



Chap11 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

11.1

软件生态系统

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



```
>>> 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



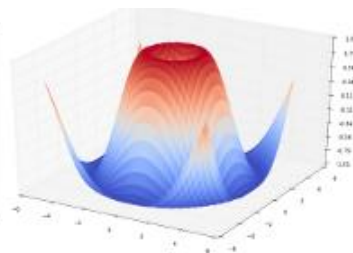
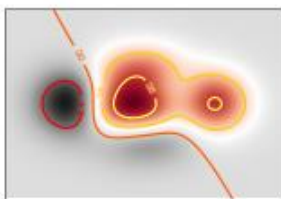
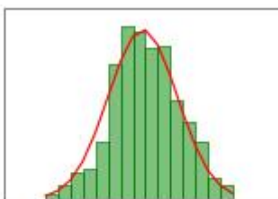
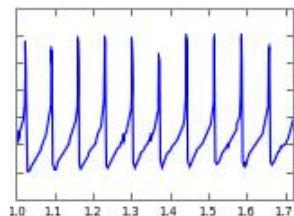
```
>>> import numpy as np
>>> from scipy import linalg
>>> arr = np.array([[1, 2], [3, 4]])
>>> linalg.det(arr)
-2.0
```

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





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

8



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)

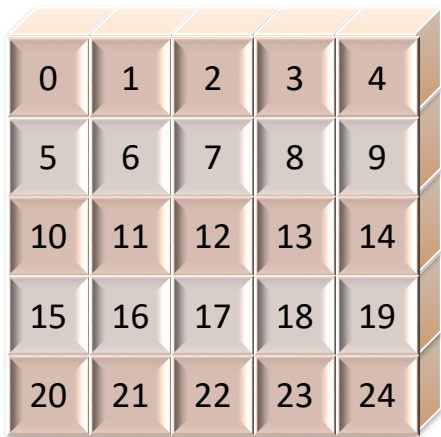


11.2

NUMPY

Python中的数组

- 用list和tuple等数据结构表示数组
 - 一维数组 `lst = [1,2,3,4]`
 - 二维数组 `lst = [[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



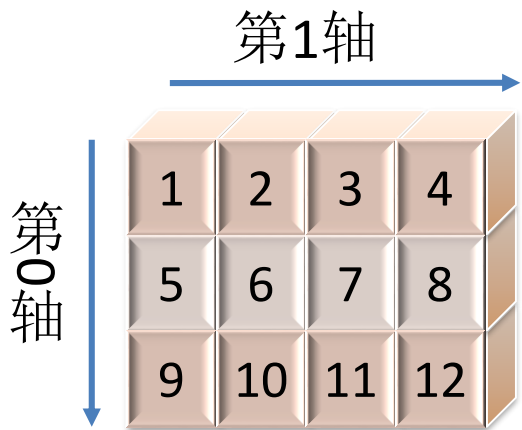
0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

- 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

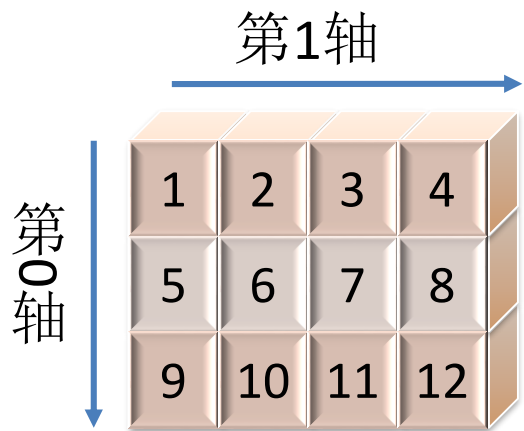
11.2.1 NDARRAY的基本特性



• ndarray数组属性

N维数组

- 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数组属性N维数组

– 基本属性

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

11.2.2 创建NDARRAY

ndarray的创建

array()函数

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)])
>>> 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
identity	linspace
logspace	mgrid
ogrid	ones
ones_like	r
zeros	zeros_like

ndarray创建函数

ndarray的创建

ones()
zeros()
arange()
linspace()

Source

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

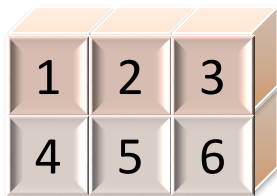
random()
fromfunction()

```
>>> np.random.random((2, 2))
array([[ 0.79777004,  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.]])
```

11.2.3 NDARRAY的操作和运算

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-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

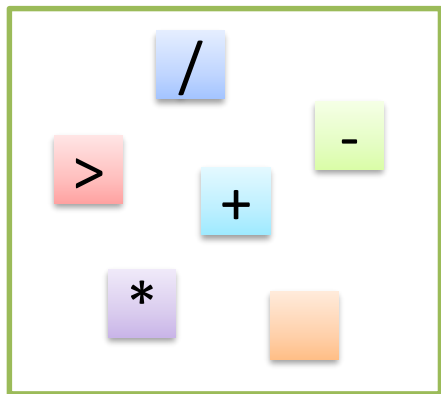
```
>>> aArray.resize(3,2)
>>> aArray
array([[1, 2],
       [3, 4],
       [5, 6]])
```

ndarray的基本操作-堆叠

A small orange 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

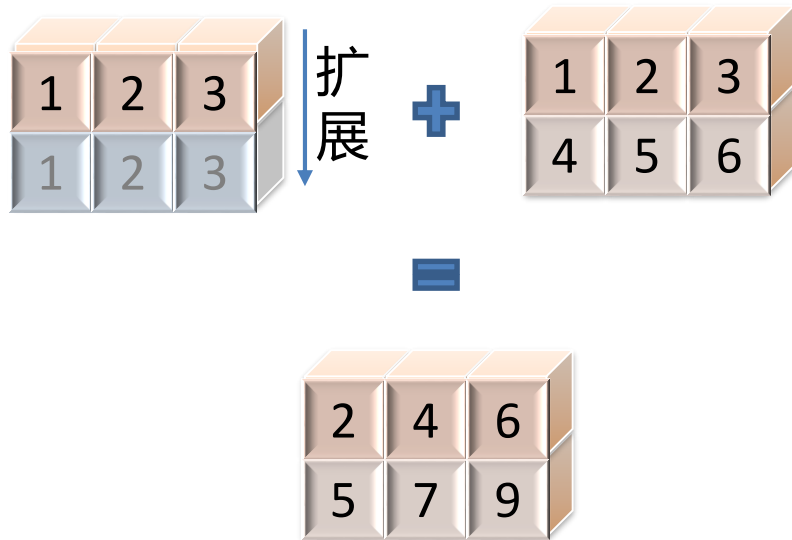
Source

```
>>> 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

11.2.4 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。

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: $t_n2 - t_n1$

11.2.5 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]]
>>> 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

常用函数示例

11.3

PANDAS


11.3.1 SERIES

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



```
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
>>> 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 = {'AXP': '86.40', 'CSCO': '122.64', 'BA': '99.44'}
>>> sindex = ['AXP', 'CSCO', 'BA', 'AAPL']
>>> aSer = pd.Series(data, index = sindex)
>>> aSer
```

AXP	86.40
CSCO	122.64
BA	99.44
AAPL	NaN

dtype: object

Source

```
>>> pd.isnull(aSer)
```


AXP	False
CSCO	False
BA	False
AAPL	True

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

```
>>> aSer = pd.Series(data, index = sinindex)
>>> aSer
AXP      86.40
CSCO    122.64
BA       99.44
AAPL      NaN
dtype: object
>>> bSer = {'AXP': '86.40', 'CSCO': '122.64', 'CVX': '23.78'}
>>> cSer = pd.Series(bSer)
>>> aSer + cSer
AAPL      NaN
AXP      86.4086.40
BA       NaN
CSCO    122.64122.64
CVX      NaN
dtype: object
```

11.3.2 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

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

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

```
>>> 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

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

score.csv - 记事本

文件(E) 编辑(E) 格式(O) 查看(V) 帮助(H)

姓名, 语文, 数学, 英语, 总分
陈纯, 88, 87, 85, 260
方小磊, 93, 88, 90, 271
王好, 82, 99, 96, 277
彭子晖, 97, 94, 84, 275
丁海斌, 97, 94, 76, 267

score.csv

DataFrame数据存取-读csv文件

File

```
# Filename: read_csv.py
```

```
>>> 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

读Yahoo
财经DJI
数据

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

- 索引
- 切片
- 专门的方法

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

- 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-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
 - 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



```
>>> 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

ix-选择列

- ix
 - 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[:, ['总分']]
```

总分

a 260

b 271

c 277

d 275

e 267

```
df.ix[:, [4]]
```

ix-选择区域

- ix
 - 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[:, '语文': '英语']
```

语文 数学 英语

a	88	87	85
b	93	88	90
c	82	99	96
d	97	94	84
e	97	94	76

```
df.ix[:,1:4]
```


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)]
```

11.3.3 SERIES和DATAFRAME

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



```
>>> 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	王好	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
```

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
```

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
```

Python财经数据接口包tushare

72

The screenshot shows the Tushare website's 'Table Of Contents' page. The left sidebar lists various data categories such as '前言' (Preface), '使用对象' (Target Audience), '使用前提' (Prerequisites), '下载安装' (Download and Install), '版本升级' (Version Upgrade), '版本信息' (Version Information), '数据源' (Data Source), '交易数据' (Transaction Data), '历史行情' (Historical Quotes), '实时行情' (Real-time Quotes), '历史分笔' (Historical Tick Data), '实时分笔' (Real-time Tick Data), '当日历史分笔' (Daily Historical Tick Data), '大盘指数行情列表' (Market Index Quote List), '大单交易数据' (Large Order Transaction Data), '投资参考数据' (Investment Reference Data), '分配数据' (Distribution Data), '业绩预告' (Earnings Forecast), '限售股解禁' (Restricted Share Unlocking), '基金持股' (Fund Holdings), '新股数据' (New Share Data), '融资融券 (沪市)' (Margin Trading and Securities Lending (Shanghai)), '融资融券 (深市)' (Margin Trading and Securities Lending (Shenzhen)), '股票分类数据' (Stock Classification Data), '行业分类' (Industry Classification), '概念分类' (Concept Classification), '地域分类' (Geographical Classification), '中小板分类' (Small and Medium Board Classification), '创业板分类' (Growth Enterprise Market Classification), '风险提示部分类' (Risk Warning Classification), '沪深300成份及权重' (CSI 300 Components and Weights), '上证50成份' (SSE 50 Components), '中证500成份' (CSI 500 Components), '截止上市股票列表' (List of Stocks as of Listing Date), '暂停上市股票列表' (List of Stocks Suspended from Listing), and '基本面数据' (Fundamental Data). The main content area features a QR code and a section titled 'TUSHARE 功能概览' (Tushare Function Overview) with a diagram showing data flow from 'Internet' (Internet) to 'Tushare core' (Tushare core) to 'Storage' (Storage). The 'Internet' box includes '上交所' (Shanghai Stock Exchange), '深交所' (Shenzhen Stock Exchange), '新三板' (New Third Market), '期货' (Futures), '期权' (Options), '融资融券' (Margin Trading and Securities Lending), and '基金' (Funds). The 'Tushare core' box includes '历史数据' (Historical Data), '实时数据' (Real-time Data), '分类数据' (Classification Data), '基本面' (Fundamentals), '宏观经济' (Macroeconomy), '网络舆情' (Network Public Opinion), '新闻事件' (News Events), and '分笔' (Tick Data). The 'Storage' box includes 'CSV/HDFS' and 'Excel/JSON' file formats, and 'DataBase / NoSQL' databases.

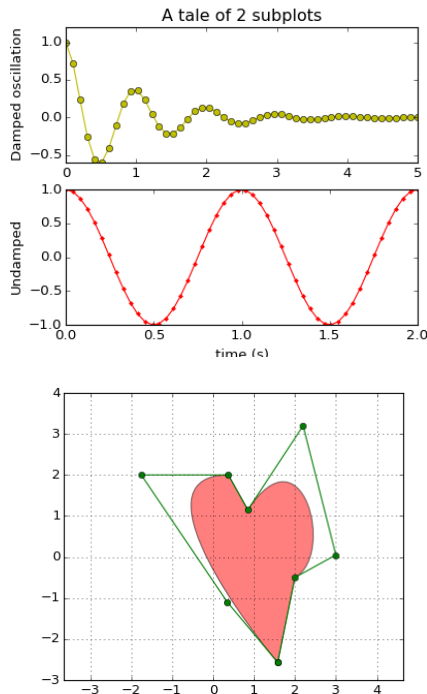
任务:

1. 找出“葛洲坝600068”2016年的股票数据;
2. 计算年开盘价平均值;

11.4

MATPLOTLIB

Matplotlib绘图



- ## Matplotlib绘图

最著名Python绘图库，主要用于二维绘图

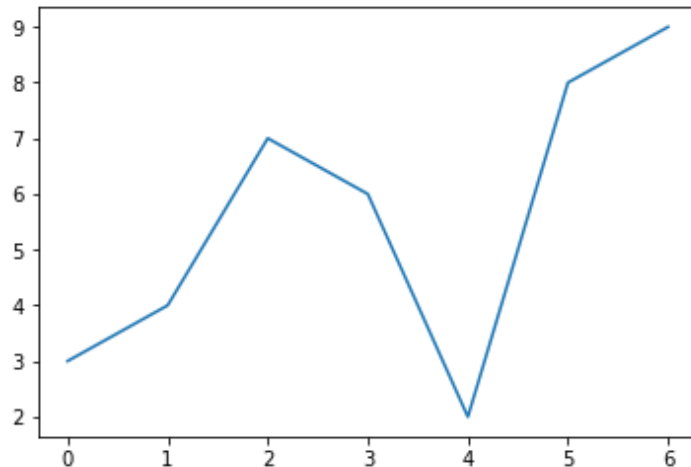
- 画图质量高
- 方便快捷的绘图模块
 - 绘图API——pyplot模块
 - 集成库——pylab模块（包含NumPy和pyplot中的常用函数）

11.4.1 MATPLOTLIB绘图基本方法

折线图



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



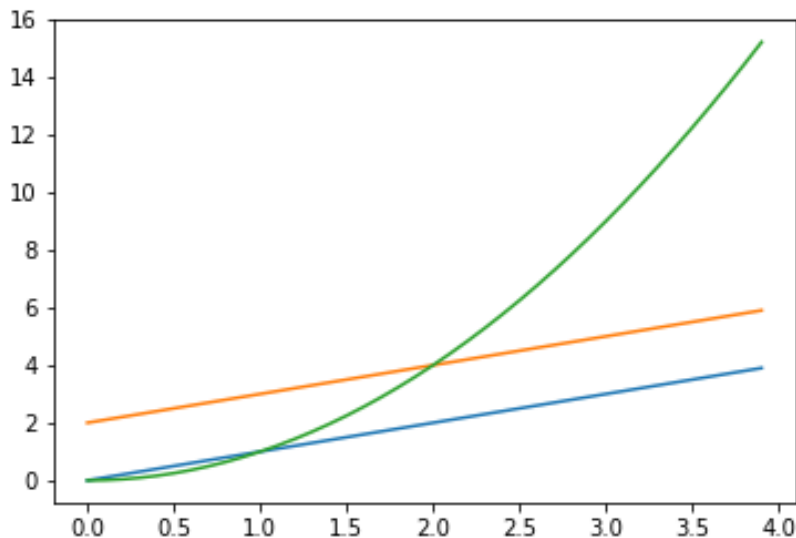
```
plt.plot(range(7), [3, 4, 7, 6, 2, 8, 9])
```

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

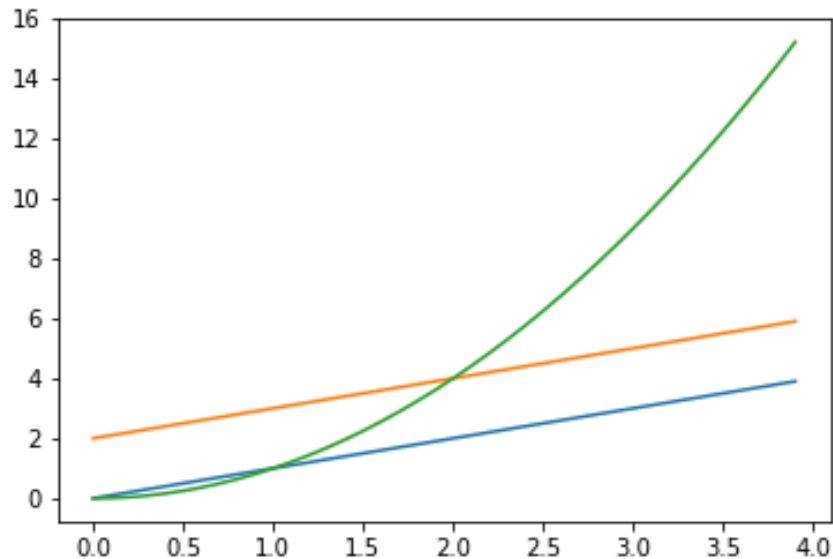
```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> t=np.arange(0.,4.,0.1)
>>> plt.plot(t, t, t, t+2, t, t**2)
```



pylab绘图

S_{ource}

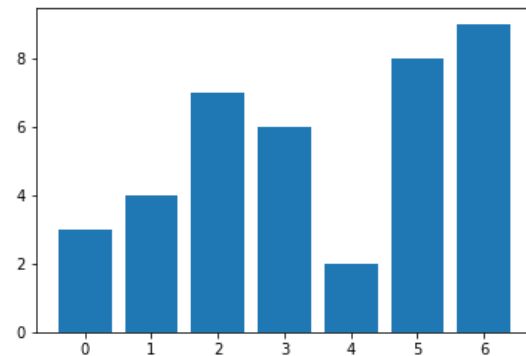
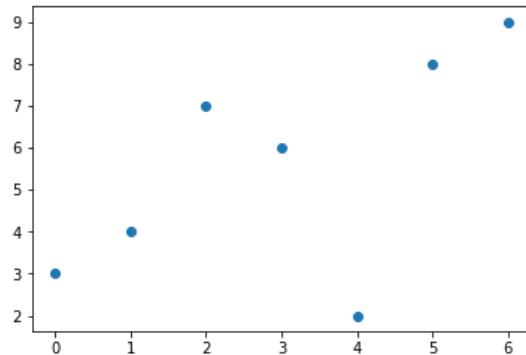
```
>>> import pylab as pl  
>>> t=pl.arange(0.,4.,0.1)  
>>> pl.plot(t, t, t, t+2, t, t**2)
```



绘制其他类型的图

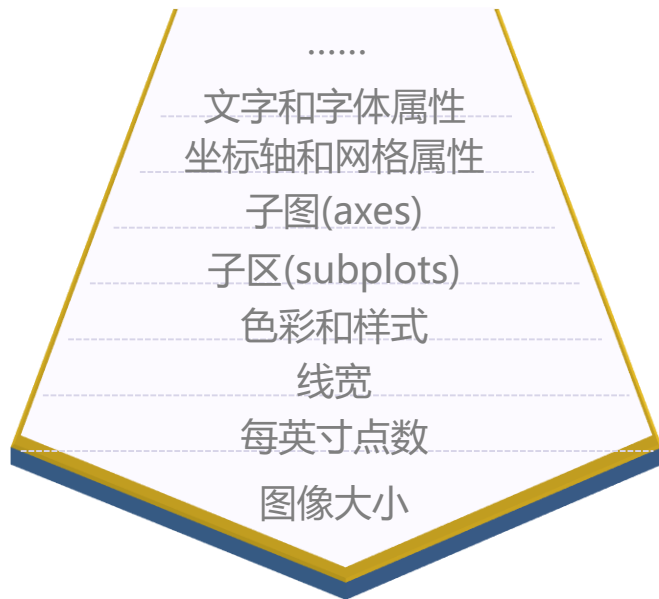
Source

```
>>> 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])
```



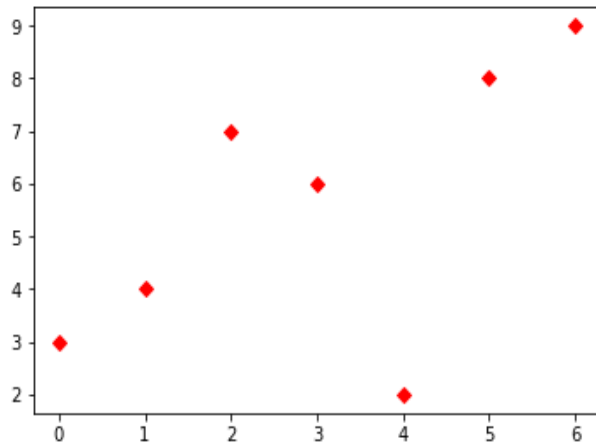
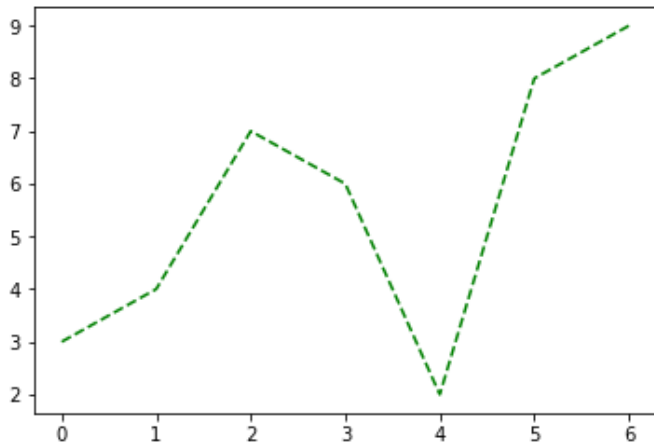
11.4.2 MATPLOTLIB图形属性控制

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
...	...

File

Filename: 2.py

```
import pylab as pl
```

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

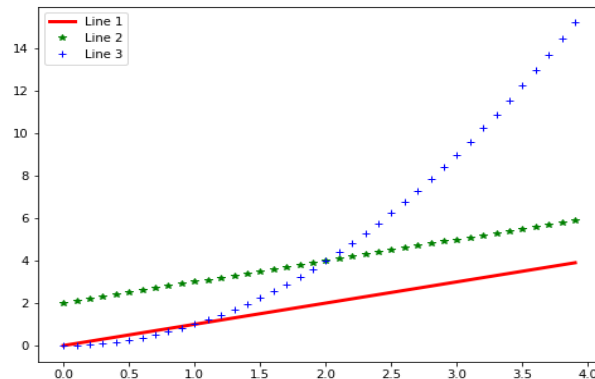
```
t=pl.arange(0.,4.,0.1)
```

```
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')
```

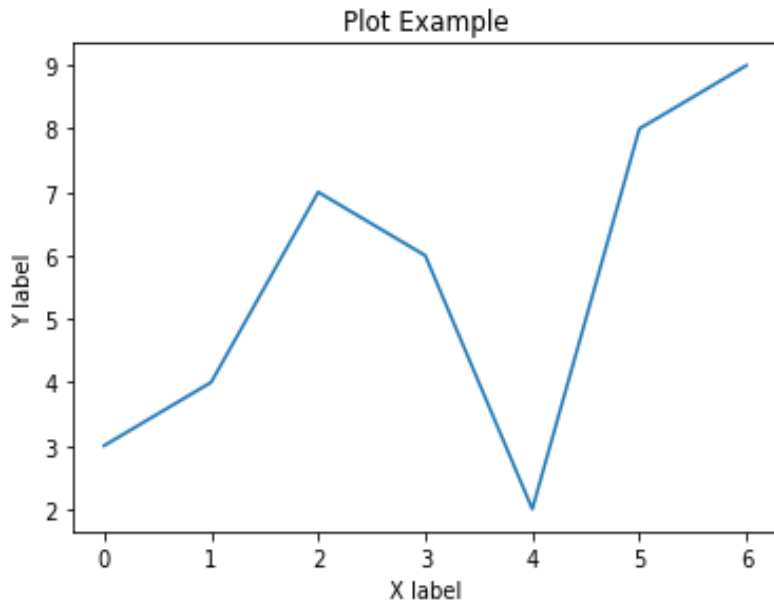


加标题：图、横轴和纵轴



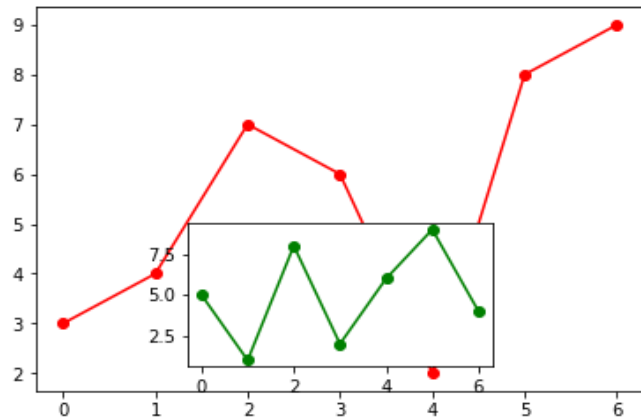
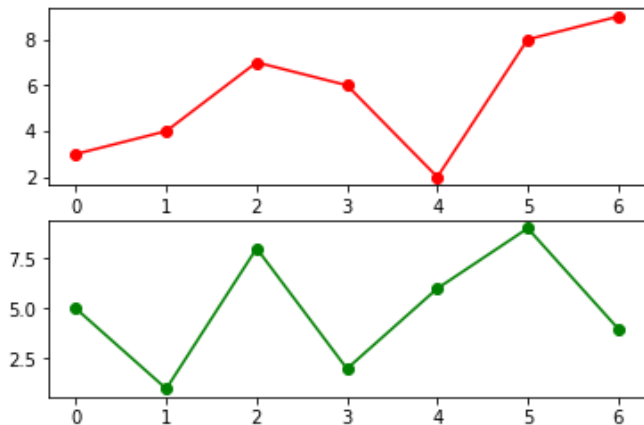
Filename: 3.py

```
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])
```

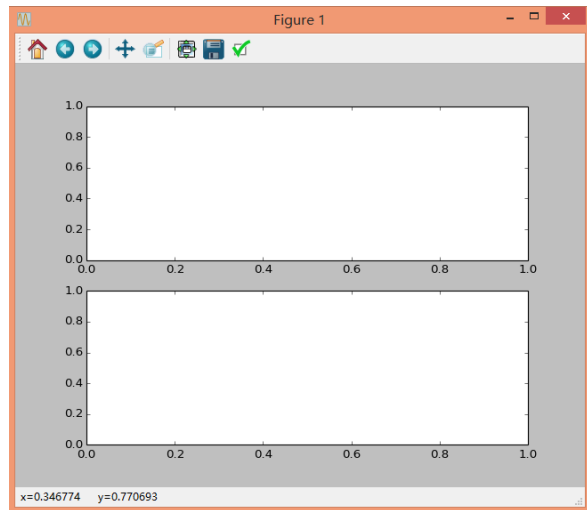


绘制子图

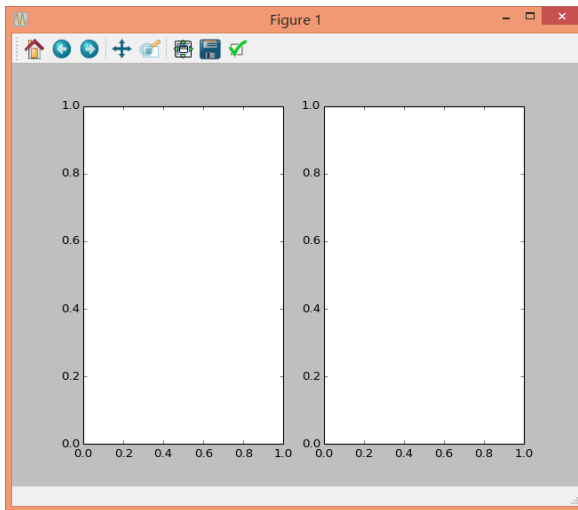
- 在Matplotlib中绘图在当前图形 (figure) 和当前坐标系 (axes) 中进行，默认在一个编号为1的figure中绘图，可以在一个图的多个区域分别绘图
- 使用subplot()函数和axes()函数



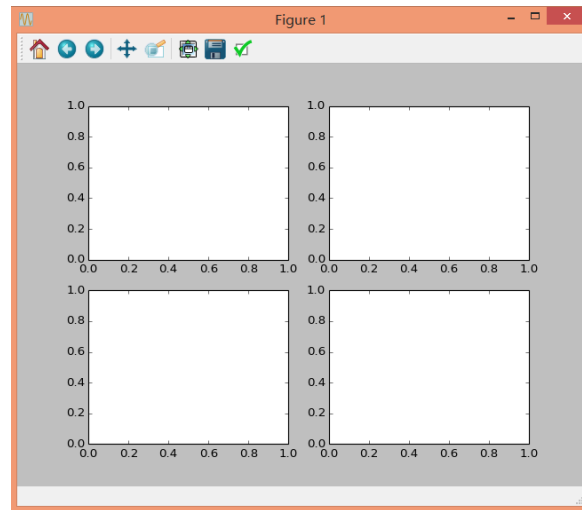
多子图-subplots



```
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
```

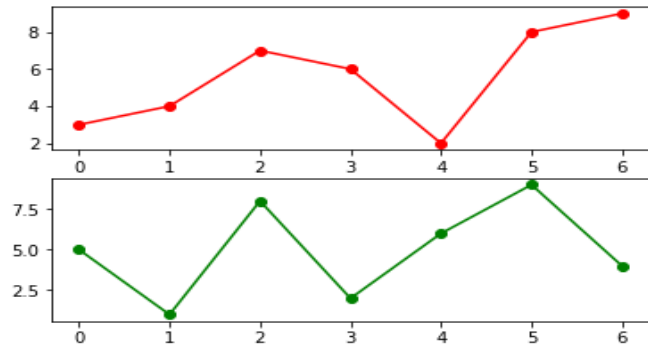
```
plt.subplot(211)        # 第一个子图
```

```
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')
```

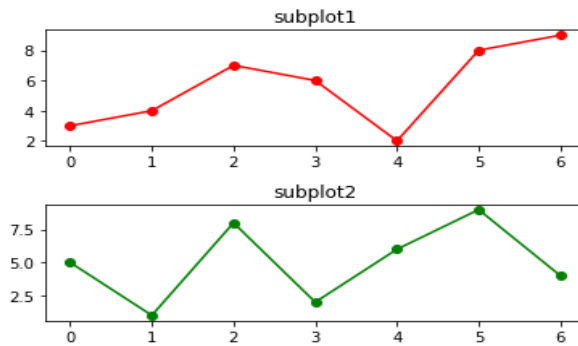
```
plt.show()
```



多子图-subplots

!
F ile
.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 = 'red', 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')
fig.savefig('d:\\abc.png')
```



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



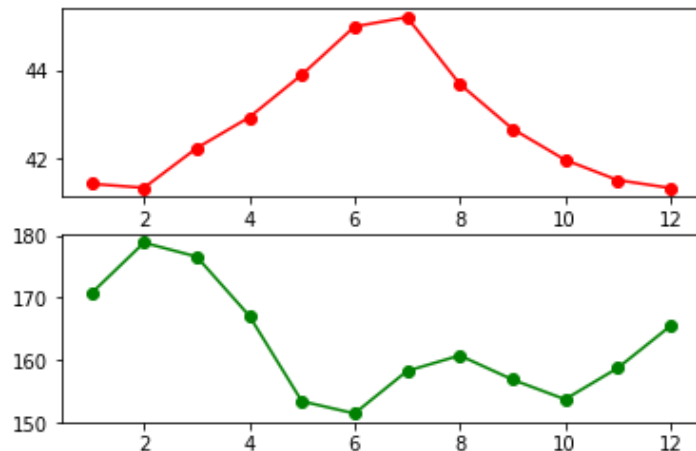
#The data of Coca-Cola and IBM is ready

```
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

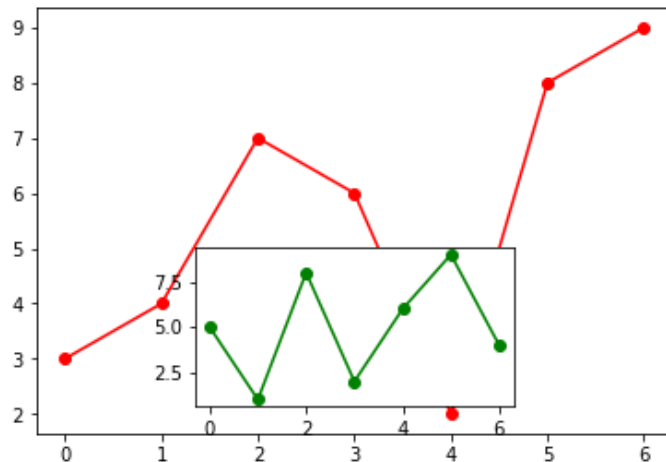
```
import matplotlib.pyplot as plt
```

```
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])
```

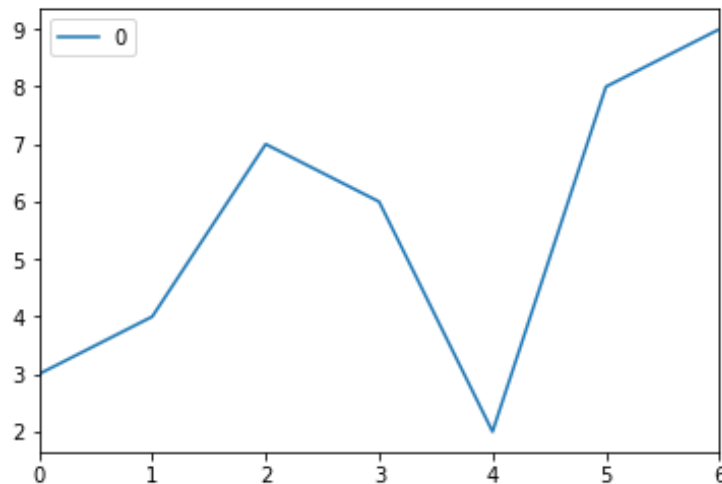
```
plt.plot(range(7), [5, 1, 8, 2, 6, 9, 4], color = 'green', marker = 'o')
```



pandas绘图

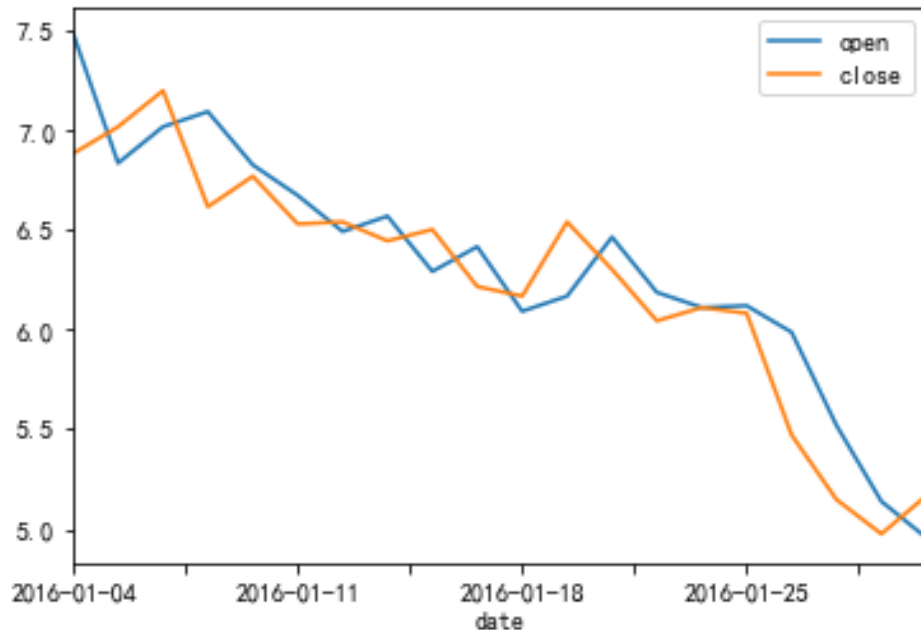
Source

```
>>> 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”2016年1
月份的股票数据
开盘价和收盘价
的折线图



```
import pandas as pd
import matplotlib.pyplot as plt
import tushare as ts
df=ts.get_k_data('600068',start='2016-01-01',end='2016-01-31')
df1= pd.DataFrame()
df1['open']=df.open
df1['close']=df.close
df1.index=df.date
df1.plot(kind='line')
```

pandas绘图

File

Filename: 6.py

```
import pandas as pd
```

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

解决图中中文显示方块问题

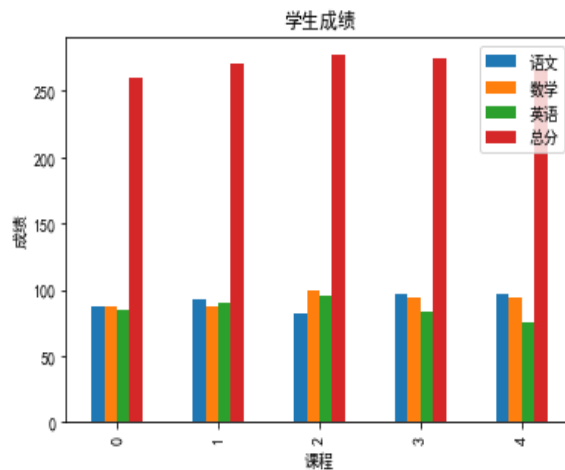
```
import matplotlib as mpl
```

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

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

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

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



pandas绘图

F
ile

Filename: 6.py

```
...  
df = pd.DataFrame(data)  
df_copy = pd.DataFrame()  
df_copy['语文'] = df.语文  
df_copy['数学'] = df.数学  
df_copy['英语'] = df.英语  
df_copy.index = df.姓名  
cht=df_copy.plot.bar(title='学生  
成绩')  
cht.set_ylabel('成绩')  
cht.set_xlabel('课程')
```

