Python 编程分析基础

金林

jinlin@zuel.edu.cn

中南财经政法大学统计与数学学院

2020-01

- Python 数据类型
- 数据的基本类型
- 标准数据结构

python 对象

1. python 创建和控制的实体称为对象 (object)，它们可以是变量、数组、字符串、函数或结构。
2. 由于 python 是一种所见即所得的脚本语言，故不需要编译。
3. 在 python 里，对象是通过名字创建和保存的。
4. 可以用 who 命令来查看当前打开的 python 环境里的对象，用 del 删除这些对象。

对象操作

1. 查看数据对象
2. 生成数据对象
3. 删除数据对象

```
who
x=10.12
who
del x
who
```

上面列出的是新创建的数据对象 x 的名称。python 对象的名称必须以一个英文字母打头，并由一串大小写字母、数字或下画线组成。

注意：python 区分大小写，比如，Orange 与 orange 数据对象是不同的。

不要用 python 的内置函数名作为对象的名称，如 who/del 等。

数值型

1. python 的基本数据类型包括数值型、逻辑型、字符型、复数型等，也可能是缺失值。
2. 数值型数据的形式是实数，可以写成整数（如 =3）、小数（如 x=1.46）/科学计数（y=1e9）的方式，该类型数据默认是双精度数据。
3. python 支持 4 种不同的数字类型：
   1. int(有符号整型)；
   2. long（长整型，也可以代表八进制和十六进制）；
   3. flont（浮点型）；
   4. complex（复数）。
4. 说明：python 中显示数据或对象内容直接用其名称，相当于执行 print 函数。

例子

```
1 n=10
2 n
```

```
1 print(”n=”,n)
```

```
1 x=10.234
2 print(x)
```

```
1 print(”x=%10.5f”%x)
```

逻辑型

1. 逻辑型数据只能取 True 或 False 值。
2. 可以通过比较获得逻辑型数据，

```
a=True;a
```

```
b=False;b
```

```
10>3
```

```
10<3
```

```
print(3)
```

字符型

1. 字符型数据的形式是夹在双引号“”或单引号’‘之间的字符串，如‘MR’。
2. 注意：一定要用英文引号，不能用中文引号“”或‘’。
3. python 语言中的 string（字符串）是由数字、字母、下画线组成的一串字符。一般形式为 s=‘I love python’ 它是编程语言中表示文本的数据类型。
4. python 字符串具有切片功能，即由左到右索引默认从 0 开始；由右到左索引默认从-1 开始。
5. 如果要实现从字符串中获取一段子字符串，可以使用变量 [头下标：尾下标]，其中下标从 0 开始算起，可以是正数或负数，也可以为空，表示取到头或尾。比如，上例中 s[7] 的值是 p，s[2 : 6] 的结果是 love.

例子

```
1 s='IlovePython';s
```

```
1 s[7]
```

```
1 s[2:6]
```

```
1 s+s
```

```
1 s*2
```

加号（＋）是字符串连接运算符，星号（*）是重复操作。

缺失值

有些统计资料是不完整的。当一个元素或值在统计的时候是“不可得到”或“缺失值”的时候，相关位置可能会被保留并且赋予一个特定的 nan（not availablenumber，不是一个数）值。任何 nan 的运算结果都是 nan。例如，float（‘nan’）就是一个实数缺失值。

```
1 float('nan')
```

数据基本类型转换

1. 需要对数据内置的类型进行转换，只须将数据类型作为函数名即可。
2. 以下几个内置的函数可以实现数据类型之间的转换。
3. 这些函数返回一个新的对象，表示转换的值。下面列出几种常用的数据类型转换方式：
   1. int (x[,base]) # 将 x 转换为一个整数
   2. float (x) # 将 x 转换为一个浮数点
   3. str (x) # 将对象 x 转换为字符串
   4. chr (x) # 将一个整数转换为一个字符
4. python 的所有数据类型都是类，可以通过 type () 查看该变量的数据类型。

- Python 数据类型
- 数据的基本类型
- 标准数据结构

介绍

1. 在内存中存储的数据可以有多种类型。
2. 例如，一个人的年龄可以用数字来存储，名字可以用字符来存储。
3. python 定义了一些标准类型，用于存储各种类型的数据，这些标准的数据类型是由前述基本类型构成的。

list (列表)

1. list (列表) 是 python 中使用最频繁的数据类型。
2. 列表可以完成大多数集合类的数据结构实现。
3. 它支持字符、数字、字符串，甚至可以包含列表 (即嵌套)。
4. 列表用 [] 标识，是一种最通用的复合数据类型。
5. python 的列表也具有切片功能，列表中值的切割也可以用到变量 [头下标：尾下标]，可以截取相应的列表，从左到右索引默认从 0 开始，从右到左索引默认从-1 开始，下标可以为空，表示取到头或尾。

例子

```
list1=[];list1
```

```
list1=['Python',786,2.23,'R',70.2]
list1
```

```
list1[0]
```

```
list1[1:3]
```

```
list1[2:]
```

```
list1*2
```

```
list1+list1[2:4]
```

加号 + 是列表连接运算符，星号 * 是重复操作。操作类似字符串。

例子

```
X=[1,3,6,4,9];X
```

```
sex=[' 女',' 男',' 男',' 女',' 男']
sex
```

```
weight=[67,66,83,68,70];
weight
```

tuple（元组）

1. 元组是另一种数据类型，类似于 list（列表）。
2. 元组用“()”标识，内部元素用逗号隔开。元组不能赋值，相当于只读列表。操作类似列表。

dictionary (字典)

1. 字典也是一种数据类型，且可存储任意类型对象。
2. 字典的每个键值对用冒号“：”分隔，每个键值对之间用逗号“，”分隔，整个字典包括在花括号 {} 中，格式如下：

   $$\text{dict}=\{\text{key1:value1,key2:value2}\}$$

3. 键必须是唯一的，但值则不必，值可以取任何数据类型，如字符串、数字或元组。
4. 字典是除列表外 python 中最灵活的内置数据结构类型。列表是有序的对象集合，字典是无序的对象集合。
5. 两者之间的区别在于：字典中的元素是通过键来存取的，而不是通过下标存取。

例子

```
{}
```

```
dict1={'name':'john','code':6734,'dept':'sales'};dict1
```

```
dict1['code']
```

```
dict1.keys()
```

```
dict1.values()
```

```
dict2={'sex': sex,'weight':weight}; dict2
```

数值分析库 numpy

在使用 numpy 库前，须加载其到内存中，语句为 import numpy，通常将其简化为 import numpy as np

一维数组（向量）

```
import numpy as np
np.array([1,2,3,4,5])
```

```
np.array([1,2,3,np.nan,5])
```

```
np.array(X)
```

```
np.arange(9)
```

```
np.arange(1,9,0.5)
```

```
np.linspace(1,9,5)
```

```
np.random.randint(1,9)
```

```
np.random.rand(10)
```

```
np.random.randn(10)
```

二维数组（矩阵）

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

```
1 A=np.arange(9).reshape((3,3));A
```

数组的操作

1. 数组的维度
2. 空数组
3. 零数组
4. 1 数组
5. 单位阵

```
A.shape
```

```
np.empty([3,3])
```

```
np.zeros((3,3))
```

```
np.ones((3,3))
```

```
np.eye(3)
```

简介

1. 在数据分析中，数据通常以变量（一维数组，python 中用序列表示）和矩阵（二维数组，python 中用数据框表示）的形式出现，
2. 下面结合 python 介绍 pandas 基本的数据操作。
3. 注意：在 python 编程中，变量通常以列表（一组数据），而不是一般编程语言的标量（一个数据）形式出现。

```
1 import pandas as pd
```

- 序列及其操作
- 数据框的读写
- 数据框的操作

序列（series）

1. 创建序列（向量、一维数组）
   1. 假如要创建一个含有 n 个数值的向量（X=x1,x2...,xn），python 中创建序列的函数是列表，这些向量可以是数字型的，也可以是字符串型的，还可以是混合型的。
   2. 特别说明：python 中显示数据或对象内容直接用其名称。
2. 生成系列

```
1 pd.Series()
```

序列 (series)

3. 根据列表构建序列

```
X=[1,3,6,4,9]
S1=pd.Series(X);S1
```

```
S2=pd.Series(weight);S2
```

```
S3=pd.Series(sex);S3
```

4. 系列合并

```
pd.concat([S2,S3],axis=0)
```

```
pd.concat([S2,S3],axis=1)
```

5. 系列切片

```
S1[2]
```

```
S3[1:4]
```

数据框（DataFrame）及基本操作

1. pandas 中的函数 DataFrame() 可用序列构成一个数据框。
2. 数据框相当于关系数据库中的结构化数据类型，传统的数据大都以结构化数据形式存储于关系数据库中，因而传统的数据分析是以数据框为基础的。
3. python 中的数据分析大都是基于数据框进行的，所以本书的分析也是以数据类型为主，向量和矩阵都可以看成数据框的一个特例。

数据框基本操作

1. 生成数据框

```
pd.DataFrame()
```

2. 根据列表创建数据框

```
pd.DataFrame(X)
```

```
pd.DataFrame(X,columns=['X'],index=range(5))
```

```
pd.DataFrame(weight,columns=['weight'],index=['A','B','C','D','E'])
```

数据框基本操作

3. 根据字典创建数据框

```
df1=pd.DataFrame({'S1':S1,'S2':S2,'S3':S3});df1
```

```
df2=pd.DataFrame({'sex':sex,'weight':weight},index=X);df2
```

4. 增加数据框列

```
df2['weight2']=df2.weight**2; df2
```

数据框基本操作

5. 删除数据框列

```
del df2['weight2']; df2
```

6. 缺失值处理

```
df3=pd.DataFrame({'S2':S2,'S3':S3},index=S1);df3
```

```
df3.isnull()
```

```
df3.isnull().sum()
```

```
df3.dropna()
#df3.dropna(how = 'all')
```

7. 数据框排序

```
df3.sort_index()
```

```
df3.sort_values(by='S3')
```

- 序列及其操作
- 数据框的读写
- 数据框的操作

pandas 读取数据集

1. 大的数据对象常常从外部文件读入，而不是在 python 中直接输入的。
2. 外部的数据源有很多，可以是电子表格、数据库、文本文件等形式。
3. python 的导入工具非常简单，但是对导入文件有一些比较严格的限制。
4. 最常使用的是 pandas 包读取数据的方式，事先须调用 pandas 包，即 import pandas。

读取 csv 格式数据

虽然 python 可以直接复制表格数据，但也可读取电子表格工作簿中的一个表格（例如，在 Excel 中将数据 Dapy-data.xlsx 的表单 [BSdata] 另存为 BSdata.csv，这时 BSdata.csv 本质上也是文本文件，是以逗号分隔的文本数据，既可以用记事本打开，也可用电子表格软件打开，是最通用的数据格式），其读取命令也最简单，如下所示。

```
1 BSdata=pd.read_csv("../data/BSdata.csv",encoding='utf-8')
2 BSdata[6:9]
```

读取 Excel 格式数据

使用 pandas 包中的 read-excel 可直接读取 Excel 文档中的任意表单数据，其读取命令也比较简单，例如，要读取 Dapy-data.xlsx 表单的 [BSdata]，可用以下命令。

```
1 BSdata=pd.read_excel('../data/DaPy_data.xlsx','BSdata');BSdata[-5:]
```

其他读取方式

从剪贴板上读取

1. 先在 Dapy-data.xlsx 数据文件的【BSdata】表中选取 A1：H52，复制，然后在 python 中读取数据。
2. BSdata 为读入 python 中的数据框名，clipboard 为剪贴板。

```
BSdata=pd.read_clipboard();
BSdata[:5]
```

读取其他统计软件的数据

- 要调用 SAS、SSPS、Stata 等统计软件的数据集，须先用相应的包，详见 python 手册。

pandas 数据集的保存

1. python 读取和保存数据集的最好方式是 csv 和 xlsx 文件格式，pandas 保存数据的命令也很简单，如下所示。

```
1 BSdata.to_csv('BSdata1.csv')
```

- 序列及其操作
- 数据框的读写
- 数据框的操作

显示基本信息

1. 数据框显示

有三种显示数据框内容的函数，即 info( )(显示数据结构)、head( )(显示数据框前 5 行)、tail() (显示数据框后 5 行)。

```
BSdata.info()
```

```
BSdata.head()
```

```
BSdata.tail()
```

显示基本信息

1. 数据框列名（变量名）

```
BSdata.columns
```

2. 数据框行名（样品名）

```
BSdata.index
```

3. 数据框维度

```
BSdata.shape
```

```
BSdata.shape[0]    # 行数
```

```
BSdata.shape[1]    # 列数
```

4. 数据框值（数组）

```
BSdata.values
```

选取变量

1. “.” 法或 [‘’] 法：这是 python 中最直观的选择变量的方法，比如，要选择数据框 BSdata 中的 “身高” 和 “体重” 变量，直接用 “BSdata. 身高” 与 “BSdata. 体重” 即可，也可用 BSdata [‘身高’] 与 [‘体重’]，该方法书写比 “.” 法烦琐，却是最不容易出错且直观的一种方法，可推广到多个变量的情形，推荐使用。
2. 下标法：由于数据框是二维数组（矩阵）的扩展，所以也可以用矩阵的列下标来选取变量数据，这种方法进行矩阵 (数据框) 运算比较方便。例如，dat.iloc [i,j] 表示数据框（矩阵）的第 i 行、第 j 列数据，dat.iloc [i,] 表示 dat 的第 i 行数据向量，而 dat.iloc[,j] 表示 dat 的第 j 列数据向量（变量）。再如，“身高” 和 “体重” 变量在数据框 BSdata 的第 3、4 两列。

```
BSdata.身高 #取一列数据，BSdata['身高']
```

```
BSdata[['身高','体重']]
```

```
BSdata.iloc[:,2]
```

```
BSdata.iloc[:,2:4]
```

提取样品

```
1 BSdata.loc[3]
```

```
1 BSdata.loc[3:5]
```

选取观测与变量

- 同时选取观测与变量数据的方法就是将提取变量和样品方法结合使用。例如，要选取数据框中男生的身高数据，可用以下语句。

```
BSdata.loc[:3,['身高','体重']]
```

```
BSdata.iloc[:3,:5]
```

条件选取

- 选取身高超过 180cm 的男生的数据，以及身高超过 180cm 且体重小于 80kg 的男生的数据，可用以下语句。

```
BSdata[BSdata['身高']>180]
```

```
BSdata[(BSdata['身高']>180)&(BSdata['体重']<80)]
```

数据框的运算

1. 生成新的数据框

   可以通过选择变量名来生成新的数据框。

```
BSdata['体重指数']=BSdata['体重']/(BSdata['身高']/100)**2
round(BSdata[:5],2)
```

2. 数据框转置.T。

```
BSdata.iloc[:3,:5].T
```

数据框的运算

- 数据框的合并 pd.concat(): 可以用 pd.concat() 将两个或两个以上的向量、矩阵或数据框合并起来，参数 axis=0 表示按行合并，axis=1 表示按列合并。
  1. 按行合并，axis=0。
  2. 按列合并，axis=1。

```
pd.concat([BSdata.身高, BSdata.体重],axis=0)
```

```
pd.concat([BSdata.身高, BSdata.体重],axis=1)
```

基本运算

1. 与 Basic 语言、VB 语言、C 语言、C++ 语言等一样，python 语言具有编程功能，但 python 是新时期的编程语言，具有面向对象的功能，同时 python 还是面向函数的语言。
2. python 是一种编程语言，它就具有常规语言的算术运算符和逻辑运算符（如表 3-1 所示），以及控制语句、自定义函数等功能。

算术运算符	含义	逻辑运算符	含义
+	加	< (<=)	小于 (小于等于)
-	减	> (>=)	大于 (大于等于)
	乘	==	等于
/	除	!=	不等于
	幂	not x	非 x
%	取模	or	或
//	整除	and	与

- 控制语句
- 函数
- 面向对象

循环语句 for

python 的 for 循环可以遍历任何序列的项目，如一个列表或一个字符串。for 循环允许循环使用向量或数列的每个值，在编程时非常有用。

for 循环的语法格式如下：for iterating_var in sequence: statements(s) python 的 for 循环比其他语言的更为强大，例如：

```
for i in range(1,5):
    print(i)
```

```
fruits = ['banana', 'apple',  'mango']
for fruit in fruits:
   print('当前水果 :', fruit)
```

```
for var in BSdata.columns:
    print(var)
```

条件语句 if/else

if/else 语句是分支语句中的主要语句，其格式如下：

```
a = -100
if a < 100:
    print(”数值小于100”)
else:
    print(”数值大于100”)
```

```
-a if a<0 else a
```

python 中有更简洁的形式来表达 if/else 语句。注意：循环和条件等语句中要输出结果，请用 print() 函数，这时只用变量名是无法显示结果的。

- 控制语句
- 函数
- 面向对象

自定义函数

1. python 与其他统计软件的区别之一是，可以随时随地自定义函数，而且可以像使用 python 的内置函数一样使用自定义的函数。
2. python 进行数据分析是基于函数和面向对象的，所有 python 的命令都是以函数形式出现的，比如读取文本数据的 read_clipoard() 函数和读取 csv 数据文件的 read_csv() 函数，以及建立序列的 Series() 函数和构建数据框 DataFrame() 函数。
3. 由于 python 是开源的，故所有函数使用者都可以查看其源代码。

自定义函数

- 定义函数的语法：

```
def 函数名（参数1，参数2，…）：
    函数体
    return 语句
```

- 下表所列是 python 中常用的数学函数。

math 中的数学函数	含义 (针对数值)	numpy 中的数学函数	含义 (针对数组)
abs(x)	数值的绝对值	len(x)	数组中元素个数
sqrt(x)	数值的平方根	sum(x)	数组中元素求和
log(x)	数值的对数	prod(x)	数组中元素求积
exp(x)	数值的指数	min(x)	数组中元素最小值
round(x,n)	有效位数 n	max(x)	数组中元素最大值
sin(x),cos(x),...	三角函数	sort(x)	数组中元素排序
		rank(x)	数组中元素秩次

自定义函数

- python 内建函数命令，可直接使用
- 要了解任何一个 python 函数，使用 help() 函数即可，例如，命令 help(sum) 或？sum 将显示 sum() 函数的使用帮助。
- 如果函数只用来计算，不需要返回结果，则可在函数中用 print 函数，这时只用变量名是无法显示结果的，见下。比如，定义一个用来求一组数据的均值的函数，可以用于 C、C++、VB 等语言相同的方式定义，但方便得多。如计算向量 X=(x1,x2,...,xn) 的均值函数：

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

自定义函数

代码如下：

```
import numpy as np
x=[1,3,6,4,9,7,5,8,2];x
```

```
def xbar1(x):
    n=len(x)
    xm=sum(x)/n
    xm

def xbar2(x):
    n=len(x)
    xm=sum(x)/n
    return(xm)

xbar1(x)
xbar2(x)
```

```
np.mean(x)
```

- 控制语句
- 函数
- 面向对象

面向对象

python 是一种面向对象的语言（一般使用者可不了解）。

前面介绍的序列（向量、一维数组），数据框（矩阵、二维数组）都是 python 的数据对象，各种 python 函数也是对象。由于 python 函数的许多计算结果都放在对象中，这使得 python 的结果通常比 SAS、SPSS 和 Stata 等数据分析软件的结果简洁，需要时才调用，为进一步分析提供了方便。

下面就通过编写一个函数的过程来简单介绍 python 的函数自定义方法和面向对象技术。如计算向量 X=（x1,x2,...,xn）的离均差平方和函数：

```
def SS1(x):
    n=len(x)
    ss=sum(x**2)-sum(x)**2/n
    return(ss)

SS1(S1)
```

```
def SS2(x):
    n=len(x)
    xm=sum(x)/n
    ss=sum(x**2)-sum(x)**2/n
    return[x**2,n,xm,ss]
```