变量命名规则

只包含字母、数字和下划线，以字母和下划线开头不可用数字开头；

变量名不能包含空格，使用下划线代替空格改开单词；

不要用python的函数和关键字命名变量；

变量名要简短又具描述性；

慎用I或O，可能被看成1和0.

upper() 将字符串全改为大写

lower() 小写

title() 以首字母大写的方式显示每个单词

lstrip(),rstrip(),strip()remove（）

str()让数字编程字符串，避免程序错误

[]表示列表

修改列表内值，和直接变量赋值一样

append（）列表末尾加元素

insert（）可在任何位置加元素

例:(.insert(0,’ducati’)）

del删除元素

例：del motorcycle[0]

pop()删除末尾的元素，pop()可以删除列表中任何位置的元素pop（0）

remove（）不知道元素的位置，只知道元素的值，删除并标注用

sort（）永久性修改列表排列顺序，按字母顺序排序

sort(reverse = True)为反字母顺序排序

print (sorted())对列表临时排序

reverse()以列表的反向顺序永久性修改列表

print(len())打印列表元素数

for语句遍历整个列表每个元素

例：for magician in magicians:

print(magician) #循环内

print(magician) #循环外

注意缩进：结束循环，提供总结性输出或接着执行程序必须完成的其他任务

代码放在for循环后，且不缩进

循环不要忘记冒号，找出单字符错误很耗时

range()生成一系列数字

例：for value in range(1,5):

使用函数list()将range()做成列表

例：list(range())

使用range可指定步长，例如输出偶数

例：list(range(2,11,2) #每两个数间距位2

如何将前10个整数的平方加入到一个列表

列表解析

例：squares = [value\*\*2 for value in range(1,11)]

print (squares)

一行代码取代三行

squares = []

for valve in range(1,11):

square = value\*\*2

squares.append(square)

print (squares)

切片

例：players = ['charles','martina','micheal','florence','eli']

print (players[0:3])

print (players[1:3])

print (players[:3])

print (players[2:])

print (players[-3:]]

遍历切片

for player in players[1;3]

print (player.title())

[:]同时省略起始索引和终止索引，包含整个列表

例：friend\_foods = my\_foods

是将friend\_foods赋值给my\_foods，而不是给整my\_foods副本储存到friend\_foods

元组 python将不能修改的值称为不可变的，不可变的列表称为元组

元组用圆括号表示，不用方括号。

虽然不能修改元组的元素，但是可以给储存元组的变量赋值

格式指南

缩进 PEP8建议每级缩进使用四个空格，提高可读性，留下足够的多级空间

行长 不要超过79个字符，影响可读性

空行 将程序的不用部分分开要用空行，不能滥用，隔开两部分用一个空行合适，不要用三四个，影响可读性

If语句

每条if语句的核心都是一个值为True或False的表达式，这种表达式就是条件测试。

条件测试值为True，Python就执行紧跟if语句的代码，如果是False，忽略这些代码

检查是否相等 赋值后Python检查区分大小写，不区分的方法：

例：car.lower() = ‘audi’

！表示不 ！= 编写程序大多数表达式都检查两个值是否相等，但有时候检查两个值是否不等效率更高

比较数字

使用and检查多个条件，都通过Ture

使用or检查多个条件，一个通过就可以Ture

in检查特定值是否在列表中

例：for ‘green peppers’ in toppings:

not in检查不在列表中用

例：for ‘green peppers’ not in toppings:

布尔表达式

if语句和for语句后都需要缩进，否则忽略

if-else语句 条件测试通时执行一个操作，并在没有通过时执行另一个操作，else语句让你能指定条件测试未通过时要执行的操作

if-else结构非常适合用于要让python执行两种操作之一的情况

if-elif-else检查超过两种情况

简洁代码

例: if age < 4:

price = 0

elif age < 18:

price = 5

else:

price = 10

print("Your admission cost is $" + str(price) + ".")

使用多个elif代码

省略else代码 if-elif结构不必须有else代码块，一些情况使用一条elif处理的情形更清晰，else是一条包罗万象的语句，只要不满足if或elif就会去执行，会引入无效、恶意数据

测试多个条件 if-elif-else结构，遇到通过的时候，就跳过余下测试。如果要检查你关心的所有条件就要每条单独用if语句然后输出

检查特殊元素 在for循环中加入if语句

确定列表不是空的 在if循环中加入for语句，if [列表]：

设置if语言格式 养成良好的格式设置习惯， ==、>=、<=等比较运算符两遍添加一个空格

字典

一个简单的字典

例：

alien\_0 = {'color': 'green', 'points': 5}

print(alien\_0['color'])

print(alien\_0['points']

访问字典中的值 依次指定字典名和放在方括号里的键。

添加键-值对 字典动态结构，可随时添加键-值对，依次指定字典名、用方括号括起的键和相关联的值

创建空字典 使用对空的花括号定义一个字典，储存用户提供的数据或在编写能自动生成大量键-值对的代码时，通常先定义一个空字典

修改字典中的值

删除字典中的值 使用del语句彻底删除，指定字典名和要删除的键

由类似对象组成的字典 需要使用多行来定义字典时，再输入左花括号后按回车键，再在下一行缩进四个空格，指定一个键-值对，并在后面加一个逗号，换行

例：

favorite\_languages = {

'jen': 'python',

'sarah': 'c',

'edward': 'ruby',

'phil': 'python',

}

在最后一个键-值对后加一个逗号也是个好做法，方便下次编辑

遍历所有键-值对

例：

user\_0 = {

'username': 'efermi',

'first': 'enrico',

'last': 'fermi',

}

for key, value in user\_0.items():

print("\nKey: " + key)

print("Value: " + value)

字典中方法items()以列表返回可遍历的(键, 值) 元组数组

方法keys()遍历字典中所有的键

方法values()遍历字典中所有的值

例：

for key,value in dictionary.items():

for key in dictionary.keys(): 可以替换成 for key in dictionary:

for value in dictionary.values():

用key作为变量检查是否在特定列表中

按顺序遍历字典中的个所有键 在for语句中，对方法dictionary.keys()调用函数sorted()

遍历列表中所有的值，产生大量的重复项，使用集合set()去重复，集合 类似于列表，但每个元素独一无二。

嵌套 将一系列字典储存在列表中，或将列表作为值储存在字典中，可以列表嵌套字典，字典嵌套字典

字典中储存列表 键-值对 键中可以是包含很多值的列表

列表和字典的嵌套层级不应太多，有时可能有更简单的解决问题的方案

函数input()让程序暂停运行，等待用户输入一些文本。获取用户输入后，Python将其储存在一个变量中，以便使用。

提示可能超过一行，例如需要指出特定输入法原因，可将提示储存在一个变量中，再将该变量传递给函数input（）。

使用 += 在变量的字符串末尾添加一个字符串,使用 = 会给变量重新赋值。