df2
     name  quantity  price  sold
3     pen     500.0    9.9  50.0
1   ruler     300.0    3.1  30.0
2  rubber     500.0    2.3  78.0
```

```
>>> df2.sort_values(by=['quantity','sold'],ascending=[True,False])
     name  quantity  price  sold
1   ruler     300.0    3.1  30.0
2  rubber     500.0    2.3  78.0
3     pen     500.0    9.9  50.0
```

DataFrame对象的算术运算

```
>>> df3 = pd.DataFrame(np.arange(15).reshape(3,5))
>>> df4 = pd.DataFrame(np.arange(16).reshape(4,4))
>>> print(df3)
    0   1   2   3   4
0   0   1   2   3   4
1   5   6   7   8   9
2  10  11  12  13  14
>>> print(df4)
    0   1   2   3
0   0   1   2   3
1   4   5   6   7
2   8   9  10  11
3  12  13  14  15
```

```
>>> print(df3+df4)
      0     1     2     3   4
0   0.0   2.0   4.0   6.0 NaN
1   9.0  11.0  13.0  15.0 NaN
2  18.0  20.0  22.0  24.0 NaN
3   NaN   NaN   NaN   NaN NaN
```

```
>>> print(df3.add(df4,fill_value=100))
       0      1      2      3      4
0    0.0    2.0    4.0    6.0  104.0
1    9.0   11.0   13.0   15.0  109.0
2   18.0   20.0   22.0   24.0  114.0
3  112.0  113.0  114.0  115.0    NaN
```

数据计算：总结

- Numpy：Python数据分析和机器学习的基础库，核心为数组运算
- Numpy：数组的概念、创建、运算、索引切片、增删改、重塑、统计分析
- Pandas：Python核心数据分析支持库
- Pandas主要数据结构：Series和DataFrame
- Pandas基本数据处理：创建、数据抽取、索引设置、数据增删改、排序