2018/1/17小结

变量名只能包含字母、数字、和下划线 变量名不能包含空格，但可以使用下划线来分隔其中的单词。 不要将python关键字和函数名用作变量名。

1.python中的标识符是区分大小写的。

2.标示符以字母或下划线开头，可包括字母，下划线和数字。

3.以下划线开头的标识符是有特殊意义的

2018/1/18 基本语法

1.在python中单引号的效用是等同于双引号，所有的字符串用单引号或者双引号。

2.在python中引入了三引号，针对特别长的文字，避免重复使用\。

3.Python是要求有强制缩进。

4.python中的变量不用声明类型，可以直接使用。

5.Python的函数声明实现是一起的。

6.在if 等语句的条件后面，必须有“：”。

2018/1/19

增加元素可以使用append()和insert()两个方法：

在列表尾部增加元素，使用append()

在列表随意位置增加元素，使用insert()

删除元素可以使用del、pop()、remove()三个方法

知道元素的位置然后将元素删除，可以使用del（)

删除列表的最后一个元素，可以使用pop()

当只知道元素值的存在，不知道在列表中的位置时，可以使用remove()删除此元素

pop()和remove()删除的元素都可以拿来再使用但是：

1、pop()是在删除元素的同时将删除的元素放进一个可以使用的变量中存储

2、remove()是先将要删除的元素放进一个变量中，然后将变量放进remove()中执行删除，也可以直接将删除的元素放进remove()执行删除

2018/1/20

组织列表方法：

使用方法sort（）对列表进行永久性排序如：

cars=['bmw', 'audi', 'toyota', 'subaru']

cars.sort()

print(cars)

【‘audi’，‘bmw’‘subaru’‘toyota】

反转列表元素的排列顺序，可使用reverse（）

cars=【‘bwm’‘audi’‘toyota’‘subaru’】

print（cars）

cars.reverse（）

print（cars）

注意，reverse（）不是指按与字母顺序相反的顺序排列列表元素，而只是反转列表元素的排列顺序：

【‘bmw’，‘audi’，‘toyota’，‘subaru’】

【‘subaru’，‘toyota’‘audi’，‘bmw’】

使用函数len（））可以快速熟悉列表长度

cars=【‘bwm’‘audi’‘toyota’‘subaru’】

len（cars）

输出结果为：4

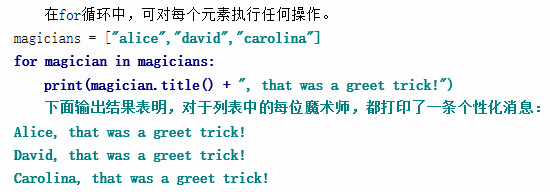
2018/1/24

For循环：需要对列表中的每个元素都执行相同的的操作时，可以使用python中的for循环。例如：

magicians = ["alice","david","carolina**"]**for magician in magicians:  
print(magician)  
运行结果：  
alice  
david  
carolina

刚开始使用循环时牢记，对列表中的每个元素，都将执行循环制定的步骤，而不管列表包含多少元素。如果列表包含一百万个元素，Python就重复执行一百万次，且通常速度非常快。

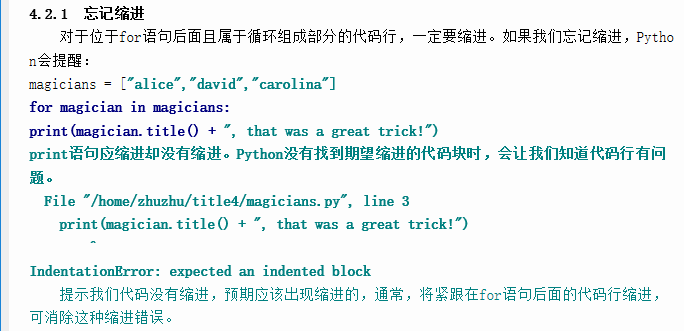
在for循环中执行更多的操作；

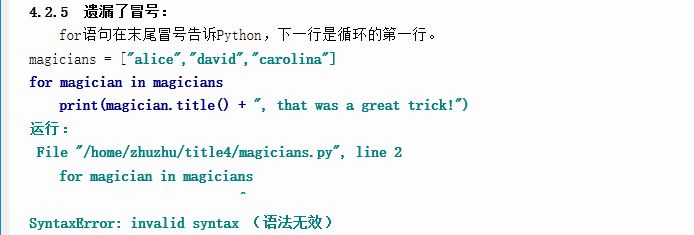


在for循环中，想包含多少行代码都可以。在代码行for magician in magicians后面，每个缩进的代码行都是循环的一部分，且将针对列表中的每个值都执行一次。因此，可对列表中能够的每个值执行任意次数的操作。

2018/1/25

避免缩进错误：





如果不小心遗漏了冒号，将导致语法错误，运行会提示无效语法。

2018/1/26

创建数值列表：

**使用函数range()：**

**for value in range(1,6):  
print(value)  
这样，输出将从1开始，到5结束：  
1  
2  
3  
4  
5  
使用range()函数时，如果输出不符合预期，尝试将指定的值加1或减1.**

**使用range()函数创建数字列表：**

numbers = list(range(1,6))  
print(numbers)  
结果如下：  
[1, 2, 3, 4, 5]  
使用函数range()时，还可指定步长。例如，下面的代码打印1~10内的奇数：

numbers = list(range(1,10,2))  
print(numbers)  
   
在这个实例中，函数range()从2开始数，然后不断地加2，直到达到或超过终值(10),因此输出的结果如下：  
[1, 3, 5, 7, 9]

两个星号(\*\*)表示乘方运算。例如：

squares = []  
**for num in range(1,11):  
square = num\*\*2  
squares. append(square)  
print(squares)**

输出结果为：[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

要使用这种语法，首先指定一个描述性的列表名，如squares；然后指定一个左方括号，并定义一个表达式，用于生成要存储得到列表中的值。在这个示  
例中，表达式为value\*\*2，它计算平方值。接下来，编写一个for循环，用于给表达式提供值，再加上右方括号。在这个实例中，for循环为for value in   
range(1,11),它将值1~10提供给表达式value\*\*2.注意，这里的for语句末尾没有冒号。

2018/1/27

元组：定义：**Python将不能修改的值称为不可变的**，而不可变的列表被称为**元组**

定义元组（元祖看起来犹如列表，但使用圆括号而不是方括号来标识。定义元祖后，就可以使用索引来访问其元素，就像访问列表元素一样。）

例如，如果有一个大小不应改变的矩形，可将其长度和宽度存储在一个元祖中，从而确保它们是不能修改的：

**dimensions.py**

dimensions = (200,50)  
    print(dimensions[0])

200  
    print(dimensions[1])

50

下面尝试修改元素dimensions中的一个元素

dimensions = (200,50)  
dimensions[0] = 250  
print(dimensions)

得到结果：

Traceback (most recent call last):  
      File "/home/zhuzhu/title4/dimensions.py", line 2, in <module>  
      dimensions[0] = 250  
    TypeError: 'tuple' object does not support item assignment

代码试图修改矩形的尺寸时，Python报告错误，这刚好符合元祖的定义。

2018/1/28

遍历元组中的所有值：

像列表一样，也可以使用for循环来遍历元组中的所有值：

    dimensions = (200,50)  
    for dimension in dimensions:  
         print(dimension)

    就像遍历列表时一样，Python返回元组中所有的元素：

    200

    50

修改元组变量：

虽然不能修改元组的元素，但可以给存储元组的变量赋值。因此，如果要修改前述矩形的尺寸，可重新定义整个元组：

  dimensions = (200,50)  
  print("Original dimensions: ")  
 for dimension in dimensions:  
  print(dimension)  
  
 dimensions = (400,100)  
  print("\nModified dimensions: ")  
  for dimension in dimensions:  
  print(dimension)

首先定义了一个元组，并将其存储的尺寸打印了出来；接下来，将一个新元组存储到变量dimensions中；然后，打印新的尺寸。这次，Python不会报告任何错误，因为给元组变量赋值是合法的：

Original dimensions:   
200  
50

Modified dimensions:

400

100

相比于列表，元组是更简单的数据结构.如果需要存储的一组值在程序的整个生命周期内都不变，可使用元组。