## Python

### 环境配置

使用NotePad++：

* “运行”增加Python运行快捷

cmd /k C:\Users\Liang.Lu\Desktop\Python\Anaconda "$(FULL\_CURRENT\_PATH)" & PAUSE & EXIT

* Python 字体颜色

### 基础语法

#### 基本概念

#### 运算符与表达式

#### 控制流

#### 函数

#### 模块

模块就是一个包含了所有你定义的函数和变量的文件，模块必须以.py为扩展名。模块可以从其他程序中‘输入’(import)以便利用它的功能。

在python程序中导入其他模块使用'import',所导入的模块必须在sys.path所列的目录中，因为sys.path第一个字符串是空串''即当前目录，所以程序中可导入当前目录的模块。

***1. 字节编译的.pyc文件***

导入模块比较费时，python做了优化，以便导入模块更快些。一种方法是创建字节编译的文件，这些文件以.pyc为扩展名。

pyc是一种二进制文件，是py文件经编译后产生的一种byte code，而且是跨平台的（平台无关）字节码，是有python虚拟机执行的，类似于java或.net虚拟机的概念。pyc的内容，是跟python的版本相关的，不同版本编译后的pyc文件是不同的。

***2. from .. import***

如果想直接使用其他模块的变量或其他，而不加'模块名+.'前缀，可以使用from .. import。例如想直接使用sys的argv，from sys import argv 或 from sys import \*

***3. 模块的\_\_name\_\_***

每个模块都有一个名称，py文件对应模块名默认为py文件名，也可在py文件中为\_\_name\_\_赋值；如果是\_\_name\_\_，说明这个模块被用户单独运行。

***4. dir()函数***

dir(sys)返回sys模块的名称列表；如果不提供参数，即dir()，则返回当前模块中定义名称列表。

del -> 删除一个变量/名称，del之后，该变量就不能再使用。

#### 数据结构

##### 列表

list是处理**一组有序**项目的数据结构，列表是可变的数据结构。列表的项目包含在方括号[]中，eg: [1, 2, 3]， 空列表[]。

判断列表中是否包含某项可以使用in， 比如 l = [1, 2, 3]; print 1 in l; #True；

支持索引和切片操作；索引时若超出范围，则IndexError；

使用函数len()查看长度；

使用del可以删除列表中的项，eg: del l[0] # 如果超出范围，则IndexError。

##### 元组

tuple和list十分相似，但是***tuple是不可变的，即不能修改tuple***，元组通过***圆括号中用逗号分割***的项定义；支持索引和切片操作；可以使用 in

查看一个元素是否在tuple中。空元组()；只含有一个元素的元组("a",) #需要加个逗号

**优点**：tuple比list速度快；对不需要修改的数据进行‘写保护’，可以是代码更安全

　 tuple与list可以相互转换，使用内置的函数list()和tuple()。

##### 字典

字典由***键值对组成，键必须是唯一***的；eg: d = {key1:value1, key2:value2}；***空字典用{}***表示；

字典中的键值对是没有顺序的，如果想要一个特定的顺序，那么使用前需要对它们排序；

d[key] = value，如果字典中已有key，则为其赋值为value，否则添加新的键值对key/value；

使用del d[key] 可以删除键值对；判断字典中是否有某键，可以使用in 或 not in.

##### 序列

**序列类型**是指容器内的元素从0开始的索引顺序访问，一次可以访问一个或者多个元素；列表、元组和字符串都是序列；序列的两个主要特点是索引操作符和切片操作符；***索引可以得到特定元素；切片可以得到部分序列***；

切片操作符中的第一个数（冒号之前）表示切片开始的位置，第二个数（冒号之后）表示切片到哪里结束。 如果不指定第一个数，Python就从序列首开始。如果没有指定第二个数，则Python会停止在序列尾。 注意，返回的序列从开始位置 开始 ，刚好在结束位置之前 结束。即开始位置是包含在序列切片中的，而结束位置被排斥在切片外。 可以用负数做切片。负数用在从序列尾开始计算的位置。

##### 绑定

当创建一个对象并给它赋一个变量的时候，这个变量仅仅“指向”那个对象，而不是表示这个对象本身！ 也就是说，***变量名指向计算机中存储那个对象的内存***，这被称作名称到***对象的绑定***。

如果要复制一个列表或者类似的序列或者其他复杂的对象（不是如整数那样的简单对象），那么必须使用切片操作符来取得拷贝。

#### 面向对象编程

　　python支持面向对象编程；类和对象是面向对象编程的两个主要方面，类创建一个新的类型，对象是这个类的实例。

　　对象可以使用普通的属于对象的变量存储数据，属于对象或类的变量被称为***域***；对象也可以使用属于类的函数，这样的函数称为类的方法；域和方法可以合称为类的属性。

　　域有两种类型--属于实例的或属于类本身；它们分别被称为实例变量和类变量。

　　类使用关键字class创建，类的域和方法被列在一个缩进块中。

　　类的方法必须有一个额外的第一个参数，但是在调用时不为这个参数赋值，这个特殊变量指对象本身，按照惯例它的名称是self，类似C#中的this。

class Animal:

pass #empty block

　　\_\_init\_\_方法　　在类的一个对象被创建时调用该方法；相当于c++中的构造函数。

　　\_\_del\_\_方法　　在类的对象被销毁时调用该方法；相当于c++中的析构函数。在使用del删除一个对象时也就调用\_\_del\_\_方法。

　　Python中所有的类成员(包括数据成员)都是public的；只有一个例外，如果使用的数据成员以双下划线为前缀，则为私有变量。

class Person:

Count = 0

def \_\_init\_\_(self, name, age):

Person.Count += 1

self.name = name

self.\_\_age = age

p = Person("peter", 25)

p1 = Person("john", 20)

print Person.Count #2

print p.name #peter

print p.\_\_age #AttributeError: Person instance has no attribute '\_\_age'

**继承**：为了使用继承，基类的名称作为一个元组跟在类名称的后面；python支持多重继承。下面是一个关于继承的例子：

class SchoolMember:

'''Represent any school member.'''

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

print "Initializing a school member."

def tell(self):

'''Tell my details'''

print "Name: %s, Age: %s, " % (self.name, self.age),

class Teacher(SchoolMember):

'''Represent a teacher.'''

def \_\_init\_\_(self, name, age, salary):

SchoolMember.\_\_init\_\_(self, name, age)

self.salary = salary

print "Initializing a teacher"

def tell(self):

SchoolMember.tell(self)

print "Salary: %d" % self.salary

class Student(SchoolMember):

'''Represent a student.'''

def \_\_init\_\_(self, name, age, marks):

SchoolMember.\_\_init\_\_(self, name, age)

self.marks = marks

print "Initializing a student"

def tell(self):

SchoolMember.tell(self)

print "Marks: %d" % self.marks

print SchoolMember.\_\_doc\_\_

print Teacher.\_\_doc\_\_

print Student.\_\_doc\_\_

t = Teacher("Mr. Li", 30, 9000)

s = Student("Peter", 25, 90)

members = [t, s]

for m in members:

m.tell()

程序输出如下：

Represent any school member.

Represent a teacher.

Represent a student.

Initializing a school member.

Initializing a teacher

Initializing a school member.

Initializing a student

Name: Mr. Li, Age: 30, Salary: 9000

Name: Peter, Age: 25, Marks: 90

#### 输入/输出

程序与用户的交互需要使用输入/输出，主要包括控制台和文件；对于控制台可以使用raw\_input和print，也可使用str类。raw\_input(xxx)输入xxx然后读取用户的输入并返回。

**1. 文件输入/输出**

　　　　可以使用file类打开一个文件，使用file的read、readline和write来恰当的读写文件。对文件读写能力取决于打开文件时使用的模式，常用模式

　　有读模式("r")、写模式("w")、追加模式("a")，文件操作之后需要调用close方法来关闭文件。

test = '''\

This is a program about file I/O.

Author: Peter Zhange

Date: 2011/12/25

'''

f = file("test.txt", "w") # open for writing, the file will be created if the file doesn't exist

f.write(test) # write text to file

f.close() # close the file

f = file("test.txt") # if no mode is specified, the default mode is readonly.

while True:

line = f.readline()

if len(line) == 0: # zero length indicates the EOF of the file

break

print line,

f.close()

**2. 存储器**

　　　　python提供一个标准的模块，成为pickle，使用它可以在一个文件中存储任何python对象，之后可以完整的取出来，这被称为持久地存储对象；还有另外一个模块成为cPickle，它的功能和pickle完全一样，只不过它是用c写的，要比pickle速度快(大约快1000倍)。

复制代码

import cPickle

datafile = "data.data"

namelist = ["peter", "john", "king"]

f = file(datafile, "w")

cPickle.dump(namelist, f)

f.close()

del namelist

f = file(datafile)

storednamelist = cPickle.load(f)

print storednamelist

#['peter', 'john', 'king']

#### 异常

* 当程序中出现某些异常的状况时，异常就发生了。python中可以使用try ... except 处理。
* 可以让try ... except 关联上一个else，当没有异常时则执行else。
* 我们可以定义自己的异常类，需要继承Error或Exception。
* try...finally

try:

f = file("test.txt")

while True:

line = f.readline()

if len(line) == 0:

break

time.sleep(2)

print line,

finally:

f.close()

print "Cleaning up..."

#### Python标准库

* **sys模块　　sys模块包含系统对应的功能**

sys.argv　　---包含命令行参数，第一个参数是py的文件名

sys.platform　　---返回平台类型

sys.exit([status])　　---退出程序，可选的status(范围：0-127)：0表示正常退出，其他表示不正常，可抛异常事件供捕获

sys.path　　　　---程序中导入模块对应的文件必须放在sys.path包含的目录中，使用sys.path.append添加自己的模块路径

sys.modules　　---This is a dictionary that maps module names to modules which have already been loaded

sys.stdin,sys.stdout,sys.stderr　　---包含与标准I/O 流对应的流对象

* **os模块　　该模块包含普遍的操作系统功能**

os.name字符串指示你正在使用的平台。比如对于Windows，它是'nt'，而对于Linux/Unix用户，它是'posix'

os.getcwd()函数得到当前工作目录，即当前Python脚本工作的目录路径

os.getenv()和os.putenv()函数分别用来读取和设置环境变量

os.listdir()返回指定目录下的所有文件和目录名

os.remove()函数用来删除一个文件

os.system()函数用来运行shell命令

os.linesep字符串给出当前平台使用的行终止符。例如，Windows使用'\r\n'，Linux使用'\n'而Mac使用'\r'

os.sep 操作系统特定的路径分割符

os.path.split()函数返回一个路径的目录名和文件名

os.path.isfile()和os.path.isdir()函数分别检验给出的路径是一个文件还是目录

os.path.existe()函数用来检验给出的路径是否真地存在

#### 其它

**1. 一些特殊的方法**

\_\_init\_\_(self,...) 这个方法在新建对象恰好要被返回使用之前被调用。

\_\_del\_\_(self) 恰好在对象要被删除之前调用。

\_\_str\_\_(self) 在我们对对象使用print语句或是使用str()的时候调用。

\_\_lt\_\_(self,other) 当使用 小于 运算符（<）的时候调用。类似地，对于所有的运算符（+，>等等）都有特殊的方法。

\_\_getitem\_\_(self,key) 使用x[key]索引操作符的时候调用。

\_\_len\_\_(self) 对序列对象使用内建的len()函数的时候调用。

**2. 综合列表**

通过列表综合，可以从一个已有的列表导出一个新的列表

**3. 函数接收元组/列表/字典**

当函数接收元组或字典形式的参数的时候，有一种特殊的方法，使用\*和\*\*前缀。该方法在函数需要获取可变数量的参数的时候特别有用。

由于在args变量前有\*前缀，所有多余的函数参数都会作为一个元组存储在args中。如果使用的是\*\*前缀，多余的参数则会被认为是一个字典的键/值对。

def powersum(power, \*args):

total = 0

for i in args:

total += pow(i, power)

return total

print powersum(2, 1, 2, 3) #14

def displaydic(\*\*args):

for key,value in args.items():

print "key:%s;value:%s" % (key, value)

displaydic(a="one", b="two", c="three")

#key:a;value:one

#key:c;value:three

#key:b;value:two

**4. lambda**

lambda语句被用来创建新的函数对象，并在运行时返回它们。lambda需要一个参数，后面仅跟单个表达式作为函数体，而表达式的值被这个

新建的函数返回。 注意，即便是print语句也不能用在lambda形式中，只能使用表达式。

**5. exec/eval**

exec语句用来执行储存在字符串或文件中的Python语句；eval语句用来计算存储在字符串中的有效Python表达式。

**6. assert**

assert语句用来断言某个条件是真的，并且在它非真的时候引发一个错误--AssertionError。

**7. repr函数**

　repr函数用来取得对象的规范字符串表示。反引号（也称转换符）可以完成相同的功能。

　注意，在大多数时候有eval(repr(object)) == object。

可以通过定义类的\_\_repr\_\_方法来控制对象在被repr函数调用的时候返回的内容。

## IPython

***主要用于交互式数据处理和利用matplotlib对数据进行可视化处理。***

1. ***类似Mathematica的HTML笔记本***
2. ***基于QT框架的GUI控制台，其中含有绘图、多行编辑以及语法高亮显示等***

## NumPy

***Numberical Python :是Python科学计算的基础包。***

***特点：***

1. ***多维数组对象ndarray；***
2. ***对数据执行元素级计算以及直接对数组执行数学运算的函数；***
3. ***读写硬盘上基于数组的数据集的工具；***
4. ***线性代数运算、傅立叶变换、以及随机数生成；***
5. ***用于将C、C++、Fortran代码集成到Python的工具；***
6. ***算法之间传递数据的容器；***

***理解NumPy数组以及面向数组的计算。***

***对于大部分数据分析，主要关注功能集中在：***

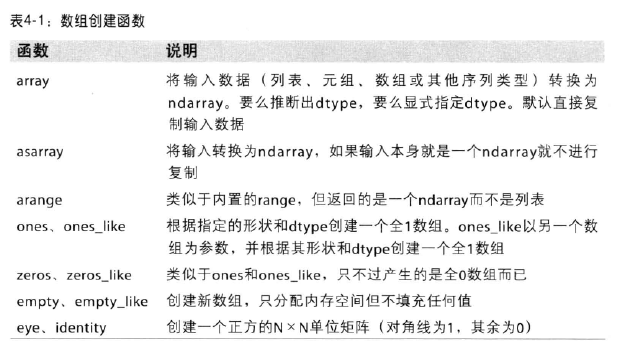
* 1. ***用于数据整理和清理、子集构造和过滤、转换等快速的矢量化数组运算；***
  2. ***常用的数组算法、如排序、唯一化、集合运算等。***
  3. ***高效描述统计和数据聚合、摘要运算***
  4. ***用于异构数据集的合并、连接运算的数据对齐和关系型数据运算；***
  5. ***将条件逻辑表述为数组表达式***
  6. ***数据的分组运算（聚合、转换、函数应用等）***

***NumPy的ndarray：一种多维数组对象***

***Numpy最重要的一个特点是其N维数组对象（即ndarray），该对象是一个快速灵活的大数据容器。***

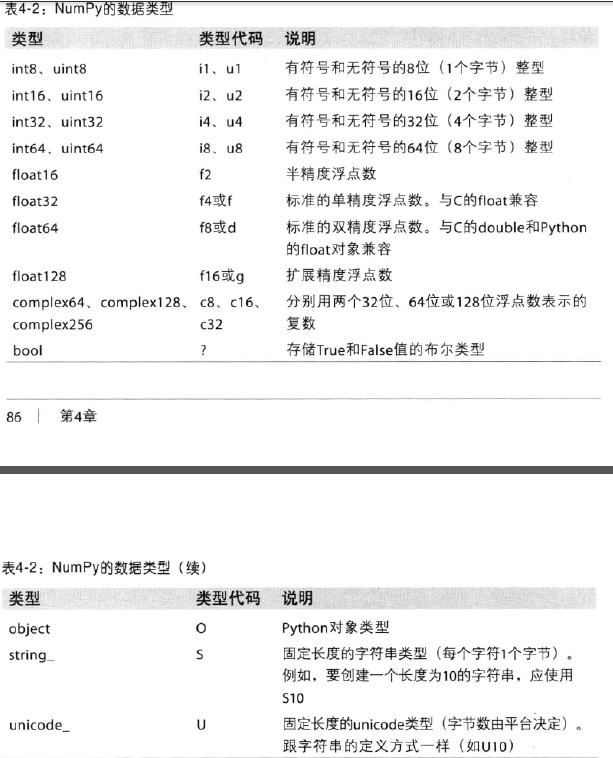
***Ndarray是一个通用的铜鼓数据多维容器，即其中的所有元素必须是相同类型的；每个数组都有一个shape（一个表示各维度大小的元组）和一个dtype（一个用于说明数组数据类型的对象）***

***数组创建函数：***



***Ndarray的数据类型***

***Dtype（数据类型）是一个特殊的对象，它含有ndarray将一块内存解释为特定数据类型所需的信息。***



***通过ndarray的astype方法显式转换为其dtype***

***如果将浮点数转换成整数，则小数部分将会被截断。***

***如果某字符串数组表示的全是数字，可以用astype将其转换为数值形式。***

***大小相等的数组之间的任何算是运算都会将运算运用到元素级。***

***不同大小的数组之间的运算叫广播（broadcasting）***

***基本的索引和切片***

***与列表最重要的区别在于，数组切片是原始数组的视图，意味着数据不会被复制。***

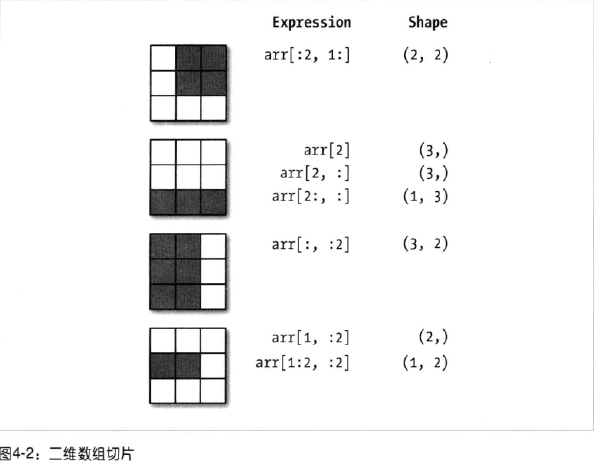
***np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])***

***括号外面的“维度”是一维、二维、三维、四维之类，而括号里面的可理解为“轴”，即“返回的低维数组含有原始高维数组某条轴上的所有数据”。***

***切片索引***

***可以在一个或多个轴上进行切片，也可以跟整数索引混合使用。***

***“只有冒号”表示选取整个轴。例如：arr2d[2,:1]***



***布尔型索引***

***布尔型数组的长度必须跟被索引的轴长度一致。***

***将布尔型数组跟切片、整数（或整数序列）混合使用。例如：data[names==’Bob’,2:]***

***可以使用不等于符号(!=),也可以通过负号（-）对条件进行否定。***

***选取三个条件中的两个组合运用多个布尔条件，使用&(和)、|（或）***

***花式索引***

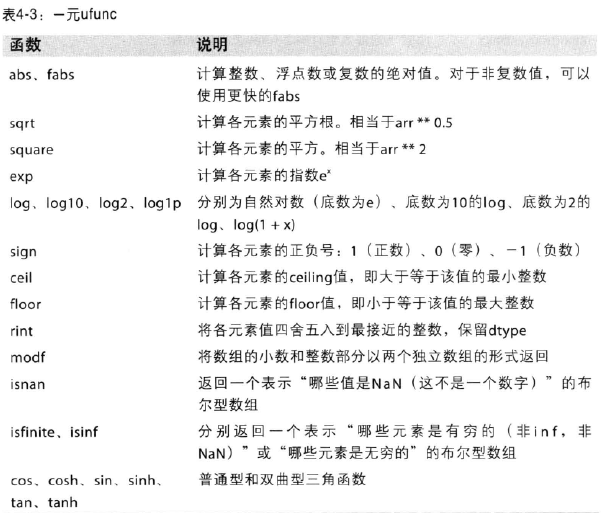
***花式索引（Fancy indexing）是一个Numpy术语，它指的是利用整数数组进行索引。***

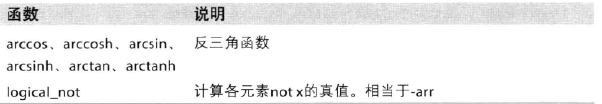
***数组转置和轴对换***

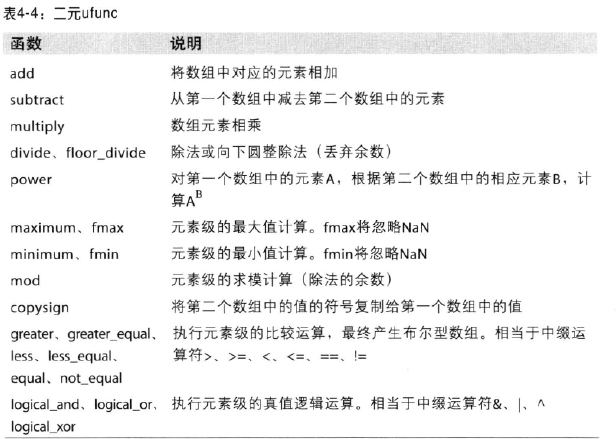
***转置（transpose）是重塑的一种特殊形式，它返回的是源数据的视图。***

***通用函数：快速的元素级数组函数***

***通用函数（即ufunc）是一种对ndarray中的数据执行元素级运算的函数。***





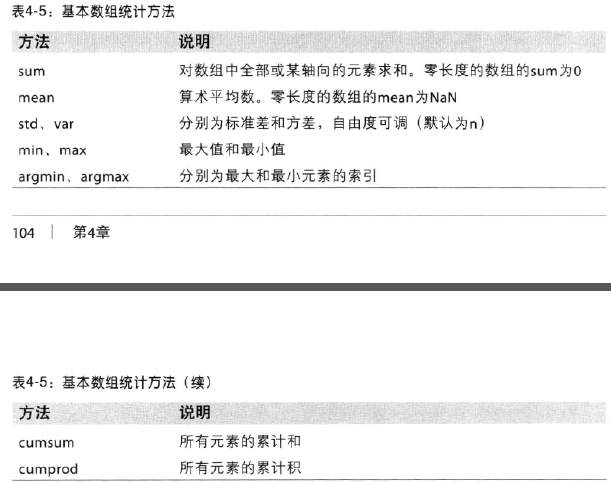


***利用数组进行数据处理***

***用数组表达式代替循环的做法，被称为矢量化。（一般来说，矢量化数组运算比等价的纯Python的方式快上一两个数量级）***

***数学和统计方法***

***通过数组上的一组数学函数对整个数组或某个轴向的数据进行统计计算。***

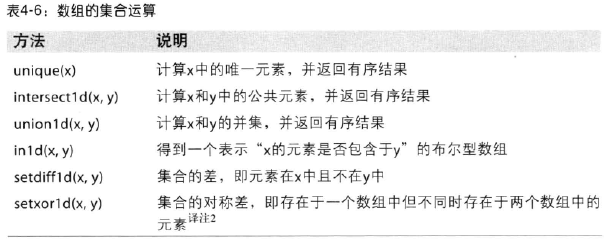


***排序***

***Numpy数组通过sort方法进行排序***

***唯一化以及其他的集合逻辑***

***NumPy提供了一些针对一维ndarray的基本集合运算。最常用的np.unique***



***将数据以二进制格式保存到磁盘***

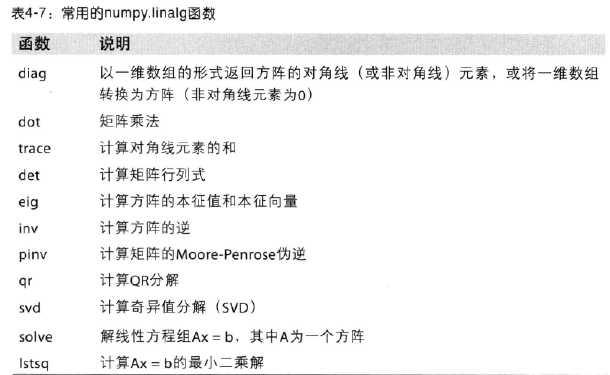
***np.save和np.load是读写磁盘数组数据的两个主要函数。默认情况下，数组是以未压缩的原始二进制格式保存在扩展名为.npy的文件中。***

***np.savez可以将多个数组保存到一个压缩文件中，将数组以关键字参数的形式传入。***

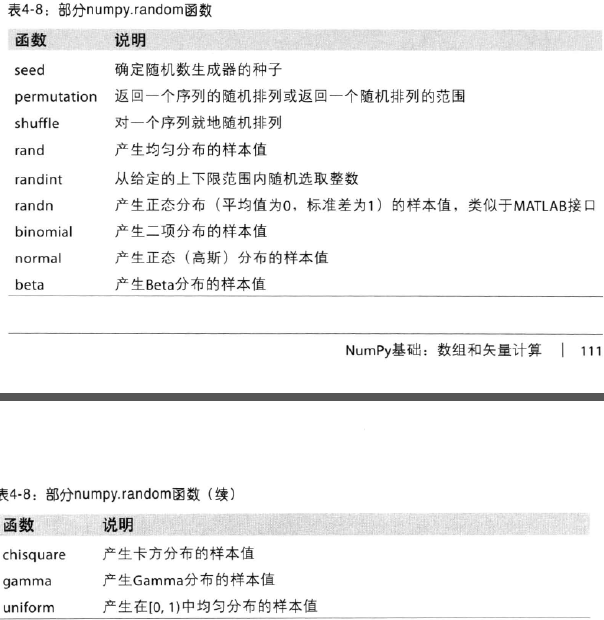
***存取文本文件***

***Loadtxt***

***线性代数***



***随机数生成***



## Pandas

***#Pandas其能做什么；#在数据分析中运用场景***

***主要数据结构：Series 和 DataFrame***

* ***Series***

***一种类似于一维数组的对象，它由一组数据以及一组与之相关的数据标签（即索引）组成。***

***\*在算术运算中自动对齐不同索引的数据。***

***Series对象本身及其索引都有一个name属性。***

***Series的索引可以通过赋值的方式进行修改。***

* ***DataFrame***

***一个表格型的数据结构，它含有一组有序的列，每列可以是不同的值类型（数值、字符串、布尔值等）***

***DataFrame既有行索引也有列索引，***

***构建DataFrame，最常用的一种是直接传入一个由等长列表或NumPy数组组成的字典。***

***data = {‘state’:[‘Ohio’,’Ohio’,’Ohio’,’Nevada’,Nevada’],***

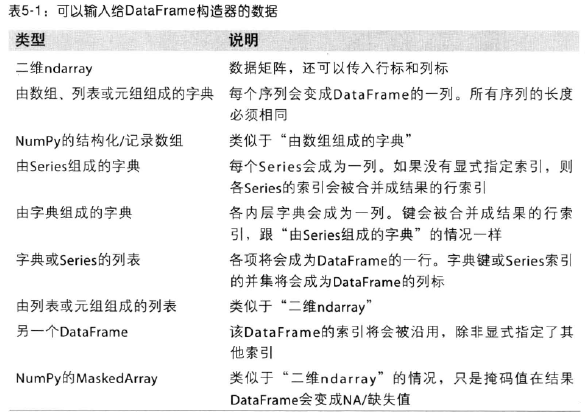
***‘year’:[2000,2001,2002,2001,2002],***

***‘pop’:[1.5,1.7,3.6,2.4,2.9]}***

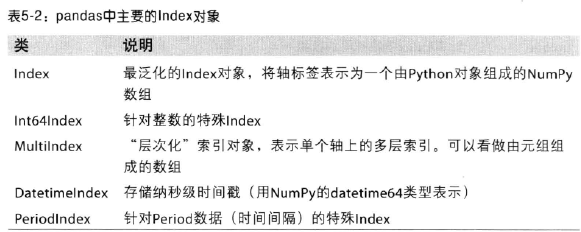
***Frame = DataFrame(data)***

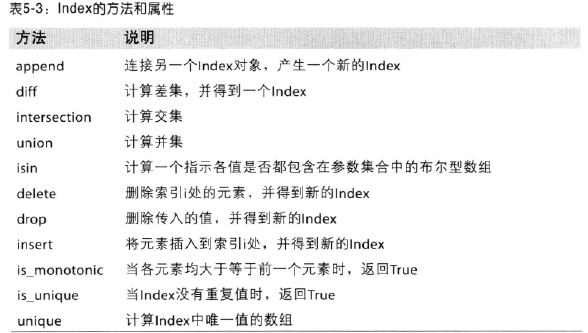
***通过索引方式返回的列只是想要数据的视图，并不是副本，因此，对返回的Series所做的任何修改都会反映在源DataFrame上。***

***另一种常见的数据形式是嵌套字典。即：外层字典的键作为列，内层键作为行索引。***

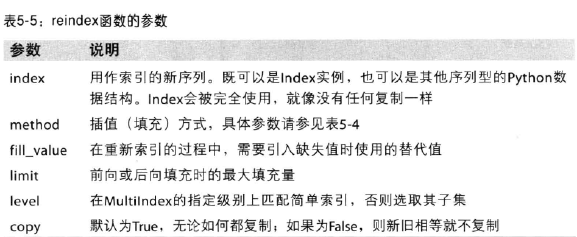


***索引对象***



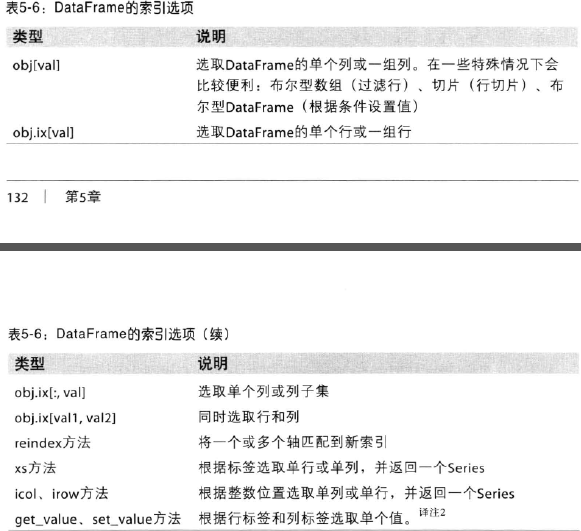


***对于DataFrame, reindex可以修改（行）索引、列，或两个都修改。***



***丢弃指定轴上的项***

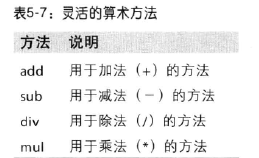
***索引、选取和过滤***



***算术运算和数据对齐***

***Pandas最重要的一个功能是，可以对不同索引的对象进行算术运算。在将对象相加时，如果存在不同的索引时，则结果的索引就是该索引对的并集。***

***在算术方法中充值***



***DataFrame 和Series之间的运算***

***函数应用和映射***

***排序和排名***

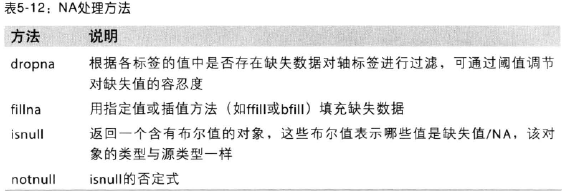
***带有重复值得轴索引***

***汇总和计算描述统计***

***相关系统与协方差***

***唯一值、值计数以及成员资格***

***处理缺失数据***



***滤除缺失数据***

***填充缺失数据***

***层次化索引：以低维度形式处理高纬度数据***

***重排分级排序***

***根据级别汇总统计***

***使用DataFrame的列***

## TuShare

1. ***处理结构化数据的大量数据结构和函数；例如：DataFrame***
2. ***兼具NumPy高性能的数据计算功能以及电子表格和关系型数据库灵活的数据处理功能；***
3. ***提供复杂精细的索引功能，方便完成重塑、切片和切块、聚合以及选取数据子集等操作。***

## Matplotlib

* 1. ***绘制数据图表的Python库；***

## Scipy

***Scipy是一组专门解决科学计算中各种标准问题域的包的集合：***

***Scipy.integrate:数据积分例程和微分方程求解器；***

***Scipy.linalg:扩展了由numpy.linalg提供的线性代数例程和矩阵分解功能；***

***Scipy.optimize：函数优化器（最小化器）以及根查找算法；***

***Scipy.signal：信号处理工具；***

***Scipy.sparse：稀疏矩阵和稀疏线性系统求解器；***

***Scipy.special：SPECFUM的包装器；***

***Scipy.stats：标准连接和离散概率分布、各种统计检验方法、以及更好的描述统计法；***

***Numpy与Scipy有机结合可替代Matlab的计算功能***

## 数据分析步骤

1. ***与外界进行交互：读取各种各样的文件格式和数据库***
2. ***准备：对数据进行清理、修整、整合、规范化、重塑、切片切块、变形等处理以便分析；***
3. ***转换：对苏聚集做一些数学和统计运算以产生新的数据集；***
4. ***建模和计算：将数据更统计模型、机器学习算法或其他计算工具联系起来；***
5. ***展示：创建交互式的或静态的图片或文字摘要***

***第二章摘要***

***用纯Python代码对时区进行计数***

***用pandas对时区进行计数***

***情景一：***

1. ***最常出现的是哪个时区？***
   1. ***按windows和非windows用户对时区统计信息进行分解***
2. ***利用绘图库（即matplotlib）生成图片***

***情景二：拥有6000名用户对4000部电影的100万评分；分为三个表：评分、用户信息和电影信息。***

1. ***按性别计算每部电影的平均得分***
2. ***过滤评分数据不够250条的电影***
3. ***了解女性观众最喜欢的电影，（降序排列）***
4. ***找出男性和女性观众分歧最大的电影？对结果列反序取出前15条记录；***
5. ***不考虑性别，找出分歧最大的电影***

***情景三：1880-2010年间全美婴儿姓名***

1. ***用births列的sex分组小计表示该年度的births总计；***
2. ***增加prop列，用于存放指定名字的婴儿数相对于总出生数的比例；***
3. ***按性别和年度统计总出生数***
4. ***分析命名趋势，绘制几个名字的曲线图***
5. ***评估命名多样性的增长（计算最流行的1000个名字所占比例，按year和sex进行聚合并绘图）***
6. ***各年出生的男孩中名字以d、n、y结尾的人数比例***
7. ***变成女孩子名字的男孩子名字（以及相反的情况）***

***第三章 IPython 一种交互式计算和开发环境 摘要***

***2001年，Fernando 启动IPython项目***

***IPython本身没有提供任何的计算或数据分析功能，其设计目的是在交互计算和软件开发两方面最大化提高生产力。***

***鼓励“执行-探索“的工作模式***

***启动ipython notebook***

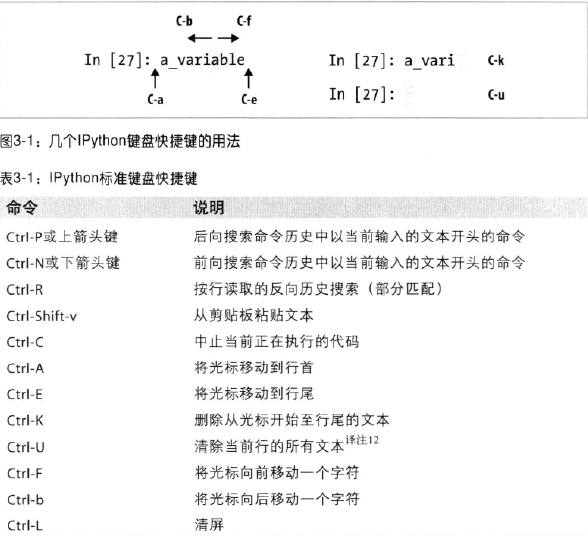
***>ipython.exe notebook***

***Tab键自动完成搜索命名空间和自动完成对象或模块属性，自动完成函数关键字参数。***

***对象内省：在变量的前面或后面加上一个问号（？）可以将有关该对象的一些通用信息显示。***

***%run命令：在IPython环境中，所有文件都可以通过%run命令当做python程序来运行。***

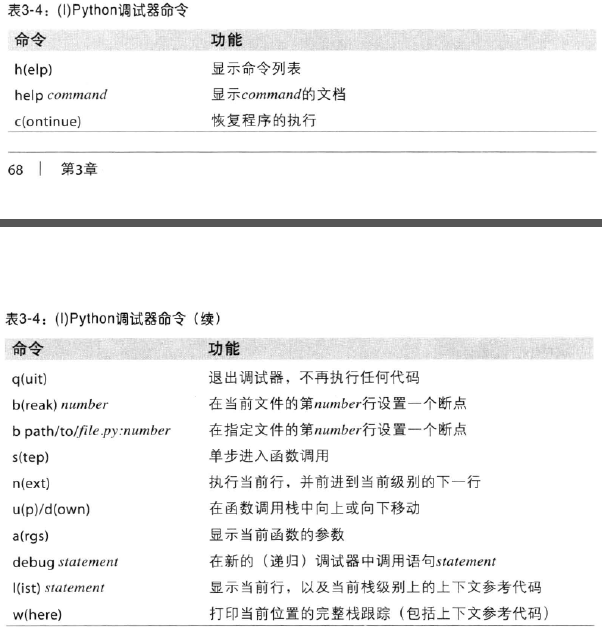
***%paste命令：***



***魔术命令***



***IPython调试器命令***



***测试代码执行时间：%time 和 %timeit***

***基本性能分析：%prun 和 %run –p***

***import numpy as np***

***from numpy.linalg import eigvals***

***def run\_experiment(niter = 100):***

***K = 100***

***results = []***

***for \_ in xrange(niter):***

***mat = np.random.randn(K,K)***

***max\_eigenvalue = np.abs(eigvals(mat)).max()***

***results.append(max\_eigenvalue)***

***return results***

***some\_results = run\_experiment()***

***print 'Largest one we saw :%s' % np.max(some\_results)***

***逐条分析函数性能 ：%lprun***

***通用语法：%lprun –f func1 –f func2 statement\_to\_profile***

***from numpy.random import randn***

***def add\_and\_sum(x,y):***

***added = x +y***

***summed = added.sum(axis = 1)***

***return summed***

***def call\_function():***

***x = randn(1000,1000)***

***y = randn(1000,1000)***

***return add\_and\_sum(x,y)***

***先用%run 运行py脚本，再用%lprun 或 %prun进行分析***

***第四章 NumPy基础：数组和矢量计算***