Python 基础语法学习笔记

（一）标识符

在 `Python` 中，所有标识符可以包括英文、数字以及下划线 `_`，但不能以数字开头。

`Python` 中的标识符是区分大小写的。

以下划线开头的标识符是有特殊意义的。以单下划线开头 `_foo` 的代表不能直接访问的类属性，需通过类提供的接口进行访问，不能用 `from xxx import *` 而导入。

以双下划线开头的 `__foo` 代表类的私有成员，以双下划线开头和结尾的 `__foo__` 代表 `Python` 里特殊方法专用的标识，如 `__init__()` 代表类的构造函数。

以下关键字不能作为变量

```
and,as,assert,break,class,continue,def,del,elif,for,from,global,if,import,in,is,
lambda,not,try,while,with,yield
```

（二）缩进

Python 的代码块不使用大括号 `{}` 来控制类，函数以及其他逻辑判断,而是用缩进来写模块。

```
if True:
    print ("True")
else:
    print ("False")
```

以下代码会报错

```
if True:
    print ("Answer")
    print ("True")
else:
    print ("Answer")
  print ("False")
```

语句中一般以新行作为语句的结束符。

但是我们可以使用斜杠 `'\'` 将一行的语句分为多行显示，如下所示：

```
total = item_one + \
        item_two + \
        item_three
```

语句中包含 `[]`, `{}` 或 `()` 括号就不需要使用多行连接符。如下：

```
number = ['one', 'two', 'three',
        'four', 'five']
```

（三）字符串

字符串都使用引号，此时单引号和双引号没有区别。但是如果字符串中有单引号的话，外面就得用双引号；如果里面有双引号，外面就用单引号；如果既有单引号又有双引号，那么用三引号。

三引号也可以多行注释代码，单行注释，用 #

```
Message = "I'm BH."
info = 'Python comments uses "#".'
NewMessage = '''Hello "python".'''
'''
注释1
'''
#注释2
```

（四）变量类型

`Python` 中变量不需要指定数据类型，直接使用等号定义即可。

`Python` 变量里面存的是内存地址，将变量赋值给另一个变量，是将内存地址赋给另一个变量。

```
counter = 100 # 赋值整型变量
miles = 1000.0 # 浮点型
name = 'new'
new_name = name
```

在内存中存储的数据可以有多种类型。

Python有五个标准的数据类型：

`Numbers` （数字）

Python支持四种不同的数字类型：

> `int`：有符号整型
>
> `long`：长整型，也可以代表八进制和十六进制（用 `L` 表示）
>
> `float`：浮点型
>
> `complex`：复数（可以使用 `complex(a,b)` 或 `a+bj` 表示）

`String` （字符串）

python的字串列表有2种取值顺序:

> 从左到右索引默认0开始的，最大范围是字符串长度少1
>
> 从右到左索引默认-1开始的，最大范围是字符串开头

`[头下标:尾下标]` 获取的子字符串包含头下标的字符，但不包含尾下标的字符。

```
>>> s = 'abcdef'
>>> s[1:5]
'bcde'
```

加号 `'+'` 是字符串连接运算符，星号 `'*'` 是重复操作。

列表截取可以接收第三个参数，参数作用是截取的步长，以下实例在索引 1 到索引 4 的位置并设置为步长为 2（间隔一个位置）来截取字符串。

`List`（列表）

列表使用类似字符串。

`Tuple`（元组）

元组使用 `()` 标识，类似列表但是不能进行二次赋值，可以理解为常量。

`Dictionary`（字典）

列表是有序的对象集合，字典是无序的对象集合。

两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取。

字典用 `"{ }"` 标识。字典由索引 `key` 和它对应的值 `value` 组成。

```
dict = {}
dict['one'] = "This is one"
dict[2] = "This is two"
tinydict = {'name': 'bh','code':1234, 'age': '114514'}
```

（五）运算符

基本与 `C` 语言相同

特殊：

运算符	描述	实例
`**`	幂 - 返回 `x` 的 `y` 次幂	`a**b` 为 10 的 20 次方，输出结果 100000000000000000000
`//`	取整除 - 返回商的整数部分（**向下取整**）	`9 // 2 = 4` , `-9 // 2 = -5`
`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。
`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。	(a and b) 返回 20。
`or`	x or y 布尔"或" - 如果 x 是非 0，它返回 x 的计算值，否则它返回 y 的计算值。	(a or b) 返回 10。
`not`	not x 布尔"非" - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。	not(a and b) 返回 False
`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

ps：`id()` 函数用于获取对象内存地址。

（六）语句

条件语句

```
if 判断条件：
    执行语句……
else：
    执行语句……
#
if 判断条件1:
    执行语句1……
elif 判断条件2:
    执行语句2……
elif 判断条件3:
    执行语句3……
else:
    执行语句4……
```

循环语句

```
while 判断条件(condition)：
    执行语句(statements)……
```

`while … else` 在循环条件为 `false` 时执行 `else` 语句块：

```
while count < 5:
   print count, " is  less than 5"
   count = count + 1
else:
   print count, " is not less than 5"
```

```
for iterating_var in sequence:
   statements(s)
```

（七）函数

你可以定义一个由自己想要功能的函数，以下是简单的规则：

> 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 functionname( parameters ):
   "函数_文档字符串"
   function_suite
   return [expression]
```

关键字参数

```
#可写函数说明
def printinfo( name, age ):
   "打印任何传入的字符串"
   print "Name: ", name
   print "Age ", age
   return

#调用printinfo函数
printinfo( age = 100, name = "bh" )
```

不定长参数

```
def functionname([formal_args,] *var_args_tuple ):
   "函数_文档字符串"
   function_suite
   return [expression]
```

加了星号 `*` 的变量名会存放所有未命名的变量参数。