Chap10 Scientific Computing and Data Processing

Python科学计算与数据分析开发基础

Department of Computer Science and Technology
Department of University Basic Computer Teaching
Nanjing University

10.1

科学计算生态系统SciPy

SciPy

特征

- 基于Python的软件生态系统(ecosystem)
- 开源
- 主要为数学、科学和工程服务

NumPy
Base N-dimensional array package

SciPy library
Fundamental library for scientific computing

Matplotlib
Comprehensive 2D Plotting

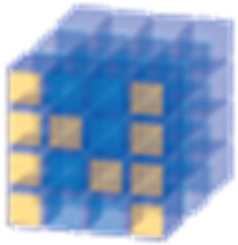
IP[y]:
IPython

IPython
Enhanced Interactive Console

Sympy
Symbolic mathematics

pandas
Data structures & analysis

NumPy



特征

- 高性能科学计算和数据分析的基础包
- 强大的ndarray对象
- 精巧的函数和ufunc函数
- 适合线性代数和随机数处理等科学计算

Source

```
>>> import numpy as np
>>> xArray = np.ones((3, 4))
```

SciPy library

特征

- Python中科学计算程序的核心包
- 有效计算numpy矩阵，让NumPy和SciPy library协同工作
- 致力于科学计算中常见问题的各个工具箱，其不同子模块有不同的应用，如插值、积分、优化和图像处理等

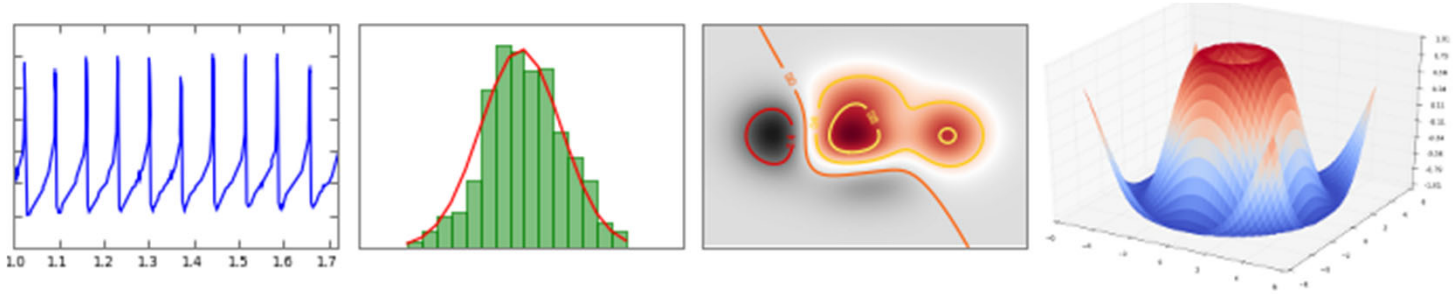
Source

```
>>> import numpy as np
>>> from scipy import linalg
>>> arr = np.array([[1, 2], [3, 4]])
>>> linalg.det(arr)
-2.0
```

Matplotlib

特征

- 基于NumPy
- 二维绘图库，简单快速地生成曲线图、直方图和散点图等形式的图
- 常用的pyplot是一个简单提供类似MATLAB接口的模块



pandas

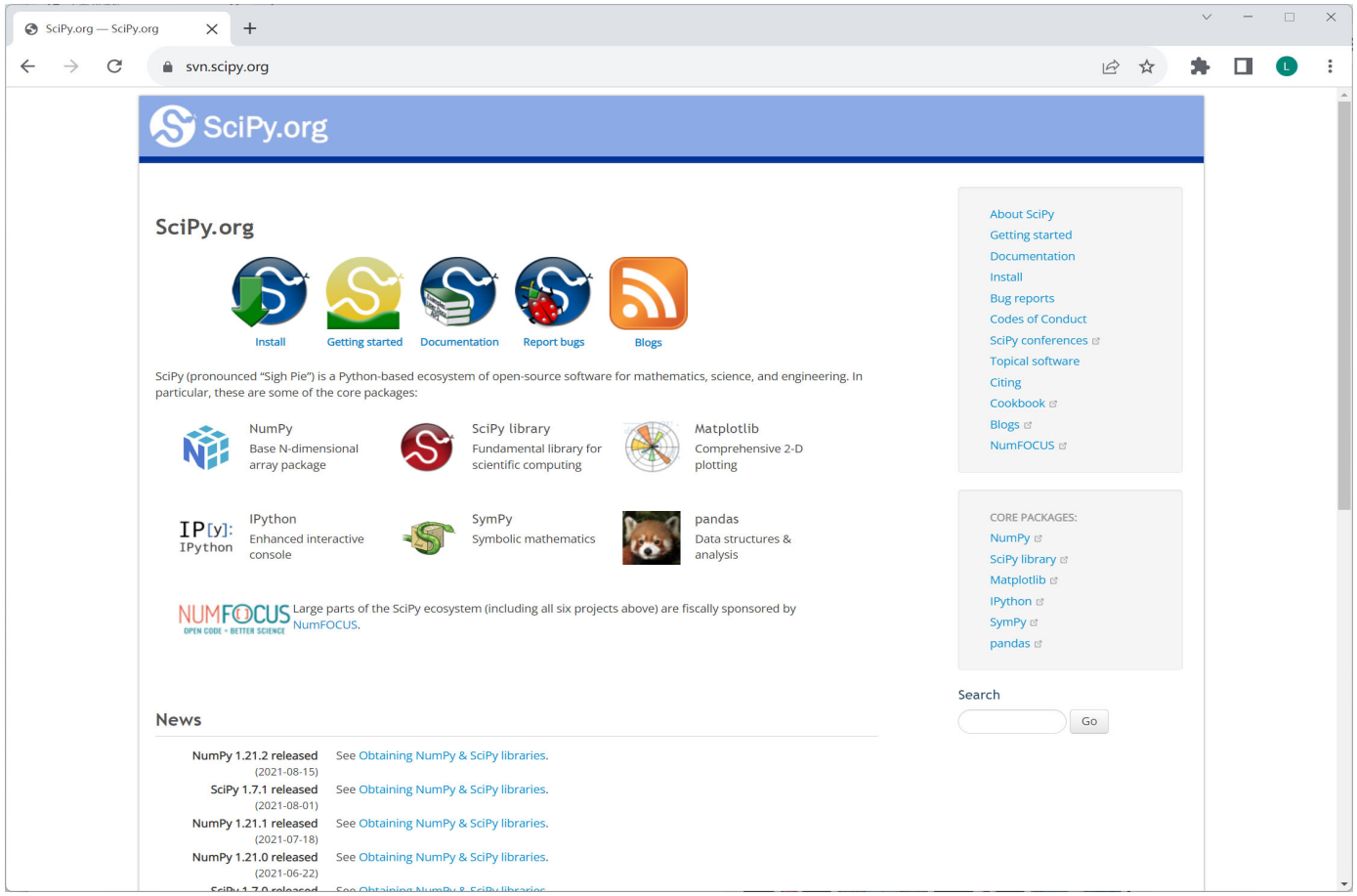
特征

- 基于 SciPy library和 NumPy
- 高效的Series和DataFrame数据结构
- 强大的可扩展数据操作与分析的Python库
- 高效处理大数据集的切片等功能
- 提供优化库功能读写多种文件格式，如CSV、 HDF5

Source

```
...
>>> df[2 : 5]
>>> df.head(4)
>>> df.tail(3)
```

Home Page


SciPy.org — SciPy.org
svn.scipy.org
SciPy.org
SciPy.org
Install
Getting started
Documentation
Report bugs
Blogs
SciPy (pronounced "Sigh Pie") is a Python-based ecosystem of open-source software for mathematics, science, and engineering. In particular, these are some of the core packages:
NumPy
Base N-dimensional array package
SciPy library
Fundamental library for scientific computing
Matplotlib
Comprehensive 2-D plotting
IPython
Enhanced interactive console
SymPy
Symbolic mathematics
pandas
Data structures & analysis
NUMFOCUS
Large parts of the SciPy ecosystem (including all six projects above) are fiscally sponsored by NumFOCUS.
News
NumPy 1.21.2 released (2021-08-15) See Obtaining NumPy & SciPy libraries.
SciPy 1.7.1 released (2021-08-01) See Obtaining NumPy & SciPy libraries.
NumPy 1.21.1 released (2021-07-18) See Obtaining NumPy & SciPy libraries.
NumPy 1.21.0 released (2021-06-22) See Obtaining NumPy & SciPy libraries.
About SciPy
Getting started
Documentation
Install
Bug reports
Codes of Conduct
SciPy conferences
Topical software
Citing
Cookbook
Blogs
NumFOCUS
CORE PACKAGES:
NumPy
SciPy library
Matplotlib
IPython
SymPy
pandas
Search
Go

Python常用的数据结构

其他数据结构?

- **SciPy中的数据结构**

  Python原有数据结构的变化

  - ndarray(N维数组)
  - Series(变长字典)
  - DataFrame(数据框)

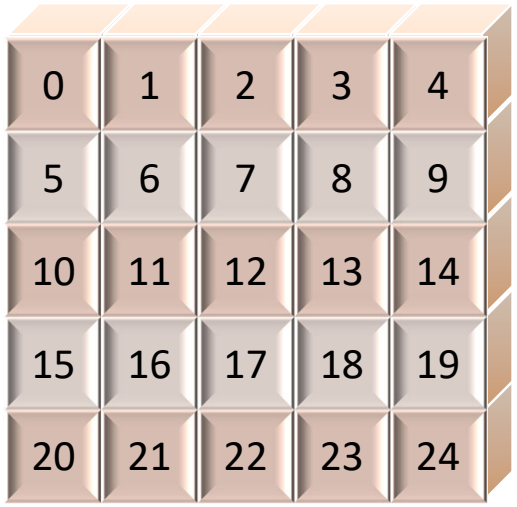
10.2

NumPy

Python中的数组

- 用list和tuple等数据结构表示数组
  - 一维数组 list = [1,2,3,4]
  - 二维数组 list = [[1,2,3],[4,5,6],[7,8,9]]
- array模块
  - 通过array函数创建数组， array.array("B", range(5))
  - 提供append、 insert和read等方法

ndarray



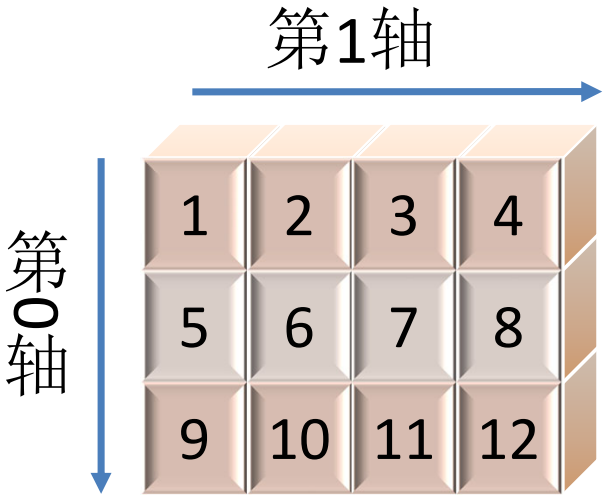
- **ndarray是什么?**

  N维数组

  - NumPy中基本的数据结构
  - 所有元素是同一种类型
  - 别名为array
  - 利于节省内存和提升计算性能
  - 有丰富的函数

10.2.1 ndarray的基本特性

ndarray基本概念



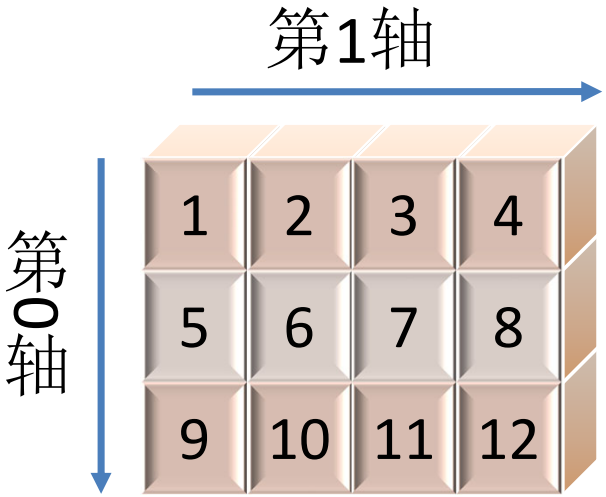
C order vs F order

- ndarray数组属性

N维数组

- 形状(shape)称为轴(axes)，轴的个数称为维数(dimension)
- 沿着第0轴和第1轴操作
  - axis = 0(按列)
  - axis = 1(按行)

ndarray基本概念



- **ndarray数组属性**

N维数组

– 基本属性

- ndarray.ndim(维数，有几个轴)
- ndarray.shape(形状，每个轴有几个元素)
- ndarray.size(一共有几个元素)
- ndarray.dtype(元素类型)
- ndarray.itemsize(元素字节大小)

10.2.2 创建ndarray

ndarray的创建

array()函数

[x.,]表示该元素是浮点数

Source

```
>>> import numpy as np
>>> aArray = np.array([1,2,3])
>>> aArray
array([1, 2, 3])
>>> bArray=np.array([(1,2,3),(4,5,6)], dtype ='float32')
>>> bArray
array([[1., 2., 3.],
       [4., 5., 6.]], dtype=float32)
>>> bArray.ndim, bArray.shape, bArray.dtype
(2, (2, 3), dtype('float32'))
```

ndarray的创建

zeros()：生成全零矩阵

ones()：生成全一矩阵

Source

```
>>> np.zeros((2, 2))
array([[ 0.,  0.],
       [ 0.,  0.]])
>>> np.ones([2, 3])
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
```

ndarray的创建

full()：根据所给元素生成满矩阵

Source

```
>>> np.full((2,2), 0)
array([[0, 0],
       [0, 0]])
>>> np.full((2,3), 1, dtype = np.int32)
array([[1, 1, 1],
       [1, 1, 1]], dtype=int32)
>>> np.full((3,3), np.pi)
array([[3.14159265, 3.14159265, 3.14159265],
       [3.14159265, 3.14159265, 3.14159265],
       [3.14159265, 3.14159265, 3.14159265]])
```

ndarray的创建

zeros_like()
ones_like()
full_like()
根据所给矩阵的shape生成全零/全一/满矩阵

Source

```
>>> x = np.array([[1,2,3], [4,5,6]], dtype = np.float32)
>>> np.zeros_like(x)
array([[0., 0., 0.],
       [0., 0., 0.]], dtype=float32)
>>> np.ones_like(x)
array([[1., 1., 1.],
       [1., 1., 1.]], dtype=float32)
>>> np.full_like(x, 5)
array([[5., 5., 5.],
       [5., 5., 5.]], dtype=float32)
```

ndarray的创建

identity()：生成单位矩阵

eye()：生成对角线为1的矩阵

Source

```
>>> np.identity(3)
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
>>> np.eye(3)
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Source

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

ndarray的创建

arange()
基于range直接生成矩阵

linspace()
基于等差序列生成矩阵

Source

```
>>> np.arange(1, 5, 0.5)
array([ 1. ,  1.5,  2. ,  2.5,  3. ,  3.5,  4. ,  4.5])
>>> np.linspace(1, 2, 10, endpoint = False)
array([ 1. ,  1.1,  1.2,  1.3,  1.4,  1.5,  1.6,  1.7,  1.8,  1.9])
```

ndarray的创建

empty()
基于shape生成未初始化矩阵

random.rand()
生成(0,1)均匀分布矩阵

random.uniform()
生成(low,high)均匀分布矩阵

random.normal()
生成(/mu,/sigma)正态分布矩阵

Source

```
>>> np.random.random((2, 2))
array([[ 0.79777004,  0.1468679 ],
       [ 0.95838379,  0.86106278]])
>>> np.random.normal(loc=3, scale=3**0.5, size=1000)
>>> np.random.normal(loc=0, scale=1, size=1000)
>>> np.random.randn(1000)
>>> np.random.uniform(low=-5,high=5,size=1000)
>>> # N(3,3), N(0,1), U[-5,5)
```

ndarray的创建

fromfunction()
基于索引号
i,j,k,....生成矩
阵元素

Source

```
>>> np.fromfunction(lambda i, j:(i+1)*(j+1), (9,9))
array([[ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9.],
       [ 2.,  4.,  6.,  8., 10., 12., 14., 16., 18.],
       [ 3.,  6.,  9., 12., 15., 18., 21., 24., 27.],
       [ 4.,  8., 12., 16., 20., 24., 28., 32., 36.],
       [ 5., 10., 15., 20., 25., 30., 35., 40., 45.],
       [ 6., 12., 18., 24., 30., 36., 42., 48., 54.],
       [ 7., 14., 21., 28., 35., 42., 49., 56., 63.],
       [ 8., 16., 24., 32., 40., 48., 56., 64., 72.],
       [ 9., 18., 27., 36., 45., 54., 63., 72., 81.]])
```

NumPy数组文件存取


```
>>> x = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> np.savetxt('arr.txt', x, fmt='%d')
>>> data = np.loadtxt('arr.txt')
```

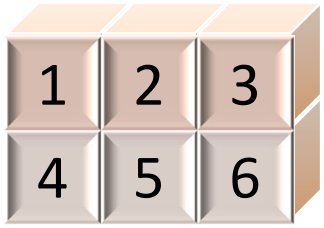
```
1 2 3
4 5 6
7 8 9
```

10.2.3 ndarray的操作和运算

ndarray的基本操作-基本索引&花式索引

索引方式	语法示例	返回结果说明	核心用途
整数索引	arr[i](1D) arr[i, j](2D)	1D 返回索引i对应的单个元素； 2D 返回第i行第j列的单个元素	获取指定位置的单个元素
切片索引	arr[i:j](1D) arr[i:j, k:l](2D)	1D 返回索引i到j-1的连续子数组；2D 返回第i到j-1行、第k到l-1列的连续子矩阵	获取连续子数组 / 子矩阵
整数数组索引	arr[[i, j, k]](1D) arr[[i, j], [k, l]](2D)	1D 返回索引i、j、k对应的离散元素； 2D 返回(i,k)、(j,l)位置的离散元素(若需子矩阵需配合维度切片)	获取离散位置的元素 / 行 / 列

ndarray的基本操作-基本索引&花式索引



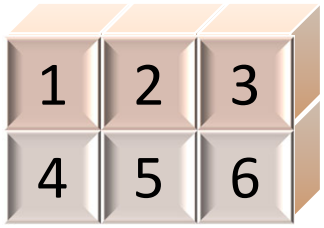
Source

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

Source

```
>>> print(aArray[[0, 2]])
[[1 2 3]
 [7 8 9]]
>>> print(aArray[[0, 2], 1])
[2 8]
```

ndarray的基本操作-切片(索引)



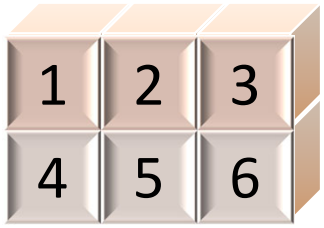
Source

```
>>> aArray = np.array([(1, 2, 3), (4, 5, 6)])
array([[1, 2, 3],
       [4, 5, 6]])
>>> print(aArray[0: 2])
[[1 2 3]
 [4 5 6]]
>>> print(aArray[:, 0: 2])
[[1 2]
 [4 5]]
>>> print(aArray[1, 0: 2])
[4 5]
```

Source

```
>>> print(aArray[::-1])
[[4 5 6]
 [1 2 3]]
>>> print(aArray[:,::-1])
[[3 2 1]
 [6 5 4]]
```

ndarray的基本操作-切片(索引)



Source

```
>>> bArray = np.ones((5, 5))
>>> bArray[1: -1, 1: -1] = 0
>>> print(bArray)
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
>>> aArray[[0, 1], :] = aArray[[1, 0], :]
>>> aArray
array([[4, 5, 6],
       [1, 2, 3]])
```

数组切片是创建副本还是视图?

ndarray的基本操作-布尔索引

索引方式	语法示例	返回结果说明	核心用途
布尔索引	arr[condition]	返回与原数组中condition为True位置对应的所有元素(统一为 1D 结构副本)	按条件筛选元素(如数值范围、奇偶性等)
np.ix_()索引	arr[np.ix_([i,j], [k,l])](2D)	返回第i、j行与第k、l列交叉组成的子矩阵(保留原数组维度结构副本)	提取离散行和离散列的不规则子矩阵
np.where()索引	arr[np.where (condition, arrayA, arrayB)]	基于condition从arrayA和arrayB的对应索引选取元素构建矩阵副本	结合条件筛选与位置获取，精准提取目标元素

ndarray的基本操作-布尔索引

Source

```
>>> aArray = np.arange(1, 101)
>>> bArray = aArray[aArray <= 50]
...
>>> aArray[(aArray % 2 == 0) & (aArray > 50)]
...
>>> aArray[(aArray % 2 == 0)]  = -1
...
>>> aArray = np.arange(1, 101)
>>> cArray = np.where(aArray % 2 == 0, -1, aArray)
>>> cArray = np.where(aArray % 2 == 0)
```

np.nonzero()
np.argwhere()

ndarray的基本操作-改变数组形状1

Source

```
>>> aArray = np.array([(1,2,3),(4,5,6)])
>>> aArray.shape
(2, 3)
>>> bArray = aArray.reshape(3,2)
>>> bArray
array([[1, 2],
       [3, 4],
       [5, 6]])
>>> aArray
array([[1, 2, 3],
       [4, 5, 6]])
```

ndarray的基本操作-改变数组形状1

Source

```
>>> aArray = np.arange(1, 17)
>>> aArray
array([ 1,  2,  3, ..., 14, 15, 16])
>>> aArray.resize(4, 4)
>>> aArray
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

-1的功能

ndarray的基本操作-改变数组形状2

Source

```
>>> aArray = np.array([(1,2,3),(4,5,6)])
>>> # 数组展平
>>> x = aArray.ravel()    # view
>>> y = aArray.flatten('F')    # copy
```

ndarray的基本操作-数组组合

Source

```
>>> bArray = np.array([1,3,7])
>>> cArray = np.array([3,5,8])
>>> np.vstack((bArray, cArray))
array([[1, 3, 7],
       [3, 5, 8]])
>>> np.hstack((bArray, cArray))
array([1, 3, 7, 3, 5, 8])
```

ndarray的基本操作-数组分割

Source

```
>>> dArray = np.arange(1,17).reshape(4,4)
>>> dArray
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
>>> np.hsplit(dArray, 2)
>>> np.vsplit(dArray, 2)
>>> np.split(dArray, 2, axis = 1)
>>> np.split(dArray, 2, axis = 0)
```

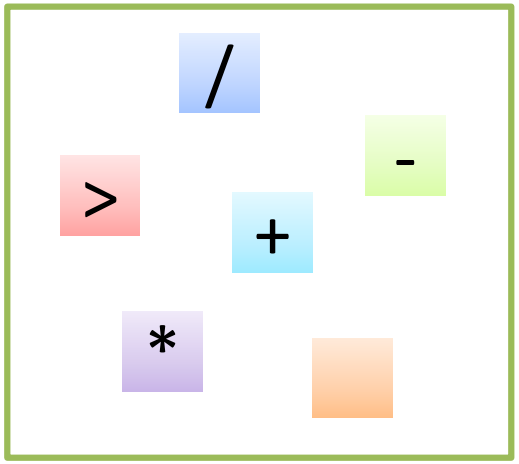
hsplit()/axis=1: 水平方向
vsplit()/axis=0: 垂直方向

ndarray的基本操作-矩阵转置

Source

```
>>> y = np.array([[7, 3], [6, 4]])
>>> y.T          # y.transpose()
array([[7, 6],
       [3, 4]])
```

ndarray的运算



利用基本运算符

Source

```
>>> aArray = np.array([(1,2,3),(4,5,6)])
>>> bArray = np.array([(2,2,2),(2,2,2)])
>>> cArray = aArray * bArray
>>> cArray
array([[2, 4, 6],
       [8, 10, 12]])
>>> aArray += bArray
>>> aArray
array([[3, 4, 5],
       [6, 7, 8]])
```

并非矩阵乘法
c[i,j] =a[i,j]*b[i,j]

ndarray的运算

Source

```
>>> x = np.array([[2,3], [3,4]])
>>> y = np.array([[7,3], [6,4]])
>>> np.dot(x, y)
array([[32, 18],
       [45, 25]])
>>> x = np.matrix([[2,3], [3,4]])
>>> y = np.matrix([[7,3], [6,4]])
>>> x * y
matrix([[32, 18],
        [45, 25]])
```

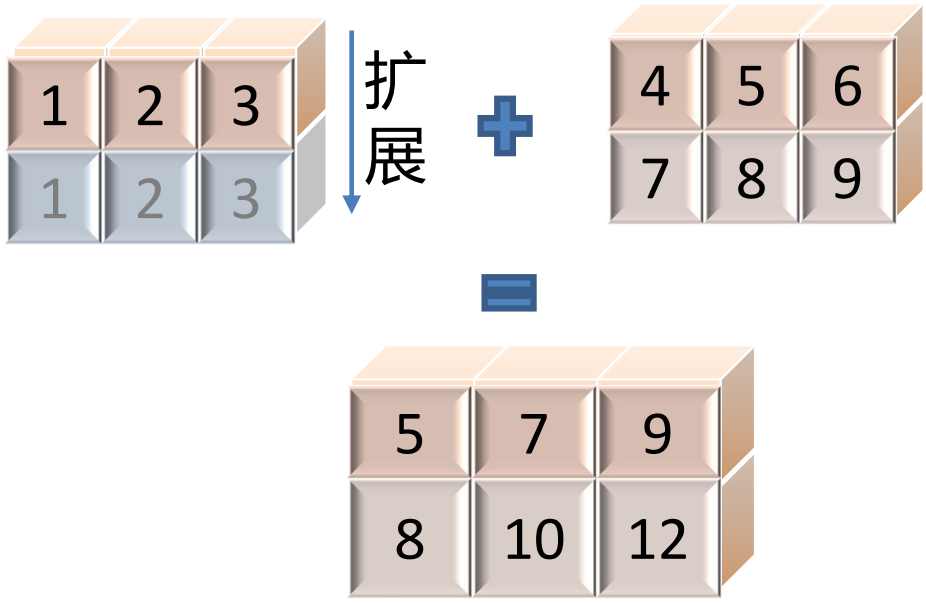
矩阵乘法

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

ndarray的运算

广播功能

较小的数组会广播到较大数组的大小，使它们的形状兼容



Source

```
>>> a = np.array([1,2,3])
>>> b = np.array([[4,5,6],[7,8,9]])
>>> a + b
array([[5, 7, 9],
       [8, 10, 12]])
```

广播

- 若有3名同学各2门课的成绩数组，请计算每门课与该门课均值(mean()方法)的差值。

```
scores = np.array([[98, 76, 87], [76, 88, 91]])
```

处理后输出：

```
array([[ 11., -11.,   0.],
       [ -9.,   3.,   6.]])
```

```
scores - scores.mean(axis = 1, keepdims = True)
```

矢量运算&广播例：计算前N项Fibonacci数列

$$F_n = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

Binet's Formula

```
import numpy as np

def fibonacci(n):
    ratio = (1 + np.sqrt(5)) / 2
    fib = (ratio**n - (-1 / ratio)* n) / np.sqrt(5)
    return fib

N = 20
n = np.arange(1, N + 1)
fib = fibonacci(n)
fib = fib.astype(int)
print(fib)
```

ndarray的运算—简单统计

Source

```
>>> aArray = np.array([(6,5,4),(3,2,1)])
>>> aArray.sum()
21
>>> np.sum(aArray >= 5)
2
>>> aArray.sum(axis = 0)
array([9, 7, 5])
>>> aArray.sum(axis = 1)
array([ 15, 6])
>>> aArray.min()      # return value
1
>>> aArray.argmin()      # return index
5
```

sum	mean
std	var
min	max
argmin	argmax
cumsum	cumprod

利用基本数组统计方法

ndarray的运算—简单统计

Source

```
>>> aArray = np.array([(6,5,4),(3,2,1)])
>>> aArray.mean()
3.5
>>> aArray.var()
2.9166666666666665
>>> aArray.std()
1.707825127659933
>>> np.cumprod(np.arange(1,11))
array([1, 2, 6, 24, 120, 720, 5040,
40320, 362880, 3628800], dtype=int32)
```

sum	mean
std	var
min	max
argmin	argmax
cumsum	cumprod

利用基本数组统计方法

ndarray的运算—统计

Source

```
>>> aArray = np.array([3, 1, 2])
>>> aArray.argsort()    # np.argsort(aArray)
array([1, 2, 0], dtype=int64)
>>> aArray[aArray.argsort()]
>>> aArray.argsort()[0]
1
>>> bArray = np.array([(6,4,5),(3,2,8)])
>>> bArray.argsort()
array([[1, 2, 0],
       [1, 0, 2]], dtype=int64)
>>> bArray.argsort(axis = 0)
array([[1, 1, 0],
       [0, 0, 1]], dtype=int64)
```

argsort

利用常用数组统计方法

a.argsort(axis=-1, kind='quicksort')

ndarray的运算—统计

Source

```
>>> Z = np.array([1, 3, 7, 3, 2, 2, 3, 2])
>>> count = np.bincount(Z)
>>> count
array([0, 1, 3, 3, 0, 0, 0, 1], dtype=int64)
# 计算Z中出现频次最高的整数
>>> count.argmax()
>>> np.nonzero(count == count.max())
(array([2, 3], dtype=int64),)
```

bincount

利用常用数组统计方法

10.2.4 ufunc函数

ndarray的ufunc函数

- ufunc(universal function，通用)是一种能对数组的每个元素进行操作的函数。NumPy内置的许多ufunc函数都是在C语言级别实现的，计算速度非常快，数据量大时有很大的优势。

Math operations: add, subtract, mod, power, log, …
Trigonometric functions: sin, cos, tan, tanh, …
Bit-twiddling functions: bitwise_and, invert, …
Comparison functions: greater, less, equal, logical_and, …
Floating functions: isinf, fabs, modf, floor, …

https://docs.scipy.org/doc/numpy/reference/ufuncs.html

ndarray的ufunc函数

File

```
# Filename: prog10-1.py
import time
import math
import numpy as np
x = np.arange(0, 100, 0.01)
t_m1 = time.perf_counter()
for i, t in enumerate(x):
        x[i] = math.pow((math.sin(t)), 2)
t_m2 = time.perf_counter()
y = np.arange(0, 100, 0.01)
t_n1 = time.perf_counter()
y = np.power(np.sin(y), 2)
t_n2 = time.perf_counter()
```

```
Running time of math: t_m2 - t_m1
Running time of numpy: t_n2 - t_n1
```

ndarray的ufunc函数

File

```
# Filename: prog10-2.py
import numpy as np
import time
arr1 = np.random.randn(10000000)
arr2 = np.random.randn(10000000)
start = time.perf_counter()
r = 0
for i in range(len(arr1)):
    r += arr1[i] * arr2[i]
print(time.perf_counter() - start, 's')
start = time.perf_counter()
r = np.dot(arr1, arr2)
print(time.perf_counter() - start, 's')
```

```
4.634805200010305 s
0.086594699983862 s
```

将标量函数转换为ufunc函数

- np.frompyfunc(func, nin, nout)
  nin: 参数个数，int类型
  nout：函数返回对象个数，int类型
- Use frompyfunc to add broadcasting to the Python function oct

```
>>> oct_array = np.frompyfunc(oct, 1, 1)
>>> oct_array(np.array((10, 30, 100)))
array(['0o12', '0o36', '0o144'],
dtype=object)
>>> np.array((oct(10), oct(30), oct(100)))
array(['0o12', '0o36', '0o144'], dtype='<U5')
```

思考：构建计算[1,1000]间所有整数的反序数ufunc函数

```
def rev(x):
    return int(str(x)[::-1])
rev_func = np.frompyfunc(rev, 1, 1)
print(rev_func(np.arange(1, 1001)))
```

10.2.5 专门的应用

ndarray的专门应用—线性代数

Source

```
>>> import numpy as np
>>> x = np.array([[1,2],
[3,4]])
>>> r1 = np.linalg.det(x)
>>> print(r1)
-2.0
>>> r2 = np.linalg.inv(x)
>>> print(r2)
[[-2.   1. ]
 [ 1.5 -0.5]]
```

dot	矩阵内积
linalg.det	行列式
linalg.inv	逆矩阵
linalg.solve	多元一次方程组求根
linalg.eig	求特征值和特征向量

常用函数示例

求解线性方程组

- 利用NumPy的linalg模块中的solve()函数

例如要求解如下二元一次方程组：

$-x_0 + 4x_1 = 10$

$2x_0 + 3x_1 = 13$

Source

```
>>> a = np.array([[-1,4], [2,3]])
>>> b = np.array([10,13])
>>> x = np.linalg.solve(a, b)
>>> x
array([ 2.,  3.])
```

求解非线性方程组

- 使用scipy.optimize模块中的fsolve()函数来解决。

例如对于如下非线性方程组：

$2x_0^2 - 3x_1 = 6$

$3x_0 + x_1^2 = 25$

```
[ 3.  4.]
```

File

```
from scipy.optimize import fsolve

def f(x):
    x0, x1 = float(x[0]), float(x[1])
    return [2*x0**2-3*x1-6, 3*x0+x1**2-25]

x = fsolve(f, [0, 0])
print(x)
```

10.3

pandas

10.3.1 Series

Series

- **基本特征**
  - 由数据和索引组成(有序字典，也称变长字典)

Series()函数

Source

```
>>> import pandas as pd
>>> aSer = pd.Series([1, 2.0, 'a'])
>>> aSer
0   1
1   2
2   a
dtype: object
```

自定义Series的index

Source

```
>>> bSer = pd.Series(['apple','peach','lemon'], index = [1,2,3])
>>> bSer
1    apple
2    peach
3    lemon
dtype: object
>>>aSer.index = [1,2,3]
>>> aSer
1    1
2    2
3    a
dtype: object
```

自定义Series的index

Source

```
>>> bSer.index        # 常进行单独赋值
Int64Index([1, 2, 3], dtype = 'int64')
>>> bSer.values
array(['apple', 'peach', 'lemon'], dtype = object)
```

Series的基本运算

Source

```
>>> cSer = pd.Series([3, 5, 7], index = ['a', 'b', 'c'])
>>> cSer['b']
5
>>> cSer * 2
a    6
b   10
c   14
dtype: int64
>>> import numpy as np
>>> np.exp(cSer)
a       20.085537
b      148.413159
c     1096.633158
dtype: float64
```

Series的基本运算

切片
基于位置
基于索引

Source

```
>>> cSer = pd.Series([3, 5, 7], index = ['a', 'b', 'c'])
>>> cSer[1: 2]
b   5
dtype: int64
>>> cSer['a': 'b']
a   3
b   5
dtype: int64
```

Series的数据对齐

Source

```
>>> data = {'apple':6, 'peach':8, 'lemon':15}
>>> sindex = ['apple', 'peach', 'orange', 'lemon']
>>> dSer = pd.Series(data, index = sindex)
>>> dSer
apple      6.0
peach      8.0
orange     NaN
lemon     15.0
dtype: float64
```

Source

```
>>> pd.isnull(dSer)
apple     False
peach     False
orange     True
lemon     False
dtype: bool
```

Series的数据对齐

- **重要功能**
  - 在算术运算中自动对齐不同索引的数据

Source

```
>>> eSer = pd.Series({'apple':6, 'peach':8, 'kiwi':20})
>>> eSer
apple     6
peach     8
kiwi     20
dtype: int64
>>> dSer + eSer
apple     12.0
kiwi       NaN
lemon      NaN
orange     NaN
peach     16.0
dtype: float64
```

10.3.2 DataFrame

DataFrame

- **基本特征**
  - 一个表格型的数据结构(称数据框)
  - 含有一组有序的列(类似于index)
  - 大致可看成共享同一个index的Series集合

	name	pay
0	Mayue	3000
1	Lilin	4500
2	Wuyun	8000

1.创建和修改DataFrame

DataFrame()函数

Source

```
>>> data = {'name': ['Mayue', 'Lilin', 'Wuyun'], 'pay': [3000, 4500, 8000]}
>>> aDF = pd.DataFrame(data)
>>> aDF
     name   pay
0   Mayue  3000
1   Lilin  4500
2   Wuyun  8000
```

DataFrame的索引和值

Source

```
>>> data = np.array([('Mayue', 3000), ('Lilin', 4500), ('Wuyun', 8000)])
>>> bDF =pd.DataFrame(data, index = range(1, 4), columns = ['name', 'pay'])
>>> bDF
    name   pay
1   Mayue  3000
2   Lilin  4500
3   Wuyun  8000
>>> bDF.index   # 重新赋值即为修改行索引
RangeIndex(start=1, stop=4, step=1)
>>> bDF.columns   # 重新赋值即为修改列索引
Index(['name', 'pay'], dtype='object')
>>> bDF.values
array([['Mayue', '3000'],
       ['Lilin', '4500'],
       ['Wuyun', '8000']], dtype=object)
```

设定DataFrame的索引

Source

```
>>> bDF = pd.DataFrame(data, columns = ['name', 'pay']).set_index('name')
>>> bDF
          pay
name
Mayue     3000
Lilin     4500
Wuyun     8000
>>> bDF = pd.DataFrame(data, index = range(1, 4), columns = ['name', 'pay'])
>>> bDF.set_index('name', inplace = True)   # 默认drop参数的值为True，表示删掉原数据列
>>> bDF
          pay
name
Mayue     3000
Lilin     4500
Wuyun     8000
```

设定DataFrame的索引

Source

```
>>> bDF = pd.DataFrame(data, index = range(1, 4), columns = ['name', 'pay'])
>>> bDF
     name    pay
1   Mayue   3000
2   Lilin   4500
3    Wuyun   8000
>>> index_new = [1, 3, 2]
>>> bDF = bDF.reindex(index = index_new)
>>> bDF
     name    pay
1   Mayue   3000
3   Wuyun   8000
2    Lilin   4500
>>> columns_new = ['pay','name']
>>> bDF = bDF.reindex(columns = columns_new)
>>> bDF
     pay    name
1   3000   Mayue
3   8000   Wuyun
2   4500   Lilin
```

修改DataFrame-添加列

```
>>> aDF
      name   pay
0    Mayue  3000
1     Lilin  4500
2   Wuyun  8000
```

Source

```
>>> aDF['tax'] = [0.05, 0.05, 0.1]
>>> aDF
      name   pay   tax
0    Mayue  3000  0.05
1     Lilin  4500  0.05
2   Wuyun  8000   0.1
```

修改DataFrame-添加行

```
>>> aDF
      name   pay   tax
0    Mayue  3000  0.05
1    Lilin  4500  0.05
2   Wuyun   8000   0.1
```

Source

```
>>> aDF.loc[5] = {'name': 'Liuxi', 'pay': 5000, 'tax': 0.05}
>>> aDF
     name   pay   tax
0   Mayue  3000  0.05
1   Lilin  4500  0.05
2  Wuyun   8000   0.1
5   Liuxi  5000  0.05
```

修改DataFrame-添加行

```
>>> aDF
      name   pay   tax
0    Mayue  3000  0.05
1    Lilin  4500  0.05
2    Wuyun  8000   0.1
5    Liuxi  5000  0.05
>>> tempDF
      name   pay   tax
7    Yeqing  7000   0.1
9   Qianjie  9500   0.1
```

Source

```
>>> aDF.append(tempDF)
      name   pay   tax
0    Mayue  3000  0.05
1    Lilin  4500  0.05
2    Wuyun  8000   0.1
5    Liuxi  5000  0.05
7    Yeqing  7000   0.1
9   Qianjie  9500   0.1
```

修改DataFrame-添加行

```
>>> aDF
      name   pay   tax
0    Mayue  3000  0.05
1    Lilin  4500  0.05
2    Wuyun  8000   0.1
5    Liuxi  5000  0.05
>>> tempDF
      name   pay   tax
7    Yeqing  7000   0.1
9   Qianjie  9500   0.1
```

Source

```
>>> pieces = [aDF, tempDF]
>>> pd.concat(pieces)
      name   pay   tax
0    Mayue  3000  0.05
1    Lilin  4500  0.05
2    Wuyun  8000   0.1
5    Liuxi  5000  0.05
7    Yeqing  7000   0.1
9   Qianjie  9500   0.1
```

删除

```
>>> aDF
   name   pay   tax
0 Mayue  3000  0.05
1  Lilin  4500  0.05
2 Wuyun  8000   0.1
5 Liuxi  5000  0.05
```

Source

```
>>> aDF.drop(5)
   name   pay   tax
0 Mayue  3000  0.05
1  Lilin  4500  0.05
2 Wuyun  8000   0.1
>>> aDF.drop('tax', axis = 1)
   name   pay
0 Mayue  3000
1  Lilin  4500
2 Wuyun  8000
5 Liuxi  5000
```

inplace = True

修改DataFrame

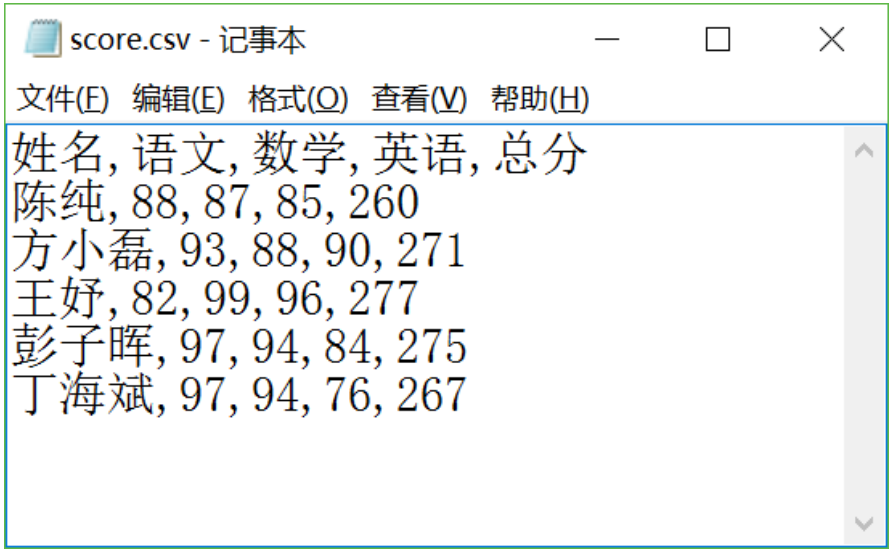
```
>>> aDF
   name   pay   tax
0 Mayue  3000  0.05
1  Lilin  4500  0.05
2 Wuyun  8000   0.1
5  Liuxi  5000  0.05
```

Source

```
>>> aDF['tax'] = 0.03
>>> aDF
   name   pay   tax
0 Mayue  3000  0.03
1  Lilin  4500  0.03
2 Wuyun  8000  0.03
5  Liuxi  5000  0.03
>>> aDF.loc[5] = ['Liuxi', 9800, 0.05]
   name   pay   tax
0 Mayue  3000  0.03
1  Lilin  4500  0.03
2 Wuyun  8000  0.03
5  Liuxi  9800  0.05
```

2.DataFrame数据文件存取

	A	B	C	D	E
1	姓名	语文	数学	英语	总分
2	陈纯	88	87	85	260
3	方小磊	93	88	90	271
4	王妤	82	99	96	277
5	彭子晖	97	94	84	275
6	丁海斌	97	94	76	267



score.csv

DataFrame数据存取-读csv文件

Source

```
>>> import pandas as pd
>>> data = pd.read_csv('score.csv', encoding = 'gb2312')
>>> data
```

	姓名	语文	数学	英语	总分
0	陈纯	88	87	85	260
1	方小磊	93	88	90	271
2	王妤	82	99	96	277
3	彭子晖	97	94	84	275
4	丁海斌	97	94	76	267

如何将姓名作为index?

DataFrame数据存取-写csv文件


```
# Filename: to_csv.py
import pandas as pd
df = pd.DataFrame(data)
df.to_csv('score_copy.csv')
```

DataFrame数据存取-读写excel文件


```
# Filename: excel_rw.py
import pandas as pd
df = pd.read_excel('score.xlsx')
df.to_excel('score.xlsx', sheet_name = 'score')
```

DataFrame数据存取-读写excel文件

	A	B	C	D	E	F
1	name	Math	Physics	English	Python	PE
2	stu1		73	92	82	95
3	stu2	92	95	88	96	85
4	stu3	90	92	93	95	90
5	stu4	83	93	86	85	60
6	stu5	87		70	93	75
7	stu6	95	87	90	98	80

Source

```
>>> import pandas as pd
>>> df = pd.read_excel('score.xlsx', index_col = 'name')
>>> df
        Math  Physics  English  Python  PE
name
stu1   NaN      73.0       92        82  95
stu2   92.0     95.0       88        96  85
stu3   90.0     92.0       93        95  90
stu4   83.0     93.0       86        85  60
stu5   87.0     NaN        70        93  75
stu6   95.0     87.0       90        98  80
```

3.数据预处理——数据清洗之重复值与缺失值处理

判断重复行 df.duplicated()

删除重复行 df.drop_duplicates([某列索引])

数据清洗——处理缺失值

	A	B	C	D	E	F
1	name	Math	Physics	English	Python	PE
2	stu1		73	92	82	95
3	stu2	92	95	88	96	85
4	stu3	90	92	93	95	90
5	stu4	83	93	86	85	60
6	stu5	87		70	93	75
7	stu6	95	87	90	98	80

Source

```
>>> df.dropna()    # 返回不包含NaN的行，df不变
      Math  Physics  English  Python   PE
name
stu2  92.0    95.0       88      96  85
stu3  90.0    92.0       93      95  90
stu4  83.0    93.0       86      85  60
stu6  95.0    87.0       90      98  80
```

数据预处理——处理缺失值

	A	B	C	D	E	F
1	name	Math	Physics	English	Python	PE
2	stu1		73	92	82	95
3	stu2	92	95	88	96	85
4	stu3	90	92	93	95	90
5	stu4	83	93	86	85	60
6	stu5	87		70	93	75
7	stu6	95	87	90	98	80

thresh参数

Source

```
>>> df.dropna(how = 'all')   # 返回不全为NaN的行，df不变
```

```
      Math  Physics  English  Python  PE
name
stu1  NaN   73.0     92       82      95
stu2  92.0  95.0     88       96      85
stu3  90.0  92.0     93       95      90
stu4  83.0  93.0     86       85      60
stu5  87.0  NaN      70       93      75
stu6  95.0  87.0     90       98      80
```

数据预处理——处理缺失值

	A	B	C	D	E	F
1	name	Math	Physics	English	Python	PE
2	stu1		73	92	82	95
3	stu2	92	95	88	96	85
4	stu3	90	92	93	95	90
5	stu4	83	93	86	85	60
6	stu5	87		70	93	75
7	stu6	95	87	90	98	80

Source

```
>>> df.fillna(0)        # 用0填充NaN
      Math  Physics  English  Python   PE
name
stu1   0.0     73.0       92      82  95
stu2  92.0     95.0       88      96  85
stu3  90.0     92.0       93      95  90
stu4  83.0     93.0       86      85  60
stu5  87.0      0.0       70      93  75
stu6  95.0     87.0       90      98  80
```

数据预处理——处理缺失值

	A	B	C	D	E	F
1	name	Math	Physics	English	Python	PE
2	stu1		73	92	82	95
3	stu2	92	95	88	96	85
4	stu3	90	92	93	95	90
5	stu4	83	93	86	85	60
6	stu5	87		70	93	75
7	stu6	95	87	90	98	80

Source

```
>>> df.fillna(df.mean()) # 用该列数据的平均值填充NaN
        Math  Physics  English  Python   PE
name
stu1  89.4    73.0     92       82  95
stu2  92.0    95.0     88       96  85
stu3  90.0    92.0     93       95  90
stu4  83.0    93.0     86       85  60
stu5  87.0    88.0     70       93  75
stu6  95.0    87.0     90       98  80
```

数据预处理——处理缺失值

	A	B	C	D	E	F
1	name	Math	Physics	English	Python	PE
2	stu1		73	92	82	95
3	stu2	92	95	88	96	85
4	stu3	90	92	93	95	90
5	stu4	83	93	86	85	60
6	stu5	87		70	93	75
7	stu6	95	87	90	98	80

Source

```
>>> df.fillna(method = 'bfill') # 指定缺失值的填充方向
      Math  Physics  English  Python   PE
name
stu1  92.0    73.0       92      82  95
stu2  92.0    95.0       88      96  85
stu3  90.0    92.0       93      95  90
stu4  83.0    93.0       86      85  60
stu5  87.0    87.0       70      93  75
stu6  95.0    87.0       90      98  80
```

3.数据预处理——处理缺失值

	A	B	C	D	E	F
1	name	Math	Physics	English	Python	PE
2	stu1		73	92	82	95
3	stu2	92	95	88	96	85
4	stu3	90	92	93	95	90
5	stu4	83	93	86	85	60
6	stu5	87		70	93	75
7	stu6	95	87	90	98	80

Source

```
>>>df.fillna(method = 'bfill',  inplace = True)
>>> df
      Math  Physics  English  Python   PE
name
stu1  92.0    73.0       92      82  95
stu2  92.0    95.0       88      96  85
stu3  90.0    92.0       93      95  90
stu4  83.0    93.0       86      85  60
stu5  87.0    87.0       70      93  75
stu6  95.0    87.0       90      98  80
```

3.数据预处理——数值规约之抽样

Source

```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> iris_df = pd.DataFrame(iris.data, columns = ['sepal length',
'sepal width', 'petal length', 'petal width'])
>>> iris_df['label'] = iris.target
>>> iris_df.sample(n = 20)    # 不放回随机抽样20条记录
>>> iris_df.sample(frac = 0.3)   # 不放回随机抽样30%的记录
```

4.数据选择

	code	name	lasttrade
0	MMM	3M	195.80
1	AXP	American Express	76.80
2	AAPL	Apple	153.06
3	BA	Boeing	180.76
4	CAT	Caterpillar	102.43
5	CVX	Chevron	106.52
6	CSCO	Cisco	31.21
7	KO	Coca-Cola	43.90
8	DIS	Disney	107.52
9	DD	E I du Pont de Nemours and Co	77.82
10	XOM	Exxon Mobil	81.93
11	GE	General Electric	28.05
12	GS	Goldman Sachs	215.39
13	HD	Home Depot	156.30
14	IBM	IBM	151.98
15	INTC	Intel	35.40
16	JNJ	Johnson & Johnson	127.00
17	JPM	JPMorgan Chase	84.78
18	MCD	McDonald's	148.15
19	MRK	Merck	63.78
20	MSFT	Microsoft	67.69
21	NKE	Nike	51.77
22	PFE	Pfizer	32.46
23	PG	Procter & Gamble	86.24
24	TRV	Travelers Companies Inc	120.79
25	UTX	United Technologies	121.16
26	UNH	UnitedHealth	172.59
27	VZ	Verizon	45.42
28	V	Visa	92.48
29	WMT	Wal-Mart	78.77

选择方式

- 选择行
- 选择列
- 选择区域
- 筛选(条件选择)

	close	high	low	open	volume
2016-05-23	63.590000	64.099998	63.560001	63.860001	3074100
2016-05-24	64.870003	65.099998	63.790001	63.790001	3946100
2016-05-25	65.309998	65.760002	65.010002	65.040001	5755900
2016-05-26	65.230003	65.370003	64.949997	65.290001	3593500
2016-05-27	65.519997	65.699997	65.330002	65.389999	3925700
2016-05-31	65.760002	65.919998	65.400002	65.699997	5256000
2016-06-01	65.910004	65.959999	65.180000	65.760002	3816000
2016-06-02	66.410004	66.410004	65.599998	65.860001	3052200
2016-06-03	65.489998	65.820000	64.769997	65.529999	4336100
2016-06-06	65.940002	66.199997	65.500000	65.550003	3915200
2016-06-07	65.889999	66.599998	65.879997	66.150002	3779500
2016-06-08	66.260002	66.580002	65.940002	65.940002	2601100
2016-06-09	65.709999	65.779999	64.900002	65.720001	3883800
2016-06-10	64.970001	65.480003	64.709999	65.260002	3939100
2016-06-13	63.669998	64.889999	63.630001	64.800003	5883400
2016-06-14	61.070000	63.660000	60.380001	63.590000	12323200
2016-06-15	61.419998	62.160000	60.860001	61.470001	5979900

DataFrame数据选择-选择行


```
>>> df
   姓名  语文  数学  英语  总分
a  陈纯   88   87   85  260
b 方小磊  93   88   90  271
c  王妤   82   99   96  277
d 彭子晖  97   94   84  275
e 丁海斌  97   94   76  267
```

- **选择行**
  - 花式索引
  - 切片(基于索引/位置)
  - 专门的方法

Source

```
>>> df[['a', 'b', 'c']]
>>> df['a': 'c']
>>> df[0: 3]
>>> df.head(3)
```

DataFrame数据选择-选择列


```
>>> df
     姓名  语文  数学  英语  总分
a    陈纯  88    87    85    260
b  方小磊  93    88    90    271
c    王妤  82    99    96    277
d  彭子晖  97    94    84    275
e  丁海斌  97    94    76    267
```

- **选择列**
  - 列名

Source

```
>>> df['姓名']
>>> df.姓名
```

不支持
df['姓名', '语文']
df['语文': '英语']

支持
df[['姓名', '语文']]

DataFrame数据选择-选择区域


```
>>> df
     姓名  语文  数学  英语  总分
a    陈纯  88  87  85  260
b  方小磊  93  88  90  271
c    王妤  82  99  96  277
d  彭子晖  97  94  84  275
e  丁海斌  97  94  76  267
```

- **选择区域**
  - 标签(loc)
  - 位置(iloc)

Source

```
>>> df.loc['b': 'd', '语文': '英语']
>>> df.iloc[1: 4, 1: 4]
```

DataFrame数据选择-选择区域


```
>>> df
     姓名  语文  数学  英语  总分
a    陈纯  88    87    85    260
b  方小磊  93    88    90    271
c    王妤  82    99    96    277
d  彭子晖  97    94    84    275
e  丁海斌  97    94    76    267
```

- **选择区域-行或列**
  - 标签(loc)
  - 位置(iloc)

Source

```
>>> df.loc['a': 'c',]
>>> df.loc[:, ['语文', '数学']]
>>> df.iloc[:, [1, 2, 3]]
```

单独列索引用[]

常用基于索引处理数据方式

Source

```
>>> temp_index = df2.index
>>> temp_index
Index(['b', 'd', 'e'], dtype='object')
>>> df.loc[temp_index]
    姓名  语文  数学  英语   总分
b  方小磊   93   88   90   271
d  彭子晖   97   94   84   275
e  丁海斌   97   94   76   267
```

DataFrame数据选择-选择区域-重新索引


```
>>> df
    姓名  语文  数学  英语  总分
a   陈纯  88  87  85  260
b  方小磊  93  88  90  271
c   王妤  82  99  96  277
d  彭子晖  97  94  84  275
e  丁海斌  97  94  76  267
```

```
>>> df.reindex(columns = [ '姓名',
'数学', '语文', '英语','总分'] #不要总分
```

Source

```
>>> df.iloc[:, [0, 2, 1, 3, 4]]
    姓名  数学  语文  英语  总分
a  陈纯  87  88  85  260
b  方小磊  88  93  90  271
c  王妤  99  82  96  277
d  彭子晖  94  97  84  275
e  丁海斌  94  97  76  267
```

```
>>> df.reindex(columns = [ '姓名',
'数学', '语文', '英语','总分']
```

DataFrame数据选择-条件筛选


```
>>> df
     姓名  语文  数学  英语  总分
a    陈纯  88  87  85  260
b  方小磊  93  88  90  271
c    王妤  82  99  96  277
d  彭子晖  97  94  84  275
e  丁海斌  97  94  76  267
```

找出索引值在'b'~'d'之间(包括'b'和'd')并且数学成绩大于等于90的学生记录

Source

```
>>> df[(df.index >= 'b') & (df.index <= 'd') & (df.数学 >= 90)]
```

DataFrame数据选择-条件筛选

Source

```
>>> A = iris_df[iris_df.label == 0].sample(frac = 0.3)
>>> B = iris_df[iris_df.label == 1].sample(frac = 0.2)
>>> C = A.append(B)
>>> C
    sepal length  sepal width  petal length  petal width  label
48           5.3          3.7           1.5          0.2      0
32           5.2          4.1           1.5          0.1      0
...
62           6.0          2.2           4.0          1.0      1
59           5.2          2.7           3.9          1.4      1
91           6.1          3.0           4.6          1.4      1
>>> C.shape
(25, 5)
```

分层抽样

DataFrame数据选择-筛选


```
>>> df
     姓名  语文  数学  英语  总分
a   陈纯   88   87   85   260
b  方小磊   93   88   90   271
c    王妤   82   99   96   277
d  彭子晖   97   94   84   275
e  丁海斌   97   94   76   267
```

1. 查找陈纯和彭子晖的成绩记录；
2. 查找所有陈姓同学的成绩记录

Source

```
>>> df[df.姓名.isin(['陈纯', '彭子晖'])]
>>> df[df['姓名'].str.contains('^陈')]
```

10.3.3 Series和DataFrame数据统计与分析

数据统计与分析

Source

```
import pandas as pd
>>> dir(pd.Series)
[..., 'head', ..., 'index', ..., 'stack', 'std', ..., 'where', ...]
>>> dir(pd.DataFrame)
[..., 'head', ..., 'index', ..., 'stack', 'std', ..., 'to_csv', ...]
```

数据统计与分析-简单统计


```
>>> df
     姓名  语文  数学  英语  总分
a    陈纯  88  87  85  260
b  方小磊  93  88  90  271
c    王妤  82  99  96  277
d  彭子晖  97  94  84  275
e  丁海斌  97  94  76  267
```

Source

```
>>> df.mean()
语文    91.4
数学    92.4
英语    86.2
总分   270.0
dtype: float64
>>> df.数学.mean()
92.4
```

数据统计与分析-排序

Source

```
>>> df.sort_values(by = '总分')
    姓名  语文  数学  英语   总分
a   陈纯    88    87    85   260
e  丁海斌    97    94    76   267
b  方小磊    93    88    90   271
d  彭子晖    97    94    84   275
c   王妤    82    99    96   277
>>> df.sort_values(by = '总分')[:3].姓名
a   陈纯
e  丁海斌
b  方小磊
Name: 姓名, dtype: object
```

数据统计与分析-简单统计

统计数学成绩大于等于90的学生每门课程(包括总分)的平均值

统计总分大于等于270的学生人数

Source

```
>>> df[(df.数学 >= 90)].mean()
语文    92.000000
数学    95.666667
英语    85.333333
总分   273.000000
dtype: float64
>>> len(df[(df.总分 >= 270)])
3
```

数据统计与分析-describe和info

Source

```
>>> df.describe()
              语文        数学        英语         总分
count   5.000000   5.000000   5.00000    5.00000
mean   91.400000  92.400000  86.20000  270.00000
std     6.426508   4.929503   7.42967    6.78233
min    82.000000  87.000000  76.00000  260.00000
25%    88.000000  88.000000  84.00000  267.00000
50%    93.000000  94.000000  85.00000  271.00000
75%    97.000000  94.000000  90.00000  275.00000
max    97.000000  99.000000  96.00000  277.00000
>>> df.info()
...
```

数据统计与分析- value_counts

Source

```
>>> df.数学.value_counts()
94    2
88    1
99    1
87    1
Name: 数学, dtype: int64
```

数据统计与分析-分组

按总分是否大于等于270为界将等级分为A和B两级

Source

```
>>> mark = ['A' if item >= 270 else 'B' for item in df.总分]
>>> df.姓名.groupby(mark).count()
等级
A    3
B    2
Name: 姓名, dtype: int64
```

```
# 或将等级列添加到数据中
>>> df['等级'] = mark
>>> df.groupby('等级').姓名.count()
```

.groups属性

数据统计与分析-分组

Source

```
>>> df.groupby('等级').mean()
        语文        数学    英语        总分
等级
A    90.666667   93.666667   90.0   274.333333
B    92.500000   90.500000   80.5   263.500000
>>> df.groupby('等级')[['语文', '英语']].mean()
        语文    英语
等级
A    90.666667   90.0
B    92.500000   80.5
```

数据统计与分析-分组

Source

```
>>> df.groupby('等级')[['语文', '英语']].agg(['mean', 'max'])
          语文       英语
        mean max  mean max
等级
A   90.666667  97  90.0  96
B   92.500000  97  80.5  85
>>> df.groupby('等级').agg({'语文': ['mean', 'max'], '英语': ['min']})
语文     英语
        mean max    min
等级
A   90.666667  97    84
B   92.500000  97    76
```

多函数聚合agg([m1, m2])方法

数据统计与分析-分组

Source

```
>>> def ranging(x):
        return x.max() - x.min()
>>> df.groupby('等级')[['语文', '英语']].agg(['mean', 'max', ranging])
 语文              英语
       mean max ranging  mean max ranging
等级
A   90.666667  97      15  90.0  96      12
B   92.500000  97       9  80.5  85       9
```

数据统计与分析-groupby&apply

apply方法可对DataFrame对象进行操作，既可作用于行或列，也可作用于Series的每一个元素上

Source

```
>>> df.groupby('等级').apply(len)
>>> df.loc[:, ['a', 'c']].apply(np.float32)
     a    c
0  3.0  1.0
1  1.0  3.0
2  8.0  1.0
3  2.0  2.0
```

数据统计与分析-透视

- 从不同层面查看表格数据的汇总信息
- pandas中常用pivot_table()函数来对数据进行透视实现分组的功能

Source

```
>>> pd.pivot_table(df, index = '等级')
          总分       数学    英语       语文
等级
A    274.333333   93.666667   90.0   90.666667
B    263.500000   90.500000   80.5   92.500000
>>> pd.pivot_table(df, values=['语文', '英语'], index = '等级',
aggfunc = [np.mean, max])
     mean          max
     英语       语文   英语  语文
等级
A    90.0  90.666667    96   97
B    80.5  92.500000    85   97
```

数据统计与分析-透视

Source

```
>>> data = pd.DataFrame({'A': ['one', 'two', 'one', 'two', 'one', 'two', 'one', 'one'], 'B': ['red',
'blue', 'white', 'red', 'red', 'blue', 'white', 'blue'], 'C': [54, 66, 78, 35, 98, 102, 105, 97]})
>>> data
    A     B    C
0  one    red   54
1  two   blue   66
2  one  white   78
...
>>> pd.pivot_table(data, values = 'C', index = 'A', columns = 'B', fill_value = 0)
>>> pd.pivot_table(data, values = 'C', index = 'A', columns = 'B', fill_value = 0, margins = True)
```

columns参数用于设置列层次的属性

数据统计与分析-合并

- 纵向连接(合并)：df1.append(df2)和pd.concat(df1, df2)
- 横向连接(合并)：pd.merge()

```
pd.merge(df_left, df_right, how='inner', on=None, left_on=None, right_on=None)
```

数据统计与分析-合并


```
>>> stus   # 学生信息表
   学号  姓名 性别  年龄
0  1001   王萌  女   18
1  1002  马小军 男    17
2  1003   张弛  男   19
3  1004   刘星  男   19
4  1005   张恒  女   18
>>> scores    # 学生成绩表
   学号  科目  成绩
0  1001  数学   98
1  1002  数学   87
2  1003  英语   85
3  1005  数学   92
4  1005  体育   85
```

```
>>> pd.merge(stus, scores)
   学号  姓名 性别  年龄  科目  成绩
0  1001   王萌   女   18   数学   98
1  1002  马小军 男    17   数学   87
2  1003   张弛   男   19   英语   85
3  1005   张恒   女   18   数学   92
4  1005   张恒   女   18   体育   85
```

数据统计与分析-corr

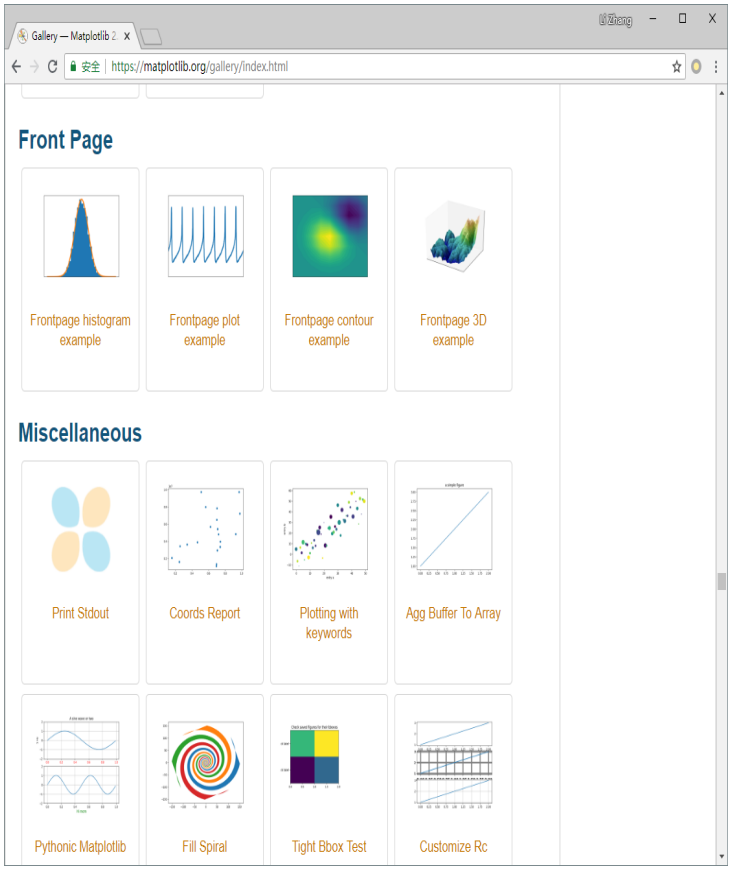
Source

```
>>> df.corr()
          语文      数学      英语      总分
语文  1.000000 -0.258843 -0.792723 -0.108978
数学 -0.258843  1.000000  0.236180  0.740275
英语 -0.792723  0.236180  1.000000  0.515971
总分 -0.108978  0.740275  0.515971  1.000000
```

10.4

Matplotlib及相关可视化包

Matplotlib绘图



- **Matplotlib绘图**

最著名Python绘图库，主要用于二维绘图

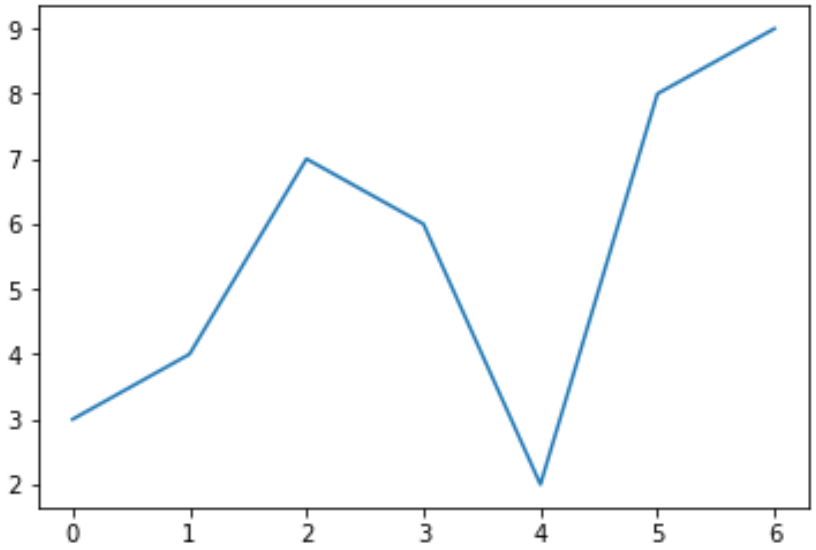
- 画图质量高
- 方便快捷的绘图模块
  - 绘图API——pyplot模块

10.4.1 Matplotlib绘图基本方法

折线图

Source

```
>>> import matplotlib.pyplot as plt
>>> plt.plot([3, 4, 7, 6, 2, 8, 9])
>>> plt.show()  # 阻塞/交互模式
```

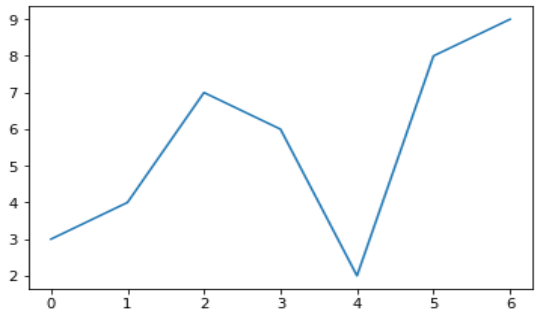



```
plt.plot(range(7), [3, 4, 7, 6, 2, 8, 9])
```

一些概念

- 画板(Figure)：默认生成一个编号为1的画板，一个画板中可包含多个坐标系
- 坐标系(Axes)：用于绘图，包括各种绘图元素(Artist)
- 手动创建一个画板并使用默认坐标系绘图
  plt.figure()
- 手动创建画板和坐标系，并设置它们的属性

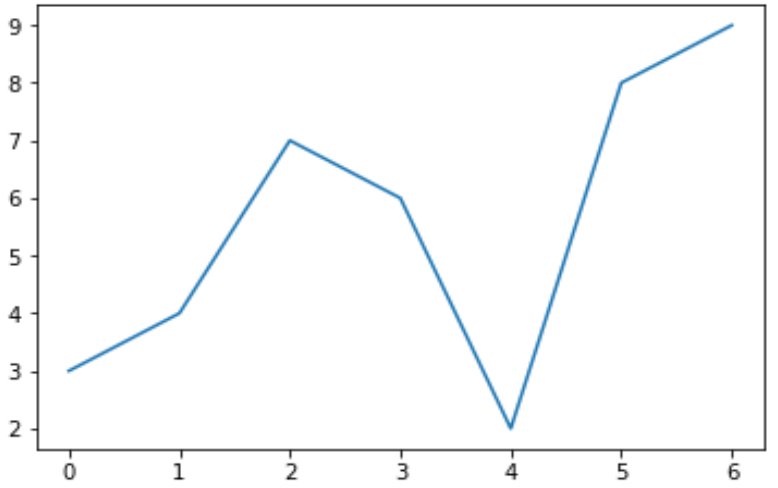
```
plt.figure(figsize=(6, 6), dpi=150)
plt.axes((0.1, 0.1, 0.8, 0.8))  # left、bottom、width、height
```



一些概念

- 使用多个画板绘制多张图(使用默认坐标系)

```
plt.figure(1)    # 可再次使用该语句调用此画板继续在其上绘图
plt.plot([3, 4, 7, 6, 2, 8, 9])
plt.figure(2)
plt.plot([3, 4, 8, 6, 2, 8, 5])
```

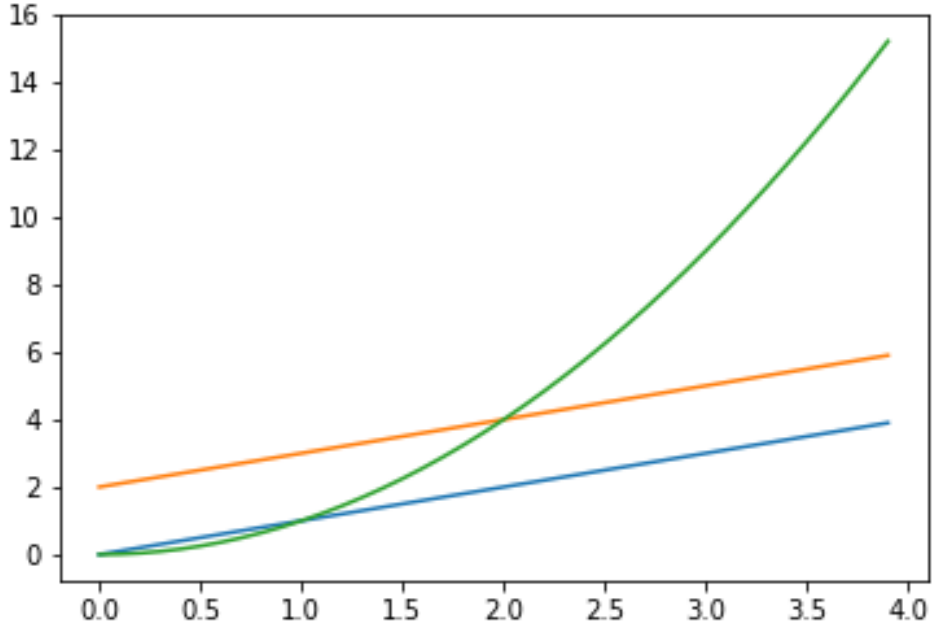


折线图-绘制多组数据

- NumPy数组也可以作为Matplotlib的参数
- 多组成对数据绘图

Source

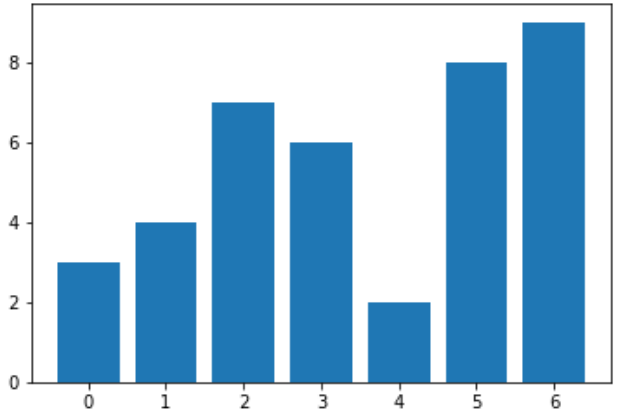
```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> t=np.arange(0.,4.,0.1)
>>> plt.plot(t, t, t, t+2, t, t**2)
```

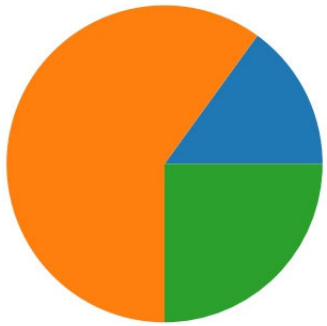


绘制其他类型的图

Source

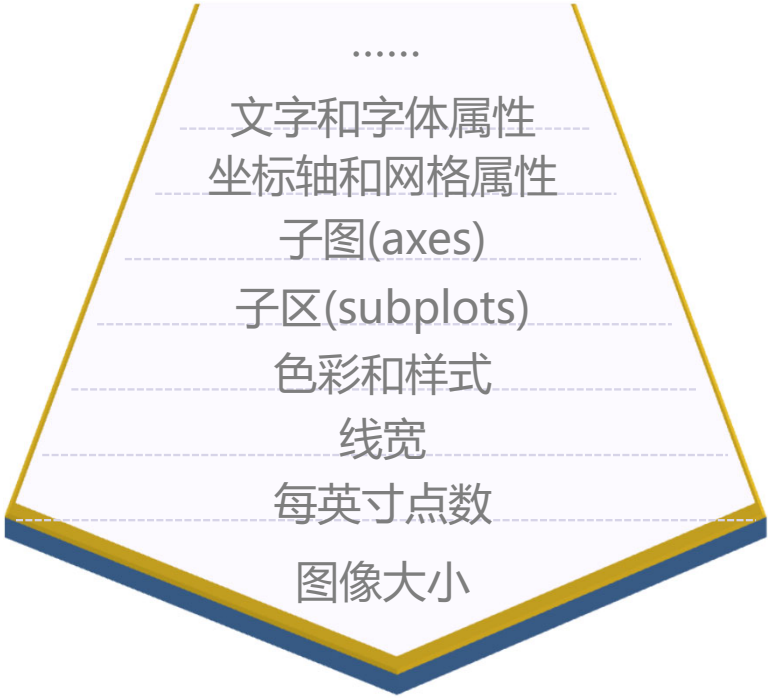
```
>>> import matplotlib.pyplot as plt
>>> plt.bar(range(7), [3, 4, 7, 6, 2, 8, 9])
>>> plt.pie(range(7), [3, 4, 7, 6, 2, 8, 9])
```





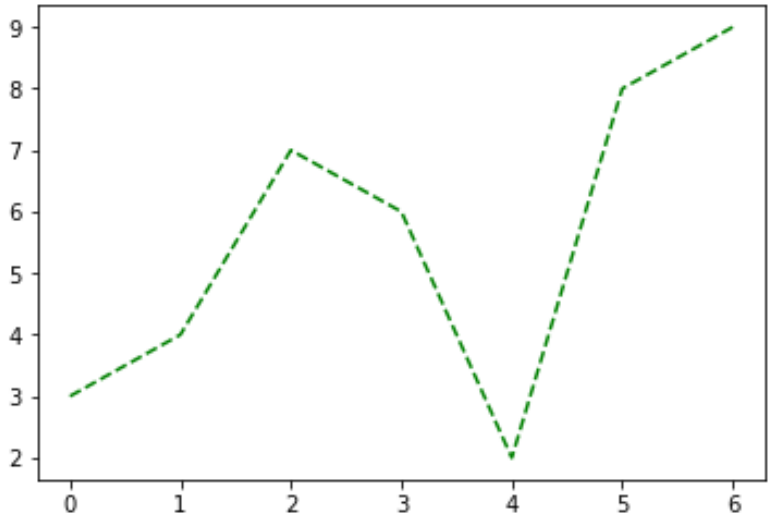
10.4.2 Matplotlib图形属性控制

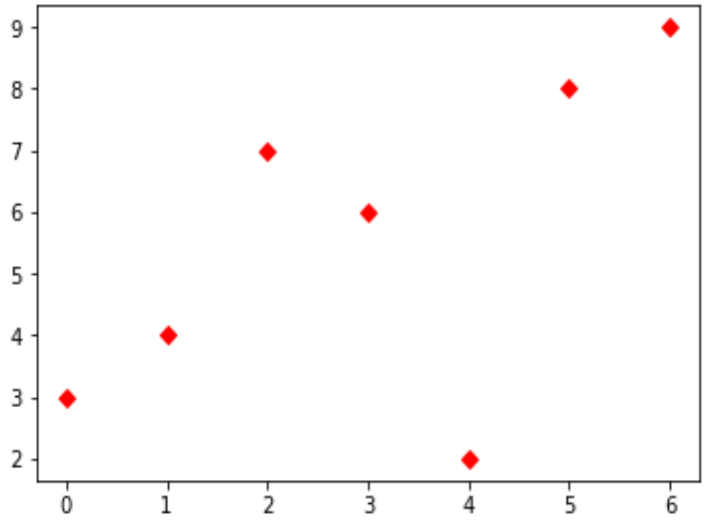
Matplotlib属性



Matplotlib可以控制的默认属性

色彩和样式






```
plt.plot(range(7), [3, 4, 7, 6, 2, 8, 9], 'g--')
plt.plot(range(7), [3, 4, 7, 6, 2, 8, 9], 'rD')
```

色彩和样式

符号	颜色
b	blue
g	green
r	red
c	cyan
m	magenta
Y	yellow
k	black
w	white

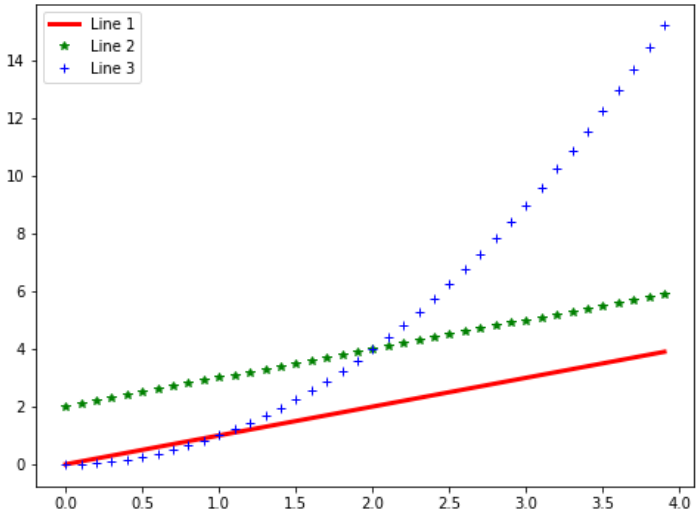
线型	描述
'-'	solid
'--'	dashed
'-.'	dash_dot
':'	dotted
'None'	draw nothing
' '	draw nothing
''	draw nothing

标记	描述
"o"	circle
"v"	triangle_down
"s"	square
"p"	pentagon
"*"	star
"h"	hexagon1
"+"	plus
"D"	diamond
…	…

多种属性

File

```
# Filename: prog10-2.py
import matplotlib.pyplot as plt
import numpy as np
plt.figure(figsize=(8,6),dpi=100)
t=np.arange(0.,4.,0.1)
plt.plot(t,t,color='red',linestyle='-',linewidth=3,label='Line 1')
plt.plot(t,t+2,color='green',linestyle='',marker='*',linewidth=3,label='Line 2')
plt.plot(t,t**2,color='blue',linestyle='',marker='+',linewidth=3,label='Line 3')
plt.legend(loc='upper left')
```

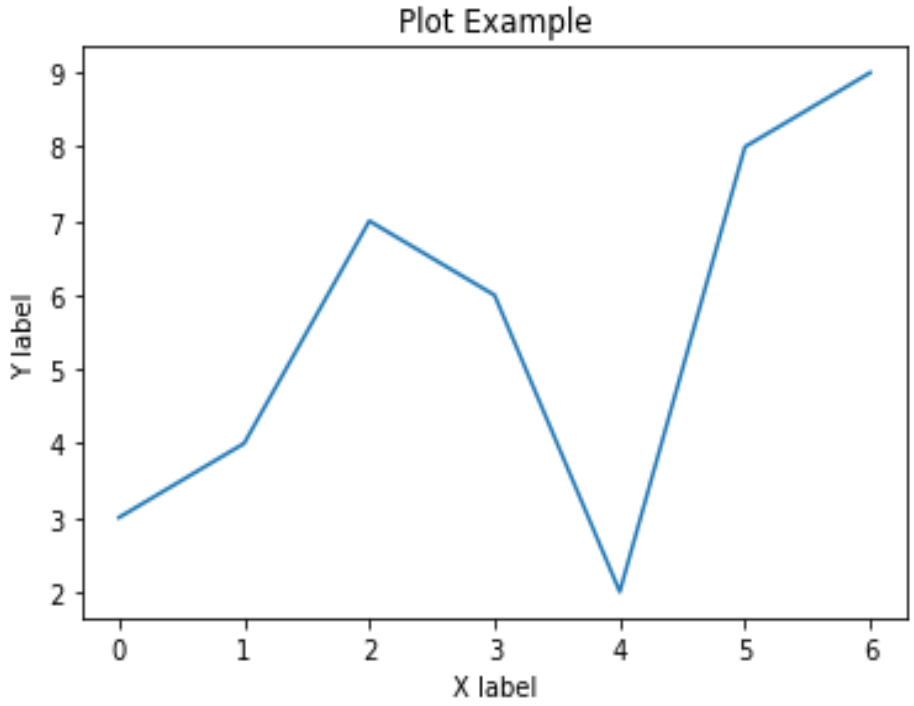


文字

加标题：图、横轴和纵轴

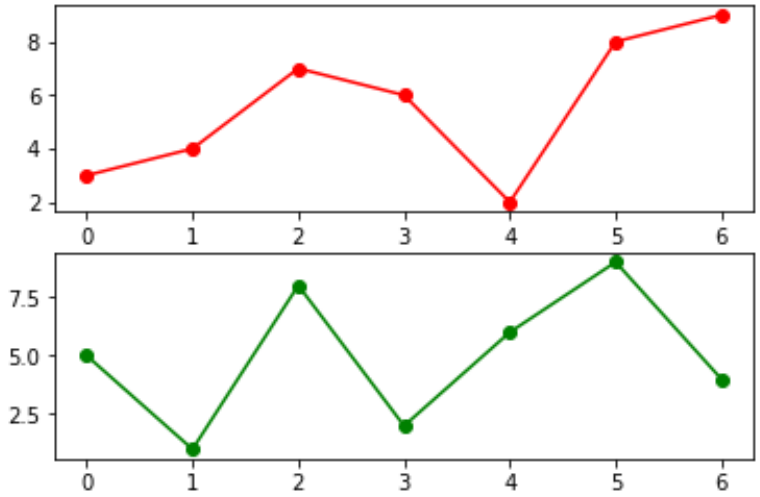
File

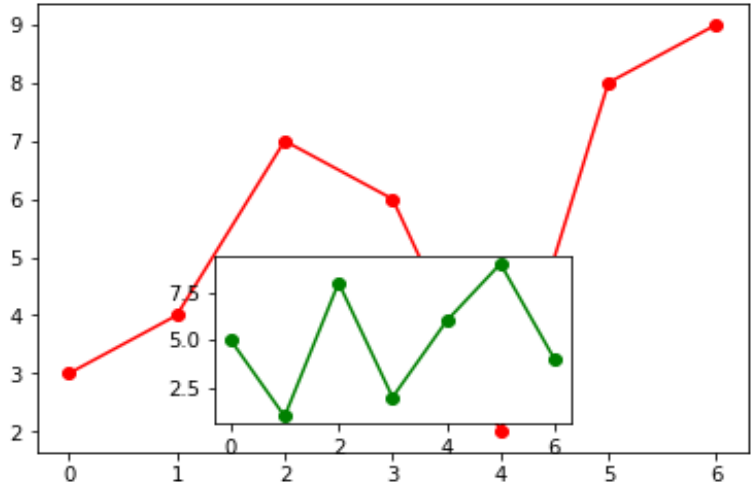
```
# Filename: prog10-3.py
import matplotlib.pyplot as plt
plt.title('Plot Example')
plt.xlabel('X label')
plt.ylabel('Y label')
plt.plot(range(7), [3, 4, 7, 6, 2, 8, 9])
```



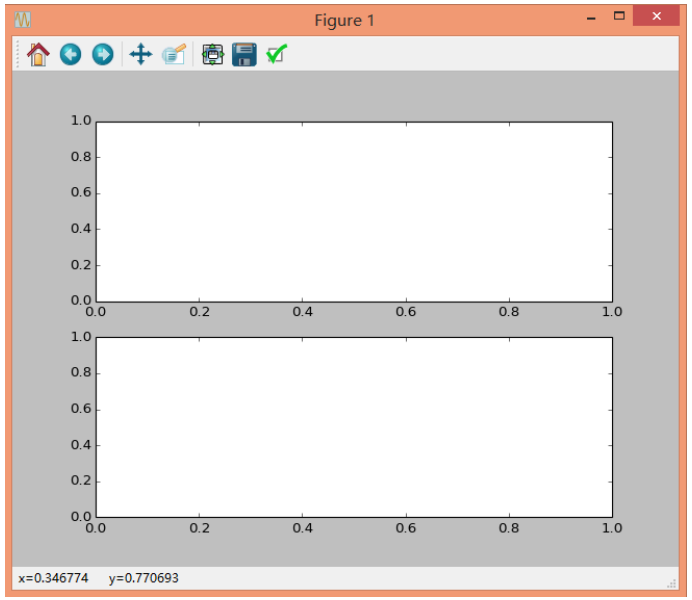
绘制子图

- 在Matplotlib中绘图在当前图形(figure)和当前坐标系(axes)中进行，默认在一个编号为1的figure中绘图，可以在一个图的多个区域分别绘图
- 使用subplot()函数和axes()函数

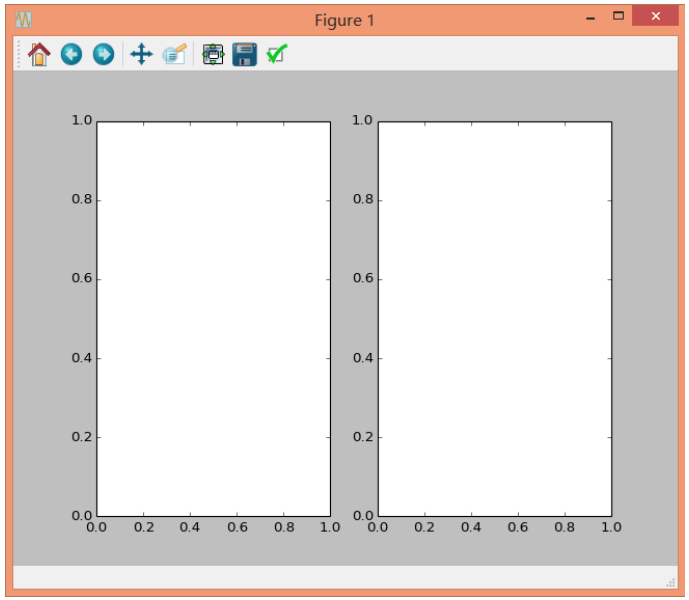




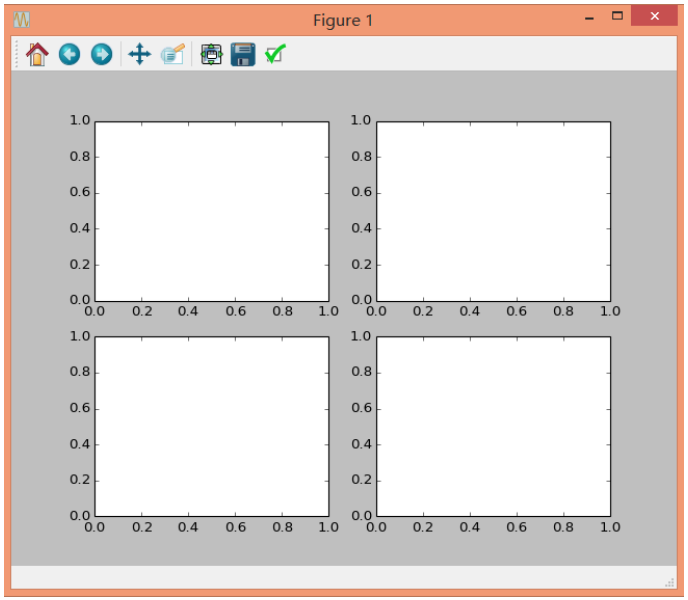
多子图-subplots



plt.subplot(211)
plt.subplot(212)



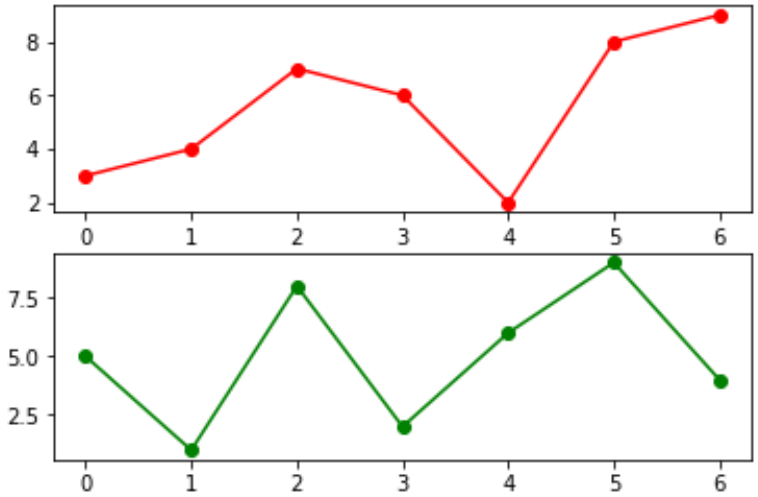
plt.subplot(121)
plt.subplot(122)



plt.subplot(221)
plt.subplot(222)
plt.subplot(223)
plt.subplot(224)

多子图-subplots

pyplot样式



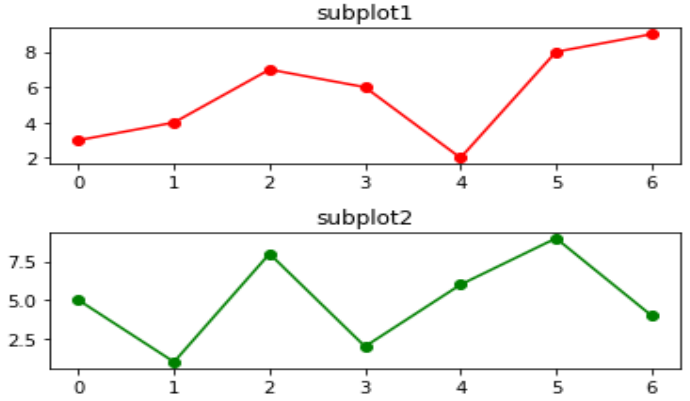
File

```
# Filename: Prog10-4.py
import matplotlib.pyplot as plt
plt.figure(1)        # 默认创建，缺省
plt.subplot(211)     # 第一个子图
plt.plot(range(7), [3, 4, 7, 6, 2, 8, 9], color = 'r', marker = 'o')
plt.subplot(212)     # 第二个子图
plt.plot(range(7), [5, 1, 8, 2, 6, 9, 4], color = 'green', marker = 'o')
```

多子图-subplots

File

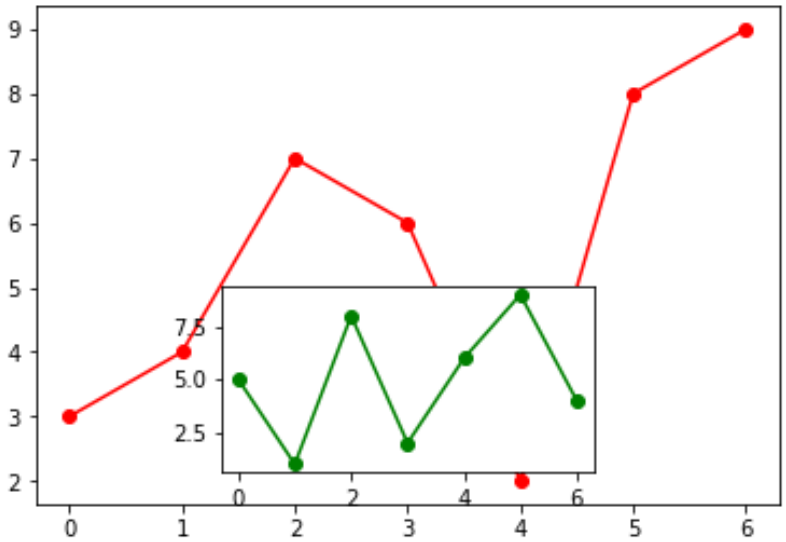
OO样式




```
# Filename: Prog10-5.py
import matplotlib.pyplot as plt
fig, (ax0, ax1) = plt.subplots(2, 1)
ax0.plot(range(7), [3, 4, 7, 6, 2, 8, 9], color = 'r', marker = 'o')
ax0.set_title('subplot1')
plt.subplots_adjust(hspace = 0.5)
ax1.plot(range(7), [5, 1, 8, 2, 6, 9, 4], color = 'green', marker = 'o')
ax1.set_title('subplot2')
```

子图-axes

axes([left,bottom,width,height])
参数范围为(0,1)



File

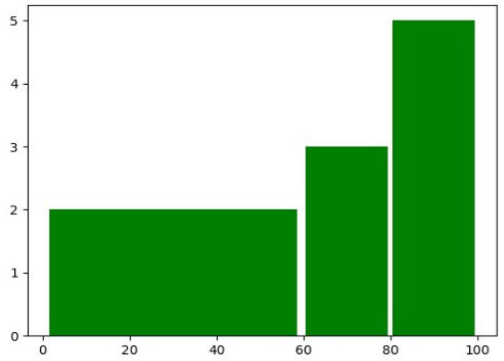
```
# Filename: Prog10-6.py
import matplotlib.pyplot as plt
plt.axes([.1, .1, 0.8, 0.8])
plt.plot(range(7), [3, 4, 7, 6, 2, 8, 9], color = 'r', marker = 'o')
plt.axes([.3, .15, 0.4, 0.3])
plt.plot(range(7), [5, 1, 8, 2, 6, 9, 4], color = 'green', marker = 'o')
```

10.4.3 常见类型图

直方图

plt.hist(x, [bins], [range])

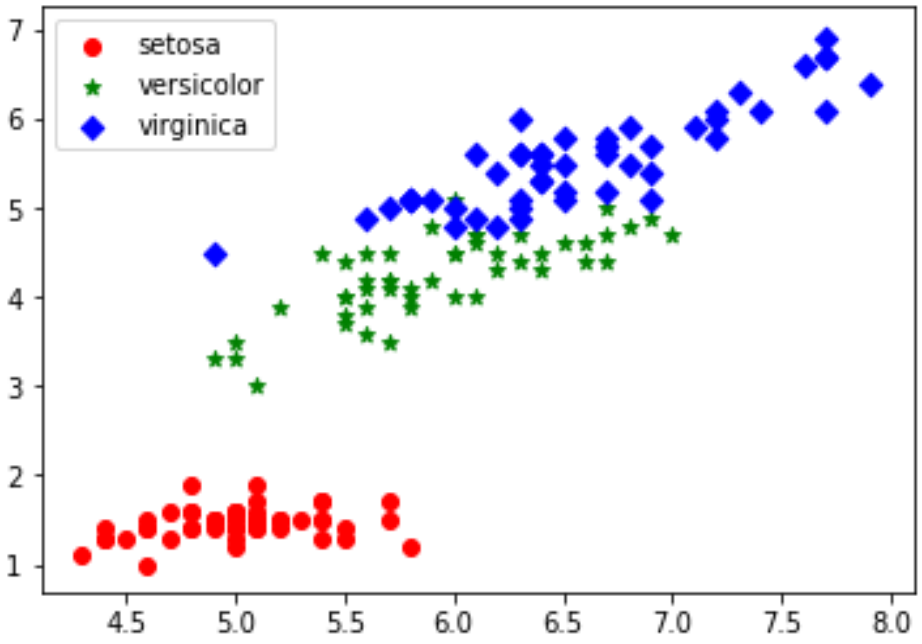
```
plt.hist(X, bins=5)    # 默认值为10
scores = np.array([86, 77, 50, 98, 95, 65, 80, 40, 61, 82])
bins = [0, 60, 80, 100]
plt.hist(scores, bins=bins, rwidth=0.95, color='g')
```



理解：

```
>>> plt.hist([1, 4, 5, 2, 6])
>>> plt.hist([1, 4, 5, 2, 6], bins=5, range=(1, 4))
```

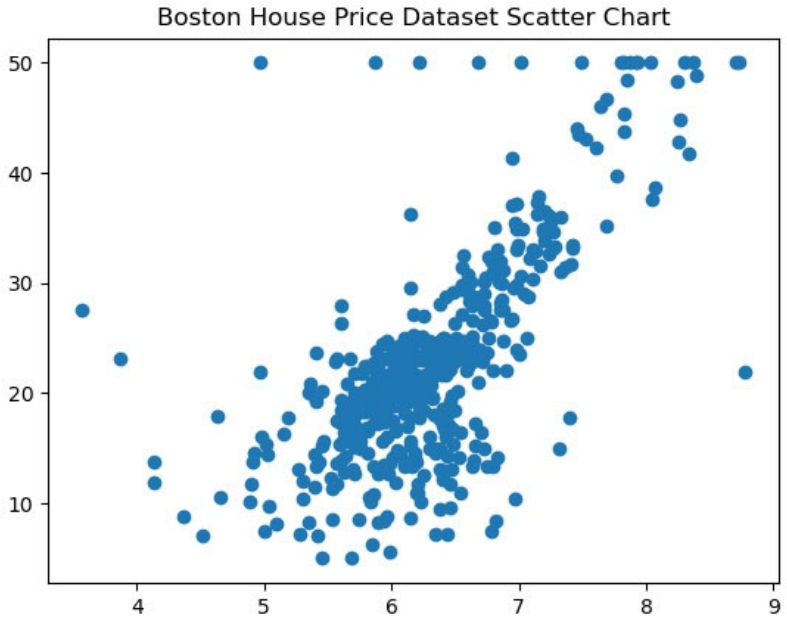
散点图




```
from sklearn import datasets
import matplotlib.pyplot as plt

iris = datasets.load_iris()
print(iris.data, iris.target)

X = [item[0] for item in iris.data]   # 获取萼片长度
Y = [item[2] for item in iris.data]   # 获取花瓣长度
plt.scatter(X[:50],Y[:50],color='red',marker='o',label='setosa')
plt.scatter(X[50:100],Y[50:100],color='green',marker='*',label='versi
color')
plt.scatter(X[-50:],Y[-50:],color='blue',marker='D',label='virginica')
plt.legend(loc='best')
plt.show()
```

数据探索阶段

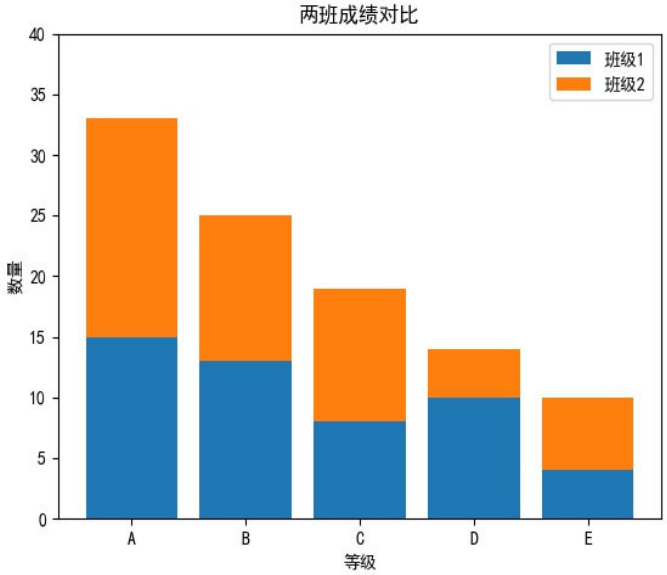
散点图



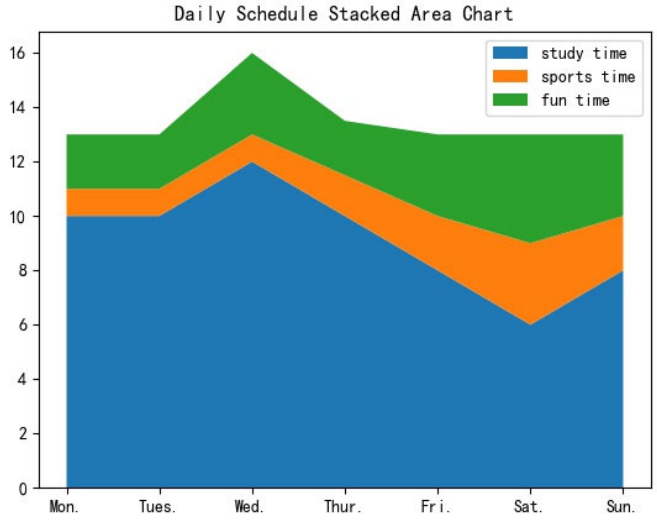

```
from sklearn import datasets
import matplotlib.pyplot as plt

boston = datasets.load_boston()
X = boston.data
y = boston.target
plt.scatter(X[:, 5], y)
plt.title('Boston House Price Dataset Scatter Chart')
plt.show()
```

堆积条形图


```
import matplotlib as mpl
import numpy as np
import matplotlib.pyplot as plt

 mpl.rcParams['font.sans-serif'] = ['SimHei']
mpl.rcParams['axes.unicode_minus'] = False
N = 5
ind = np.arange(N)
stu_counts1 = np.array([15, 13, 8, 10, 4])
stu_counts2 = np.array([18, 12, 11, 4, 6])
plt.bar(ind, stu_counts1, label = '班级1')
plt.bar(ind, stu_counts2, bottom = stu_counts1, label = '班级2')
plt.title('两班成绩对比')
plt.xlabel('等级')
plt.ylabel('数量')
plt.xticks(ind, ('A', 'B', 'C', 'D', 'E'))
plt.yticks(np.arange(0, 41, 5))
plt.legend(loc = 'best')
plt.show()
```



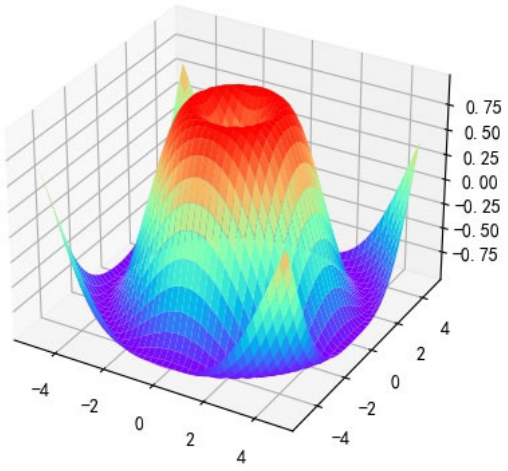
堆积面积图




```
import matplotlib.pyplot as plt

x = ['Mon.', 'Tues.', 'Wed.', 'Thur.', 'Fri.', 'Sat.',
'Sun.']
y1 = [10, 10, 12, 10, 8, 6, 8]
y2 = [1, 1, 1, 1.5, 2, 3, 2]
y3 = [2, 2, 3, 2, 3, 4, 3]
labels = ['study time', 'sports time', 'fun time']
plt.stackplot(x, y1, y2, y3, labels = labels)
plt.title('Daily Schedule Stacked Area Chart')
plt.legend(loc = 'best')
plt.show()
```

3D图

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D

X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

fig = plt.figure()
ax = plt.subplot(projection = '3d')
ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap='rainbow')
plt.show()
```

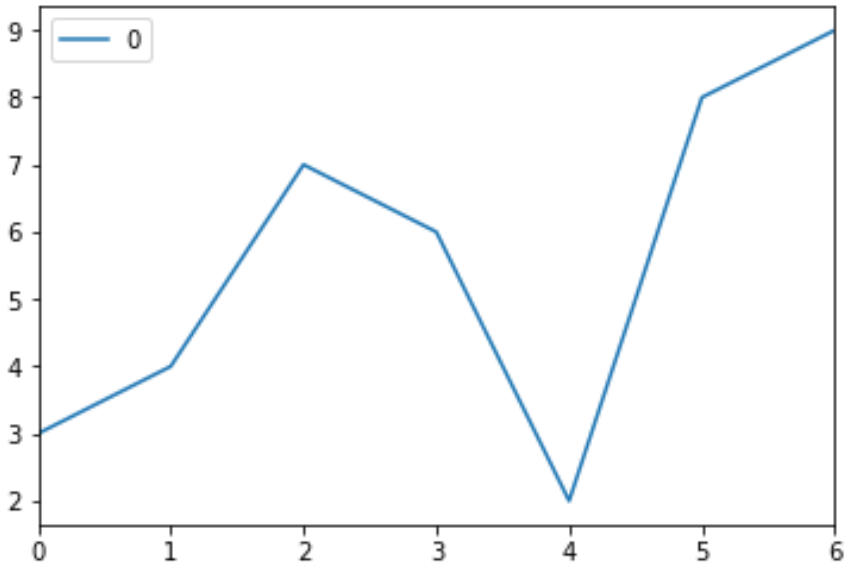


10.4.4 其他相关可视化库

pandas绘图

Source

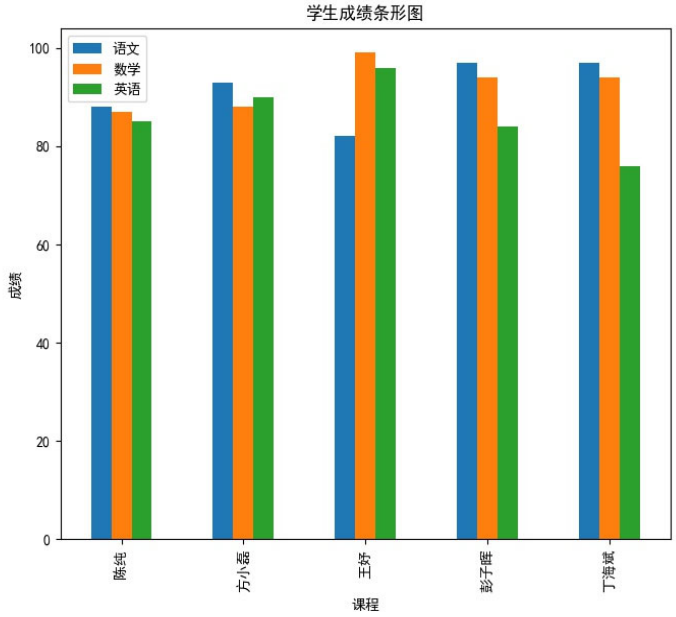
```
>>> import pandas as pd
>>> data = [3, 4, 7, 6, 2, 8, 9]
>>> pDF = pd.DataFrame(data)
>>> pDF.plot()
```



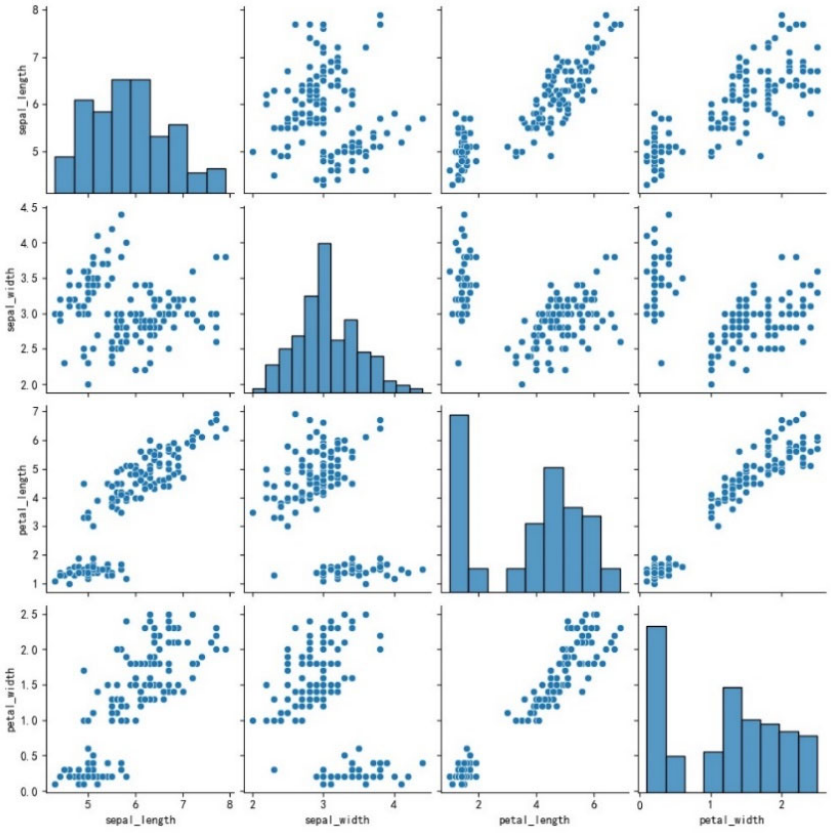
pandas绘图

File

```
# Filename: Prog10-15.py
import pandas as pd

df = pd.read_csv('score.csv', index_col = '
姓名', encoding = 'gb2312')
ax = df.iloc[:, :3].plot(kind = 'bar')
ax.set_title('学生成绩条形图')
ax.set(xlabel = '课程', ylabel = '成绩')
plt.show()
```

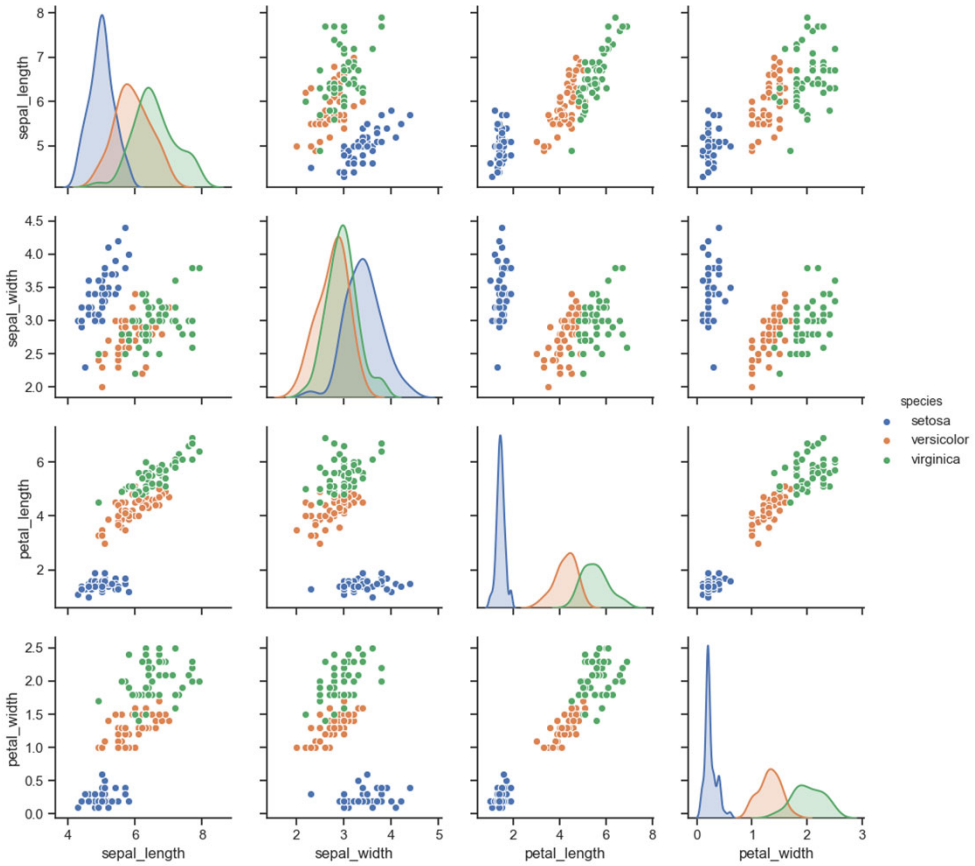


seaborn — 多变量图



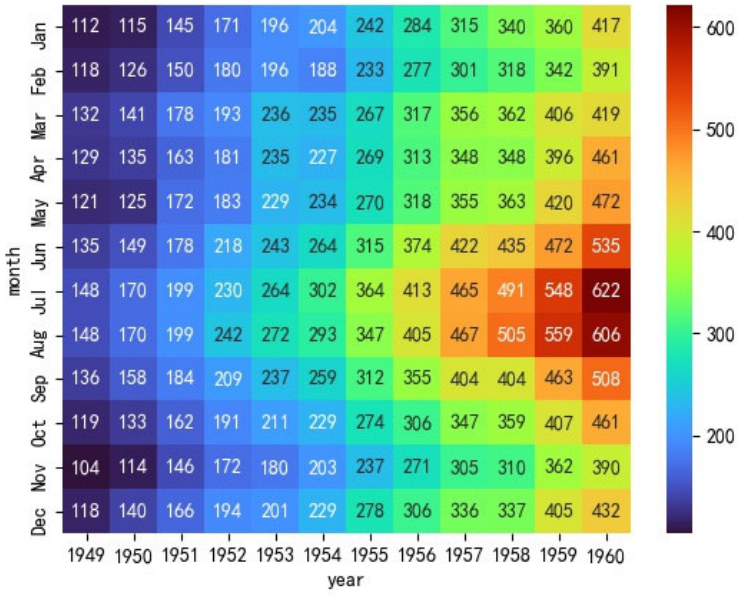

```
import seaborn as sns
import matplotlib.pyplot as plt

iris_df = sns.load_dataset('iris')
sns.pairplot(iris_df)
plt.show()
```

seaborn — 多变量图




```
import seaborn as sns

sns.set(style="ticks")

df = sns.load_dataset("iris")
sns.pairplot(df, hue="species")
plt.show()
```

seaborn — 热力图




```
import seaborn as sns
import matplotlib.pyplot as plt

flights_df = sns.load_dataset('flights')
flights = flights_df.pivot('month', 'year', 'passengers')
sns.heatmap(flights, annot = True, fmt = 'd', cmap =
'turbo')
plt.show()
```

10.5 小结

小结

- SciPy简介
- NumPy包
- pandas包
- Matplotlib及其他相关可视化包