专业·创新·增值

金程教育
GOLDEN FUTURE

Python编程基础知识

Python Programming Basics

纪慧诚
金程教育资深培训讲师
CFA FRM RFP

CONTENTS

PROFESSIONAL · LEADING · VALUE-CREATING

说明：本节Python编程基础教程主要针对Python3.x版本

专业来自101%的投入！

First Step forward python

- Fundemental calculator
  - ✓ IPython can work as a calculator on a fundemental level.
  - ✓ Basical operators such as + , - , * , / , // , % , ** , = and () for grouping. *算数运算符*
  - ✓ Python build-in numerical modules such as math(not for complex), cmath (especially for complex), random(for random numbers) ,etc..
  - ✓ In these modules, there are functions such as abs(x) , ceil(x), exp(x), fabs(x), floor(x), etc..

- **Magic Command**
  - 什么是Magic Command？
    - ✓ 由IPython Kernel 提供的，针对jupyter notebook用户的简洁高效的命令。
  - Magics 主要有两种语法:
    - ✓ **Line magics:** 以 '%' 开始，该行后面都为指令代码，参数用空格隔开。
    - ✓ **Cell magics:** 以 '%%' 开始, 后面的整个单元（Cell）都是指令代码。

      注意，%%魔法操作符只在Cell的第一行使用，而且不能嵌套、重复（一个Cell只有一个）。极个别情况下可以堆叠。
  - 输入 %lsmagic可以获得所有Magic操作符的列表。
  - 输入%matplotlib inline可以在jupyter notebook中输出图表
  - 部分常见的Magic command（见下页）：

➢ Magic command（续）

命令	说明
%quickref	显示 IPython 的快速参考
%magic	显示所有魔术命令的详细文档
%debug	从最新的异常跟踪的底部进入交互式调试器
%hist	打印命令的输入（可选输出）历史
%pdb	在异常发生后自动进入调试器
%paste	执行剪贴板中的 Python 代码
%cpaste	打开一个特殊提示符以便手工粘贴待执行的 Python 代码
%reset	删除 interactive 命名空间中的全部变量/名称
%page OBJECT	通过分页器打印输出 OBJECT
%run script.py	在 IPython 中执行一个 Python 脚本文件
%prun statement	通过 cProfile 执行 statement，并打印分析器的输出结果
%time statement	报告 statement 的执行时间
%timeit statement	多次执行 statement 以计算系综平均执行时间
%who、%who_ls、%whos	显示 interactive 命名空间中定义的变量，信息级别/冗余度可变
%xdel variable	删除 variable，并尝试清除其在 IPython 中的对象上的一切引用

➢ 基本数据类型

- Python 3.x有五个标准的数据类型，分为有符号整数型(int) ，浮点型(float)，复数型(complex)，布尔型(bool)，字符串(string)。其中前3个又称为数字数据类型/数值类型（ Numbers ）。
  - ✓ **int型：**0、1、-1、1009、-290等等；Python3中 整型没有限制大小，可以当作 Long 类型使用，因此Python3 没有 Python2中 的 Long 类型。在Python2中long型用数尾的“L”注明：87887L、-90990L、234567L等等；
  - ✓ **float型：**2.7788、3.277、8.88等等；
  - ✓ **complex 类型：** （用a + bj, 或者complex(a,b)表示，如 4+2j、-9+20j、56+7j等等；
  - ✓ **bool型：**True、False；
  - ✓ **string型：** ’Hello World！’ 等等。

String字符串

- 运算符

操作符	描述
+	字符串连接
*	重复输出字符串
[]	通过索引获取字符串中字符
[:]	截取字符串中的一部分

操作符	描述
in	成员运算符 - 如果字符串中包含给定的字符返回 True
not in	成员运算符 - 如果字符串中不包含给定的字符返回 True
r/R	原始字符串 - 原始字符串：所有的字符串都是直接按照字面的意思来使用，没有转义特殊或不能打印的字符。 原始字符串除在字符串的第一个引号前加上字母"r"（可以大小写）以外，与普通字符串有着几乎完全相同的语法。
%	格式字符串

转义字符

转义字符	描述
\(在行尾时)	续行符
\\	反斜杠符号
\'	单引号
\"	双引号
\a	响铃
\b	退格(Backspace)
\e	转义
\000	空

Python的运算符

- 算术运算符
- 比较（关系）运算符
- 赋值运算符
- 逻辑运算符
- 成员运算符
- 身份运算符

Python的运算符

- 比较（关系）运算符

运算符	描述
==	等于 - 比较对象是否相等
!=	不等于 - 比较两个对象是否不相等
>	大于 - 返回x是否大于y
<	小于 - 返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。注意，这些变量名的大写。
>=	大于等于 - 返回x是否大于等于y。
<=	小于等于 - 返回x是否小于等于y。

Python的运算符

- 赋值运算符

运算符	描述
=	简单的赋值运算符
+=	加法赋值运算符
-=	减法赋值运算符
*=	乘法赋值运算符
/=	除法赋值运算符
%=	取模赋值运算符
**=	幂赋值运算符
//=	取整除赋值运算符

Python的运算符

- 逻辑运算符

运算符	逻辑表达式	描述
and	x and y	布尔"与" - 如果x为False，x and y返回False，否则它返回y的计算值。
or	x or y	布尔"或" - 如果x是True，它返回x的值，否则它返回y的计算值。
not	not x	布尔"非" - 如果x为True，返回False。如果x为False，它返回True。

Python的运算符

- 成员运算符：能否在指定序列中找到限定的值

运算符	描述	实例
in	如果在指定的序列中找到值返回 True，否则返回 False。	x 在 y 序列中，如果 x 在 y 序列中返回 True。
not in	如果在指定的序列中没有找到值返回 True，否则返回 False。	x 不在 y 序列中，如果 x 不在 y 序列中返回 True。

Python的运算符

- 身份运算符：用于比较两个对象的存储单元

运算符	描述	实例
is	is 是判断两个标识符是不是引用自一个对象	x is y, 类似 id(x) == id(y) , 如果引用的是同一个对象则返回 True，否则返回 False
is not	is not 是判断两个标识符是不是引用自不同对象	x is not y ， 类似 id(a) != id(b)。如果引用的不是同一个对象则返回结果 True，否则返回 False。

基本数据结构（底层）

- **列表list**：用[]来定义，可以包含不同类型的对象，可以增减元素，可以跟其他的列表结合或者把一个列表拆分。

```
aList=[123, 4.56,'abc', 7-9j,['inner','list']]
```

- **元组tuple**：用()来定义，可以包含不同类型的对象，但是是不可变的，不可以增减元素，可以重新赋值。

```
aTuple=(123, 4.56,'abc', 7-9j,['inner','list'])
```

基本数据结构（底层）

- **集合set**：分为set() 可变集合和frozenset() 不可变集合，有集合的属性。

```
a = set("aaabbcdde11223")
print(a)
```

```
{'a', '3', 'e', 'd', '1', '2', 'c', 'b'}
```

- **字典dict**：用{}来定义的键值对key-value pairs，也叫关联数组。

```
aDict={'name':'Megan','age':18}
```

➢ String字符串（续）

- 格式化输出

符号	描述
%s	格式化字符串
%d	格式化整数
%u	格式化无符号整型
%f	格式化浮点数字，可指定小数点后的精度
%e	用科学计数法格式化浮点数
%E	作用同%e，用科学计数法格式化浮点数

CONTENTS

PROFESSIONAL · LEADING · VALUE-CREATING

说明：本节Python编程基础教程主要针对Python3.x版本

专业来自101%的投入！

importing

- From…import 语句 Python 的 from 语句让你从模块中导入一个指定的部分到当前命名空间中。

```
from numpy import exp, arange
```

- From…import* 语句 把一个模块的所有内容全都导入到当前的命名空间也是可行的，只需使用如下声明

```
from math import *
```

- 当然我们更多地使用以下的方法

```
import math
```

Importing（续）

- 缩写一个包的名称以便于编程

```
import numpy as np
np.array()...
```

- 调用一个本地的包，比如下例中的support

```
import support
support.print_func('Quantitative Finance !')
Hello, Quantitative Finance !
```

CONTENTS

PROFESSIONAL · LEADING · VALUE-CREATING

说明：本节Python编程基础教程主要针对Python3.x版本

专业来自101%的投入！

- 一个例子：质数函数编制的一般过程

- **什么是函数？**
  - 用来实现特定的或相关联功能的代码段。函数可重复调用。
- **自定义函数的简单规则有：**
  - 函数代码块以 **def** 关键词开头，后接函数标识符名称和圆括号 ()。
  - 任何传入参数和自变量必须放在圆括号中间，圆括号之间可以用于定义参数。
  - 函数的第一行语句可以选择性地使用文档字符串—用于存放函数说明。
  - 函数内容以冒号起始，并且缩进。
  - return [表达式] 结束函数，选择性地返回一个值给调用方。不带表达式的return相当于返回None。

```
def funcname(args1,args2,...):
    f(args1,args2)
    [return aValue]
```

参数传递

- 函数调用过程中涉及到参数传递，形式参数（简称形参，formal parameter）和实际参数（简称实参，real parameter）
- 形式参数：定义函数时出现的参数；
- 实际参数：调用函数时出现的参数。

调用函数时可使用的四种正式参数类型

- 以下四种参数从不同角度描述了参数的特征，并不是分类
  - ✓ 必需参数
  - ✓ 关键字参数
  - ✓ 默认参数
  - ✓ 不定长参数

必备参数

- 若形式参数中没有默认值，且不是不定长参数（ *tupleargs, **dictargs ），必须在实际函数调用中进行赋值

```
def funcA(a,b=10,*c):
    print(a+b)
    print(c)
funcA()
```

```
TypeError: funcA() missing 1 required positional argument: 'a'
```

关键字参数

- 除了其他语言中常见的位置参数外，Python还支持关键字参数
- 关键字参数无所谓顺序
- 关键字参数与位置参数可以同时使用

```
def funcA(a, b, c):
  print(a,b,c)
funcA(1, c = 3, b = 2)
```

```
1 2 3
```

默认参数

parameter = expression（default value）

- 当参数有缺省值时，在调用时实际参数的赋值可能被省略，该参数以默认值替代。
- 在函数定义时，如果一个参数有一个缺省值，那么接下来的参数也必须有缺省值。

不定长参数

```
def funcname(args1,*tupleargs, **dictargs ):
```

- 函数定义时，认为在实际函数调用中可能出现参数列表中难以预计或者多余的实际参数。
- 函数调用时，位置参数和关键字参数可以同时使用
- *tupleargs：针对可变数量的参数，以tuple的形式进行存储，否则出现TypeError

```
def funcA(a, b, *c):
  print a
  print b
  print "length of c is: %d " % len(c)
  print c
funcA(1, 2, 3, 4, 5, 6)
```

```
1
2
length of c is: 4
(3, 4, 5, 6)
```

不定长参数（续）

- **dictargs：针对参数名字没有在形式参数中出现的实际参数，以dictionary的形式进行存储，否则出现TypeError

```
def funcB (a, **b):
  print a
  for x in b:
     print x + ": " + str(b[x])
funcB(66, c='Hello', b=88)
```

```
66
c: Hello
b: 88
```

全局变量和局部变量

- 定义在函数内部的未声明全局作用域的变量拥有一个局部作用域，定义在函数外的拥有全局作用域。
- 局部变量只能在其被声明的函数内部访问，而全局变量可以在整个程序范围内访问。
- 函数内部的变量若声明是全局变量，则作用域不限于本函数
- 全局变量的声明需要使用“global”语句，且须分开赋值

```
global variable
```

- 在函数外未声明global的变量，被默认为全局变量
- **函数调用**：在调用函数时，所有在函数内声明的变量名称都将被加入到作用域中

➢ 全局变量和局部变量（续）

```
d=4
def funcA(a,b,c):
    global f
    f=1
    e=a+b+c+d
    print(e)
funcA(1,c = 3, b = 2)
print(f)
print(a)
```

```
10
1
NameError: name 'a' is not defined
```

匿名函数lambda

- 匿名函数lambda对简单的函数定义进行了简化

```
lambda argument list: expression
```

- Lambda内部的函数结构

```
def name(arguments):
    return expression
```

CONTENTS

PROFESSIONAL · LEADING · VALUE-CREATING

专业来自101%的投入!

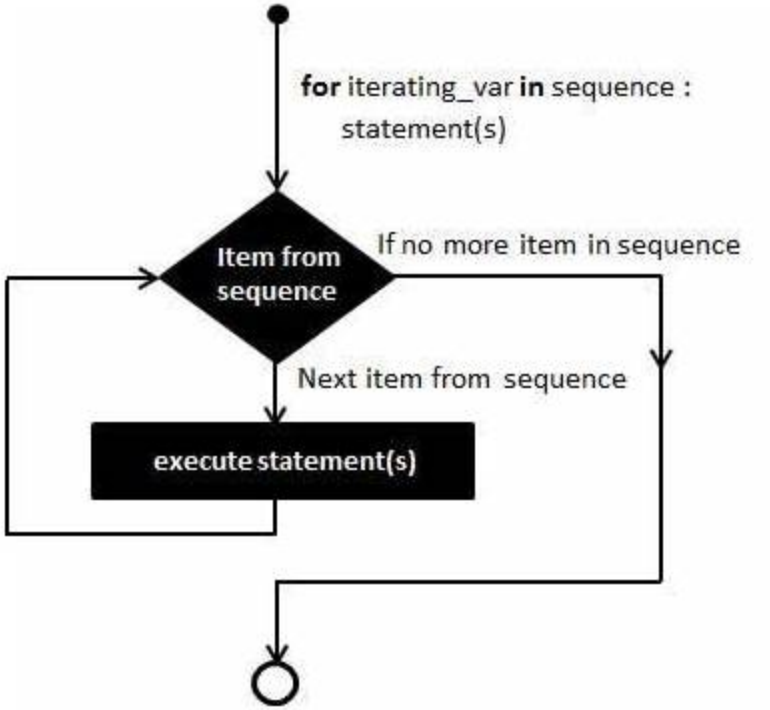
- **控制结构：**
  - 条件语句：if
  - 循环语句：for , while
  - 特殊语句：break , continue

- **异常处理**
  - 常用语句：try , with

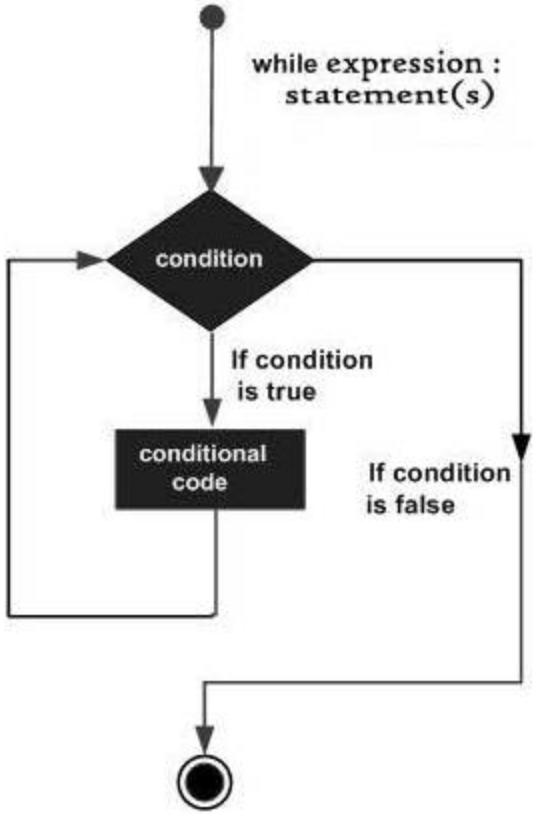
条件语句If

```
if expression : suite
        (elif expression : suite )*
        [else : suite]
```

- **循环语句：for**



- 循环语句：while



特殊语句：break

- break: 在循环结构中出现Break语句的时候，能将该循环强制停止，然后退出该循环
- continue:强制停止循环中的这一次执行，直接跳到下一次执行。

➢ 异常处理：try

- python中try-except-else-finally语句的完整格式如下所示：

```
try:
    Normal execution block
except A:
    Exception A handle
except B:
    Exception B handle
except:
    Other exception handle
else:
    if no exception, get here
finally:
    print("finally")
```

- ✓ 正常执行的程序在try下面的Normal execution block执行块中执行，在执行过程中如果发生了异常，则中断当前在Normal execution block中的执行跳转到对应的异常处理块中开始执行；
- ✓ python从第一个except X处开始查找，如果找到了对应的exception类型则进入其提供的exception handle中进行处理，如果没有找到则直接进入except块处进行处理。except块是可选项，如果没有提供，该exception将会被提交给python进行默认处理，处理方式则是终止应用程序并打印提示信息；
- ✓ 如果在Normal execution block执行块中执行过程中没有发生任何异常，则在执行完Normal execution block后会进入else执行块中（如果存在的话）执行。
- ✓ 无论是否发生了异常，只要提供了finally语句，最后一步总是执行finally所对应的代码块。

Thank you!