

Python 教程

2018-10-16

金融工程 | 专题报告

Python: 从入门到精通 (基础篇)

报告要点

■ Python 配置

Anaconda 是 Python 的一个专注于数据分析的版本, 其下的 conda 可以用来管理 Python 的包和虚拟环境, 在安装包的同时会自动安装相应的依赖, 并且可以做到虚拟环境的创建以隔离不同要求的项目。

■ 基础操作

Python 是一种面向对象的程序设计语言, 包含常用的数值型、字符型数据类型, 也可以通过控制流语句对执行模块进行控制。

■ numpy 与 pandas

numpy 是 Python 数值计算最重要的基础包, 它在一个连续的内存块中储存数据, 独立于其他 Python 内置对象, 且在整个数组上直接进行运算而不需要 Python 的 for 循环, 所以可以高效处理大数组的数据。pandas 基于 numpy 数组格式构建, 专门用来处理表格和混杂数据。

■ 画图

matplotlib 可以为 Python 构建一个 MATLAB 式的绘图接口, 可以绘制常见的散点图、折线图、柱状图、分布图等, 且可将图片导出为常见的 JPG、PNG、PDF 等格式。

■ 模型运用

由于 Python 高效的处理数据特性, 在数学建模方面 Python 也有很好的运用。statsmodels 可以建立包括统计检验、回归分析、时间序列等常用的统计模型, scipy 用于处理科学计算中常见的问题, 如插值、积分、聚类、优化, sklearn 和 tensorflow 可以建立机器学习和深度学习模型。

■ 简单爬虫

爬虫就是请求网站并提取数据的自动化程序, Python 强大的数据处理功能, 在整理爬取的数据上有一定的优势, 在处理发起请求、获取内容、解析内容上也有很多包方便处理。

分析师 覃川桃

☎ (8621) 61118766

✉ qinct@cjsc.com.cn

执业证书编号: S0490513030001

联系人 郑起

☎ (8621) 61118706

✉ zhengqi2@cjsc.com

相关研究

《国证食品指数投资价值分析》2018-8-29

《追涨的艺术和艺术地追涨》2018-8-13

《动量视角下的因子轮动》2018-7-28

风险提示: 代码的运行会因配置环境的不同产生报错。

目录

Python 的配置	4
Anaconda 的安装	4
Python 环境的配置	5
基础操作	6
使用 ipython notebook	6
基本语言和功能	7
List 的基本操作	7
Dict 和 Set 的基本操作	8
时间日期的基本操作	8
控制流	9
numpy 和 pandas	9
numpy 的基本操作	10
pandas 的基本操作	11
画图	14
模型运用	15
线性回归	15
机器学习	16
组合优化	17
简单爬虫	18

图表目录

图 1: Anaconda 的安装步骤 1	4
图 2: Anaconda 的安装步骤 2	5
图 3: conda 环境操作 1	6
图 4: conda 环境操作 2	6
图 5: 进入 ipython notebook	7
图 6: Python 基本语言和功能	7
图 7: List 的基本操作	8
图 8: Dict 和 Set 的基本操作	8
图 9: 时间日期的基本操作	9
图 10: Python 控制流语句	9
图 11: numpy 的基本操作 1	10
图 12: numpy 的基本操作 2	10
图 13: numpy 的矩阵计算	11
图 14: pandas 的数据结构	11

图 15: pandas 的数据选取	12
图 16: pandas 的数据赋值	12
图 17: pandas 的数据结构化处理 1	12
图 18: pandas 的数据结构化处理 2	13
图 19: pandas 的数据运算 1	13
图 20: pandas 的数据运算 2	13
图 21: pandas 中的 SQL 操作	14
图 22: pandas 的数据存取	14
图 23: matplotlib 画图	15
图 24: matplotlib 画图结果	15
图 25: statsmodels 线性回归	16
图 26: sklearn 实现普通线性回归	16
图 27: sklearn 实现套索回归	17
图 28: sklearn 实现随机森林	17
图 29: scipy 实现组合优化	18
图 30: 网站 1 部分 html 源码	18
图 31: 爬取网站 1 表格数据	19
图 32: 网站 2 部分 html 源码	19
图 33: 爬取网站 2 标题数据	19
表 1: conda 命令	5

Python 的配置

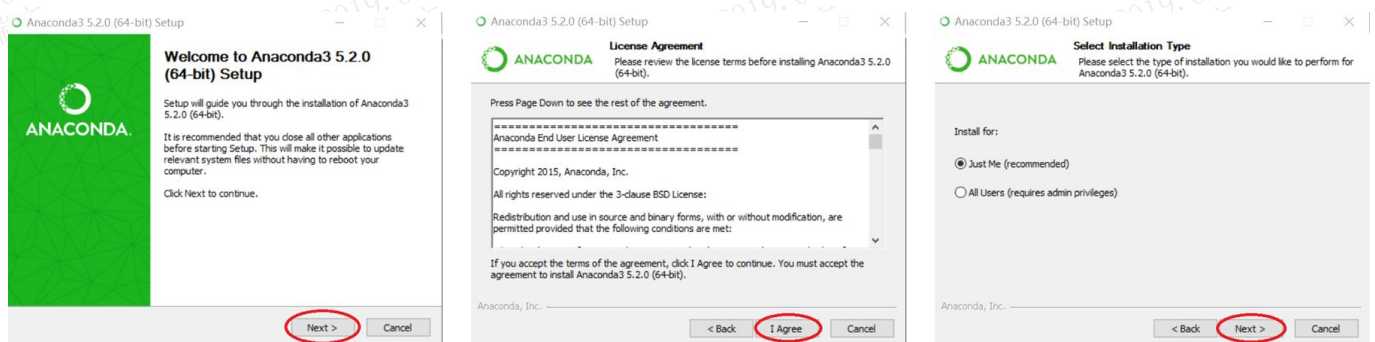
自 1991 年诞生以来, Python 已经成为了最受欢迎的动态编程语言之一, 和传统计算机语言相比, Python 有着易于学习、开发效率高、功能实现齐全等特点, 它为我们提供了非常完善的基础代码库, 覆盖了网络、文件、GUI、数据库、文本等大量内容, 尤其随着近些年来很多 Python 库的不断改良 (如 pandas、scikit-learn), 使其成为目前在数据科学领域中最广泛使用的语言之一。Python 可以处理如**统计分析、数学建模、机器学习模型建立、网页爬虫**等任务, 近期热门的深度学习框架更是以 Python 的开发效率为考量, 提供了相关的 API 接口, 其中最著名的就是 Google 开发的 tensorflow。本文从 Python 的安装配置开始, 介绍 Python 的一些基础知识。

Anaconda 的安装

Anaconda 是 Python 的一个发行版本, 包含了大部分常用的 Python 包, 专注于数据分析。其下的 conda 可以用来管理 Python 的包和虚拟环境, 方便安装、更新、卸载工具包, 在安装包的同时会自动安装相应的依赖, 并且可以做到虚拟环境的创建以隔离不同要求的项目。

Anaconda 可以直接从官网 www.anaconda.com 中进行下载, 选择与自己系统匹配且熟悉的 Python 版本, 按照图 1 和图 2 的步骤进行安装。

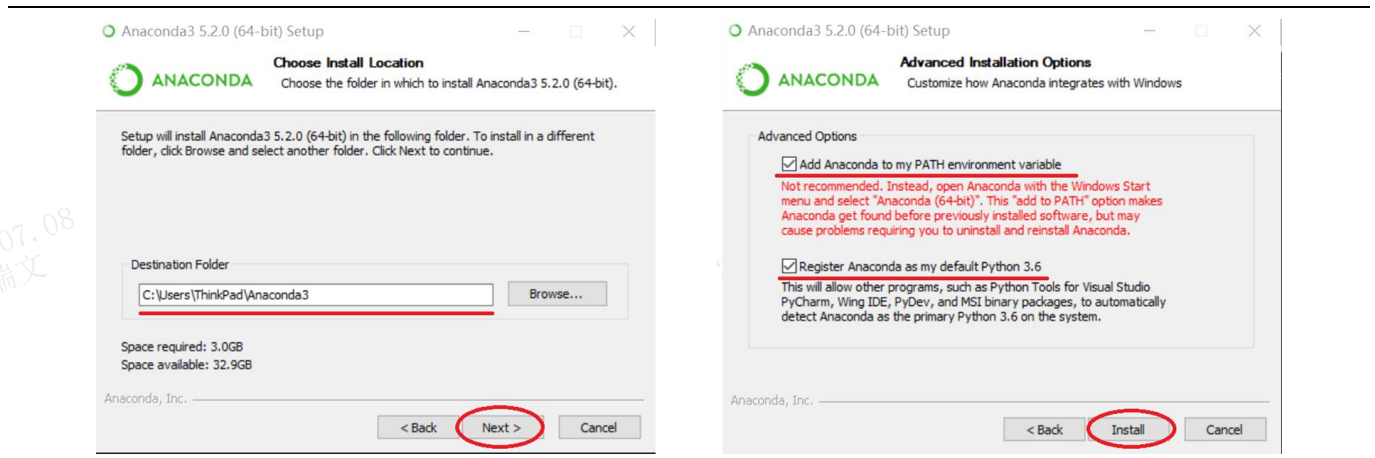
图 1: Anaconda 的安装步骤 1



资料来源: 长江证券研究所

在设置高级选项时, 建议将两个选项均打勾。第一个选项是为了配置 Anaconda 的环境变量, 使我们在命令行可以直接使用 conda 命令; 第二个选项是为我们设置默认的 Python 环境, 即 Anaconda 自带的 Python, 方便我们配置一些 API 接口, 如 Wind 的 Python 量化接口。

图 2: Anaconda 的安装步骤 2



资料来源：长江证券研究所

Python 环境的配置

在安装了 Anaconda 之后，就可以利用 conda 对 Python 的环境进行相应的配置，以实现不同工程的管理。conda 可以在命令行完成创建环境、切换环境、移除环境、查看环境、安装包、卸载包、更新包、查看包等任务。表 1 列举出了 conda 相关操作的命令语句。

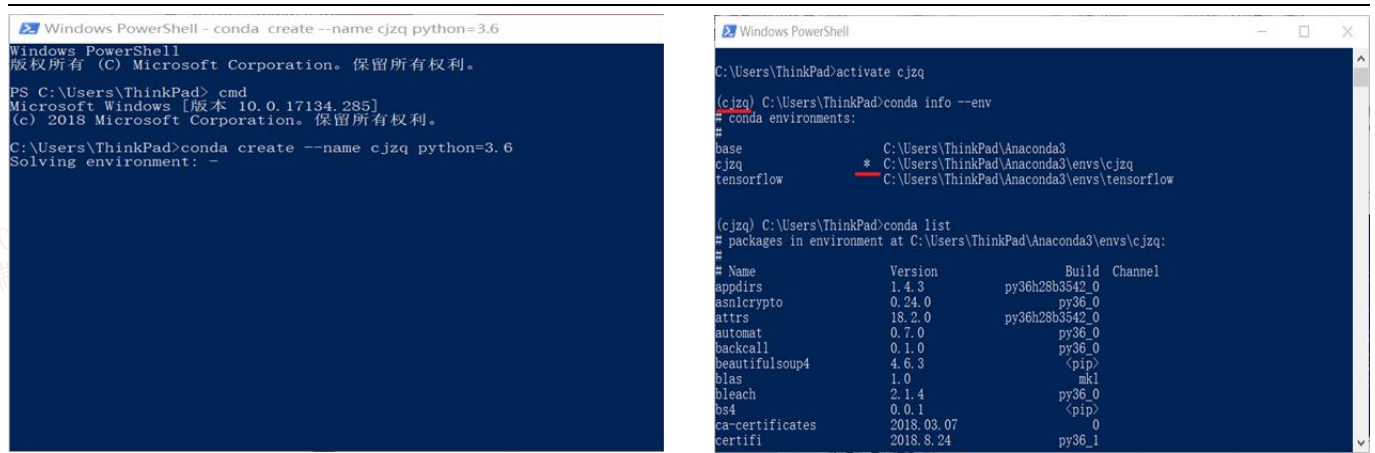
表 1: conda 命令

	操作	命令
环境管理	创建新的环境	conda create --name cjzq Python=3.6
	查看本地环境	conda info --env
	激活环境	activate cjzq
	移除环境	conda remove -n cjzq --all
包管理	当前环境下的包	conda list
	安装相关包	conda install numpy
	更新相关包	conda update numpy; conda update --all
	移除相关包	conda uninstall numpy

资料来源：长江证券研究所

在图 3、图 4 中展示了 conda 在命令行中创建虚拟环境、进入环境、安装包的相关操作界面，在图 4 中我们以 numpy 为例，展示了 Python 包的安装过程。常用的包有 pandas 和 numpy，负责数据处理；matplotlib 和 seaborn，负责数据可视化；scipy 和 statsmodel，负责相关数学运算；scikit-learn，负责建立机器学习模型；urllib 和 bs4，负责网络爬虫；notebook，一个基于网页的交互式 shell。

图 3: conda 环境操作 1



```

Windows PowerShell - conda create --name cjzq python=3.6
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

PS C:\Users\ThinkPad> cmd
Microsoft Windows [版本 10.0.17134.285]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\ThinkPad>conda create --name cjzq python=3.6
Solving environment: -

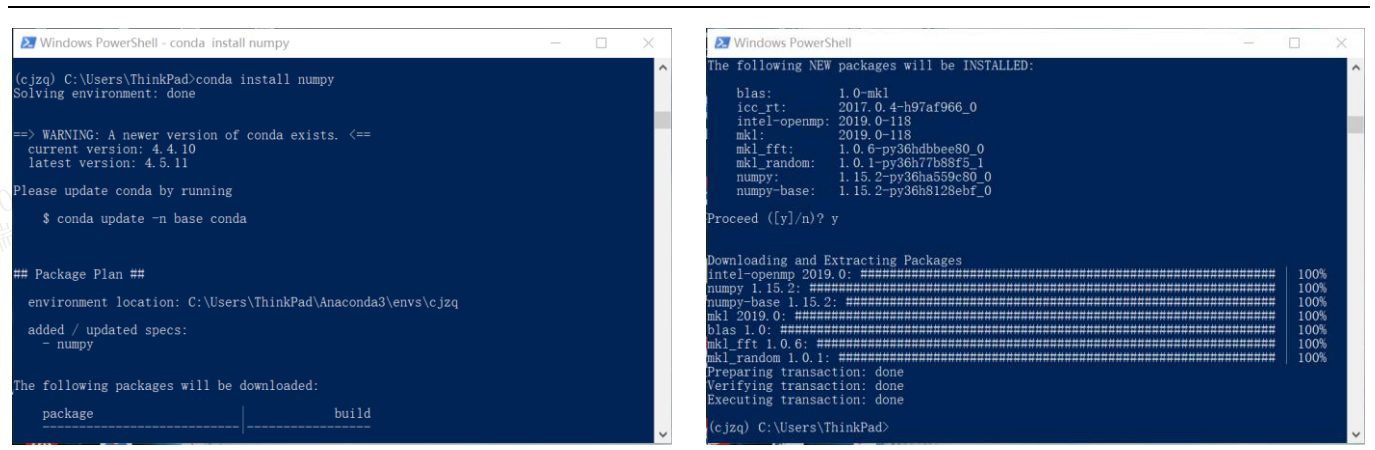
C:\Users\ThinkPad>activate cjzq

(cjzq) C:\Users\ThinkPad>conda info --env
# conda environments:
#
base                  C:\Users\ThinkPad\Anaconda3
cjzq                  * C:\Users\ThinkPad\Anaconda3\envs\cjzq
tensorflow            C:\Users\ThinkPad\Anaconda3\envs\tensorflow

(cjzq) C:\Users\ThinkPad>conda list
# packages in environment at C:\Users\ThinkPad\Anaconda3\envs\cjzq:
#
# Name                    Version            Build                Channel
#-----
appdirs                   1.4.3              py36h28b3542_0      py36_0
asn1crypto                0.24.0             py36_0
attrs                     18.2.0             py36h28b3542_0      py36_0
automat                   0.7.0              py36_0
backcall                  0.1.0              py36_0
beautifulsoup4            4.6.3              <pip>
blas                      1.0                mk1
bleach                    2.1.4              py36_0
bs4                       0.0.1              <pip>
ca-certificates           2018.03.07         0
certifi                   2018.8.24          py36_1
    
```

资料来源：长江证券研究所

图 4: conda 环境操作 2



```

Windows PowerShell - conda install numpy
(cjzq) C:\Users\ThinkPad>conda install numpy
Solving environment: done

==> WARNING: A newer version of conda exists. <==
current version: 4.4.10
latest version: 4.5.11

Please update conda by running
$ conda update -n base conda

## Package Plan ##

environment location: C:\Users\ThinkPad\Anaconda3\envs\cjzq
added / updated specs:
- numpy

The following packages will be downloaded:

package ----- build
numpy 1.15.2: ##### 100%
numpy-base 1.15.2: ##### 100%
mk1 2019.0: ##### 100%
blas 1.0: ##### 100%
mk1-fft 1.0.6: ##### 100%
mk1-random 1.0.1: ##### 100%

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(cjzq) C:\Users\ThinkPad>

The following NEW packages will be INSTALLED:
blas: 1.0-mk1
icc_rt: 2017.0.4-h97af966_0
intel-openmp: 2019.0-113
mk1: 2019.0-113
mk1-fft: 1.0.6-py36hdbbbee80_0
mk1-random: 1.0.1-py36h77b88f5_1
numpy: 1.15.2-py36ha559c80_0
numpy-base: 1.15.2-py36h8128ebf_0

Proceed ([y]/n)? y

Downloading and Extracting Packages
numpy 1.15.2: ##### 100%
numpy-base 1.15.2: ##### 100%
mk1 2019.0: ##### 100%
blas 1.0: ##### 100%
mk1-fft 1.0.6: ##### 100%
mk1-random 1.0.1: ##### 100%

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(cjzq) C:\Users\ThinkPad>
    
```

资料来源：长江证券研究所

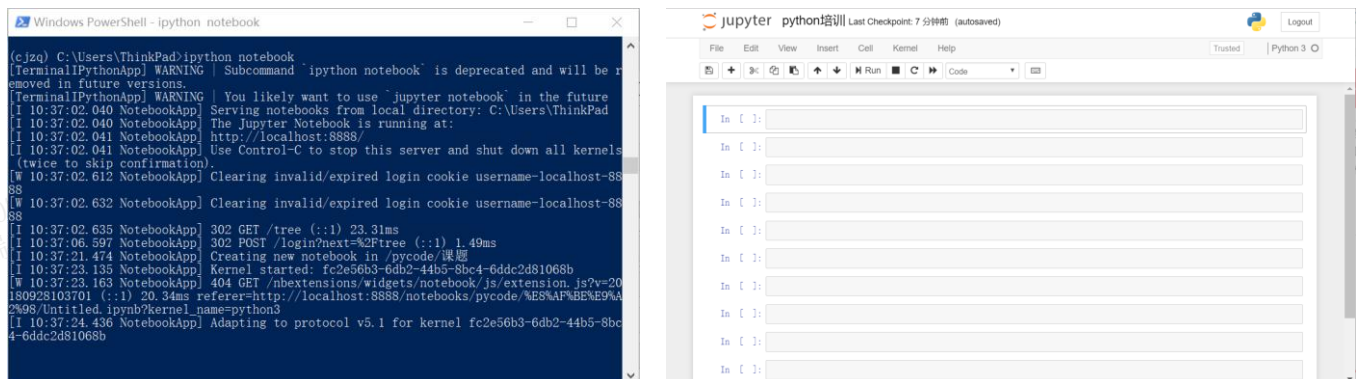
基础操作

Python 作为一门面向对象编程的、交互式、解释性语言，可以实现相应的计算机程序开发功能，其设计的程序有很强的可读性。本节将从介绍 Python 语言中的常规操作开始，包括基本的数据类型和控制流语句，并在后文继续介绍数据处理方面的内容。

使用 ipython notebook

ipython notebook 是一个强大的交互式 shell，它基于浏览器实现交互式操作，输入代码后可以看到运行的结果，支持代码、文本、数学运算、内嵌画图等，并可以将结果保存为 .py、.ipynb、.html、.pdf 等常用格式的文件。其进入方式和界面如图 5 展示。后文的 Python 功能介绍和代码展示，均基于 ipython notebook。

图 5：进入 ipython notebook



资料来源：长江证券研究所

基本语言和功能

本节主要介绍 Python 的基础数据类型有数字 (num)、字符串 (str)、列表 (list)、元组 (tuple)、字典 (dictionary) 和集合 (set)。Python 中直接以等号给变量赋值，其中数字类型用于储存数值，支持整型、长整型、浮点型和复数型；字符串一般以单引号或双引号进行创建，在 Python 中不支持单字符类型，单字符也是一个字符串；Python 中有多种序列的内置类型，较常用的有列表和字典；除此之外，还有集合类型。图 6 展示了基本的打印功能和上述六种数据的基本赋值方法。

图 6：Python 基本语言和功能

```
print("hello, python!")
print('%10s' % 'hello, python!')
print('%1f' % 1.23)
print('%d' % 1.23)

hello, python!
hello, pyt
1.2
1

# 多个类型的赋值，数值、字符、列表、元组、字典、集合
num1, num2, str1, str2, list1, list2, tuple1, tuple2, dict1, dict2, set1, set2 = 1, 1/3, "a", 'hello, python!', \
    [], [1, 2, 3], 0, (1, 2, 3), {}, {'a': 1, 'b': 2}, set(), {1, 2, 3}

# 格式化输出及相关信息
print("The value and type of num1 and num2 are {0}, {1} and {2}, {3}".format(num1, type(num1), num2, type(num2)))
print("The value and type of str1 and str2 are {0}, {1} and {2}, {3}".format(str1, type(str1), str2, type(str2)))
print("The value and type of list1 and list2 are {0}, {1} and {2}, {3}".format(list1, type(list1), list2, type(list2)))
print("The value and type of tuple1 and tuple2 are {0}, {1} and {2}, {3}".format(tuple1, type(tuple1), tuple2, type(tuple2)))
print("The value and type of dict1 and dict2 are {0}, {1} and {2}, {3}".format(dict1, type(dict1), dict2, type(dict2)))
print("The value and type of set1 and set2 are {0}, {1} and {2}, {3}".format(set1, type(set1), set2, type(set2)))

The value and type of num1 and num2 are 1, <class 'int'> and 0.3333333333333333, <class 'float'>
The value and type of str1 and str2 are a, <class 'str'> and hello, python!, <class 'str'>
The value and type of list1 and list2 are [], <class 'list'> and [1, 2, 3], <class 'list'>
The value and type of tuple1 and tuple2 are (), <class 'tuple'> and (1, 2, 3), <class 'tuple'>
The value and type of dict1 and dict2 are {}, <class 'dict'> and {'a': 1, 'b': 2}, <class 'dict'>
The value and type of set1 and set2 are set(), <class 'set'> and {1, 2, 3}, <class 'set'>
```

资料来源：长江证券研究所

List 的基本操作

List 是 Python 中最常见的序列数据类型，是一些遍历、循环操作的基础，作为一个方括号内的逗号分隔值出现，可以通过下标引用元素。和很多语言中的容器不同，Python 中的列表不需要元素为相同的数据类型。图 7 展示了包括 List 的赋值、添加元素、删除元素等简单操作。

图 7: List 的基本操作

```
# list的赋值
l1 = [0, 1, 2, 3, 4]
print(l1)
l2 = [5] * 10
print(l2)
l3 = list(range(10))
print(l3)
l4 = [pow(i, 2) for i in range(10)] # 条件表达式
print(l4)

# list的添加、删除操作
print(l1)
l1.append(5)
print(l1)
l1.extend([3, 4])
print(l1)
l1.sort(reverse=False)
print(l1)
l1.reverse()
print(l1)
print(l1[2])
print(l1[2:4])
print(l1[-2:])

[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4, 5]
[0, 1, 2, 3, 4, 5, 3, 4]
[5, 4, 4, 3, 3, 2, 1, 0]
4
[4, 3]
[1, 0]

l1 = [0, 1, 2, 3, 3, 4, 4, 5]
# remove删除首个符合条件的元素
l1.remove(4)
print(l1)
l1 = [0, 1, 2, 3, 3, 4, 4, 5]
# pop按照索引删除，并返回索引对应的元素
a = l1.pop(4)
print(a)
print(l1)
# del按照索引删除，并可以删除多个元素
l1 = [0, 1, 2, 3, 3, 4, 4, 5]
del(l1[4])
print(l1)
l1 = [0, 1, 2, 3, 3, 4, 4, 5]
del(l1[4:6])
print(l1)
# 利用条件表达式删除多个满足条件的元素
l1 = [0, 1, 2, 3, 3, 4, 4, 5]
l1 = [i for i in l1 if i != 3]
print(l1)

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

资料来源：长江证券研究所

Dict 和 Set 的基本操作

Dict 和 Set 是另外两种可变容器模型，其中 Dict 以键对值的形式进行存储，每个键和值之间用冒号分割作为一个元素，每个元素之间用逗号分割，所有元素包括在大括号内。Set 为集合，即无序不重复元素集合，也用大括号包含所有元素。图 8 展示了常用的 Dict 和 Set 操作。

图 8: Dict 和 Set 的基本操作

```
d1 = {'a': 1, 'b': 2, 'c': 3, 1: 'a', 2: 'b'}
print(d1.keys(), d1.values())
print("通过dict索引和i返回两个对应的元素i和i".format('a', 1, d1['a'], d1[1]))

# dict的赋值
d1['d'] = 4
d1['a'] = 4
print(d1)

dict_keys(['a', 'b', 'c', 1, 2]) dict_values([1, 2, 3, 'a', 'b'])
通过dict索引a和1返回两个对应的元素1和a
{'a': 4, 'b': 2, 'c': 3, 1: 'a', 2: 'b', 'd': 4}

# dict的删除用del和pop，类似list
d1 = {'a': 1, 'b': 2, 'c': 3, 1: 'a', 2: 'b'}
del d1['b']
print(d1)
d1 = {'a': 1, 'b': 2, 'c': 3, 1: 'a', 2: 'b'}
a = d1.pop('b')
print(a, d1)

{'a': 1, 'c': 3, 1: 'a', 2: 'b'}
2 {'a': 1, 'c': 3, 1: 'a', 2: 'b'}

a = {'a', 'b', 'c', 'd'}
b = {'c', 'd', 'e', 'f'}
# 交集
print(a & b)
# 并集
print(a | b)
# 差集
print(a - b)
a.add('z')
print(a)
a.remove('d')
print(a)

{'d', 'c'}
{'d', 'c', 'f', 'a', 'e', 'b'}
{'a', 'b'}
{'d', 'c', 'a', 'z', 'b'}
{'c', 'a', 'z', 'b'}
```

资料来源：长江证券研究所

时间日期的基本操作

由于金融数据数据源的不同，相同数据类型的数据却有不同格式，如日期。Python 可以通过时间和日期的操作，将格式统一，转换成固定形式的字符串、整数或时间戳，还可以进行获取当前时间、时间计算等操作。图 9 展示了 Python 中常用的时间日期操作。

图 9：时间日期的基本操作

```

date1 = date(2005, 5, 28)
time1 = time(22, 30, 50)
datetime1 = datetime(2005, 5, 28, 22, 30, 50)
print("The value and type of date1, time1 and datetime1 are {0} {1}, {2} {3} and {4} {5}.".format(date1, type(date1), time1, type(time1), datetime1, type(datetime1)))

The value and type of date1, time1 and datetime1 are 2005-05-28 <class 'datetime.date'>, 22:30:50 <class 'datetime.time'> and 2005-05-28 22:30:50 <class 'datetime.datetime'>

date2 = date1.strftime('%Y-%m-%d') # str(date1)
time2 = time1.strftime('%H:%M:%S') # str(time1)
datetime2 = datetime1.strftime('%Y-%m-%d %H:%M:%S') # str(datetime1)
print("The value and type of date2, time2 and datetime2 are {0} {1}, {2} {3} and {4} {5}.".format(date2, type(date2), time2, type(time2), datetime2, type(datetime2)))

The value and type of date2, time2 and datetime2 are 2005-05-28 <class 'str'>, 22:30:50 <class 'str'> and 2005-05-28 22:30:50 <class 'str'>

date3 = datetime.strptime(date2, '%Y-%m-%d').date()
time3 = datetime.strptime(time2, '%H:%M:%S').time()
datetime3 = datetime.strptime(datetime2, '%Y-%m-%d %H:%M:%S')
print("The value and type of date3, time3 and datetime3 are {0} {1}, {2} {3} {4} {5}.".format(date3, type(date3), time3, type(time3), datetime3, type(datetime3)))
print("比较日期1系列和日期3系列是否等价 {0} {1} {2}.".format(date1 == date3, time1 == time3, datetime1 == datetime3))

The value and type of date3, time3 and datetime3 are 2005-05-28 <class 'datetime.date'>, 22:30:50 <class 'datetime.time'> and 2005-05-28 22:30:50 <class 'datetime.datetime'>
比较日期1系列和日期3系列是否等价 True True True

date4 = date(2015, 6, 15)
time4 = time(8, 53, 23)
datetime4 = datetime(2015, 6, 15, 8, 53, 23)
timedelta1 = date4 - date1
timedelta2 = datetime4 - datetime1
print(timedelta1)
print(timedelta2)

3670 days, 0:00:00
3669 days, 10:22:33
    
```

资料来源：长江证券研究所

控制流

Python 中的程序代码按照顺序从上向下执行，为了改变程序的运行逻辑或顺序，以及循环实现代码块，可以通过控制流语句控制代码的执行方式。Python 中有三种常用的控制流，一种为分支结构，由 if 语句组成，两种为循环结构，分别由 for 和 while 语句组成。

图 10 展示了常用控制流语句的使用方法。

图 10：Python 控制流语句

```

num = 3
if num % 4 == 0:
    if num == 4:
        print("为整数4")
    else:
        print("可以整4整除")
elif num % 2 == 0:
    print("可以被2整除")
else:
    print("为奇数")

为奇数

num_list, even_list, odd_list = [1, 3, 5, 10, 92, 523], [], []
while len(num_list) > 0:
    num = num_list.pop()
    if (num % 2 == 0):
        even_list.append(num)
    else:
        odd_list.append(num)
print(even_list)
print(odd_list)

[92, 10]
[523, 5, 3, 1]

sums = 0
for i in range(10):
    if i % 3 == 0:
        sums += i
print(sums)
print(sum([i for i in range(10) if i % 3 == 0]))

for i in "python":
    if i == 't':
        print("目标字母存在")
        break

18
18
目标字母存在

prime_list, i = [], 2
while(i < 100):
    j = 2
    while(j <= i/j):
        if i % j == 0:
            break
        j = j + 1
    if (j > i/j):
        prime_list.append(i)
    i = i + 1
print(prime_list)

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]
    
```

资料来源：长江证券研究所

numpy 和 pandas

Python 在计算机程序开发方面有着较高的效率，在处理科学计算、数据分析和数据科学方面也有着方便快速的特点，而相关数据处理的基础便是 numpy 和 pandas 两个包，它们均采用 C 语言编写，有速度快操作简单的特点，后文介绍的数学建模的相关包也都大部分基于它们构建的数据结构。本节主要介绍它们的基本功能，首先这里要介绍 Python 中包的导入，有三种方式：

- `import name`; 命令为导入名为 `name` 中的所有方法和函数, 在之后的程序中以 `name.fun` 的形式引用包的内容。
- `import name as abbr`; 命令为导入名为 `name` 中的所有方法和函数, 并以 `abbr` 作为其简称, 在之后的程序中以 `abbr.fun` 的形式引用包的内容。
- `from name import *`; 命令为导入名为 `name` 中所有方法和函数, 在之后的程序中以 `fun` 的形式引用包的内容。

本文以第二种形式进行包的导入, 简化包中内容的引用, 同时可以避免不同包之间的函数重名。

numpy 的基本操作

numpy 是 Python 数值计算最重要的基础包, 大部分提供数据科学计算的包都以 numpy 的数组作为构建基础。因为 numpy 是在一个连续的内存块中储存数据, 独立于其他 Python 内置对象, 且在整个数组上直接进行运算而不需要 Python 的 for 循环, 可以高效处理大数组的数据。图 11 展示了 numpy 中数组的常用操作。

图 11: numpy 的基本操作 1

```
data = np.random.randint(0, 10, (4, 5))
print("构造数组的维数为 {}, 维度为 {}".format(data.ndim, data.shape))
print(data)
# 和标量的运算为对所有元素做同一处理, 数组运算时, 为同一位置元素进行处理
print(data / 10)
print(data + 5)

构造数组的维数为2, 维度为(4, 5)
[[8 9 2 3 8]
 [3 4 2 2 2]
 [7 3 7 3 8]
 [5 9 4 5 8]]
[[0.8 0.9 0.2 0.3 0.8]
 [0.3 0.4 0.2 0.2 0.2]
 [0.7 0.3 0.7 0.3 0.8]
 [0.5 0.9 0.4 0.5 0.8]]
[[13 14 7 8 13]
 [ 8 9 7 7 7]
 [12 8 12 8 13]
 [10 14 9 10 13]]

data = np.array([[1., 2., 3.], [4., 5., 6.]])
print(data)
print(data * data)
print(data + data)

[[1. 2. 3.]
 [4. 5. 6.]]
[[1. 4. 9.]
 [16. 25. 36.]]
[[2. 4. 6.]
 [8. 10. 12.]]

# reshape为根据顺序重塑数组, -1代表自适应
data = np.arange(0, 2.4, 0.1).reshape(4,-1)
# 以二维数组为例, 每个维度遵从基本的索引方法
print("选取数组的坐标(1,1)元素为 {}, 选取数组的第2列为 {}".format(data[0,0],
print(data[np.ix_([0,1], [2,3,5])])
print(data[[0,1], [3,5]])

选取数组的坐标(1,1)元素为0.0, 选取数组的第2列为[0.1 0.7 1.3 1.9]
[[0.2 0.3 0.5]
 [0.8 0.9 1.1]]
[0.3 1.1]
```

资料来源: 长江证券研究所

图 12: numpy 的基本操作 2

```
# 数组的赋值
old_values = data[0].copy()
data[0] = 5
print(data)
data[0] = old_values
old_values = data[np.ix_([0,1], [2,3,5])]
data[np.ix_([0,1], [2,3,5])] = np.ones(6).reshape(2,3)
print(data)

[[5. 5. 5. 5. 5. 5.]
 [0.6 0.7 0.8 0.9 1. 1.]
 [1.2 1.3 1.4 1.5 1.6 1.7]
 [1.8 1.9 2. 2.1 2.2 2.3]]
[[0. 0.1 1. 1. 0.4 1.]
 [0.6 0.7 1. 1. 1. 1.]
 [1.2 1.3 1.4 1.5 1.6 1.7]
 [1.8 1.9 2. 2.1 2.2 2.3]]

# 结合外部条件选取特定的元素
data = np.random.randn(7, 5)
index_name = np.array(['000001.SZ', '600617.SH', '000001.SZ', '300010.SZ',
                        '600617.SH', '002421.SZ', '000001.SZ'])
print(data[index_name == '000001.SZ', :])
print(data[np.where(index_name == '000001.SZ'), :])
print(np.where(index_name == '000001.SZ', 1, 2))

[[-1.24872857 -0.73812491 -2.53287989 1.09871372 -0.15054773]
 [-0.63985089 -0.07322705 -0.04754616 -0.28613658 0.76308134]
 [0.88953853 -1.58322985 0.1631085 1.45170928 -0.85697831]]
[[-1.24872857 -0.73812491 -2.53287989 1.09871372 -0.15054773]
 [-0.63985089 -0.07322705 -0.04754616 -0.28613658 0.76308134]
 [0.88953853 -1.58322985 0.1631085 1.45170928 -0.85697831]]
[[1 2 1 2 2 1]]

# 数组的相关计算
data = np.random.randn(2, 3)
data1 = np.random.randn(2, 3)
print(data)
print(data1)
print(np.dot(data.T, data))
print(np.exp(data))
print(np.maximum(data, data1))
print(np.sum(data, axis = 1))

[[-0.41199298 0.15267569 -0.63644252]
 [-1.34919013 -0.85714314 1.67081013]]
[[-0.63644252 -0.41199298 0.15267569]
 [-1.34919013 -0.85714314 1.67081013]]
[[-0.63644252 -1.34919013]
 [-0.41199298 -0.85714314]
 [0.15267569 1.67081013]]
[[2.22537309 1.41865891 -2.35140983]
 [1.41865891 0.90443257 -1.49502474]
 [-2.35140983 -1.49502474 2.81491634]]
[[0.5291716 0.66232893 1.16494711]
 [0.2594503 0.42437273 5.31647306]]
[[2.61525643 -0.41199298 1.27304492]
 [-1.34919013 -0.30879299 1.67081013]]
[-0.89575981 -0.53552315]]

np.save("D:\\DATA\\课题\\python培训\\data1.npy", data)
data = np.load("D:\\DATA\\课题\\python培训\\data1.npy")
```

资料来源: 长江证券研究所

numpy 的数组是一个快速灵活的大数据集容器，可以利用这种数组对整块数据执行一些数学运算，其中线性代数的矩阵运算为 numpy 下的 linalg 模块，可以进行如 QR 分解、线性方程组求解、特征根和特征向量求解等运算。图 13 展示了 numpy 中线性代数矩阵运算的常用操作。

图 13: numpy 的矩阵计算

```
# 线性代数
data = np.random.randn(2, 3)
# 点乘
mat = data.T.dot(data)
print(mat)
# 逆
print(np.linalg.inv(mat))
# QR 分解
q, r = np.linalg.qr(mat)
print(q)
print(r)

[[ 2.90049496  0.23014396 -0.65178828]
 [ 0.23014396  0.85493284 -0.11976298]
 [-0.65178828 -0.11976298  0.15200153]]

[[ -7.93473804e+15 -2.95665303e+15 -3.63541080e+16]
 [-2.95665303e+15 -1.10170935e+15 -1.35462837e+16]
 [-3.63541080e+16 -1.35462837e+16 -1.66560992e+17]]

[[ -0.97275839  0.09248733  0.21257282]
 [-0.07718492 -0.9938654  0.07920898]
 [ 0.21859459  0.06064378  0.97392953]]

[[ -2.98172185e+00 -3.16041933e-01  6.76503128e-01]
 [ 0.00000000e+00 -8.35665644e-01  6.79640759e-02]
 [ 0.00000000e+00  0.00000000e+00  1.20444727e-17]]

# SVD 分解
u, s, v = np.linalg.svd(data)
print(u)
print(s)
print(v)
# 线性方程组
a = np.array([[3, 1], [1, 2]])
b = np.array([9, 8])
x = np.linalg.solve(a, b)
print(x)
# 特征根特征向量
w, v = np.linalg.eig(np.diag((1, 2, 3)))
print(w)
print(v)

[[ 0.09896578  0.99509084]
 [ 0.99509084 -0.09896578]]
[ 0.99509084  0.91205838]
[ 1.75373283  0.91205838]
[ -0.96887582 -0.11231016  0.22060388]
[ -0.12685599  0.99051116 -0.05286958]
[ 0.21257282  0.07920898  0.97392953]]
[2. 3.]
[1. 2. 3.]
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

资料来源：长江证券研究所

pandas 的基本操作

pandas 基于 numpy 数组格式构建，专门用来处理表格和混杂数据，可以对数据快速进行清洗、分析，并保证了数据计算的高效。pandas 有两大基本数据结构：Series 和 DataFrame，Series 是一种类似于二维数组的对象，由 numpy 数据类型的数组和与之相关的数据标签（即索引）组成；DataFrame 是一个表格型的数据结构，由 numpy 数据类型的数组和与之对应的行索引、列索引组成，也可以看作是多组 Series 数据的合成。图 14 展示了 Series 和 DataFrame 的简单建立过程和其数据结构。

图 14: pandas 的数据结构

```
# pandas 中两个基本数据：Series 和 DataFrame
se = pd.Series([1, 2, 3, 4], index = [1, 2, 3, 4])
print(se)

# Series 的 index
print(se.index)
# dict 格式可以转换为 Series 格式
d = {'a': 3000, 'b': 4000, 'c': 5000, 'd': 6000}
se = pd.Series(d)
print(se)

df = pd.DataFrame(np.random.randn(4, 5), columns = ['a', 'b', 'c', 'd', 'e'])
print(df)
# DataFrame 的 index 和 columns，DataFrame 的每一列和每一行都是 Series 属性
print(df.index, df.columns)

# DataFrame 的索引
print(df['a'])
# iloc 用于选取 DataFrame 的特定行
print(df.iloc[0])
# dict 格式也可以转换为 DataFrame 格式
d = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada', 'Nevada'],
     'year': [2000, 2001, 2002, 2001, 2002, 2003],
     'pop': [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}
df = pd.DataFrame(d)
print(df.head(3))

0 -0.799468
1 -0.523579
2 -0.194974
3 0.983053
Name: a, dtype: float64
a -0.799468
b -0.096657
c -1.713748
d 1.295196
e 0.486324
Name: 0, dtype: float64
state year pop
0 Ohio 2000 1.5
1 Ohio 2001 1.7
2 Ohio 2002 3.6
```

资料来源：长江证券研究所

由于 Series 和 DataFrame 的数据由 numpy 的数组构成，所以对于数据的操作和 numpy 类似，图 15 和图 16 分别对 pandas 中选取特定数据和数据赋值给出了相关展示。

图 15: pandas 的数据选取

```
# DataFrame选取特定条件的
df = pd.DataFrame(np.random.randn(4,5), index = [1,3,5,2], columns = ['a', 'b', 'c', 'd', 'e'])
# iloc用于选取特定行数和列数的数据
print(df.iloc[2, [2,4]])
# loc用于选取特定index和columns
print(df.loc[[1, 3], ['a', 'e']])
# loc还可以用于选取特定条件的数据
print(df.loc[(df['a'] > 0) & (df['e'] < 0)])
print(df.loc[:, (df.iloc[0] > 0) & (df.iloc[1] < 0)])
# reindex用于选取特定已有的index, 没有会报错
print(df.reindex([1,2,3]))
```

	c	e			
1	1.431523	1.174209			
3	-0.470742	-0.466155			
	a	e			
1	0.525592	1.174209			
3	0.747547	-0.466155			
	a	b	c	d	e
3	0.747547	-0.658429	-0.470742	-0.219523	-0.466155
5	1.183531	0.434923	-1.126447	1.412356	-0.439096
	c	e			
1	1.431523	1.174209			
3	-0.470742	-0.466155			
5	-1.126447	-0.439096			
2	1.973935	-0.336457			
	a	b	c	d	e
1	0.525592	-1.592360	1.431523	-2.315102	1.174209
2	-0.324626	1.023367	1.973935	-0.995197	-0.336457
3	0.747547	-0.658429	-0.470742	-0.219523	-0.466155

资料来源: 长江证券研究所

图 16: pandas 的数据赋值

```
# DataFrame的赋值
d = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada', 'Nevada'],
      'year': [2000, 2001, 2002, 2001, 2002, 2003],
      'pop': [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}
df = pd.DataFrame(d, index = [1,2,3,4,5,6])
print(df)

# 简单的赋值
df['pop'] = 3
df['debt'] = range(6)
print(df)
```

	state	year	pop
1	Ohio	2000	1.5
2	Ohio	2001	1.7
3	Ohio	2002	3.6
4	Nevada	2001	2.4
5	Nevada	2002	2.9
6	Nevada	2003	3.2

```
# 直接将series赋值给一列
se = se = pd.Series(np.random.randn(5), index = [1,2,4,5,7])
df['old'] = se
print(df)

# 条件性赋值
df['new'] = 0
df.loc[df['old'] < 0, 'new'] = -1
df.loc[df['old'] > 0, 'new'] = 1
print(df)
```

	state	year	pop	debt	old
1	Ohio	2000	3	0	-0.921457
2	Ohio	2001	3	1	0.510676
3	Ohio	2002	3	2	NaN
4	Nevada	2001	3	3	-0.195567
5	Nevada	2002	3	4	1.585790
6	Nevada	2003	3	5	NaN

	state	year	pop	debt	old	new
1	Ohio	2000	3	0	-0.921457	-1
2	Ohio	2001	3	1	0.510676	1
3	Ohio	2002	3	2	NaN	0
4	Nevada	2001	3	3	-0.195567	-1
5	Nevada	2002	3	4	1.585790	1
6	Nevada	2003	3	5	NaN	0

资料来源: 长江证券研究所

针对 DataFrame 的行索引和列索引, 图 17 和图 18 给出了相关数据结构化处理的展示。

图 17: pandas 的数据结构化处理 1

```
# 填充空值
print(df.fillna(0.0))
# print(df.fillna(method = 'ffill'))
print(df.dropna())
df['old'] = df['old'].fillna(0.0)
print(df)
```

	state	year	pop	debt	old	new	new1
1	Ohio	2000	3	0	-0.921457	-1	-1
2	Ohio	2001	3	1	0.510676	1	1
3	Ohio	2002	3	2	0.000000	0	0
4	Nevada	2001	3	3	-0.195567	-1	-1
5	Nevada	2002	3	4	1.585790	1	1
6	Nevada	2003	3	5	0.000000	0	0

```
# DataFrame的数据结构化处理
# 转置
print(df.T)
# index的相关处理
df.index.name = 'index'
df.index = range(len(df))
print(df)
```

	state	year	pop	debt	old	new	new1
0	Ohio	2000	3	0	-0.548668	-1	-1
1	Ohio	2001	3	1	0.472348	1	1
2	Ohio	2002	3	2	0.000000	0	0
3	Nevada	2001	3	3	-1.002770	-1	-1
4	Nevada	2002	3	4	1.428989	1	1
5	Nevada	2003	3	5	0.000000	0	0

资料来源: 长江证券研究所

图 18: pandas 的数据结构化处理 2

```
# 重新赋予index
df = df.set_index('year')
print(df)
df = df.reset_index()
print(df)
```

	state	pop	debt	old	new	new1
year						
2000	Ohio	3	0	-0.548668	-1	-1
2001	Ohio	3	1	0.472348	1	1
2002	Ohio	3	2	0.000000	0	0
2001	Nevada	3	3	-1.002770	-1	-1
2002	Nevada	3	4	1.428989	1	1
2003	Nevada	3	5	0.000000	0	0

```
# df的列行数据提取
df_copy = df.pivot(columns='state', index='year', values='old')
print(df_copy)
print(df_copy.unstack().reset_index())
```

	state	Nevada	Ohio
year			
2000		NaN	-0.548668
2001		-1.002770	0.472348
2002		1.428989	0.000000
2003		0.000000	NaN

	state	year
0	Nevada	2000
1	Nevada	2001
2	Nevada	2002
3	Nevada	2003
4	Ohio	2000
5	Ohio	2001
6	Ohio	2002
7	Ohio	2003

资料来源：长江证券研究所

图 19 和图 20 展示了 pandas 中数据最简单的相关运算，以及 apply 函数的操作。apply 函数是 pandas 中非常重要的方法，它避免了在处理数据时使用 python 的 for 循环结构，而采用多线程的并行处理方式，极大的提高了数据运算的速度。

图 19: pandas 的数据运算 1

```
# DataFrame的相关运算
df1 = pd.DataFrame(np.arange(9).reshape((3, 3)), columns=list('bcd'), index=['Ohio', 'Texas', 'Colorado'])
df2 = pd.DataFrame(np.arange(12).reshape((4, 3)), columns=list('bde'), index=['Utah', 'Oregon', 'Texas', 'Utah'])
print(df1)
print(df2)
print(df1 + df2)
print(df1.add(df2, fill_value=0))
```

	b	c	d
Ohio	0.0	1.0	2.0
Texas	3.0	4.0	5.0
Colorado	6.0	7.0	8.0

	b	d	e
Utah	0.0	1.0	2.0
Ohio	3.0	4.0	5.0
Texas	6.0	7.0	8.0
Oregon	9.0	10.0	11.0

	b	c	d	e
Colorado	NaN	NaN	NaN	NaN
Ohio	3.0	NaN	6.0	NaN
Oregon	NaN	NaN	NaN	NaN
Texas	9.0	NaN	12.0	NaN
Utah	NaN	NaN	NaN	NaN

	b	c	d	e
Colorado	6.0	7.0	8.0	NaN
Ohio	3.0	1.0	6.0	5.0
Oregon	9.0	NaN	10.0	11.0
Texas	9.0	4.0	12.0	8.0
Utah	0.0	NaN	1.0	2.0

```
# 多行的apply处理
print(df[['pop', 'old']].apply(lambda x: x.max() - x.min()))
print(df[['pop', 'old']].apply(lambda x: x.max() - x.min(), axis=1))
# 除了返回series之外，还可以返回series
def fun(x):
    return pd.Series([x.min(), x.max()], index=['min', 'max'])
print(df[['pop', 'old']].apply(fun))
# applymap用于整个Dataframe
print(df[['pop', 'old']].applymap(lambda x: 1 if x > 0 else 0))
```

	pop	old
pop	2.100000	1.573915
old	1.573915	2.100000
dtype: float64		
1	1.039472	1.966931
2	1.966931	3.000000
3	0.000000	4.220307
4	2.203074	5.159306
5	1.593016	0.000000
6	0.000000	0.000000
dtype: float64		
pop	old	
min	1.5	-0.266931
max	3.6	1.306994

	pop	old
1	1	1
2	1	0
3	1	0
4	1	1
5	1	1
6	1	0

资料来源：长江证券研究所

图 20: pandas 的数据运算 2

```
# DataFrame的计算
df = pd.DataFrame([[1.4, np.nan], [7.1, -4.5], [np.nan, np.nan], [0.75, -1.3], [4, 2], [2, 3]])
print(df)
# 均值、方差、最大、总计
print(df.mean(skipna=True))
print(df.max(axis=1))
```

	one	two
a	1.40	NaN
b	7.10	-4.5
c	NaN	NaN
d	0.75	-1.3
e	4.00	2.0
f	2.00	9.0
one	3.05	
two	1.30	
dtype: float64		
a	1.40	
b	7.10	
c	NaN	
d	0.75	
e	4.00	
f	9.00	
dtype: float64		

```
print(df.std())
# 描述统计
print(df.describe())
# 协方差矩阵
print(df.corr())
```

	one	two
one	2.570019	5.778697
two	5.778697	2.570019
dtype: float64		
count	5.000000	4.000000
mean	3.050000	1.300000
std	2.570019	5.778697
min	0.750000	-4.500000
25%	1.400000	-2.100000
50%	2.000000	0.350000
75%	4.000000	3.750000
max	7.100000	9.000000

	one	two
one	1.000000	-0.519169
two	-0.519169	1.000000

资料来源：长江证券研究所

pandas 也可以像 SQL 语言一样，对结构化数据进行相关的查询操作，图 21 展示了 pandas 中的 SQL 功能操作语法。

图 21: pandas 中的 SQL 操作

```
# 合并数据集
df1 = pd.DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'a', 'b'], 'data1': range(7)})
df2 = pd.DataFrame({'key': ['a', 'b', 'd'], 'data2': range(3)})
print(df1)
print(df2)
print(pd.merge(df1, df2, on='key', how='outer'))
# print(pd.merge(df1, df2, left_on='key', right_on='key', how='outer'))

# sql中的sort & groupby
d = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
     'year': [2000, 2001, 2002, 2001, 2002, 2003],
     'pop': [1.5, 1.7, 3.6, 2.4, 2.9, 3.2]}
df = pd.DataFrame(d, index = [1,2,3,4,5,6])
df['old'] = pd.Series(np.random.randn(6), index = [1,2,4,5,7])
print(df)
print(df.sort_values('year'))
print(df.groupby('year')[['pop', 'old']].mean())

key data1
0 b 1
1 b 2
2 a 3
3 c 3
4 a 4
5 a 5
6 b 6
key data2
0 a 0
1 b 1
2 d 2
key data1 data2
0 b 0.0 1.0
1 b 1.0 1.0
2 b 6.0 1.0
3 a 2.0 0.0
4 a 4.0 0.0
5 a 5.0 0.0
6 c 3.0 NaN
7 d NaN 2.0

state year pop old
1 Ohio 2000 1.5 -0.989758
2 Ohio 2001 1.7 -0.915027
3 Ohio 2002 3.6 NaN
4 Nevada 2001 2.4 -0.452760
5 Nevada 2002 2.9 -0.491168
6 Nevada 2003 3.2 NaN
state year pop old
1 Ohio 2000 1.5 -0.989758
2 Ohio 2001 1.7 -0.915027
4 Nevada 2001 2.4 -0.452760
3 Ohio 2002 3.6 NaN
5 Nevada 2002 2.9 -0.491168
6 Nevada 2003 3.2 NaN
year
2000 1.50 -0.989758
2001 2.05 -0.683894
2002 3.25 -0.491168
2003 3.20 NaN
```

资料来源：长江证券研究所

pandas 可以对接多种类型的数据，如 csv、excel、html、sas 数据等，还可以连接 MySQL 等数据库，图 22 展示了最简单的 csv 数据的读取和储存。

图 22: pandas 的数据存取

```
#数据的读取和保存
df = pd.read_csv(open("D:\\DATA\\课题\\python培训\\000300.csv"), index_col = 0)
print(df.head(4))
df['date'] = pd.to_datetime(df['date']).astype(str).apply(lambda x: x[:4] +
x[5:7] + x[-2:]).astype(int)
print(df.head(4))
df.to_csv("D:\\DATA\\课题\\python培训\\300.csv", index = False, encoding = 'gbk')

date wind_code sec_name
No.
1 2018/9/28 000001.SZ 平安银行
2 2018/9/28 000002.SZ 万科A
3 2018/9/28 000060.SZ 中金岭南
4 2018/9/28 000063.SZ 中兴通讯
date wind_code sec_name
No.
1 20180928 000001.SZ 平安银行
2 20180928 000002.SZ 万科A
3 20180928 000060.SZ 中金岭南
4 20180928 000063.SZ 中兴通讯
```

资料来源：长江证券研究所

画图

matplotlib 可以为 Python 构建一个 MATLAB 式的绘图接口，可以绘制常见的散点图、折线图、柱状图、分布图等，可将绘制的图形导出为常见的 JPG、PNG、PDF 等格式。其核心思想是通过将图构建为一个对象，对其中的属性通过方法进行赋值、操作。图 23 展示了 matplotlib 画图的常用操作，图 24 展示了画图效果。

图 23: matplotlib 画图

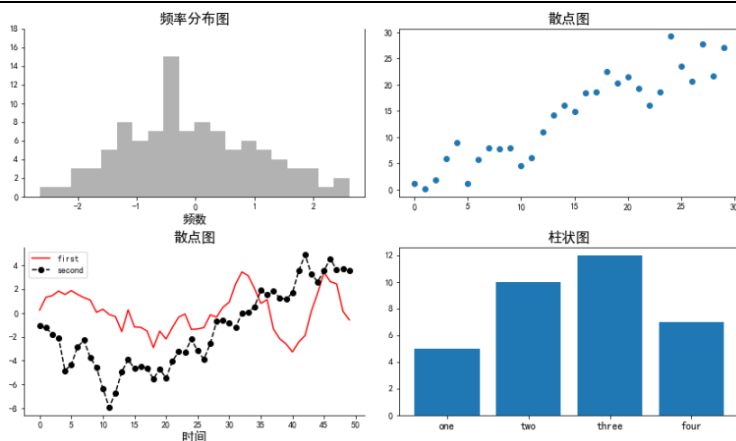
```
# 绘图应用
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
p = sns.color_palette()

# 设置字体
plt.rcParams['axes.unicode_minus'] = False
plt.rcParams['font.sans-serif'] = ['SimHei']

# 设置画布
fig = plt.figure()
# 画布大小
fig.set_size_inches(14, 8)
# 设置子画布
ax1 = fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(222)
ax3 = fig.add_subplot(2, 2, 3)
ax4 = fig.add_subplot(224)
# 分布图
ax1.hist(np.random.randn(100), bins=20, color='k', alpha=0.3)
# 设置x轴标题
ax1.set_xlabel('频率分布图', fontsize=16)
# 取消上框线
ax1.spines['top'].set_color('none')
# 设置x轴标题
ax1.set_xlabel('频率', fontsize=14)
# 散点图
ax2.scatter(np.arange(30), np.arange(30) + 3 * np.random.randn(30))
ax2.set_title('散点图', fontsize=16)
ax3.plot(np.random.randn(50).cumsum(), color='r', label='first')
ax3.plot(np.random.randn(50).cumsum(), color='k', linestyle='--', marker='o', label='sec')
ax3.set_xticks(list(range(0, 55, 5)))
ax3.set_title('散点图', fontsize=16)
# 设置图例
ax3.legend(loc='best')
ax3.spines['top'].set_color('none')
ax3.spines['right'].set_color('none')
ax3.set_xlabel('时间', fontsize=14)
# 柱状图
ax4.bar(range(4), [5, 10, 12, 7])
ax4.set_xticks(list(range(4)))
# 设置x轴标题
ax4.set_xlabel('one', 'two', 'three', 'four', rotation=0, fontsize=12)
ax4.set_title('柱状图', fontsize=16)
plt.subplots_adjust(left=0.1, bottom=None, right=None, top=None, wspace=0.1, hspace=0.3)
```

资料来源：长江证券研究所

图 24: matplotlib 画图结果



资料来源：长江证券研究所

模型运用

和数据处理对应，Python 在数学建模方面也有着很好的运用，本节主要从科学计算、传统的统计模型和传统的机器学习模型的建立上，介绍三个常用的包和相关功能的实现。

线性回归

statsmodels 是 Python 中的一个统计分析库，它包括统计检验、回归分析、时间序列等常用统计模型和相应结果分析，图 25 展示了线性回归的例子，像 R 一样 statsmodel 的回归可以展示相关的指标和检验结果，如变量的 t 值、模型的 R 方等。

图 25: statsmodels 线性回归

```
x = np.concatenate((np.linspace(1, 100, 100).reshape(100, -1), np.random.randint(0, 20, (100, 3)), np.random.randn(100, 2), np.repeat(1, 100).reshape(100, -1)), axis = 1)
# y = 1.5 * x1 + 2 * x2 - 3 * x3 + 6 * x4 - 40 * x5 + 100 * x6 + 4.6
beta = np.array([1.5, 2, -3, 6, -40, 100, 4.6])
y = np.dot(x, beta) + np.random.normal(size = 100)
# 线性回归
lr = sm.OLS(y, x)
res = lr.fit()
res.summary(alpha = 0.05)
```

OLS Regression Results

Dep. Variable:	y	R-squared:	1.000
Model:	OLS	Adj. R-squared:	1.000
Method:	Least Squares	F-statistic:	2.528e+05
Date:	Mon, 08 Oct 2018	Prob (F-statistic):	1.53e-193
Time:	11:21:40	Log-Likelihood:	-135.42
No. Observations:	100	AIC:	284.8
Df Residuals:	93	BIC:	303.1
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
x1	1.5001	0.004	425.338	0.000	1.493	1.507
x2	1.9887	0.017	118.326	0.000	1.955	2.022
x3	-3.0326	0.017	-182.790	0.000	-3.066	-3.000
x4	6.0090	0.019	319.267	0.000	5.972	6.046
x5	-39.9679	0.104	-385.372	0.000	-40.194	-39.782
x6	100.0280	0.095	1047.959	0.000	99.838	100.218
const	4.9546	0.372	13.321	0.000	4.216	5.693

Omnibus: 0.282 Durbin-Watson: 1.998
Prob(Omnibus): 0.868 Jarque-Bera (JB): 0.093
Skew: -0.079 Prob(JB): 0.955
Kurtosis: 3.053 Cond. No. 229

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

资料来源：长江证券研究所

机器学习

随着 sklearn(scikit-learn)、tensorflow、pytorch 和 keras 等包的完善，Python 成为在机器学习、深度学习开发中最受欢迎的语言之一。sklearn 是目前传统机器学习建模较常用的包，可以实现线性模型、树模型、聚类模型、卡尔曼滤波等。图 26、图 27 和图 28 分别给出了普通线性回归、套索回归和随机森林的简单实现。sklearn 中相关的线性模型可以给出模型的拟合参数，也可以对数据给出预测，但不会直接展示模型拟合的评价指标和变量的检验结果。

图 26: sklearn 实现普通线性回归

<pre> from sklearn.linear_model import LinearRegression x = np.concatenate((np.linspace(1, 100, 100).reshape(100, -1), np.random.randint(0, 20, (100, 3)), np.random.randn(100, 2), np.repeat(1, 100).reshape(100, -1)), axis = 1) # y = 1.5 * x1 + 2 * x2 - 3 * x3 + 6 * x4 - 40 * x5 + 100 * x6 + 4.6 beta = np.array([1.5, 2, -3, 6, -40, 100, 4.6]) e = np.random.normal(size = 100) y = np.dot(x, beta) + e x = x[:, :-1].copy() lr = LinearRegression(fit_intercept=True) lr.fit(x, y) print(lr.coef_) print(lr.intercept_) print(lr.score(x, y)) </pre>		<pre> [1.50535069 1.99069773 -3.00932759 5.96546934 -40.15903241 99.9254935] 4.910119068429751 0.9999405145940288 </pre>	
--	--	---	--

资料来源：长江证券研究所

图 27: sklearn 实现套索回归

```
from sklearn.linear_model import Lasso

x = np.concatenate((np.linspace(1, 100, 100).reshape(100, -1), np.random.randint(0, 20, (100, 3)),
                    np.random.randn(100, 2), np.repeat(1, 100).reshape(100, -1)), axis = 1)
# y = 1.5 * x1 + 2 * x2 - 3 * x3 + 6 * x4 - 40 * x5 + 100 * x6 + 4.6
beta = np.array([1.5, 2, -3, 6, -40, 100, 4.6])
e = np.random.normal(size = 100)
y = np.dot(x, beta) + e

x = np.concatenate((x[:, :-1].copy(), np.random.randint(0, 20, (100, 2))), axis = 1)

lasso = Lasso(alpha=1.0)
lasso.fit(x, y)

print(lasso.coef_)
print(lasso.intercept_)

[ 1.50268196e+00  2.00465081e+00 -2.98686103e+00  5.98747394e+00
 -3.86701388e+01  9.90852792e+01 -1.57005576e-02  0.00000000e+00]
4.567432937052132
```

资料来源：长江证券研究所

图 28: sklearn 实现随机森林

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification

X, y = make_classification(n_samples=1000, n_features=4, n_informative=2,
                          n_redundant=0, random_state=0, shuffle=False)

print(X.shape)
print(np.unique(y))

rf = RandomForestClassifier(n_estimators=100, max_depth=2)
rf.fit(X, y)

print(rf.feature_importances_)
print(rf.predict([[0, 0, 0, 0]]))

(1000, 4)
[0 1]
[0.12847387 0.79928183 0.0365843  0.03566   ]
[1]
```

资料来源：长江证券研究所

组合优化

scipy 用于处理科学计算中常见的问题，如插值、积分、聚类、优化，且可以计算一些简单的统计量。图 29 给出了组合优化的一个例子，已知股票收益和协方差矩阵，找出限定条件下最优收益的组合权重。

图 29: scipy 实现组合优化

```

from scipy.optimize import minimize

def CalReturn(weight, ret_matrix):
    weight = np.matrix(weight)
    return (weight * ret_matrix)[0,0]

def CalRisk(weight, cov_matrix):
    weight = np.matrix(weight)
    return (weight * cov_matrix * weight.T)[0, 0]

# 限制条件一，仓位配置不超过1
def constraint1(weight):
    return np.sum(weight) - 1.0

# 限制条件二，收益不得低于阈值
def constraint2(weight, ret_matrix, ret):
    weight = np.matrix(weight)
    return CalReturn(weight, ret_matrix) - ret

# 限制条件三，风险不得高于阈值
def constraint3(weight, cov_matrix, std):
    weight = np.matrix(weight)
    return std - CalRisk(weight, cov_matrix)

# 优化目标，使得在限制条件下，收益最大化
def ReturnObjective(weight, param):
    ret_matrix = param[0]
    weight = np.matrix(weight)
    return CalReturn(weight, ret_matrix)

# 协方差矩阵和收益矩阵
Cov = np.matrix([23.375 70.30; 37.5 122.72 13.5; 70.72 321 -32; 30 13.5 -32 52']) / 100
Ret = np.matrix([14; 12; 15; 7]) / 100

# 初始化权重
weight0 = np.matrix(np.repeat(1 / Cov.shape[0], Cov.shape[0]))
# 权重的范围，0到1之间
b = (0.0, 1.0)
bnds = tuple([b for i in range(Cov.shape[0])])

# 三个限制条件
cons1 = ['type': 'eq', 'fun': constraint1]
cons2 = ['type': 'ineq', 'fun': constraint2, 'args': [Ret, 0]]
cons3 = ['type': 'ineq', 'fun': constraint3, 'args': [Cov, 1]]
cons = [cons1, cons2, cons3]

# 优化器
res = minimize(ReturnObjective, weight0, args=[Ret], method='SLSQP',
               bounds=bnds, constraints=cons, options={'disp': True}, tol=1e-10)
weight_final = np.asmatrix(res.x)

print(weight_final)

Optimization terminated successfully. (Exit mode 0)
Current function value: -0.13959596959473872
Iterations: 12
Function evaluations: 73
Gradient evaluations: 12
[[[ 6.00909411e-01  1.26497876e-01  2.12592712e-01  1.56077388e-16]]]

```

资料来源：长江证券研究所

简单爬虫

爬虫就是请求网站并提取数据的自动化程序，由于 Python 强大的数据处理功能，以及对接网站相关包的完善，使得其在获取网页数据、整理爬取的数据上有一定的优势。爬虫一般可以分为四个步骤：发起请求、获取相应内容、解析内容、整理数据，常用的包有 urllib、BeautifulSoup、scrapy 等。由于爬虫的前三个部分都与网页本身有关，所以除了 Python 语言的处理外，还要对 html 语言有一定了解，图 30 和图 32 分别给出了带有表格数据、标题数据的两个网站的部分 html 结构，图 31 和图 33 分别爬取了两个网站对应 html 部分的相关数据。

图 30: 网站 1 部分 html 源码

```

$.each(list,function(index,array){ //遍历json数据列
    if (array.num) { array.num = '-'; }
    if (array.tclose) { array.tclose = '-'; }
    if (array.yld_mon) { array.yld_mon = '-'; }
    if (array.class_assets) { array.class_assets = '-'; }
    if (array.class_hot) { array.class_hot = '-'; }
    if (array.class_region) { array.class_region = '-'; }
    if (array.class_currency) { array.class_currency = '-'; }
    if (array.class_classify) { array.class_classify = '-'; }
    if (array.is_custom == 1) { array.is_custom = "是" } else { array.is_custom = "" }
    tr = <tr>;
}

total: 11, page_size: "50", total_page: 2, list: [...]]

v0: {index_code: "930903", index_name: "中国黄金集团", base_date: "2012-12-21 00:00:00", base_point: "1000 M", class_assets: "总资产", class_classfy: "股票", class_currency: "人民币", class_easale: "权益", class_elasly: "负债", class_accruency: "成本", class_shot: null, class_eregion: "China mainland", class_eserve: "SST Index", class_hot: null, class_region: "美国", class_seriex: "中金所沪深300", index_c_fullname: "中国黄金集团"}

```

资料来源：长江证券研究所

图 31: 爬取网站 1 表格数据

```
from urllib import request, parse
from lxml import etree
import json
import numpy as np
import pandas as pd

url = 'http://www.csindex.com.cn/cn-3X/index/index?class_id=3Xclass_1-3Xclass_1-1'
htmlstr = request.urlopen(url).read().decode(encoding='utf-8')
print(type(htmlstr))

<class 'str'>

html = etree.HTML(htmlstr)
print(html)

<Element html at 0x13e7452f1c8>

cols = html.xpath("//table[@id='table']/tr/text()")
print(cols)

['指数代码', '指数名称', '成分股数量', '最新收盘价', '1个月收益率(%)',
'资产类别', '热点', '地区覆盖', '币种', '定制定制', '指数类型']
```

资料来源：长江证券研究所

图 32: 网站 2 部分 html 源码

p 250

457.9 x 14.67

```

<div class="grid-16-8 clearfix">
  <div class="article">
    <div class="opt mod"></div>
    <div class="grid_view">
      <div>
        <div class="item">
          <div class="pic"></div>
          <div class="info">
            <div class="hd">
              <a href="http://www.125205212.com">
                <span class="next">
                  <span class="count">(共250条)</span>
                </div>
              </a>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>

```

资料来源：长江证券研究所

图 33: 爬取网站 2 标题数据

```
htmlstr = request.urlopen(url).read().decode(encoding='utf-8')
def handle_html(movie_list, htmlstr):
    html = etree.HTML(htmlstr)
    movie_list.extend(html.xpath("//ol[@class='grid_view']//div[@class='hd']//span[@class=
    if len(html.xpath("//span[@class='next']/a/text(0)")) != 0:
        handle_html(movie_list, request.urlopen(url + html.xpath("//span[@class='next']//
    return movie_list
movie_name_list = handle_html([], htmlstr)
import requests
from bs4 import BeautifulSoup

url = "http://www.baidu.com/index.html"
header = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4398.97 Safari/537.36"}
html = requests.get(url, headers = header)
def handle_html(movie_list, html):
    soup = BeautifulSoup(html.content)
    movie_list_soup = soup.find('ol', attrs={'class': 'grid_view'})
    if movie_list_soup != None:
        for movie_i in movie_list_soup.find_all("li"):
            detail = movie_i.find('div', attrs={'class': 'hd'})
            movie_name = detail.find('span', attrs={'class': 'title'}).getText()
            movie_list.append(movie_name)
        next_page = soup.find('span', attrs={'class': 'next'}).find('a')
        if next_page:
            handle_html(movie_list, requests.get(url + next_page['href'], headers = header)
    return movie_list
movie_name_list = handle_html([], html)
```

资料来源：长江证券研究所

投资评级说明

行业评级	报告发布日后的 12 个月内行业股票指数的涨跌幅度相对同期沪深 300 指数的涨跌幅为基准，投资建议的评级标准为：		
看好	：	相对表现优于市场	
中性	：	相对表现与市场持平	
看淡	：	相对表现弱于市场	
公司评级	报告发布日后的 12 个月内公司的涨跌幅度相对同期沪深 300 指数的涨跌幅为基准，投资建议的评级标准为：		
买入	：	相对大盘涨幅大于 10%	
增持	：	相对大盘涨幅在 5%~10%之间	
中性	：	相对大盘涨幅在-5%~5%之间	
减持	：	相对大盘涨幅小于-5%	
无投资评级	：由于我们无法获取必要的资料，或者公司面临无法预见结果的重大不确定性事件，或者其他原因，致使我们无法给出明确的投资评级。		

联系我们

上海

浦东新区世纪大道 1198 号世纪汇广场一座 29 层（200122）

武汉

武汉市新华路特 8 号长江证券大厦 11 楼（430015）

北京

西城区金融街 33 号通泰大厦 15 层（100032）

深圳

深圳市福田区中心四路 1 号嘉里建设广场 3 期 36 楼（518048）

重要声明

长江证券股份有限公司具有证券投资咨询业务资格，经营证券业务许可证编号：10060000。

本报告的作者是基于独立、客观、公正和审慎的原则制作本研究报告。本报告的信息均来源于公开资料，本公司对这些信息的准确性和完整性不作任何保证，也不保证所包含信息和建议不发生任何变更。本公司已力求报告内容的客观、公正，但文中的观点、结论和建议仅供参考，不包含作者对证券价格涨跌或市场走势的确定性判断。报告中的信息或意见并不构成所述证券的买卖出价或征价，投资者据此做出的任何投资决策与本公司和作者无关。

本报告所载的资料、意见及推测仅反映本公司于发布本报告当日的判断，本报告所指的证券或投资标的的价格、价值及投资收入可升可跌，过往表现不应作为日后的表现依据；在不同时期，本公司可发出与本报告所载资料、意见及推测不一致的报告；本公司不保证本报告所含信息保持在最新状态。同时，本公司对本报告所含信息可在不发出通知的情形下做出修改，投资者应当自行关注相应的更新或修改。

本公司及作者在自身所知范围内，与本报告中所评价或推荐的证券不存在法律法规要求披露或采取限制、静默措施的利益冲突。

本报告版权仅仅为本公司所有，未经书面许可，任何机构和个人不得以任何形式翻版、复制和发布。如引用须注明出处为长江证券研究所，且不得对本报告进行有悖原意的引用、删节和修改。刊载或者转发本证券研究报告或者摘要的，应当注明本报告的发布人和发布日期，提示使用证券研究报告的风险。未经授权刊载或者转发本报告的，本公司将保留向其追究法律责任的权利。