第三章：Python 基础

目录

[3.1 语句和语法 1](#_Toc350420388)

[3.2 变量赋值 2](#_Toc350420389)

[3.3标识符 3](#_Toc350420390)

[3.3基本风格指南 5](#_Toc350420391)

[3.5内存管理 7](#_Toc350420392)

[3.6将引入的其他模块中常用的变量重新赋予一个本地别名(ls = os.linesep)不仅可以避免冗长的变量名, 又可以提高加载速度(因为现在是本地变量了) 8](#_Toc350420393)

## 3.1 语句和语法

1、 (#)表示之后的字符为 Python 注释

在商业项目中一定不要忘记注释，比如对一个函数的注释：函数的功能，函数的各个参数的意思，函数的返回值。在什么输入的情况下做什么样的输出。更新时间，更新人，更新时间，做了哪些更新。

怎么写中文注释：

我在python脚本中输出中文的时候，他会有问题：

支持的方法是：#coding=utf-8,或者别加了，翻译过来加到这个上面。

推荐的代码宽度：一行八十个字符，提高代码的可读性。

2、换行 (\n) 是标准的行分隔符（通常一个语句一行）

3、反斜线 ( \ ) 继续上一行

4、分号 ( ; )将两个语句连接在一行中

Python中同一行只能写一个语句，但是有的时候语句比较短的时候，你可以把两个语句用这种方式写在一行。如：



但是不推荐这样做，哪怕再短，我们也不用这个东西。每种语言他有自己的思想。学一种语言对人的思想带来的改变是很大的。学任何一门语言你去抓他的思想，真正的去学算法，网络的底层设计等，这更能走向更高的境界吧。

5、冒号 ( : ) 将代码块的头和体分开

6、语句（代码块）用缩进块的方式体现

语句(代码块)用缩进块方式体现: 同一个代码组的代码必须严格左对齐..由于不同的editor制表符宽度不一, 为了使代码跨平台, 推荐使用4个空格缩进。Python推荐使用空格，这样便于跨平台。避免使用制表符，但是我们习惯使用制表符，目前对我的开发暂时无任何影响。其实在eclipse里面，他会自动缩进的。

7、不同的缩进深度分隔不同的代码块

8、 Python 文件以模块的形式组织

每一个Python脚本文件都可以被当成是一个模块。模块以磁盘文件的形式存在。当一个模块变得过大，并且驱动了太多功能的话，就应该考虑拆一些代码出来另外建一个模块。模块里的代码可以是一段直接执行的脚本，也可以是一堆类似库函数的代码，从而可以被别的模块导入(import)调用。记得我们在上一章中曾提到过，模块可以包含直接运行的代码块、类定义、函数定义或这几者的组合。

## 3.2 变量赋值

1、赋值并不是直接将一个值赋给一个变量，尽管你可能根据其它语言编程经验认为应该如此。在 Python 语言中，对象是通过引用传递的。在赋值时，不管这个对象是新创建的，还是一个已经存在的，都是将该对象的引用（并不是值）赋值给变量。

2、Python 的赋值语句不会返回值：如果你比较熟悉 C， 你会知道赋值语句其实是被当成一个表达式（可以返回值）。不过这条并不适合于 Python, Python的赋值语句不会返回值。类似下面的语句在 Python 中是非法的：

>>> x = 1

>>> y = (x = x + 1) # assignments not expressions! File "<stdin>", line 1

y = (x = x + 1)

^

SyntaxError: invalid syntax

3、链式赋值没问题， 看（本章稍后部分会给出更多的例子）：

>>> y = x = x + 1

>>> x, y

(2, 2)

4、增量赋值

x += 1

5、多重赋值

>>> x = y = z = 1

>>> x

1

>>> y

1

>>> z

1

在上面的例子中，一个值为 1 的整数对象被创建，该对象的同一个引用被赋值给 x、y 和z 。也就是将一个对象赋给了多个变量。当然，在 Python 当中，将多个对象赋给多个变量也是可以的。

6、“多元”赋值（将多个对象赋给多个变量也是可以的。）

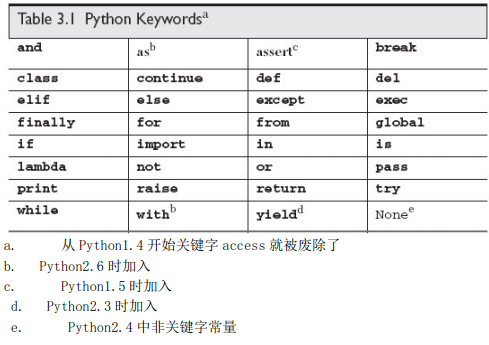
这不是官方Python术语，而是我们将 "mul-tuple"连在一起自造的。因为采用这种方式赋值时， 等号两边的对象都是元组。多元赋值: a, b, c = 1, 2, ‘string’, 建议使用tuple（元组）的方式进行多元赋值: (a, b, c) = (1, 2, ‘string’)，python推荐使用这种方法。

1. swap操作: x, y = y, x

## 3.3标识符

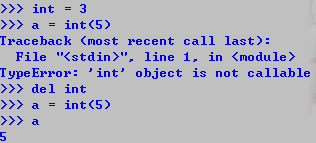
1、不推荐使用关键字集合和还有称为 built-in 标识符集合，虽然它们不是保留字，但是不推荐使用这些特别的名字。

2、python的关键字列表和iskeyword()函数在keyword模块, 方便查阅。



3、内建: built-in可以看作是系统保留字….对于一些内建函数需要覆盖(重定义, 替换)…built-in是\_\_builtins\_\_模块的成员, 该模块由解释器自动导入内建就是我的python解释器自己实现了很多的模块，例如string、int等，我的内建函数如type，尽量不要使用用他们命名，不然内部实现的会被覆盖。不要使用int、string、list等去给你的变量命名。除非你想自己定义int，然后使用自己的。

如下：



你覆盖了全局里的int。

4、下划线: 作为变量前缀和后缀指定特殊变量。在python中没有，私有的、公开的、受保护的等这些区别。

\_xxx: 不用’from module import\*’导入

\_\_xxx\_\_: 系统定义名字

\_xxx: 类中的私有变量名

注意：

避免用下划线作为变量名的开始

因为下划线对解释器有特殊的意义，而且是内建标识符所使用的符号，我们建议程序员避免用下划线作为变量名的开始。一般来讲，变量名\_xxx 被看作是“私有的”，在模块或类外不可以使用。当变量是私有的时候，用\_xxx 来表示变量是很好的习惯。因为变量名\_\_xxx\_\_对Python 来说有特殊含义，对于普通的变量应当避免这种命名风格。

## 3.3基本风格指南

1、注释

2、文档

3、缩进

4、选择标识符名称

5、模块结构和布局

1）模块组织

用模块来合理组织你的 Python 代码是简单又自然的方法。你应该建立一种统一且容易阅读的结构，并将它应用到每一个文件中去。下面就是一种非常合理的布局：

# (1) 起始行(Unix)

通常只有在类 Unix 环境下才使用起始行，有起始行就能够仅输入脚本名字来执行脚本，无需直接调用解释器。

# (2) 模块文档

简要介绍模块的功能及重要全局变量的含义，模块外可通过 module.\_\_doc\_\_ 访问这些内容。

# (3) 模块导入

导入当前模块的代码需要的所有模块；每个模块仅导入一次（当前模块被加载时）；函数内部的模块导入代码不会被执行， 除非该函数正在执行。

# (4) 变量定义

这里定义的变量为全局变量，本模块中的所有函数都可直接使用。从好的编程风格角度说， 除非必须，否则就要尽量使用局部变量代替全局变量，如果坚持这样做，你的代码就不但容易维护，而且还可以提高性能并节省内存。

# (5) 类定义

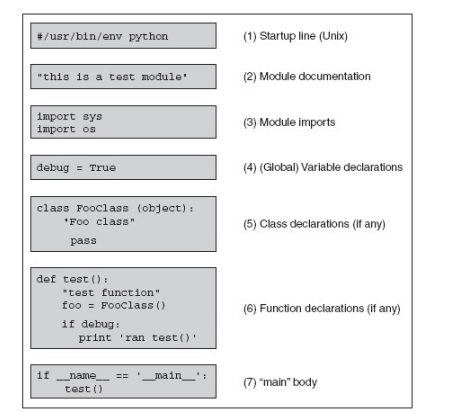
所有的类都需要在这里定义。当模块被导入时 class 语句会被执行, 类也就会被定义。类的文档变量是 class.\_\_doc\_\_。

# (6) 函数定义

此处定义的函数可以通过 module.function()在外部被访问到，当模块被导入时 def 语句会被执行， 函数也就都会定义好，函数的文档变量是 function.\_\_doc\_\_。

# (7) 主程序

无论这个模块是被别的模块导入还是作为脚本直接执行，都会执行这部分代码。通常这里不会有太多功能性代码，而是根据执行的模式调用不同的函数。



2）\_\_name\_\_用于指示模块应该如何被加载, 如果值是”\_\_main\_\_”说明是主模块, 如果是模块名, 说明是被导入的。每个模块都有一个\_name\_属性。

3）主模块测试代码

def main():

# business process

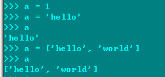
if(\_\_name\_\_ == ‘\_\_main\_\_’)：

main()

main函数的名字随便，如main1等等，你只需要在if（\_name\_==’\_main\_’）中调用一下即可。

## 3.5内存管理

1、弱类型 – 动态类型。不需要事先声明，并且无需指定类型。



1. programmer不用关心内存管理
2. 变量名会被‘回收’

4、del语句能够直接释放资源（资源是实际的对象，一个名字只能指向一个资源，但是一个资源可被多个名字指向）

* 1. 变量未初始化不允许使用
  2. 引用计数: 对于一个内存堆中的对象, 有多少个指针引用它..引用计数就是多少, 引用计数为0时, 该对象可以被垃圾回收器回收。
     1. 增加
        1. 对象创建
        2. 别名创建(引用赋值)
        3. 参数传递(引用传值)
        4. 被容器引用
     2. 减少
        1. 引用生命周期结束
        2. 对象别名被显示销毁 del y
        3. 对象别名被赋予其他引用
        4. 窗口对象本身被销毁
     3. del
        1. 从现在的名字空间中删除一个别名
        2. 对象的引用计数减一
  3. 垃圾回收: 有对象引用计数为0, 对象被显示告知需要销毁, 有内存消耗大户存在导致系统压力较大时, 垃圾回收机制运行, 清理需要回收的内存区域…垃圾回收机制还有一个循环垃圾回收器, 确保释放循环引用对象(a引用b, b引用a, 导致其引用计数永远不为0)

## 3.6将引入的其他模块中常用的变量重新赋予一个本地别名(ls = os.linesep)不仅可以避免冗长的变量名, 又可以提高加载速度(因为现在是本地变量了)