

人工智能程序设计

M2 科学计算与数据分析基础

SciPy生态系统之核心包

张 莉



人工智能程序设计

0 软件生态系统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



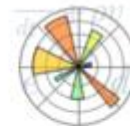
NumPy

Base N-dimensional array
package



SciPy library

Fundamental library for
scientific computing



Matplotlib

Comprehensive 2D Plotting



IPython

Enhanced Interactive Console



Sympy

Symbolic mathematics

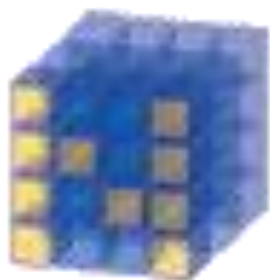


pandas

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

- 基于NumPy，是科学计算核心库
- 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

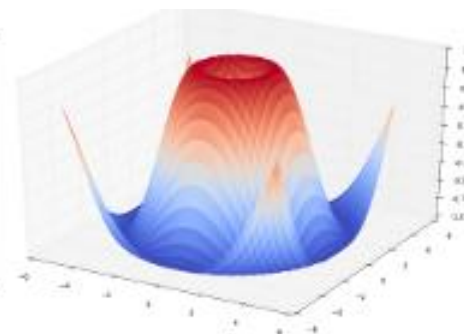
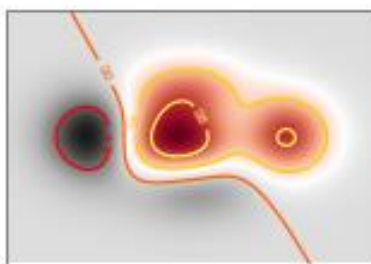
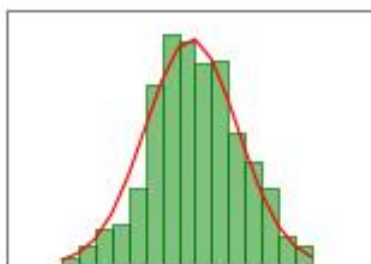
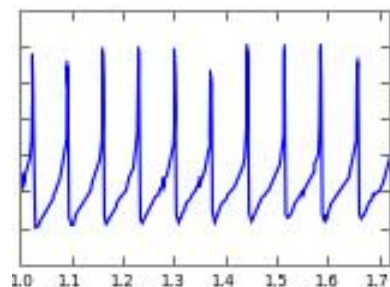



>>> import numpy as np
>>> from scipy import linalg
>>> arr = np.array([[1, 2], [3, 4]])
>>> linalg.det(arr)
-2.0

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



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

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)

1

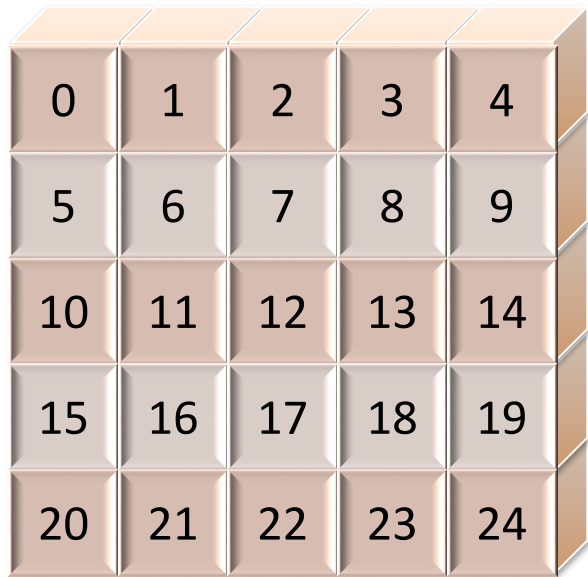
人工智能程序设计

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



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

- ndarray是什么?

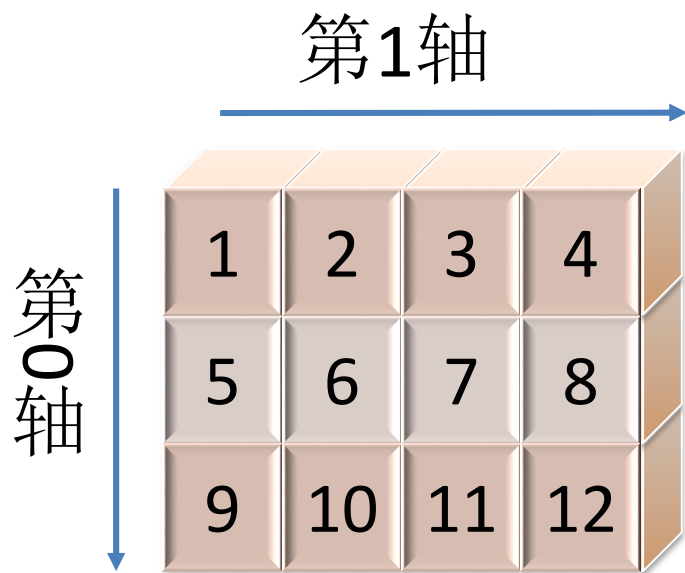
N维数组

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

NumPy

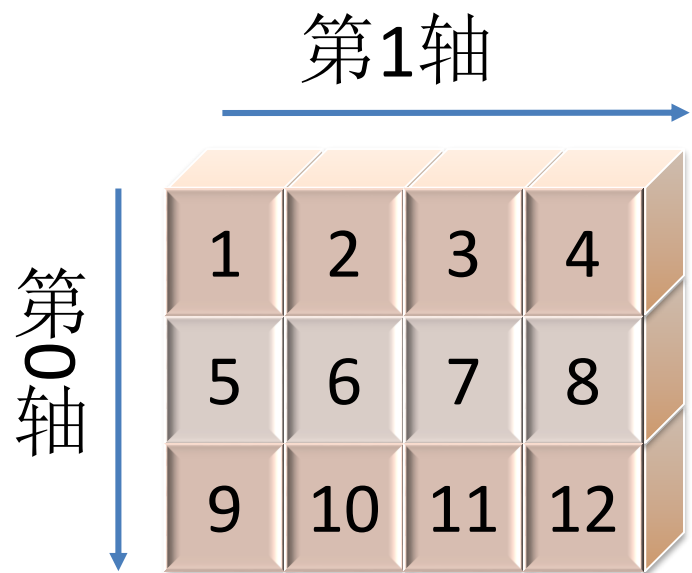
1. ndarray的基本特性
2. 创建ndarray
3. ndarray的操作与运算
4. ufunc函数
5. 专门的应用

ndarray基本概念



- ndarray数组属性
 - 维度(dimensions)称为轴(axes), 轴的个数称为秩(rank)
 - 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

- 基本属性

- ndarray.ndim (秩)
 - ndarray.shape (维度)
 - ndarray.size (元素总个数)
 - ndarray.dtype (元素类型)
 - ndarray.itemsize (元素字节大小)

NumPy

1. ndarray的基本特性
- 2. 创建ndarray**
3. ndarray的操作与运算
4. ufunc函数
5. 专门的应用

ndarray的创建

Source

array()函数

```
>>> import numpy as np
>>> aArray = np.array([1,2,3])
>>> aArray
array([1, 2, 3])
>>> bArray = np.array([(1,2,3),(4,5,6)])
>>> bArray
array([[1, 2, 3],
       [4, 5, 6]])
>>> bArray.ndim, bArray.shape, bArray.dtype
(2, (2, 3), dtype('int32'))
```

ndarray的创建

arange	array
copy	empty
empty_like	eye
fromfile	fromfunction
full	identity
linspace	logspace
mgrid	ogrid
ones	ones_like
r	zeros
zeros_like	...

ndarray创建函数

ndarray的创建

```
zeros()
ones()
full()
zeros_like()
ones_like()
full_like()
```



```
>>> np.zeros((2, 2))
array([[ 0.,  0.],
       [ 0.,  0.]])
>>> np.ones([2, 3])
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
>>> np.full((3, 3), np.pi)
...
>>> x = np.array([[1, 2, 3], [4, 5, 6]], dtype = np.float32)
>>> np.ones_like(x)
...
```

ndarray的创建

identity()
eye()

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.]])
>>> np.eye(3, k = 1)
...
```

ndarray的创建

arange()
linspace()



```
>>> 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

Source

empty()
random()
fromfunction()

```
>>> np.empty((2, 2))
array([[9.90263869e+067, 8.01304531e+262],
       [2.60801200e-310, 1.99392167e-077]])

>>> np.random.random((2, 2))
array([[ 0.797777004,  0.1468679 ],
       [ 0.95838379,  0.86106278]])

>>> 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

1. ndarray的基本特性
2. 创建ndarray
- 3. ndarray的操作与运算**
4. ufunc函数
5. 专门的应用

ndarray的基本操作-切片

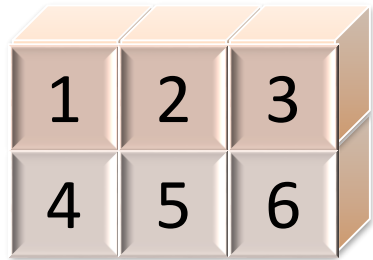


1	2	3
4	5	6

Source

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


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)
...
```

ndarray的基本操作-改变数组形状

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]])
```

Source

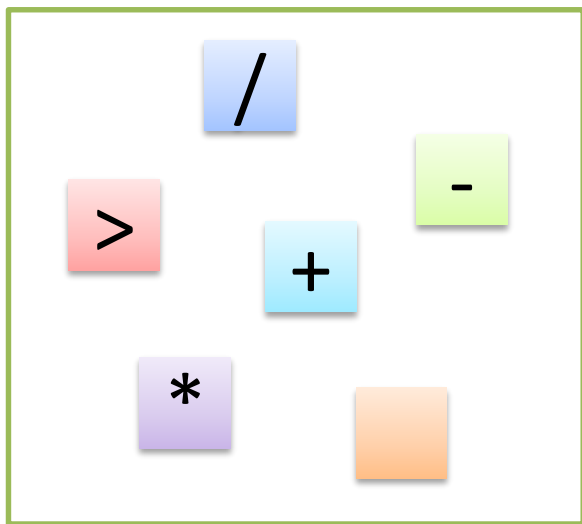
```
>>> aArray.resize(3,2)
>>> aArray
array([[1, 2],
       [3, 4],
       [5, 6]])
```

ndarray的基本操作-堆叠

A speech bubble icon containing the word "Source" in orange text.

```
>>> 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的运算



利用基本运算符

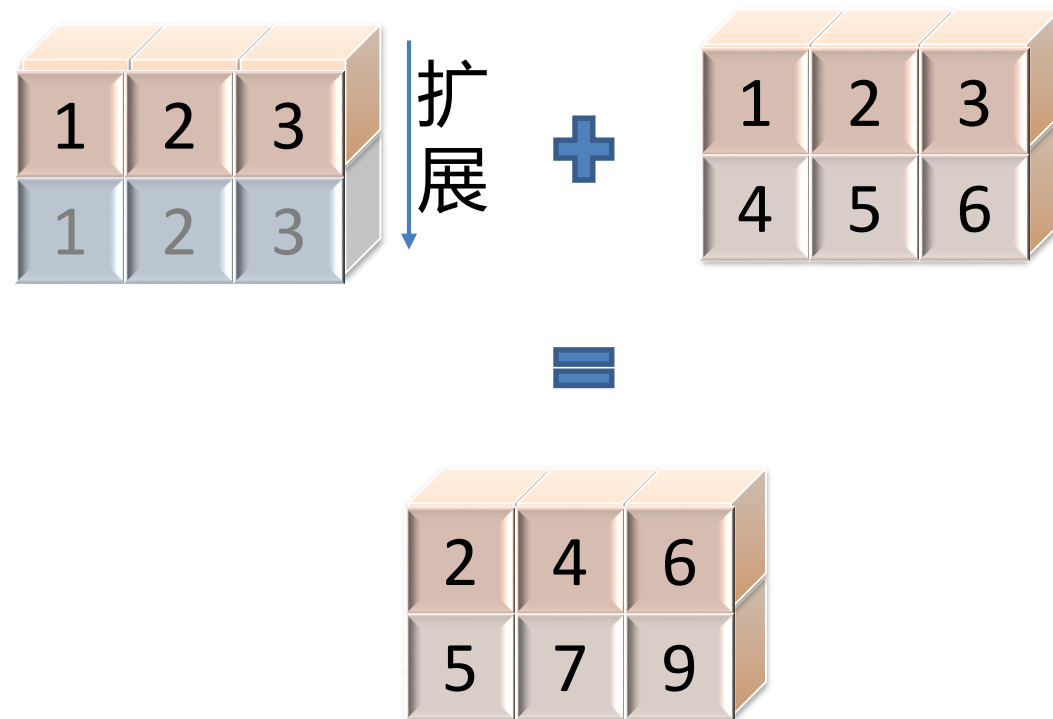
Source

```
>>> aArray = np.array([(5,5,5),(5,5,5)])
>>> bArray = np.array([(2,2,2),(2,2,2)])
>>> cArray = aArray * bArray
>>> cArray
array([[10, 10, 10],
       [10, 10, 10]])
>>> aArray += bArray
>>> aArray
array([[7, 7, 7],
       [7, 7, 7]])
```

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([[1,2,3],[4,5,6]])
>>> a + b
array([[2, 4, 6],
       [5, 7, 9]])
```

ndarray的运算—简单统计

Source

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

```
>>> aArray.sum()
```

```
21
```

```
>>> 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—统计

Source

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

```
>>> aArray.mean()
```

```
3.5
```

```
>>> aArray.var()
```

```
2.9166666666666665
```

```
>>> aArray.std()
```

```
1.707825127659933
```

sum	mean
std	var
min	max
argmin	argmax
cumsum	cumprod

利用基本数组统计方法

NumPy

1. ndarray的基本特性
2. 创建ndarray
3. ndarray的操作与运算
- 4. ufunc函数**
5. 专门的应用

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。

add, all, any, arange, apply_along_axis, argmax, argmin, argsort, average, bincount, ceil, clip, conj, corrcoef, cov, cross, cumprod, cumsum, diff, dot, exp, floor, ...

ndarray的ufunc函数

File

Filename: 1.py

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

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

NumPy

1. ndarray的基本特性
2. 创建ndarray
3. ndarray的操作与运算
4. ufunc函数
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]]
>>> r3 = np.dot(x, x)
>>> print(r3)
[[ 7 10]
 [15 22]]
```

Scipy中的
linalg
模块

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

人工智能程序设计

2 PANDAS

pandas

1. Series

2. DataFrame

3. 基于Series和DataFrame
的数据统计和分析


4. 金融数据包应用

Series

- 基本特征

- 类似一维数组的对象
- 由数据和索引组成（有序字典，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

A small orange speech bubble icon containing the word "Source" in orange text.

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



```
>>> 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
```

pandas

1. Series

2. DataFrame

3. 基于Series和DataFrame
的数据统计和分析

4. 金融数据包应用

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

创建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)
```

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

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

DataFrame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

- 选择行

- 索引
- 切片
- 专门的方法

```
>>> 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	王好	82	99	96	277
d	彭子晖	97	94	84	275
e	丁海斌	97	94	76	267

Source

```
>>> df['a': 'c']
```

```
>>> df[0: 3]
```

```
>>> df.head(3)
```

DataFrame数据选择-选择列

- 选择列
– 列名



```
>>> df['姓名']
```

```
>>> df.姓名
```

不支持

```
df['姓名', '语文']
```

```
df['语文': '英语']
```

```
>>> 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

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]]
```

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	王好	82	99	96	277
d	彭子晖	97	94	84	275
e	丁海斌	97	94	76	267

- 选择区域-单个值

- 标签 (loc或at)
- 位置 (iloc或iat)



```
>>> df.at['b', '数学']
```

```
>>> df.iat[1, 2]
```

ix-选择行

- **ix** 不推荐使用

loc和iloc的混合

```
>>> 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.ix['a'] # 或df.ix[0]
```

姓名 陈纯

语文 88

数学 87

英语 85

总分 260

Name: a, dtype: object

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)]
```

pandas

1. Series
2. DataFrame
- 3. 基于Series和DataFrame
的数据统计和分析**
4. 金融数据包应用

数据统计与分析



```
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	王好	82	99	96	277
d	彭子晖	97	94	84	275
e	丁海斌	97	94	76	267



```
>>> 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



```
>>> 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-简单统计与筛选

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



```
>>> 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-简单统计与筛选

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['等级'] = mark  
>>> df
```

	姓名	语文	数学	英语	总分	等级
a	陈纯	88	87	85	260	B
b	方小磊	93	88	90	271	A

...

```
>>> df.groupby('等级').姓名.count()
```

等级

A 3

B 2

```
Name: 姓名, dtype: int64
```

数据统计与分析

任务

- 产生数据的描述性信息
- 相关性分析



皮尔逊(Pearson)相关分析

$$r_{xy} = \frac{\sum (X - \bar{X})(Y - \bar{Y})}{(\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2})(\sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2})}$$

约束条件:

1. 两个变量间有线性关系
2. 均是连续变量
3. 变量均符合正态分布,且二元分布也符合正态分布
4. 两个变量独立

- [维基百科] 假设五个国家的国民生产总值分别是1、2、3、5、8（单位10亿美元），又假设这五个国家的贫困比例分别是11%、12%、13%、15%、18%。

x均值: 3.8

y均值: 0.138

$$(1-3.8)*(0.11-0.138)=0.0784$$

$$(2-3.8)*(0.12-0.138)=0.0324$$

$$(3-3.8)*(0.13-0.138)=0.0064$$

$$(5-3.8)*(0.15-0.138)=0.0144$$

$$(8-3.8)*(0.18-0.138)=0.1764$$

$$0.0784+0.0324+0.0064+0.0144+0.1764=0.308$$

$$0.308/(5.549775*0.05549775)=1$$

$$(1-3.8)^2=7.84$$

$$(2-3.8)^2=3.24$$

$$(3-3.8)^2=0.64$$

$$(5-3.8)^2=1.44$$

$$(8-3.8)^2=17.64$$

$$7.84+3.24+0.64+1.44+17.64=30.8$$

$$30.8^{0.5}=5.549775$$

$$0.00308^{0.5}=0.05549775$$

皮尔逊(Pearson)相关分析

$$\frac{\sum (X - \bar{X})(Y - \bar{Y})}{(\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2})(\sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2})}$$

```
x = np.array([1,2,3,5,8])  
y = np.array([0.11,0.12,0.13,0.15,0.18])  
x_mean = np.mean(x)  
y_mean = np.mean(y)
```

```
u = sum((x-x_mean)*(y-y_mean))    # np.dot(x-x_mean, y-y_mean)  
l = math.sqrt(sum((x-x_mean)**2))*math.sqrt(sum((y-y_mean)**2))  
print(u / l)
```

皮尔逊(Pearson)相关分析

```
import pandas as pd
```

```
x = [1,2,3,5,8]
```

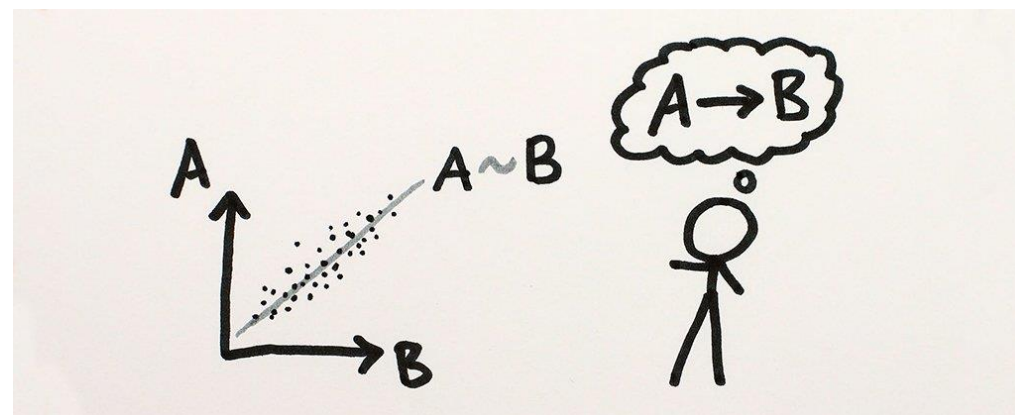
```
y = [0.11,0.12,0.13,0.15,0.18]
```

```
df = pd.DataFrame()
```

```
df['x'] = x
```

```
df['y'] = y
```

```
print(df.corr())
```



pandas

1. Series
2. DataFrame
3. 基于Series和DataFrame的数据统计和分析
4. 金融数据包应用

Python财经数据接口包Tushare

Tushare 0.4.3 documentation

Table Of Contents

- 前言
- 致谢
- 使用对象
- 使用前提
- 下载安装
- 版本升级
- 版本信息
- 友情链接
- 交易数据
- 历史行情
- 复权数据
- 实时行情
- 历史分笔
- 实时分笔
- 当日历史分笔
- 大盘指数行情列表
- 大单交易数据
- 投资参考数据
- 分配预案
- 业绩预告
- 限售股解禁
- 基金持股
- 新股数据
- 融资融券 (沪市)
- 融资融券 (深市)
- 股票分类数据
- 行业分类
- 概念分类
- 地域分类
- 中小板分类
- 创业板分类
- 风险警示板分类
- 沪深300成份及权重
- 上证50成份股
- 中证500成份股
- 终止上市股票列表
- 暂停上市股票列表
- 基本面数据

前言

Tushare是一个免费、开源的python财经数据接口包。主要实现对股票等金融数据从数据采集、清洗加工到数据存储的过程，能够为金融分析人员提供快速、整洁、和多样的便于分析的数据，为他们在数据获取方面极大地减轻工作量，使他们更加专注于策略和模型的研究与实现上。考虑到Python pandas包在金融量化分析中体现出的优势，Tushare返回的绝大部分的数据格式都是pandas DataFrame类型，非常便于用pandas/NumPy/Matplotlib进行数据分析和可视化。当然，如果您习惯了用Excel或者关系型数据库做分析，您也可以通过Tushare的数据存储功能，将数据全部保存到本地后进行分析。应一些用户的请求，从0.2.5版本开始，Tushare同时兼容Python 2.x和Python 3.x，对部分代码进行了重构，并优化了一些算法，确保数据获取的高效和稳定。

Tushare从发布到现在，已经帮助很多用户在数据方面降低了工作压力，同时也得到很多用户的反馈，Tushare将一如既往的用免费和开源的形式分享出来，希望对有需求的人带来一些帮助。如果您觉得Tushare好用并有所收获，请通过微博、微信或者网站博客的方式分享出去，让更多的人了解和使用它，使它能在大家的使用过程中逐步得到改进和提升。Tushare还在不断的完善和优化，后期将逐步增加港股、期货、外汇和基金方面的数据，所以，您的支持和肯定才是Tushare坚持下去的动力。

Tushare的数据主要来源于网络，如果在使用过程碰到数据无法获取或发生数据错误的情况请联系我，如果有什么好的建议和意见，也请及时联系我，在此谢过。如果在pandas/NumPy技术上有问题，欢迎加入“pandas数据分析”QQ群：297882961（已满），Tushare用户群：658562506，我会和大家一起帮忙为您解决。另外，请扫码关注“挖地兔”的微信公众号，定期会发布Tushare的最新动态及有价值的金融数据分析与处理方面的教程和文章。



从最新本开始，tushare将接受第三方数据的接入，欢迎供应商通过微信公众号“挖地兔”与我联系。

TUSHARE 功能概览



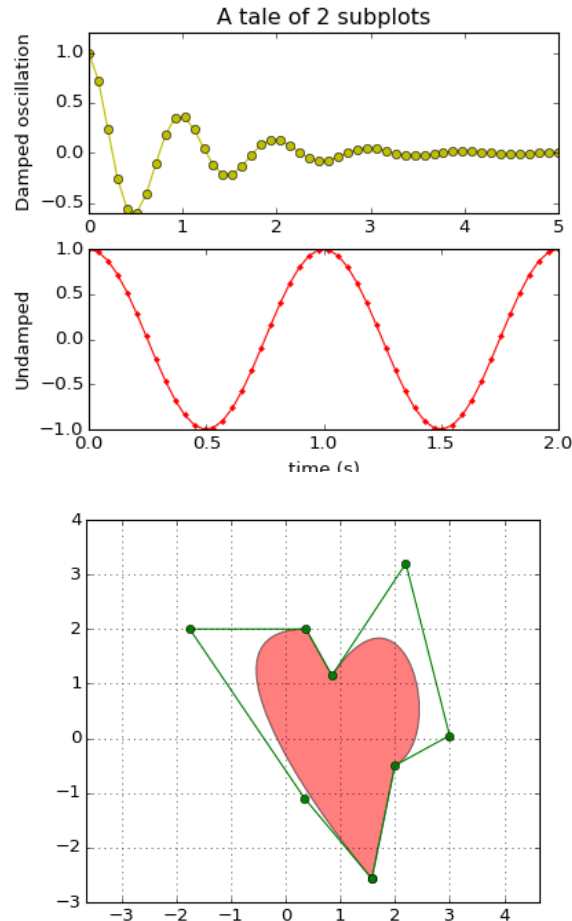
任务：

- 1.找出“葛洲坝600068” 2018年上半年的股票数据；
2. 计算年开盘价平均值；
- 3.按时间顺序排列。

人工智能程序设计

3 MATPLOTLIB

Matplotlib绘图



- **Matplotlib绘图**

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

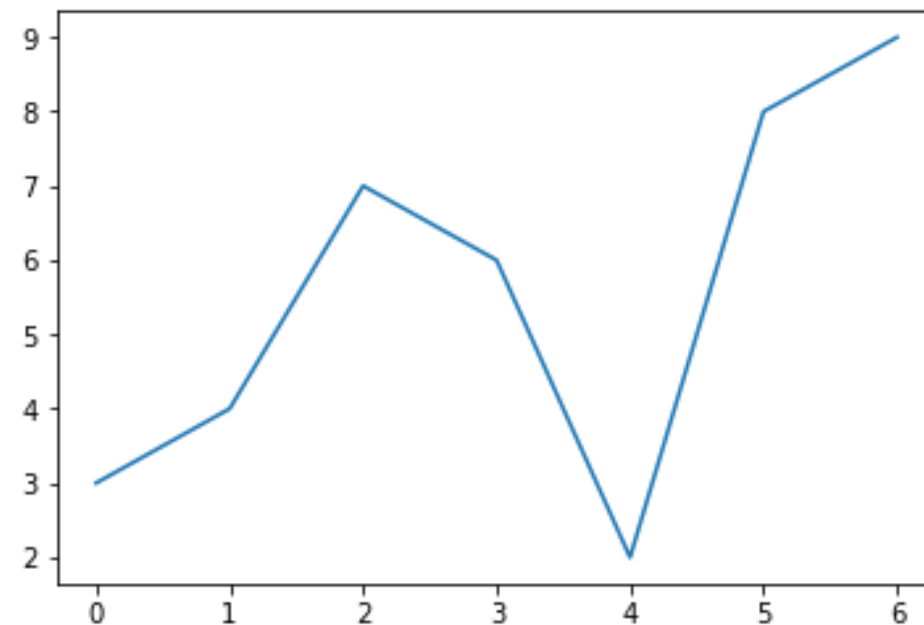
Matplotlib

1. 绘图基本方法
2. 图形属性控制
3. 基于pandas的绘图

折线图

Source

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



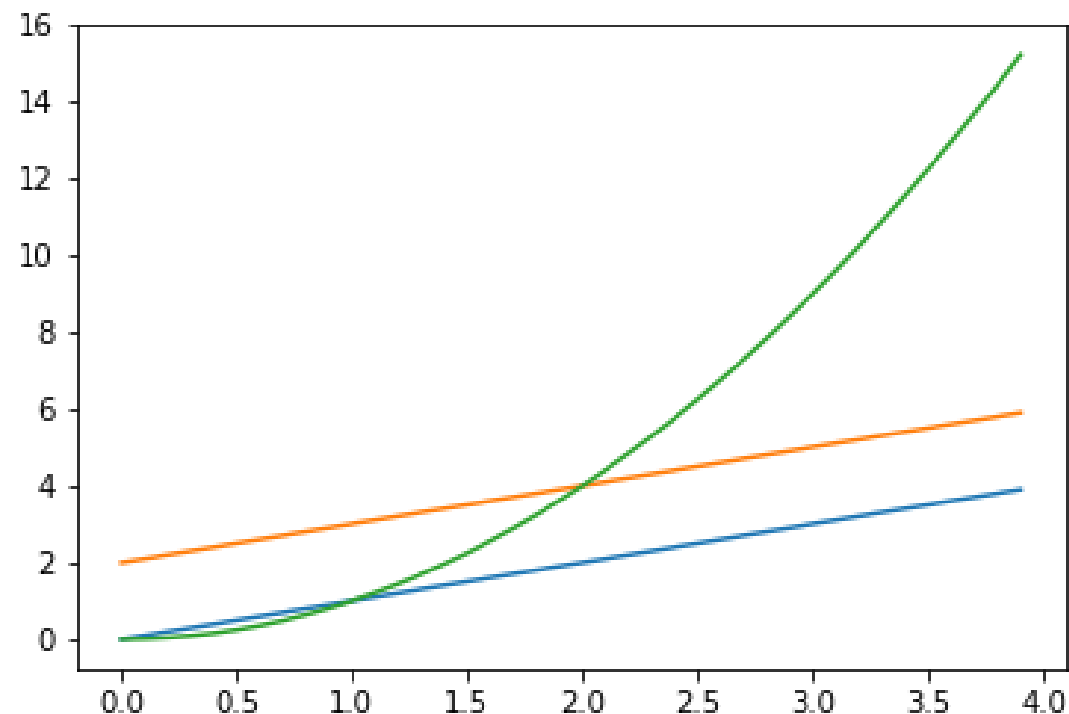
```
plt.plot(range(7), [3, 4, 7, 6, 2, 8, 9])
```

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

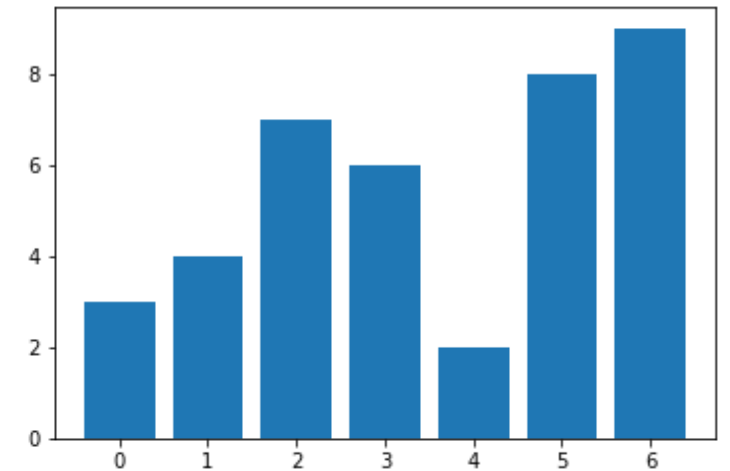
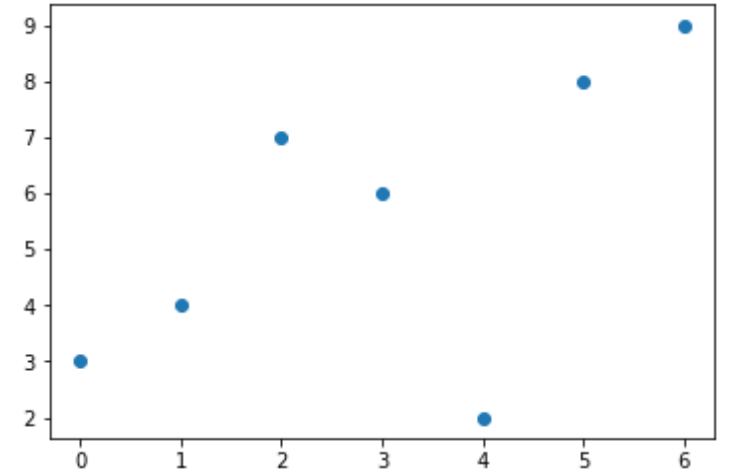
```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> t=np.arange(0.,4.,0.1)
>>> plt.plot(t, t, t, t+2, t, t**2)
```



绘制其他类型的图

Source

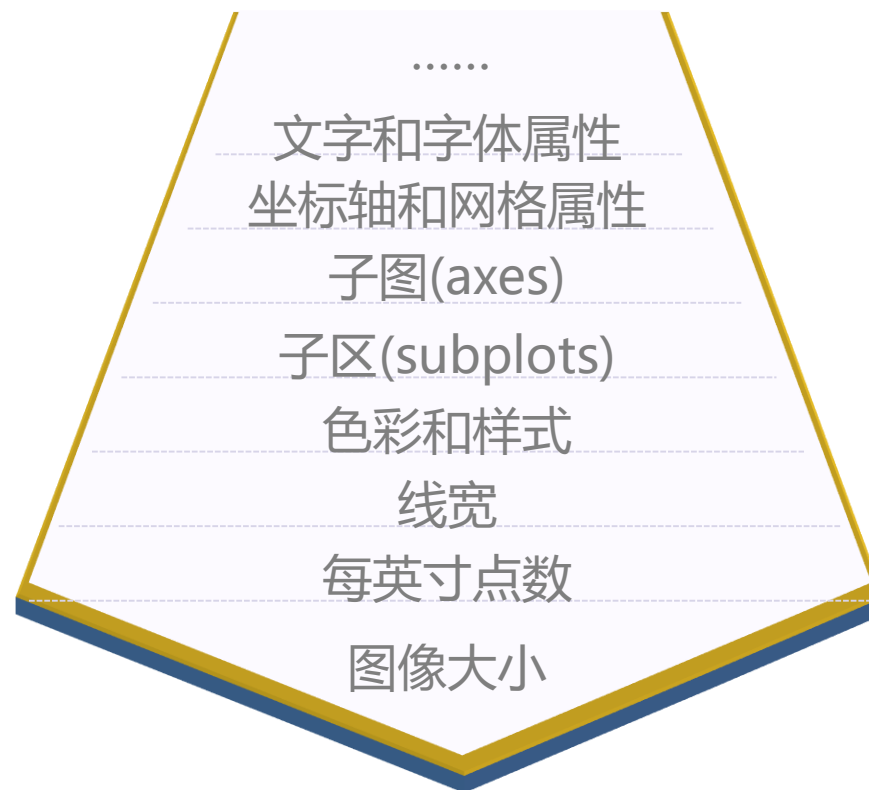
```
>>> import matplotlib.pyplot as plt  
>>> plt.scatter(range(7), [3, 4, 7, 6, 2, 8, 9])  
>>> plt.bar(range(7), [3, 4, 7, 6, 2, 8, 9])
```



Matplotlib

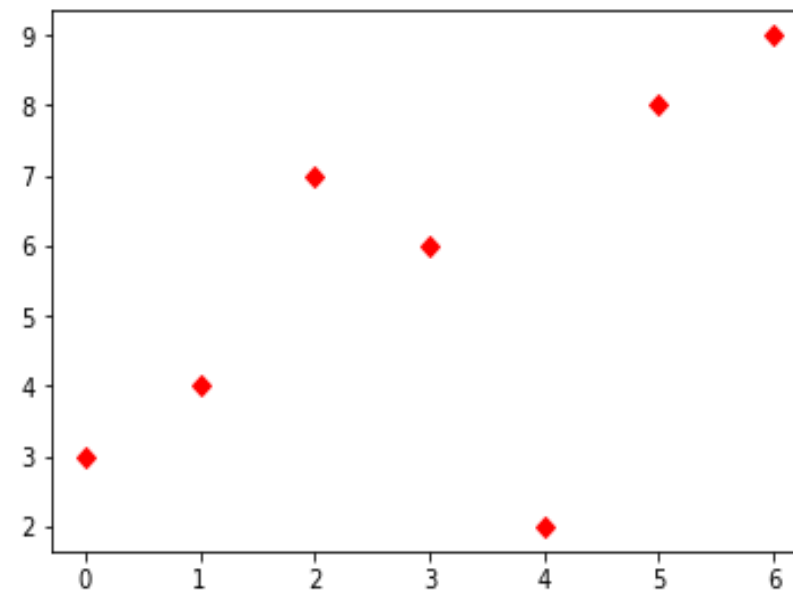
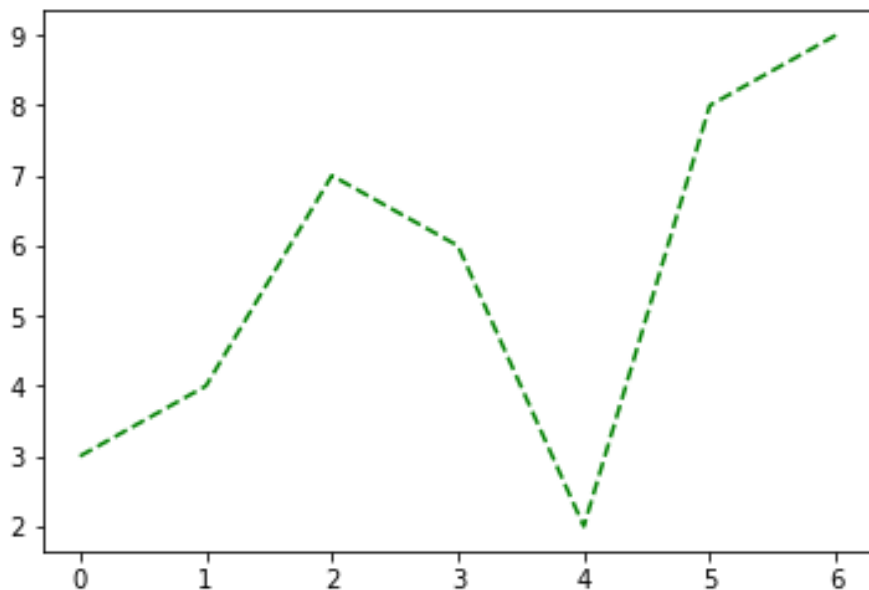
1. 绘图基本方法
- 2. 图形属性控制**
3. 基于pandas的绘图

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: 2.py

```
import pylab as pl
```

```
import numpy as np
```

```
pl.figure(figsize=(8,6),dpi=100)
```

```
t=np.arange(0.,4.,0.1)
```

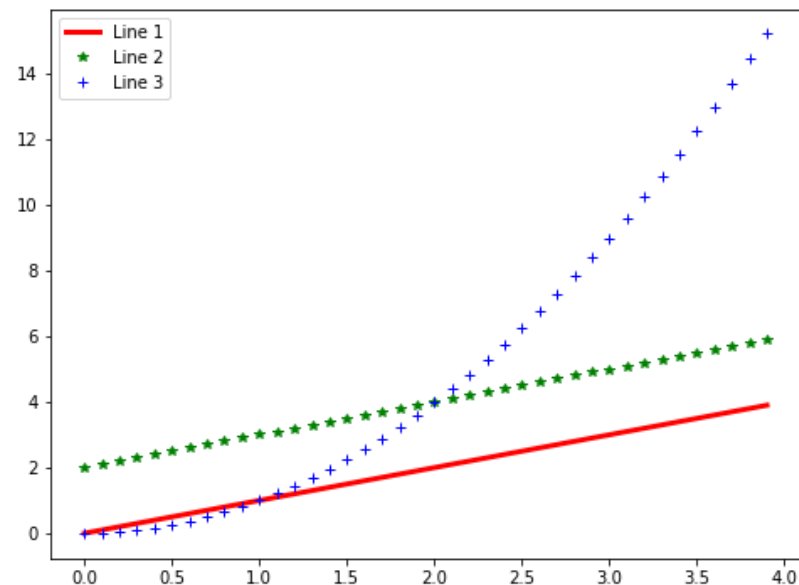
```
pl.plot(t,t,color='red',linestyle='-',linewidth=3,label='Line 1')
```

```
pl.plot(t,t+2,color='green',linestyle='',marker='*',linewidth=3,label='Line 2')
```

```
pl.plot(t,t**2,color='blue',linestyle='',marker='+',linewidth=3,label='Line 3')
```

```
pl.legend(loc='upper left')
```

```
plt.show()
```



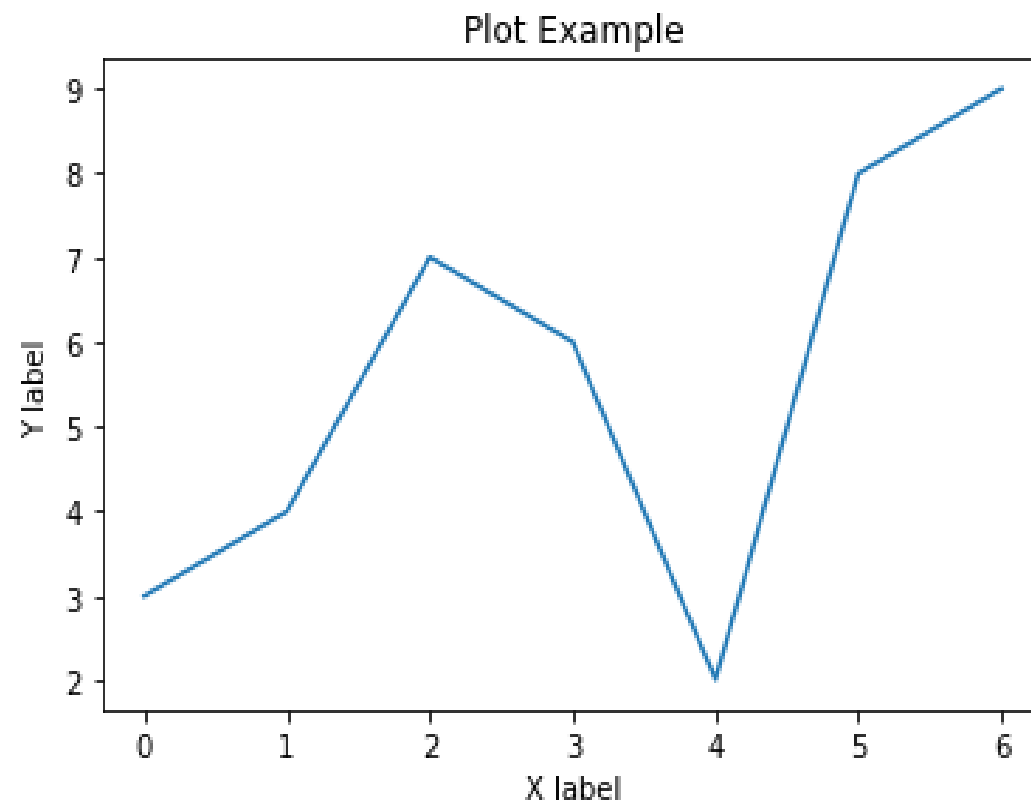
文字

加标题：图、横轴和纵轴



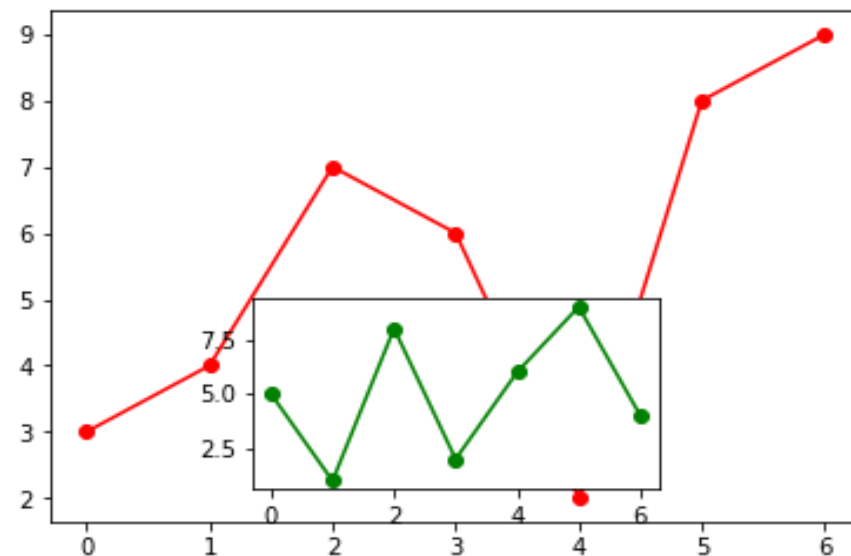
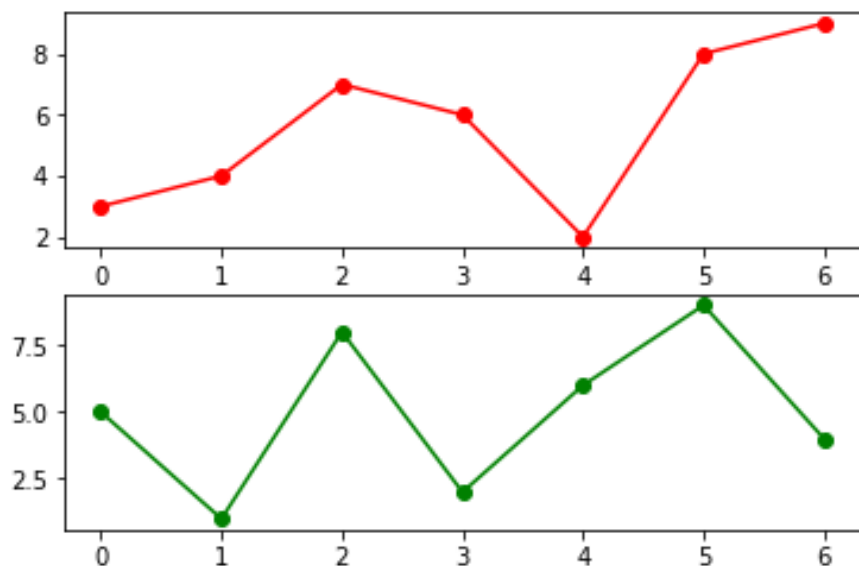
Filename: 3.py

```
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])
```

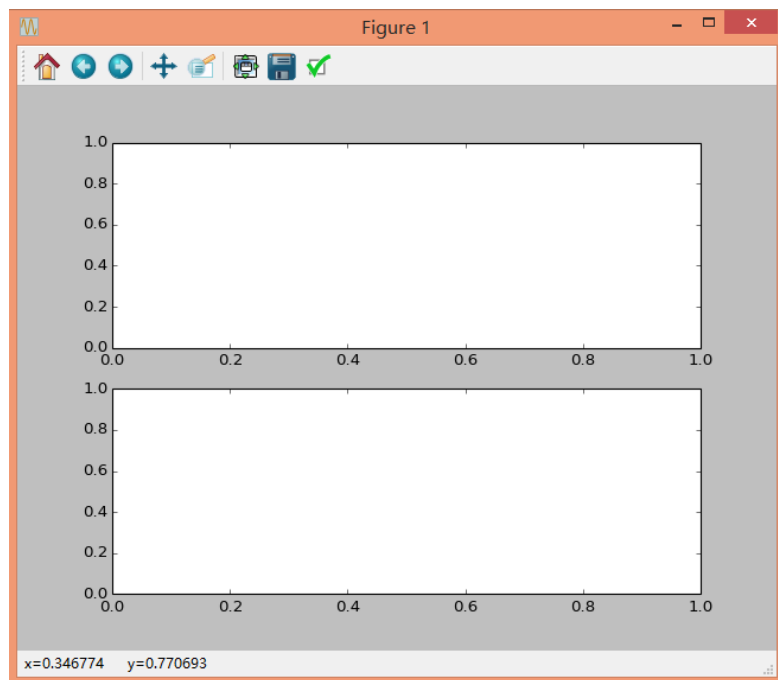


绘制子图

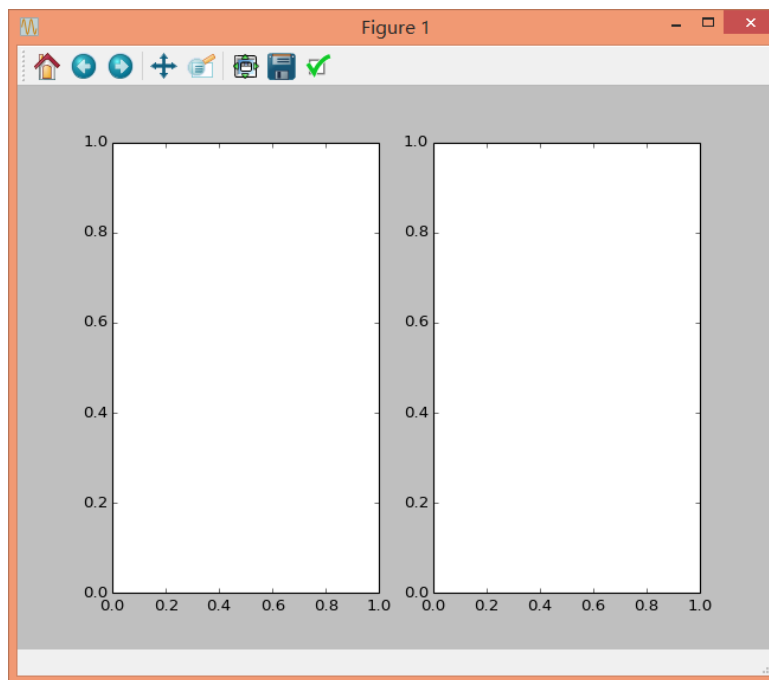
- 在Matplotlib中绘图在当前图形（figure）和当前坐标系（axes）中进行，默认在一个编号为1的figure中绘图，可以在一个图的多个区域分别绘图
- 使用subplot()函数和axes()函数



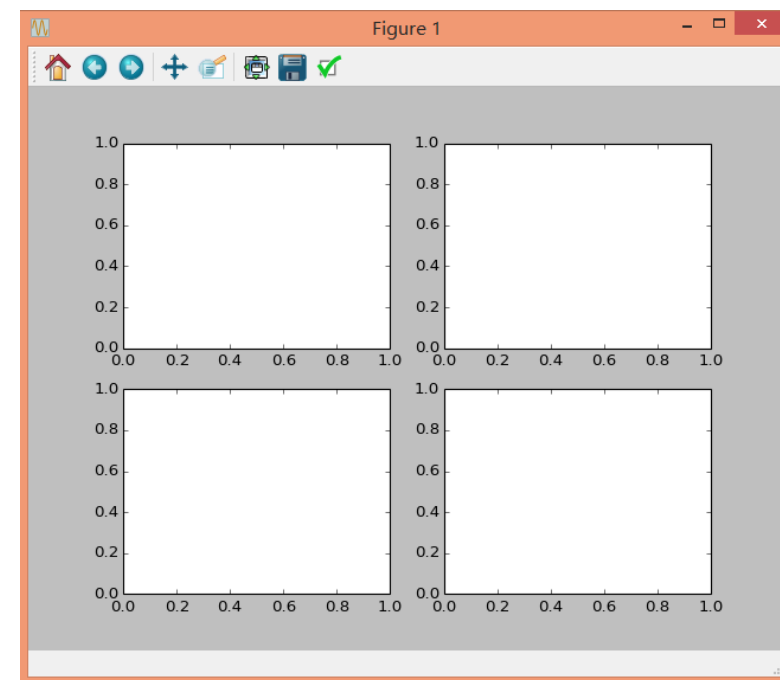
多子图-subplots



```
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

File

Filename: 3.py

```
import matplotlib.pyplot as plt
```

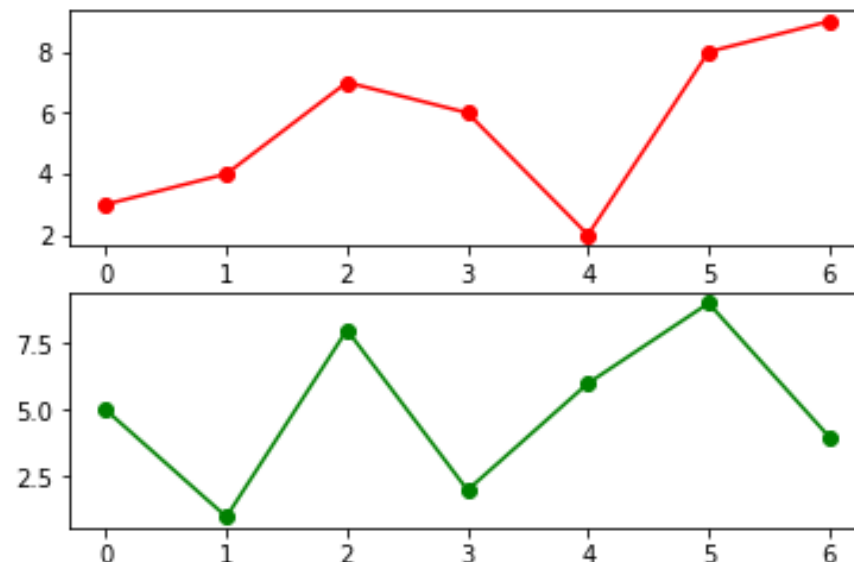
```
plt.figure(1)      # 默认创建, 缺省
```

```
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')
```

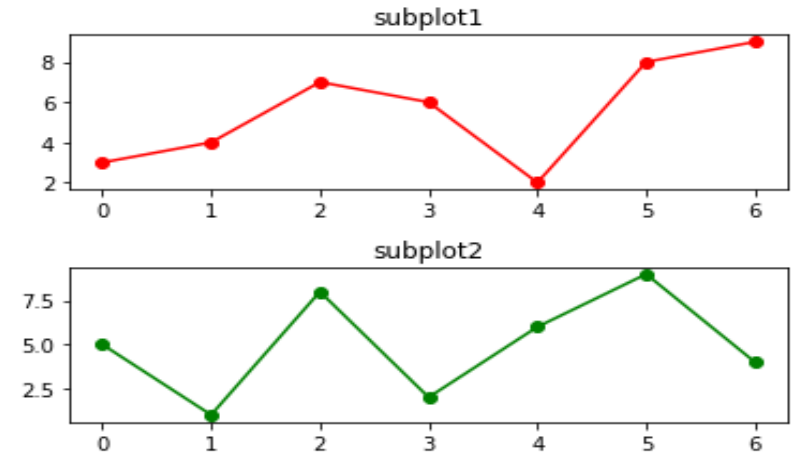


多子图-subplots

File

Filename: 4.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')
```



多子图-subplots

将可口可乐公司和IBM公司近一年来股票收盘价的月平均价绘制在一张图中

File

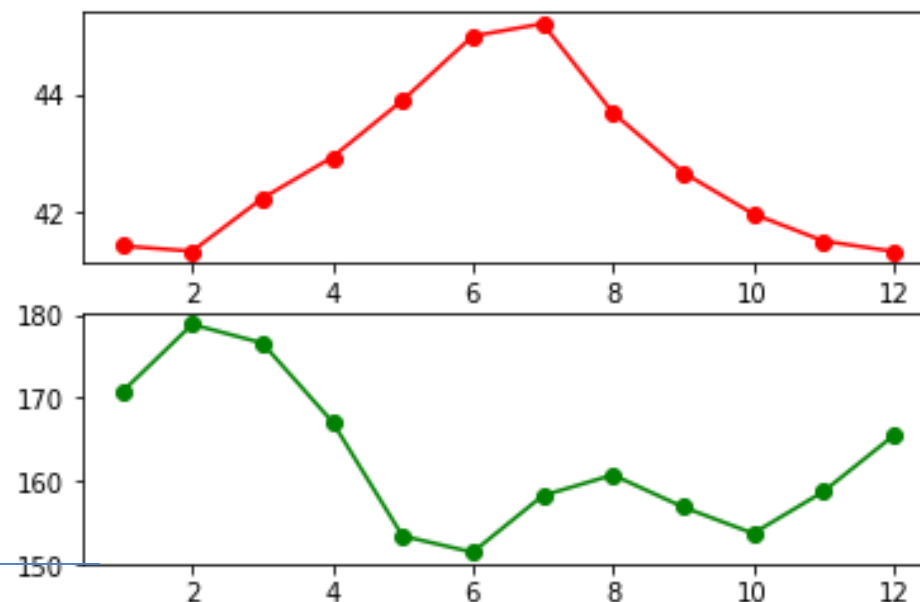
#The data of Coca-Cola and IBM is ready

```
plt.subplot(211)
```

```
plt.plot(x, y, color = 'r', marker = 'o')
```

```
plt.subplot(212)
```

```
plt.plot(xi, yi, color = 'green', marker = 'o')
```



子图-axes

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

File

Filename: 5.py

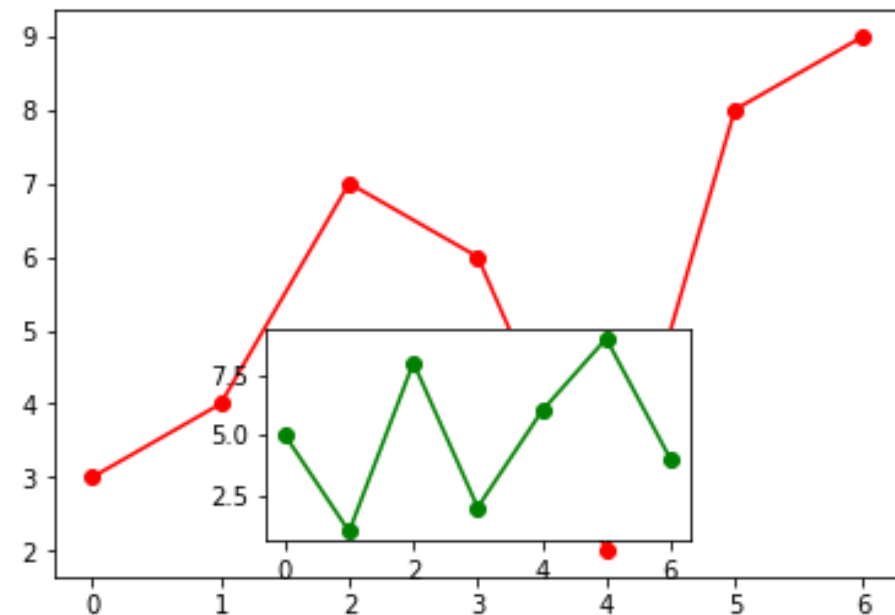
```
import matplotlib.pyplot as plt
```

```
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')
```



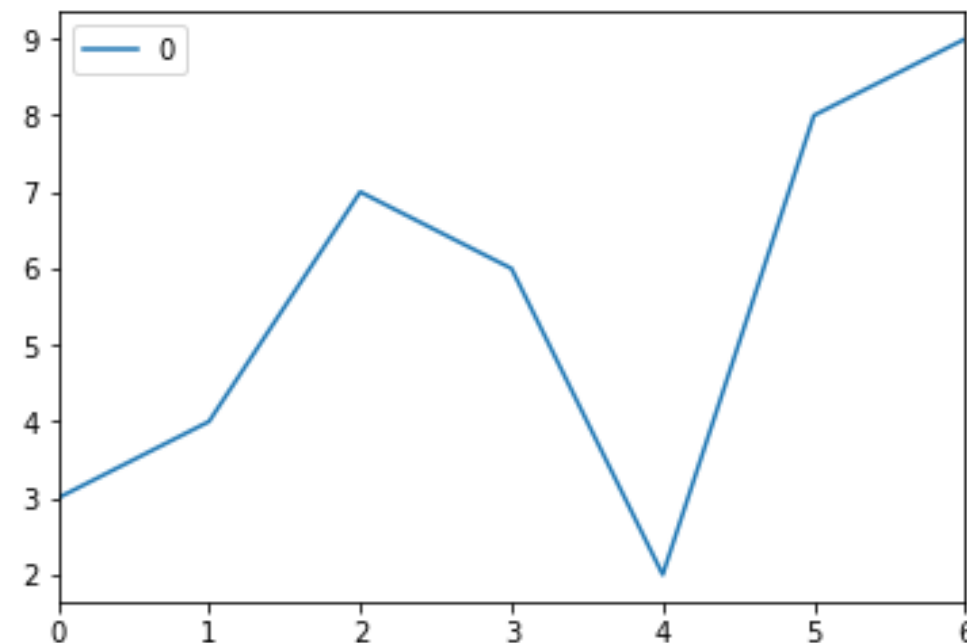
Matplotlib

1. 绘图基本方法
2. 图形属性控制
- 3. 基于pandas的绘图**

pandas绘图

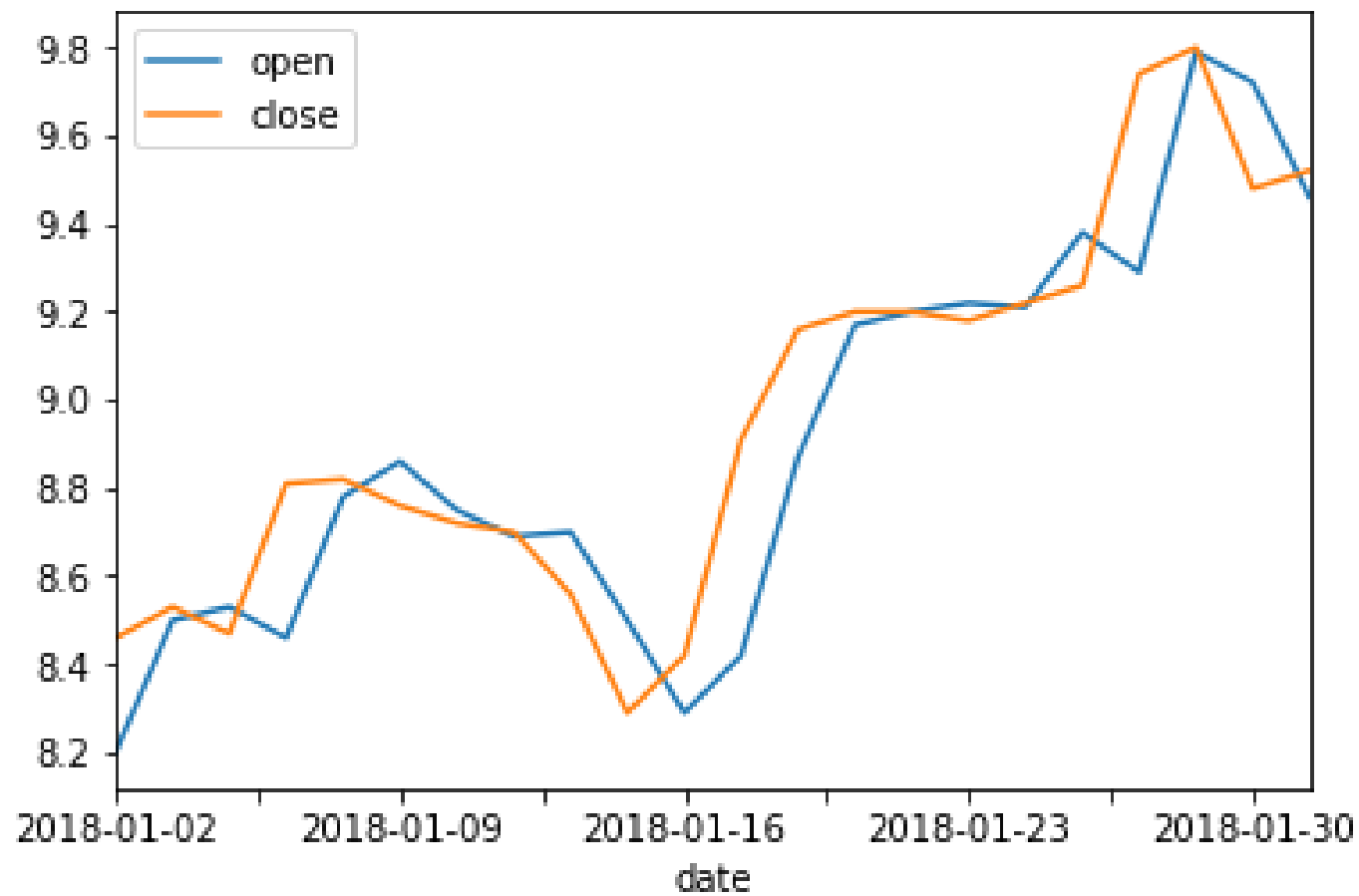
Source

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



股票数据绘制

绘制“葛洲坝
600068”2018年1
月份的股票数据开
盘价和收盘价的折
线图



pandas绘图

F_{ile}

Filename: 6.py

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

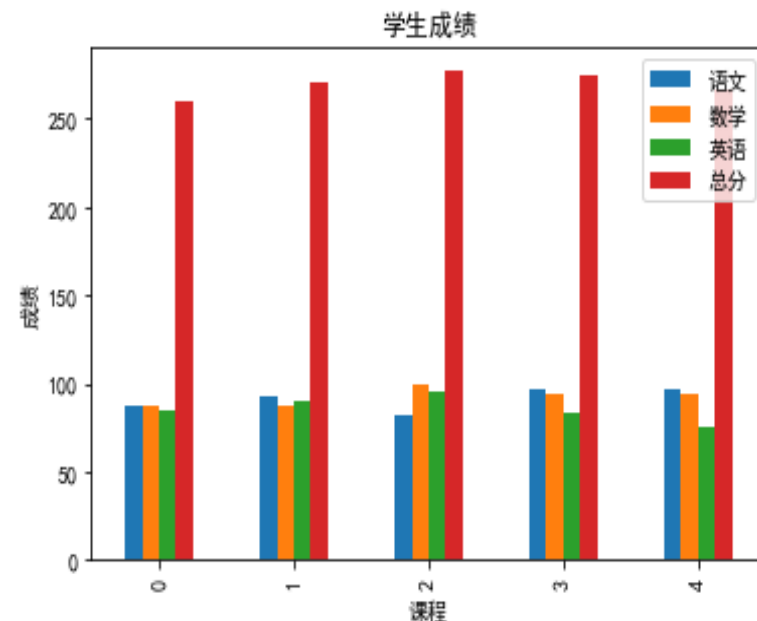
解决图中中文显示方块问题

```
import matplotlib as mpl
```

```
mpl.rcParams['font.sans-serif'] = ['SimHei']
```

```
df = pd.read_csv('score.csv', encoding = 'gb2312')
```

```
df.plot(kind = 'bar')
```



pandas绘图



Filename: 6.py

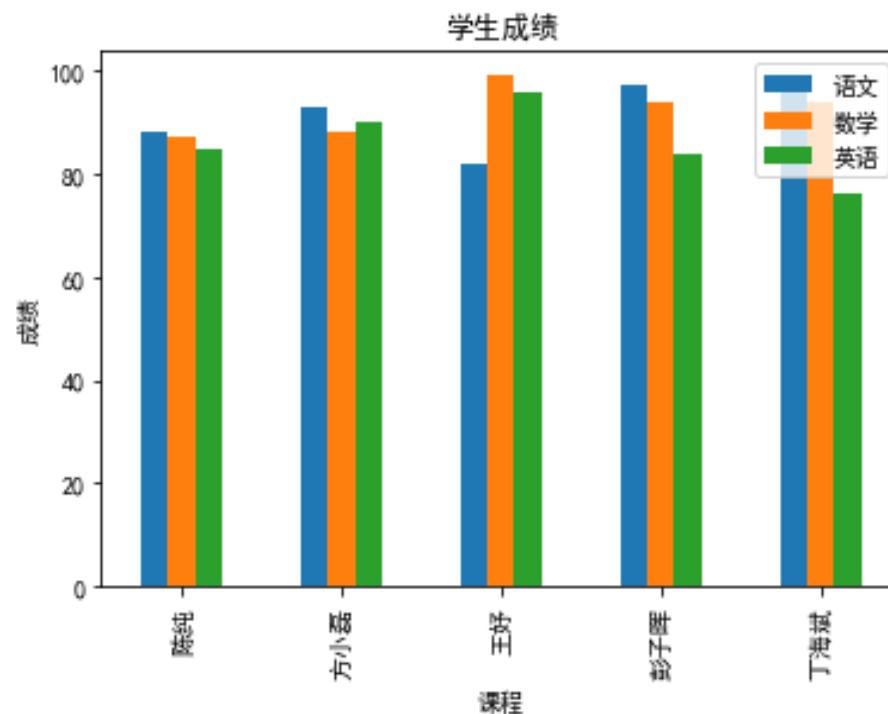
...

```
df = pd.DataFrame(data)
```

```
df_copy = df.iloc[:, :4]
```

```
Ax = df_copy.plot(kind='bar', title='学生成绩')
```

```
Ax.set(xlabel='课程', ylabel='成绩')
```

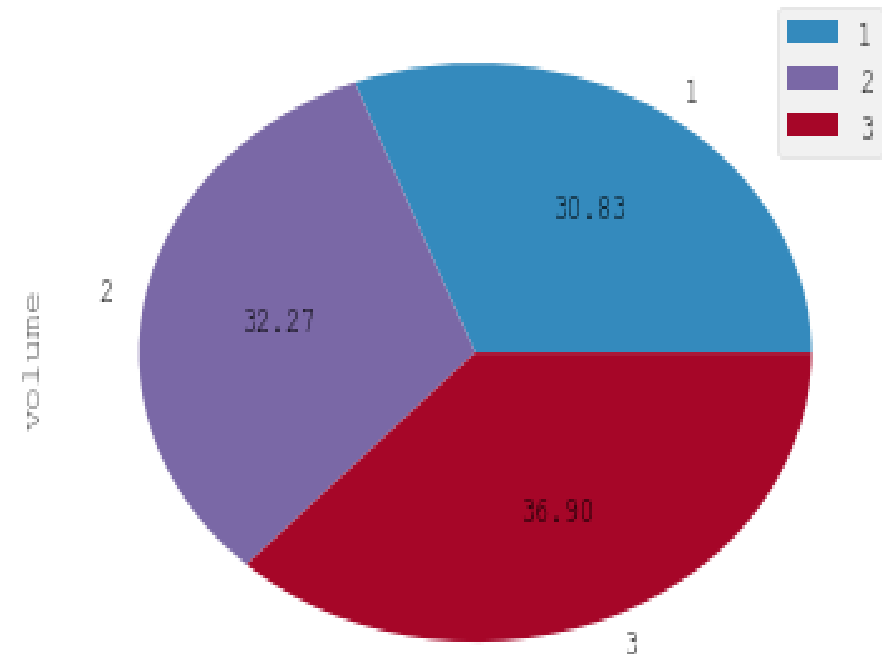


pandas控制图像形式

Intel公司本年度前3个月每个月股票收盘价的占比

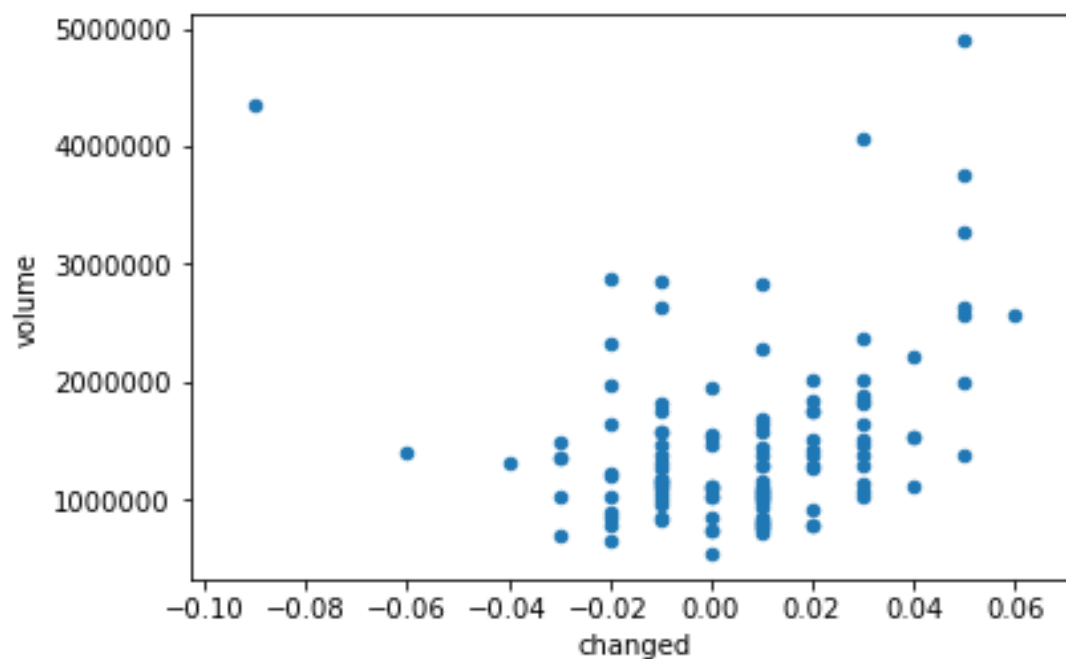
```
quotesINTC.plot()
```

```
quotesINTC.plot(kind = 'pie',  
subplots = True, autopct = '%.2f')
```



散点图中的信息

中国银行（股票代码是601988）股票在2017年1月1日至6月30日间的基本历史数据，绘制这半年期间每日收盘价与开盘价之差与当日成交量之间的散点图



M2 小结

00 SciPy生态系统

01 NumPy

02 pandas

03 Matplotlib