**DAY 1**

1.python 中的 and 从左到右计算表达式，若所有值均为真，则返回最后一个值，若存在假，返回第一个假值；or 也是从左到有计算表达式，返回第一个为真的值；其中数字 0 是假，其他都是真；字符 "" 是假，其他都是真；

2.is和==的区别:

# is用于判断两个变量是否引用同一个对象,==用于判断两个变量的值是否相等

3.字符串不可以改变

"""

输出结果name=jay是因为name变量重新指向"jay"对象,而name2=jack是因为,虽然定义了name2=name,

但是实际上只是把name2这个变量指向了"jack"对象,所以虽然name的指向发生了变化,但是并不影响name2的指向.

"""

# name = "jack"

# name2 = name

# name = "jay"

# print(name)

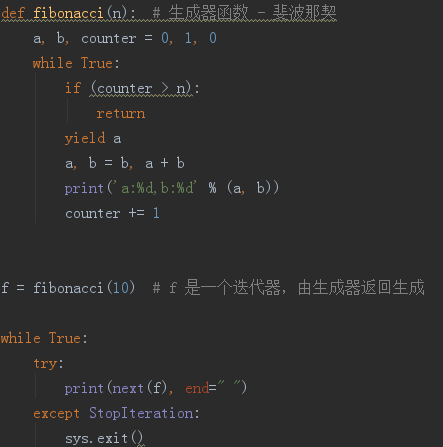
# print(name2)

4.getpass模块在pycharm客户端中使用不和谐,在windows小黑板下可以正常使用

**DAY 2**

1.在 Python 中，使用了 yield 的函数被称为生成器（generator）。

跟普通函数不同的是，生成器是一个返回迭代器的函数，只能用于迭代操作，更简单点理解生成器就是一个迭代器。在调用生成器运行的过程中，每次遇到 yield 时函数会暂停并保存当前所有的运行信息，返回yield的值。并在下一次执行 next()方法时从当前位置继续运行。



2.在 python 中，strings, tuples, 和 numbers 是不可更改的对象，而 list,dict 等则是可以修改的对象。

2.1.不可变类型：变量赋值 a=5 后再赋值 a=10，这里实际是新生成一个 int 值对象 10，再让 a 指向它，而 5 被丢弃，不是改变a的值，相当于新生成了a。

2.2.可变类型：变量赋值 la=[1,2,3,4] 后再赋值 la[2]=5 则是将 list la 的第三个元素值更改，本身la没有动，只是其内部的一部分值被修改了。

python 函数的参数传递：

2.3.不可变类型：类似 c++ 的值传递，如 整数、字符串、元组。如fun（a），传递的只是a的值，没有影响a对象本身。比如在 fun（a）内部修改 a 的值，只是修改另一个复制的对象，不会影响 a 本身。

2.4.可变类型：类似 c++ 的引用传递，如 列表，字典。如 fun（la），则是将 la 真正的传过去，修改后fun外部的la也会受影响

**python 中一切都是对象，严格意义我们不能说值传递还是引用传递，我们应该说传不可变对象和传可变对象。简而言之一句话,不可变的传的都是值,可变的传的都是对象本身**

3.函数参数分为:必需参数,关键字参数,默认参数,不定长参数

3.1.必须参数: 必需参数须以正确的顺序传入函数。调用时的数量必须和声明时的一样。

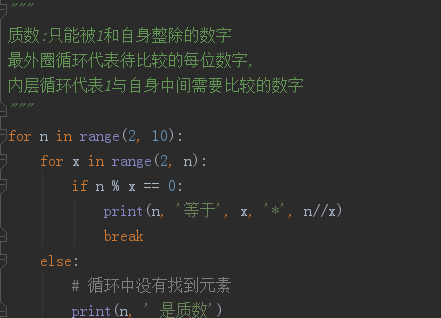
3.2.关键字参数:关键字参数和函数调用关系紧密，函数调用使用关键字参数来确定传入的参数值.使用关键字参数允许函数调用时参数的顺序与声明时不一致，因为 Python 解释器能够用参数名匹配参数值。

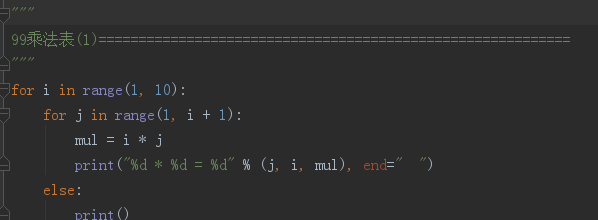
3.3.默认参数: 调用函数时，如果没有传递参数，则会使用默认参数。

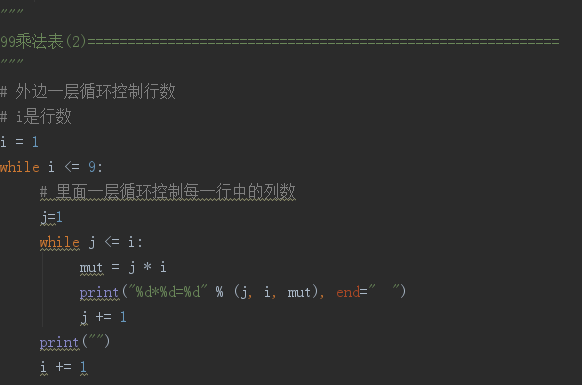
3.4.不定长参数:你可能需要一个函数能处理比当初声明时更多的参数。这些参数叫做不定长参数，和上述2种参数不同，声明时不会命名。加了星号（\*）的变量名会存放所有未命名的变量参数。如果在函数调用时没有指定参数，它就是一个空元组。我们也可以不向函数传递未命名的变量。

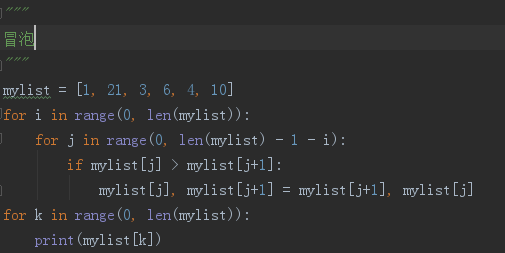
**DAY 3**

一些算法和练习:



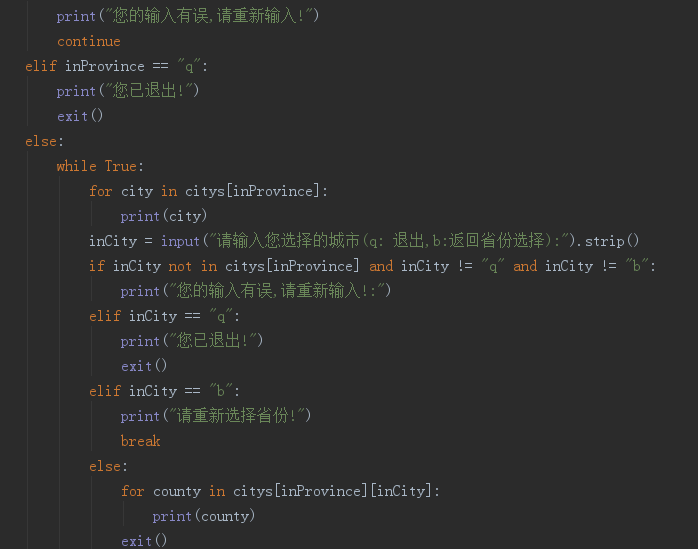




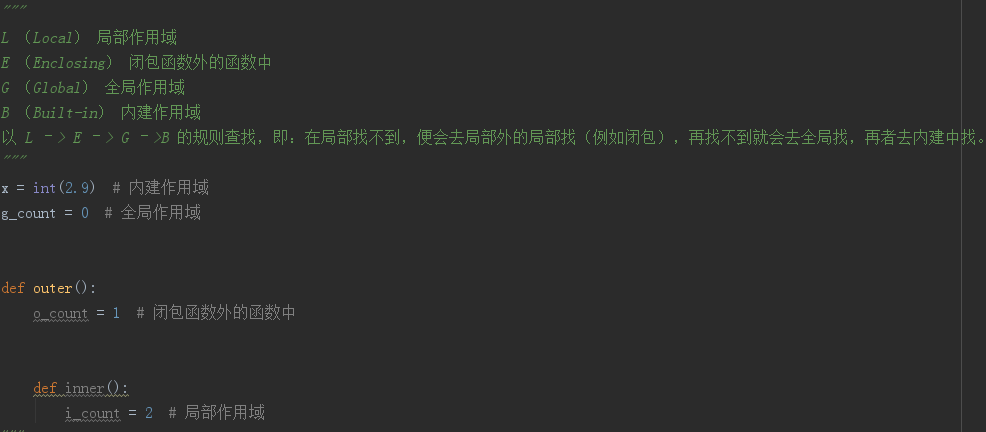


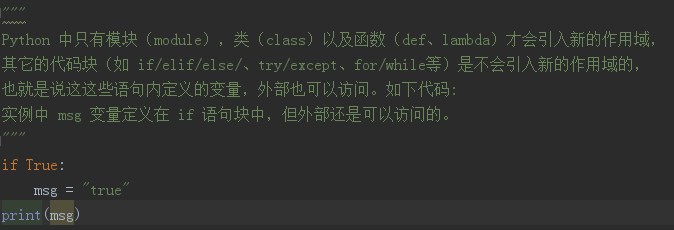
练习:





**DAY 4**





**DAY 5**

**List列表:**

list = [[321, 3213], 312321, 3213, 123, 123123, 1235324, 234234]

# 往尾部添加对象

list.append(222)

print(list)

# 添加对象,第一个参数为添加的对象在列表中的位置,第二个参数为对象

list.insert(0, 333)

print(list)

# 计算数量,只会搜索最外层的列表

print('count:', list.count(3213))

# 出现的位置,默认查找第一个,只搜索最外层列表

print(list.index(3213))

# 将所有的3213替换成111

for i in range(list.count(3213)):

list[list.index(3213)] = 111

print(list)

# 拼接合并

list1 = ['打算', '大声道', 111]

list.extend(list1)

print(list)

print(list1)

# 反转

list.reverse()

print(list)

# 排序(3.x里数字和字符串不可以一起排,2.x里可以一起排)

list2 = [321, 3213, 312321, 3213, 123, 123123, 1235324, 234234]

list2.sort()

print(list2)

# 移除列表对象,默认最后一个,参数为移除的对象坐标

list3 = [[321, 3213], 312321, 3213, 123, 123123, 1235324, 234234]

list3.pop(0)

print('list3:', list3)

"""

复制列表

如果列表内部嵌套列表,只copy最外层列表,内部嵌套的多层列表都共享.

也就是说列表内部如果嵌套列表,实际上最外层列表里存的是内部列表的内存指针,该指针指向内存中的内部列表,

这样做的好处就是,如果内部列表数据量很大的话,可以节省内存空间

"""

# 借助copy库进行深copy,可以完全复制

import copy

list4 = [321, 3213, 312321, [3213, [123, 6767]], 123123, 1235324, 234234]

list5 = list4.copy()

list6 = copy.deepcopy(list4)

list4[0] = 9999999999

list4[3][0] = 8888888888

list4[3][1][0] = 7777777

print('list4:', list4)

print('list5:', list5)

print('list6:', list6)

print('list4[3].id:', id(list4[3])) # 39429192

print('list5[3].id:', id(list5[3])) # 39429192

print('list6[3].id:', id(list6[3])) # 39450696

**List列表扩展与思考:**

*"""*

*字符串下,name2变量等于name变量,改变name变量值,实际上就是把name变量重新指向另一个内存地址,而name2变量不变..*

*而在列表下,list5变量等于list4变量,改变list4变量里的任一值,但是list4变量的指针指向不变,list5的指针指向也不变,*

*但是数据发生变化,这是因为这两个变量都指向同一个内存地址*

*"""*

name = "aaa"

name2 = name

name = "bbbb"

print(id(name))

print(id(name2))

list4 = [321, 3213, 312321, [3213, [123, 6767]], 123123, 1235324, 234234]

list5 = list4

list4[3][1][0] = 88888

print(list4)

print(list5)

print(id(list4))

print(id(list5))

list6 = []

list7 = []

for i in range(0, 10):

list6.append(9999)

for j in range(0, 10):

list7.append(list6)

list7[0][0] = 0

print('list6:', list6)

"""

此处list7内部的所有列表的0位置的数据都变成了0,这是因为虽然list7[0][0] = 0这样操作,

实际上list6与list7共享该列表的内存地址,修改的是同一份列表

"""

print('list7:', list7)

print(id(list6))

print(id(list7[0]))

graph = [[] for i in range(10)]

for a in range(0, 10):

for b in range(0, 10):

graph[a].append(9999)

graph[0][0] = 0

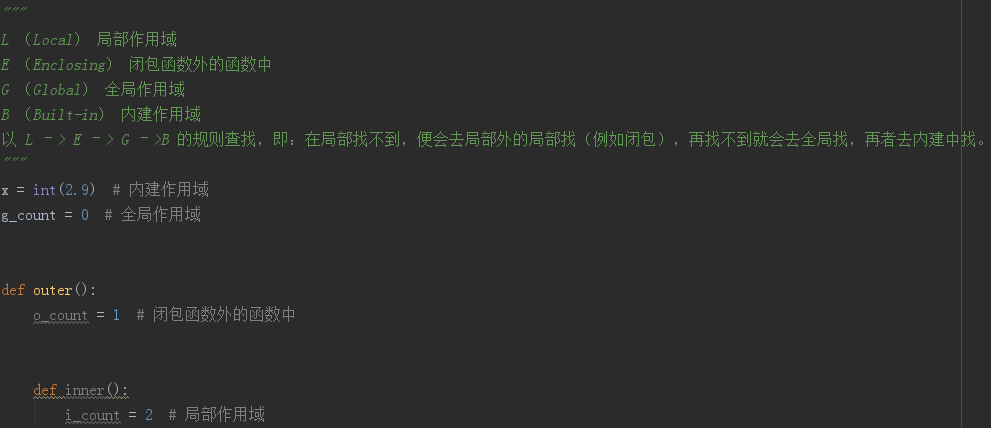
print(graph)

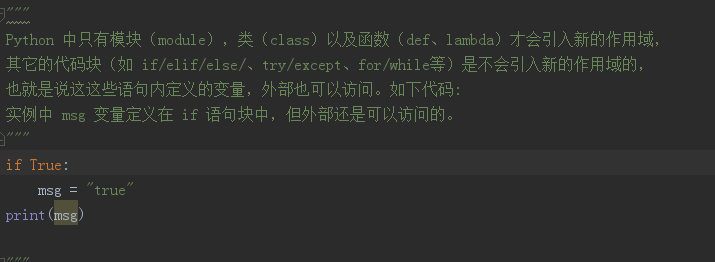
print(id(graph[1]))

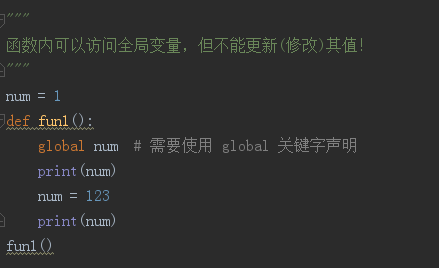
print(id(graph[2]))

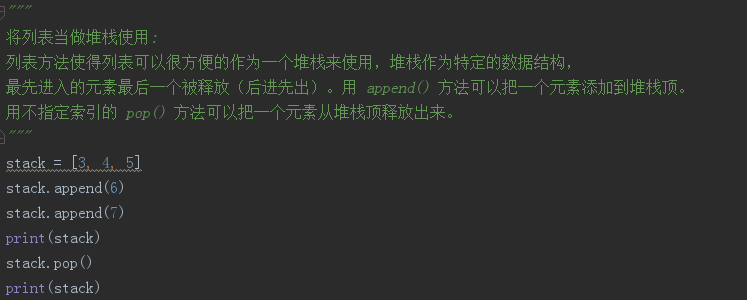
**DAY 6**

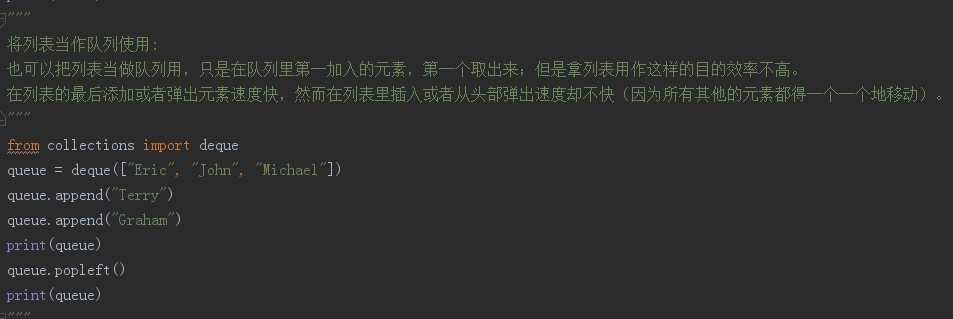
**数据结构及List其他应用:**

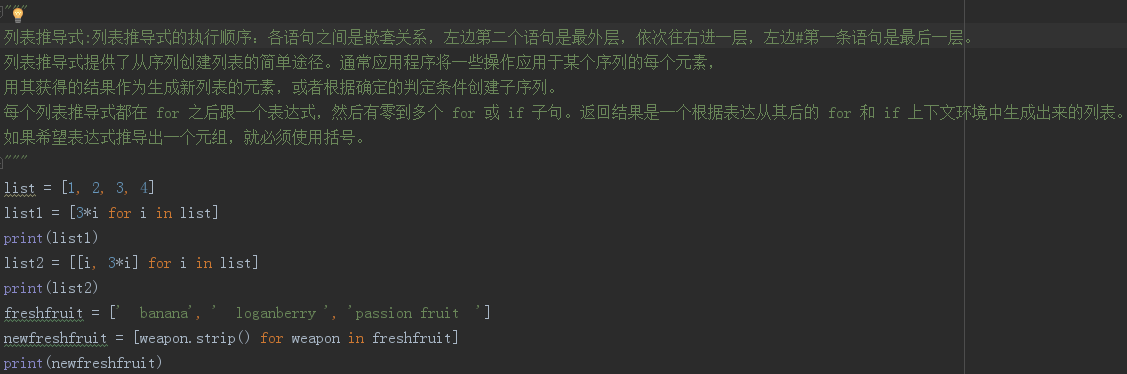


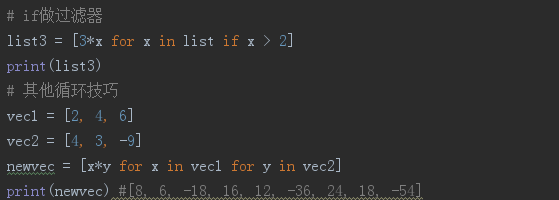


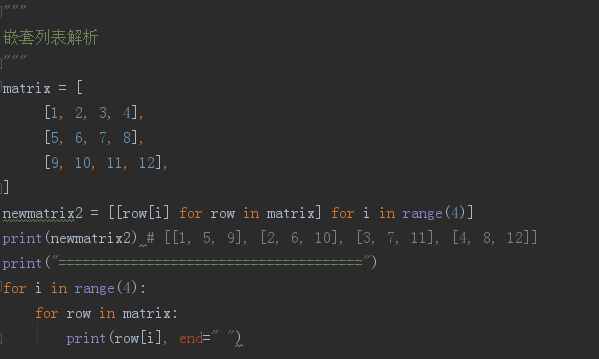






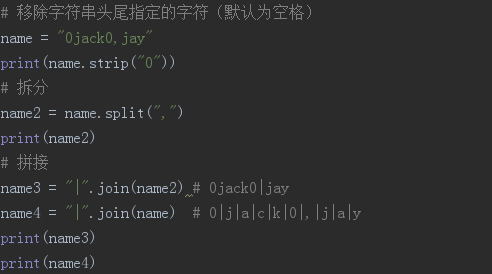






**DAY 7**

**String一些用法:**





**51CTO视频教学购物车自写练习:**

*"""*

*购物车demo*

*"""*

# 商品字典

goods = {1: {

"name": "矿泉水",

"price": 1.00,

"count": 5

}, 2: {

"name": "泡面",

"price": 4.00,

"count": 10

}, 3: {

"name": "火腿肠",

"price": 2.50,

"count": 5

}, 4: {

"name": "卫生纸",

"price": 9.78,

"count": 2

}

}

goods\_buy = {}

print("商品列表:")

for id in goods:

print(str(id) + ':' + goods[id]["name"] + ' 价格:' + str(goods[id]["price"]) + ' 库存:' + str(goods[id]["count"]))

print()

while True:

# 选择的商品编号

choose\_for\_buyer = input("请选择商品(输入编号即可,按q退出):").strip()

if choose\_for\_buyer in ['q', 'Q']:

print("您已退出!")

print("您的选择如下:".center(40,"\*"))

if len(goods\_buy) == 0:

print("您没有选择任何商品!")

else:

# 总价

sum\_price = 0.00

for id\_for\_buyer in goods\_buy:

print(goods\_buy[id\_for\_buyer]["name"] + ":" + str(goods\_buy[id\_for\_buyer]["price"]) + " " +

str(goods\_buy[id\_for\_buyer]["count"]))

sum\_price += goods\_buy[id\_for\_buyer]["price"] \* goods\_buy[id\_for\_buyer]["count"]

print()

print("总价为:\033[31;1m[%f]\033[0m" % sum\_price)

break

elif not choose\_for\_buyer.isdigit():

print("您的输入有误!")

elif int(choose\_for\_buyer) not in goods.keys():

print("您的选择有误,请重新选择!")

continue

else:

while True:

# 选择的商品数量

count\_for\_buyer = input("请输入购买数量(u:返回上一级):")

if count\_for\_buyer in ["u", "U"]:

break

elif not count\_for\_buyer.isdigit():

print("您的输入有误!")

continue

elif int(count\_for\_buyer) > goods[(int(choose\_for\_buyer))]["count"]:

print("超过库存!")

continue

else:

try:

# 叠加选择的商品数量(初始数量+选择数量)

goods\_buy\_count = goods\_buy[choose\_for\_buyer]["count"] + int(count\_for\_buyer)

except KeyError:

# 报异常说明不存在,所以初始数量设为0

goods\_buy\_count = 0 + int(count\_for\_buyer)

# 将用户选择加入购物车字典

goods\_buy.setdefault(int(choose\_for\_buyer), {"name": goods[(int(choose\_for\_buyer))]["name"],

"price": goods[(int(choose\_for\_buyer))]["price"],

"count": goods\_buy\_count})

# 对选择的商品库存进行减少操作

goods[(int(choose\_for\_buyer))]["count"] -= int(count\_for\_buyer)

print("商品列表:")

for id2 in goods:

print(str(id2) + ':' + goods[id2]["name"] + ' 价格:' + str(goods[id2]["price"]) + ' 库存:' + str(goods[id2]["count"]))

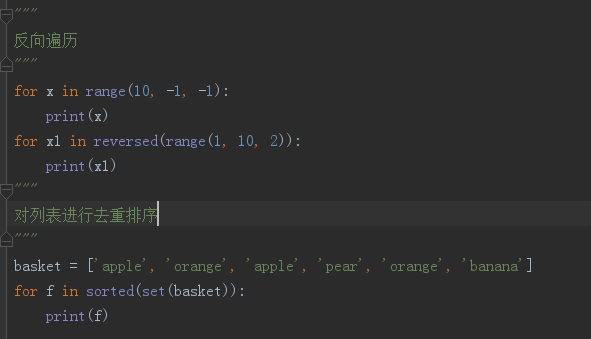
print()

break

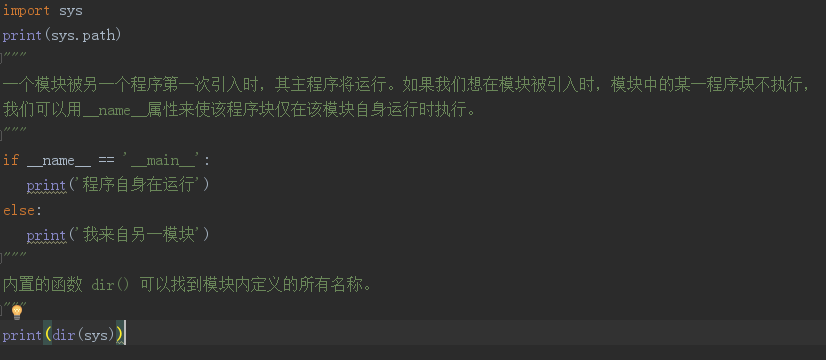
**DAY 8**

**List的一些其他用法:**





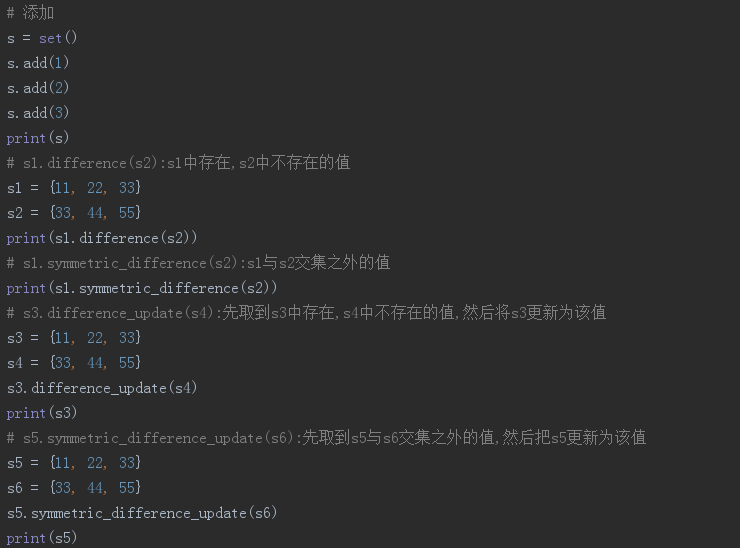
**模块:**

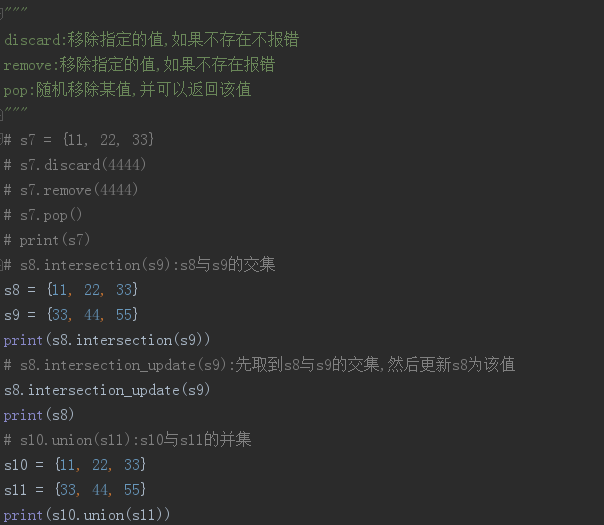


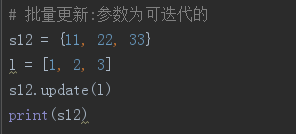
模块的相关详细资料可以查看网址:<http://www.runoob.com/python3/python3-module.html>

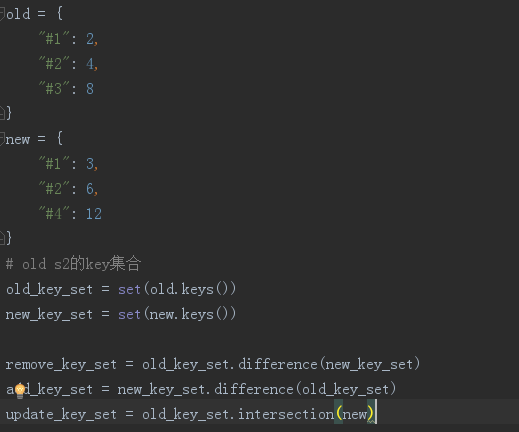
**DAY 9**

**Set:**

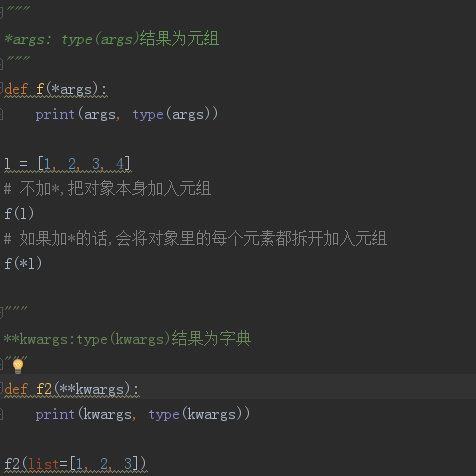


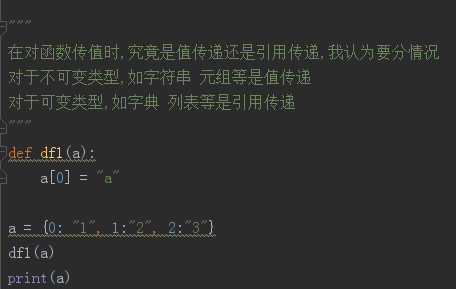






**函数:**



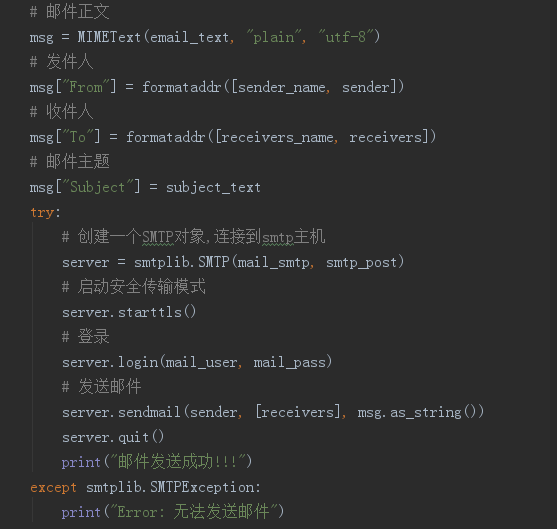


**一些练习:**

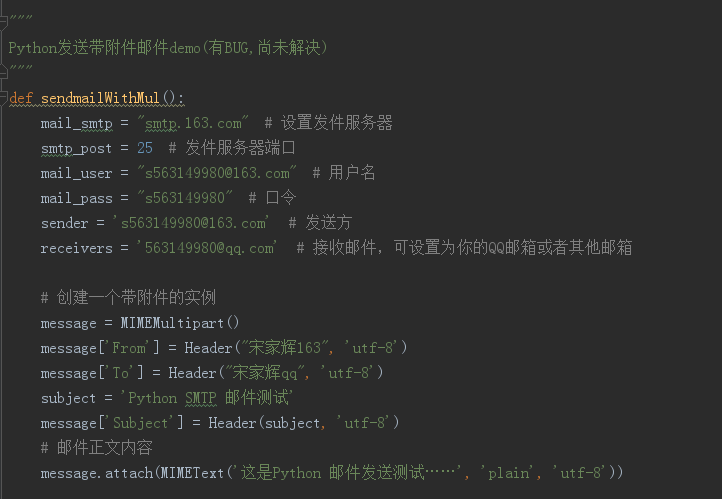
**1.发送邮件:**

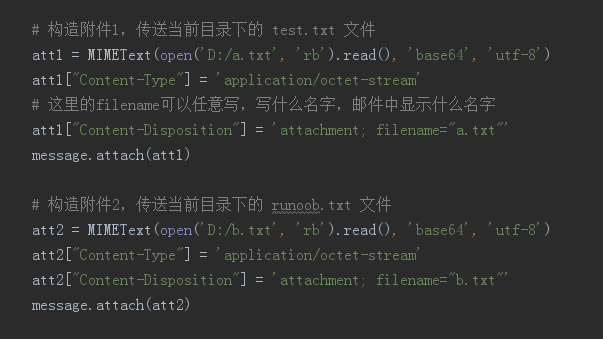
**1.1**

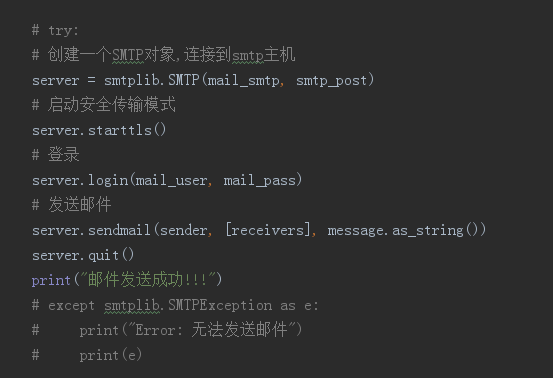




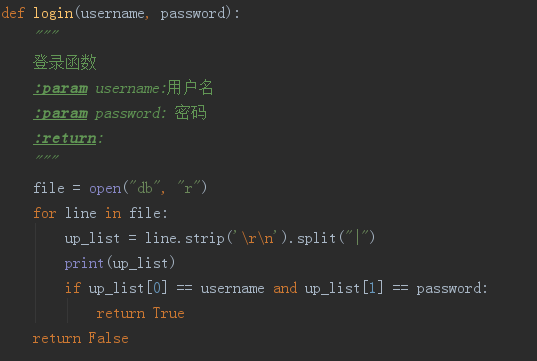
**1.2**

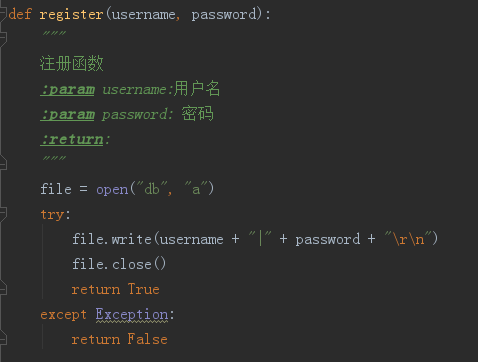


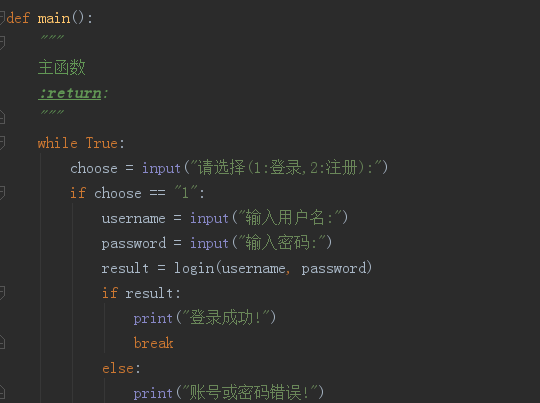


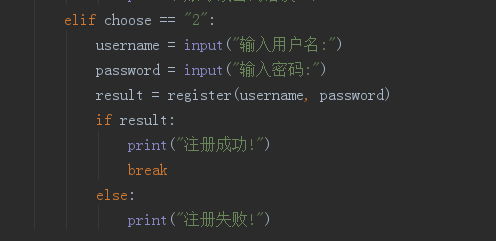


**2.登录注册:**





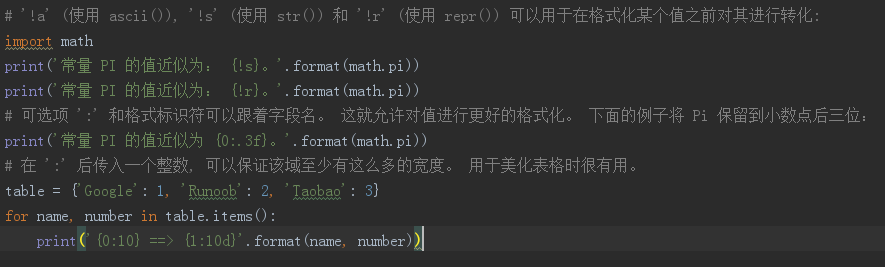




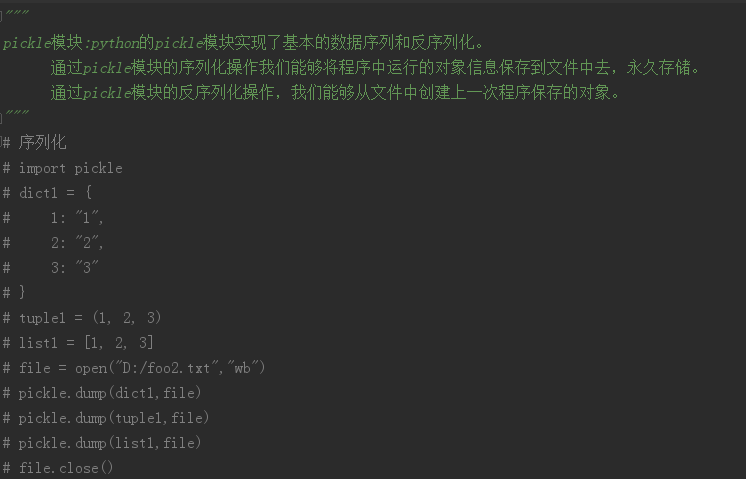
**DAY 10**

**输出格式化:**



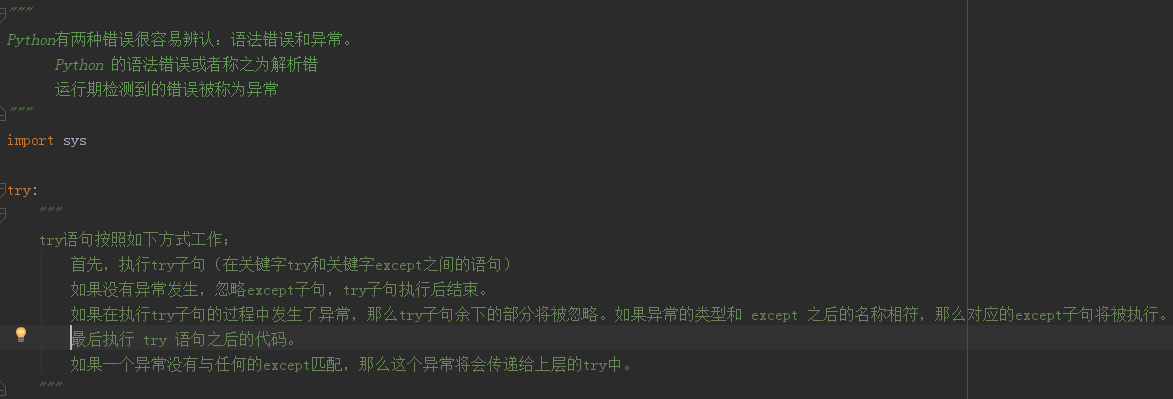


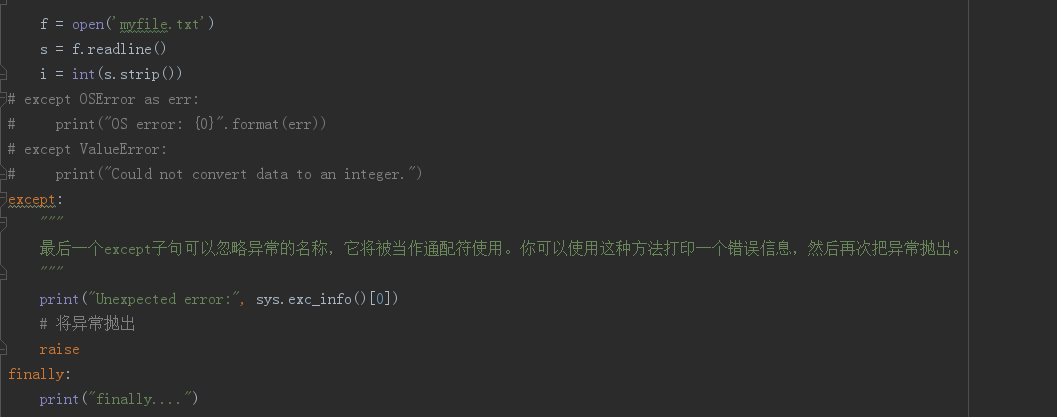
**序列化与反序列化:**



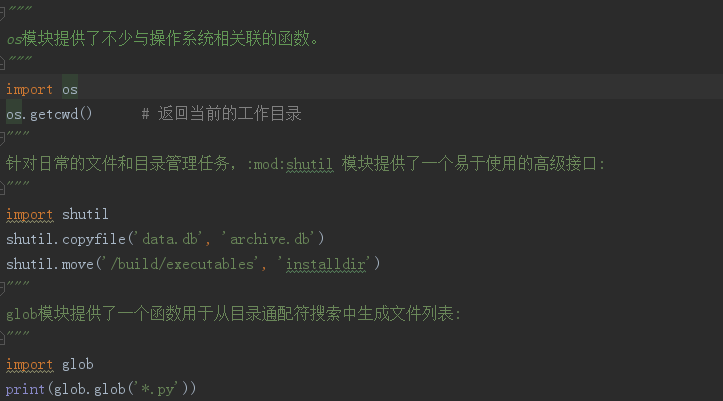


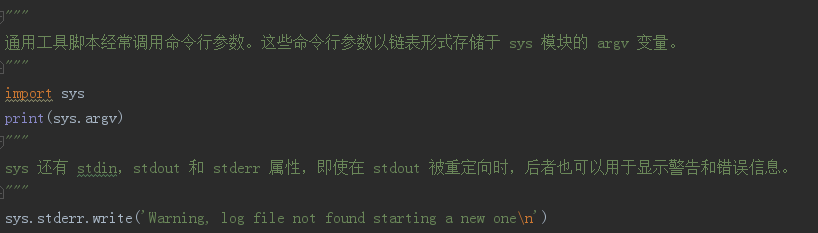
**错误与异常:**

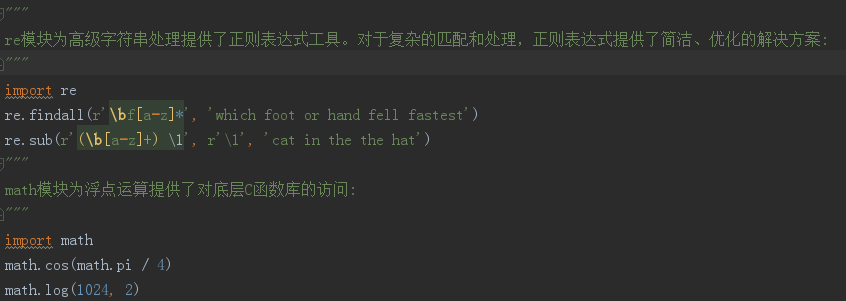


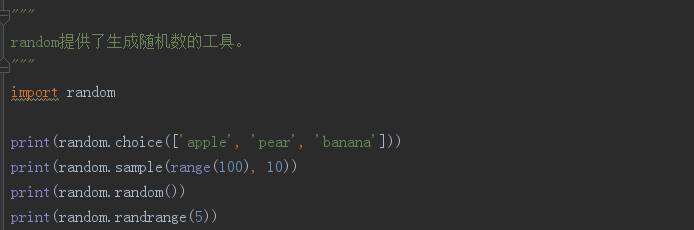


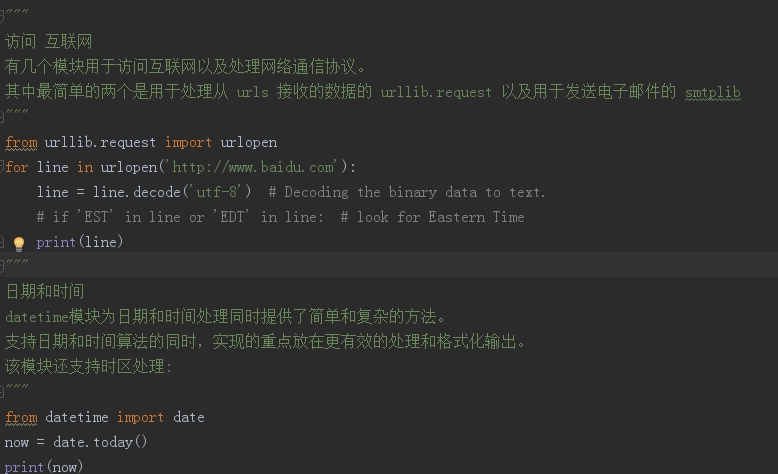
**python提供的一些其他模块:**

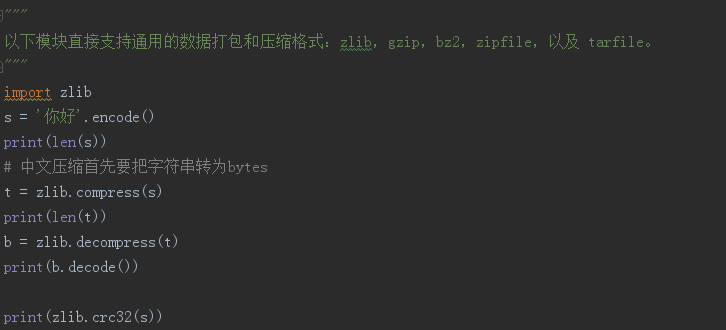


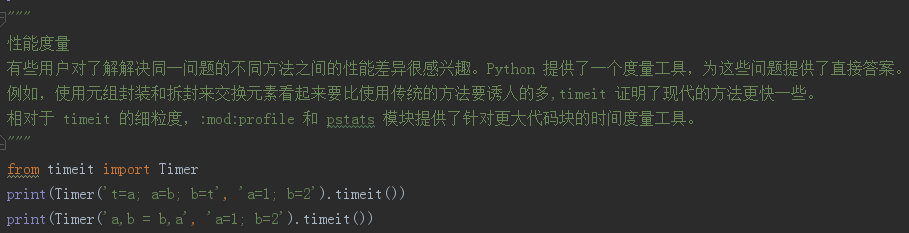


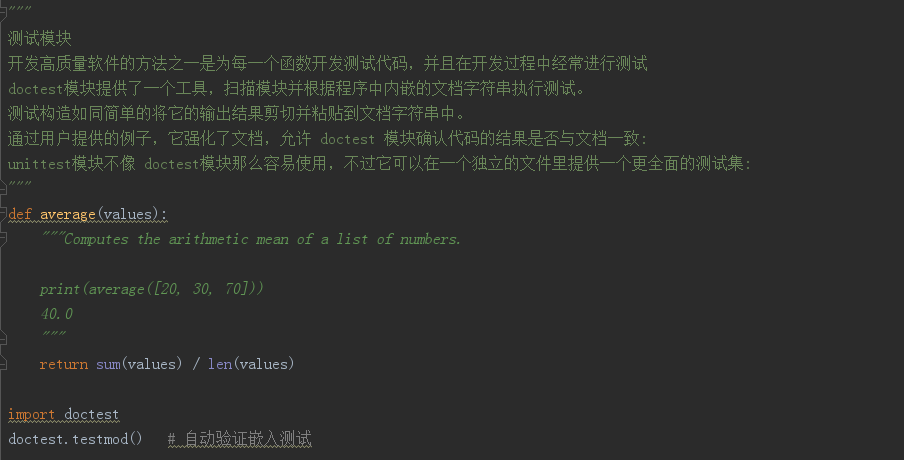






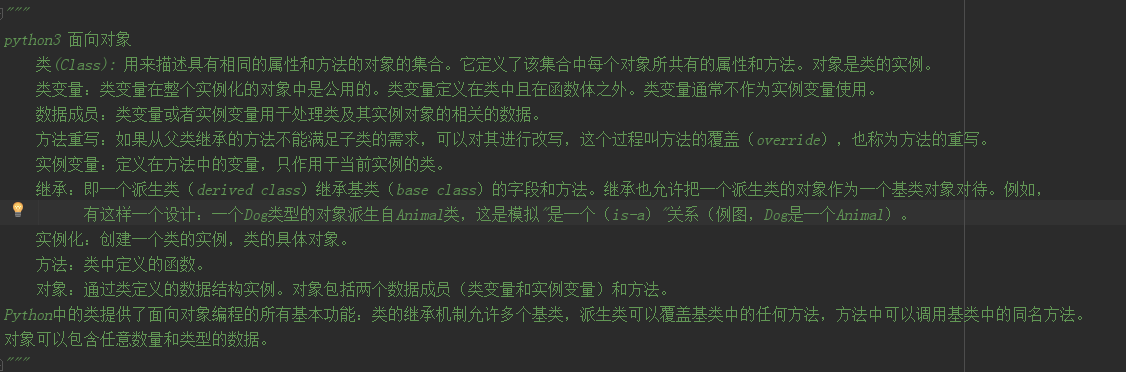


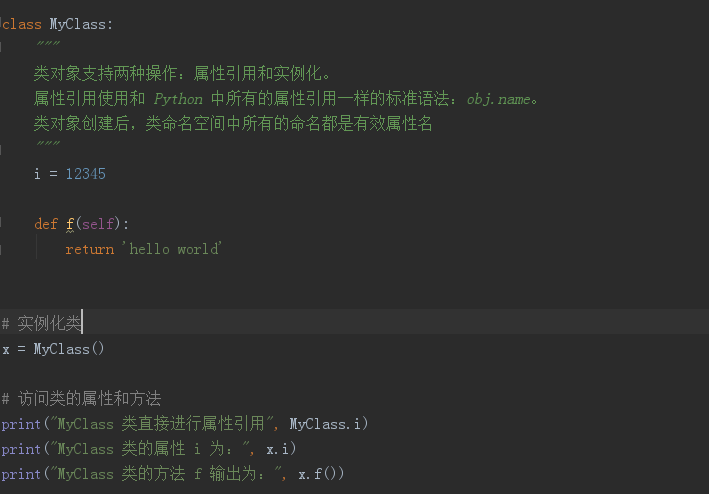


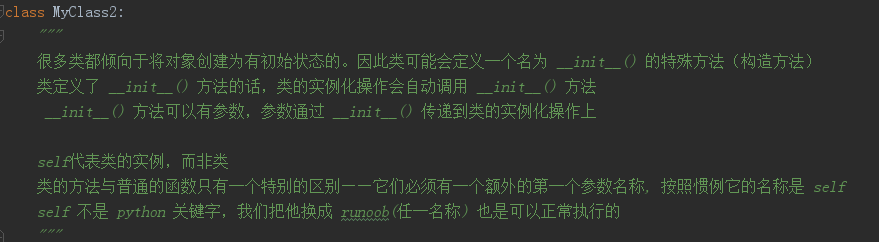


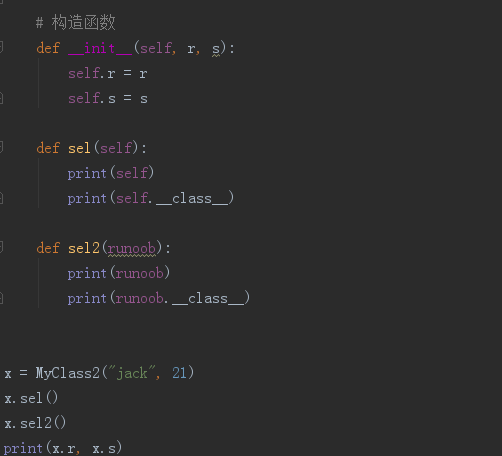
**DAY 11**

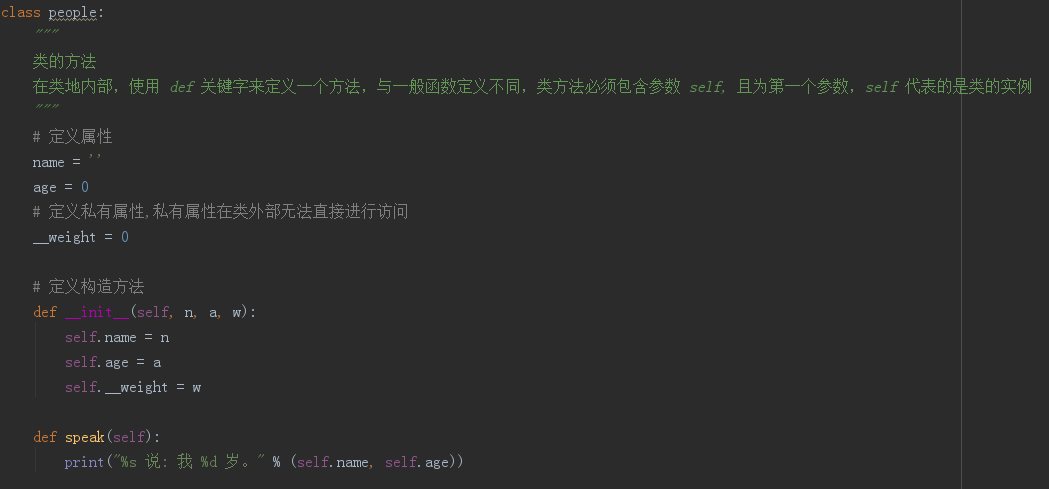
**面向对象:**

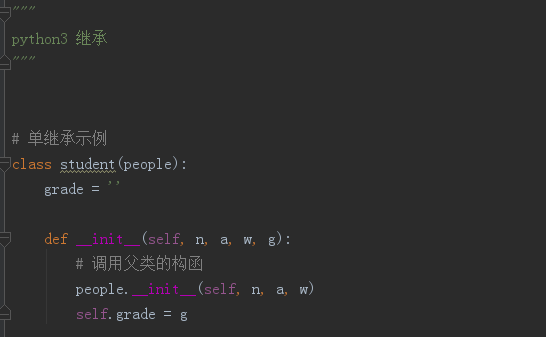


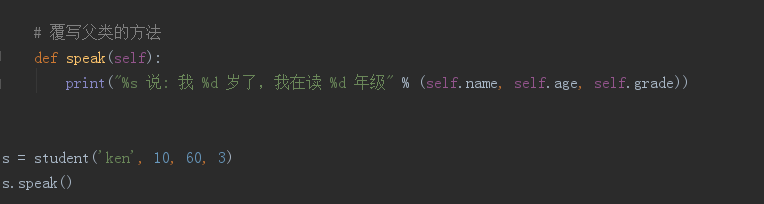


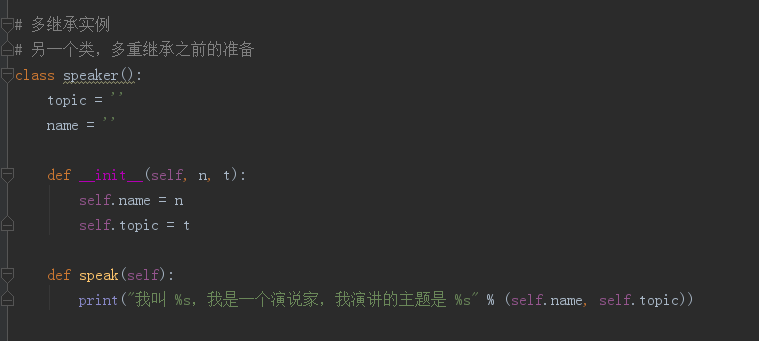


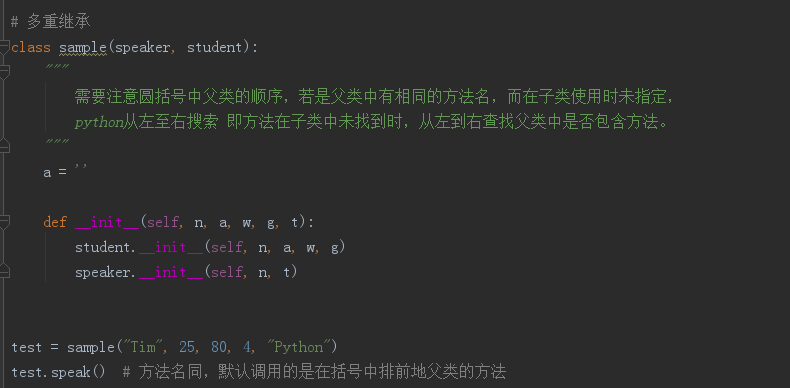












**DAY 12**

**遇到问题:**

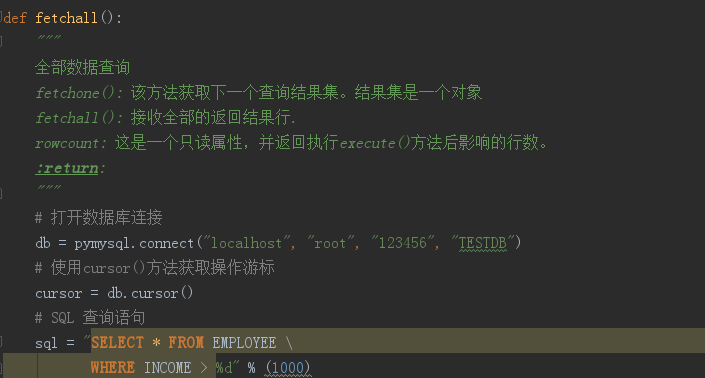
**cmd中pip安装PyMySQL第三方模块,在cmd中可以正常import,但是在pycharm中不可以import**

**解决办法:在pycharm中:Setting-->Project interpreter-->+中安装PyMySQL**

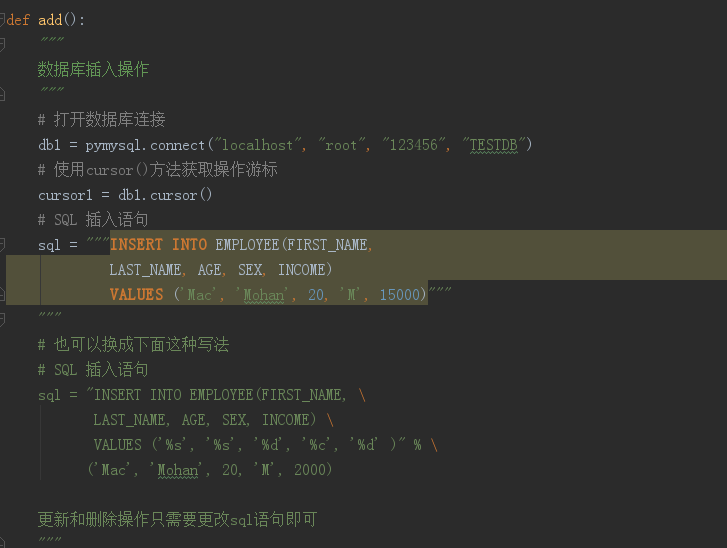
**该问题应该还有其他的解决办法,目前未知,导致该问题的原因也未知.**

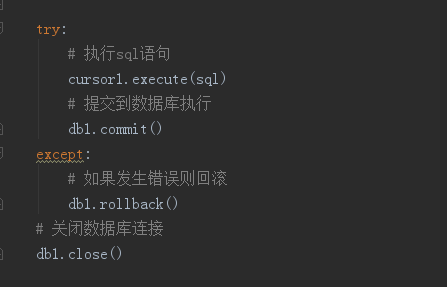
**PyMySQL:**



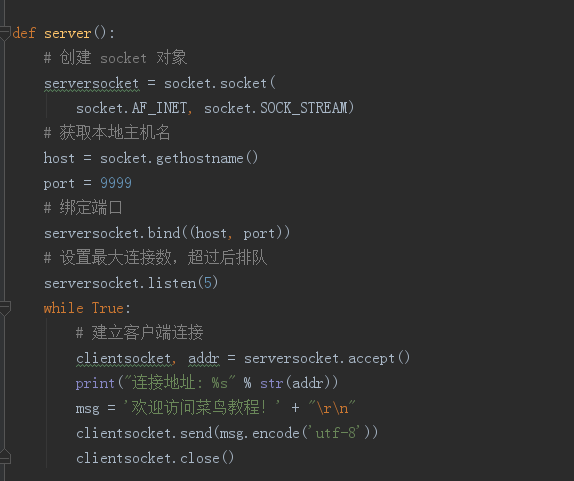


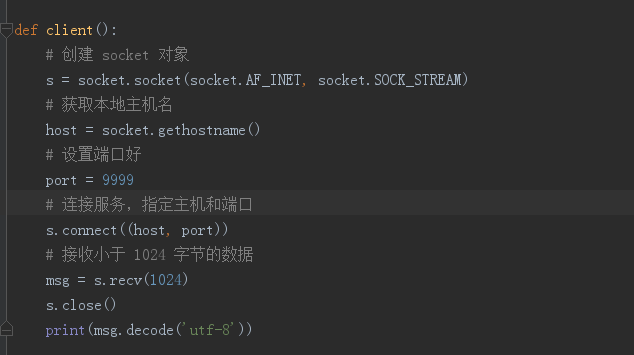






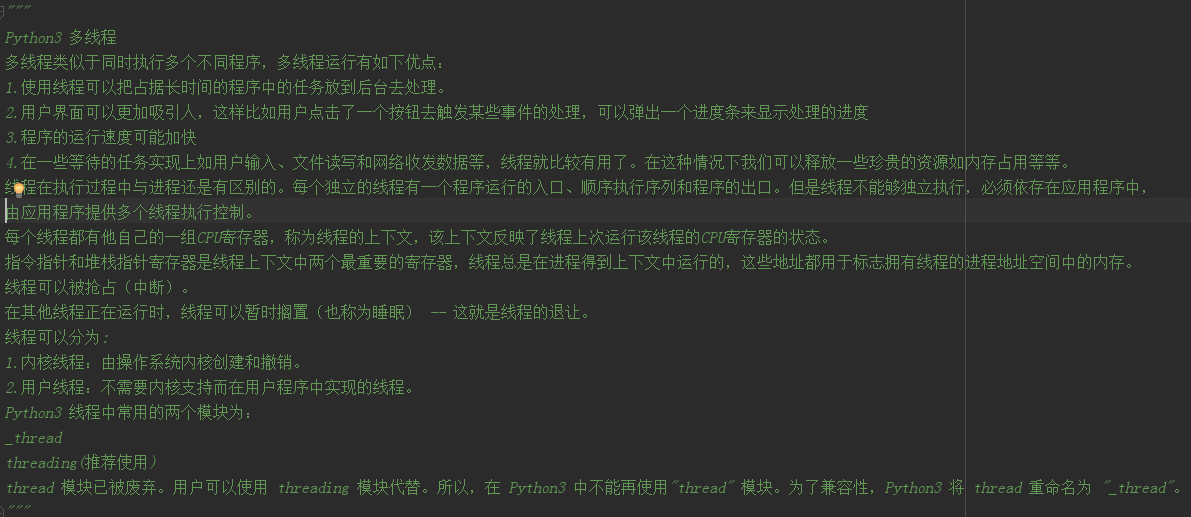
**socket:**

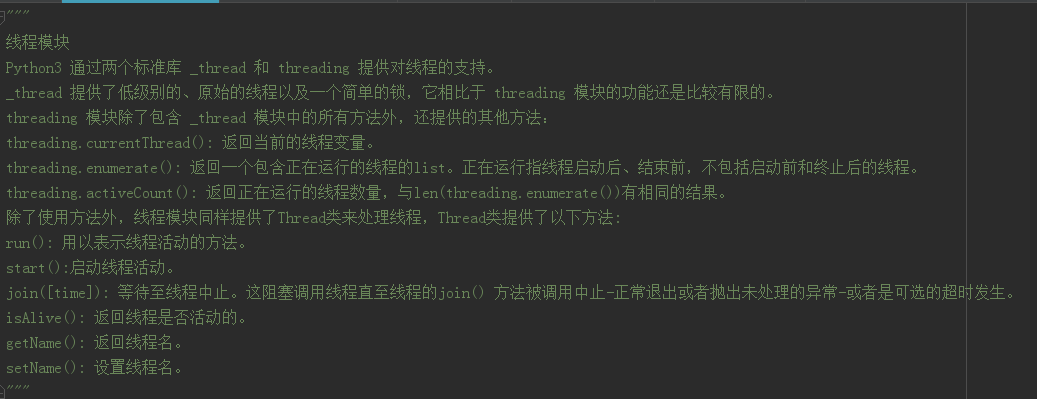




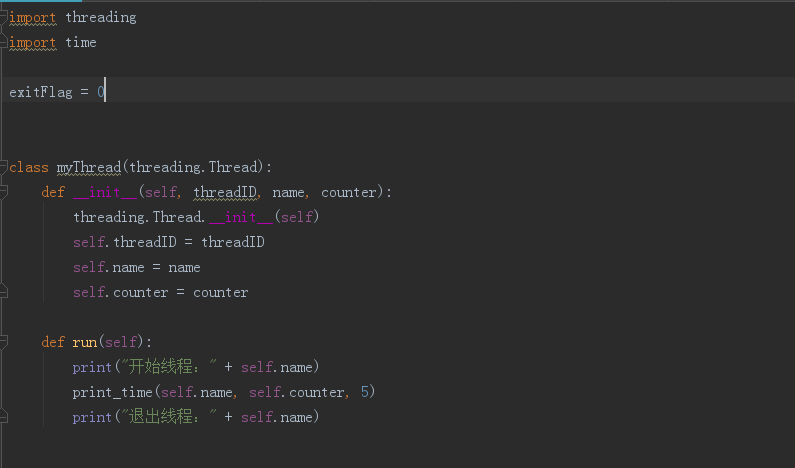
**DAY 13**

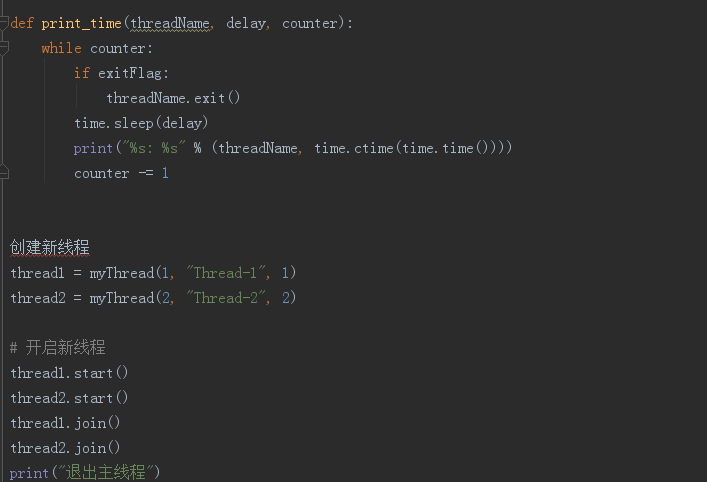
**python线程:**

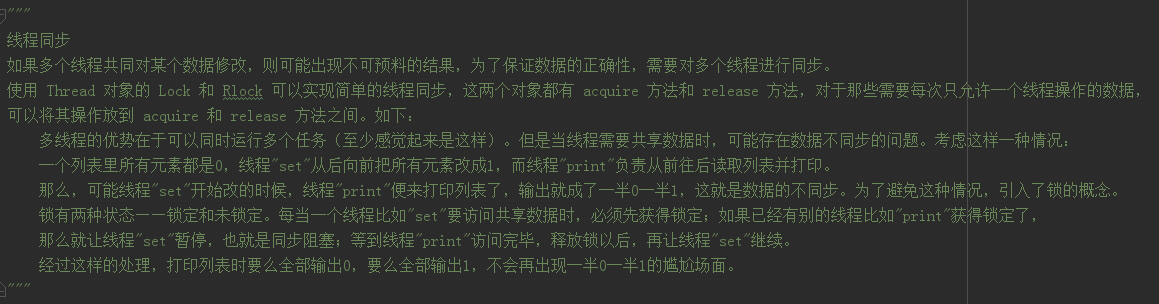


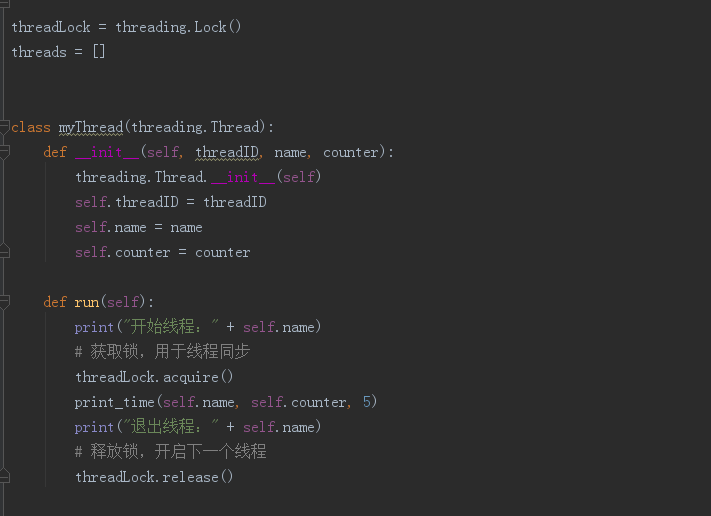


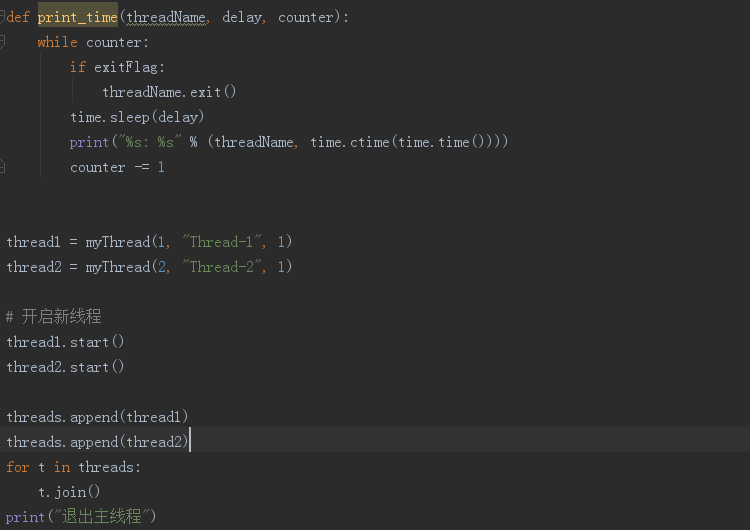


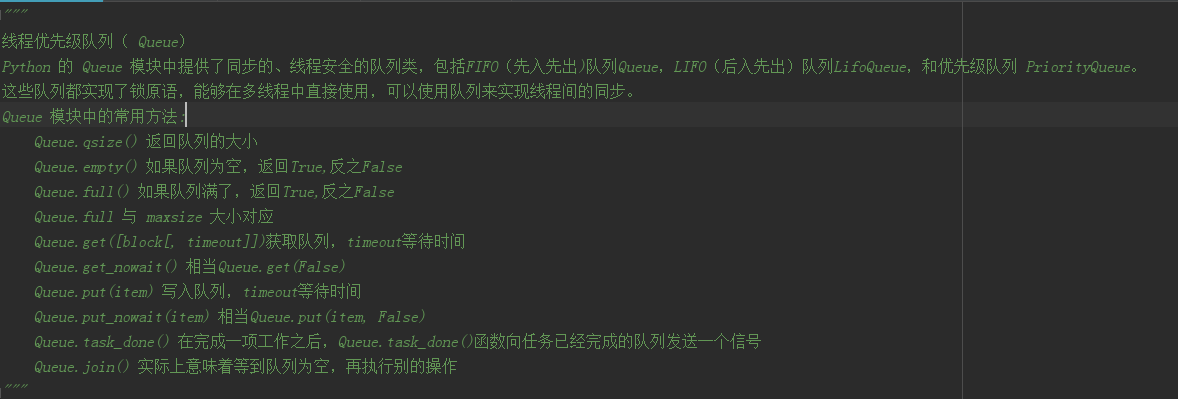


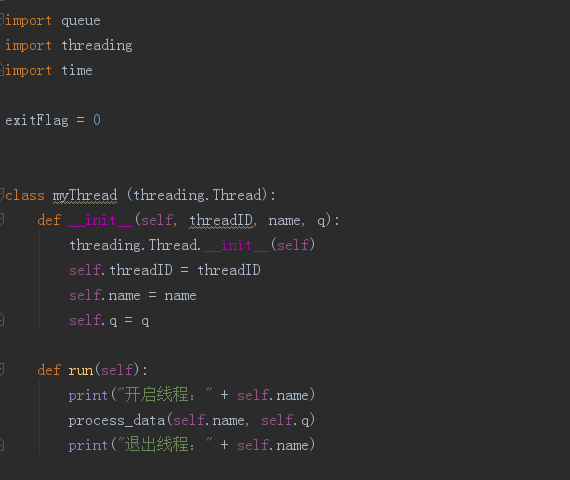


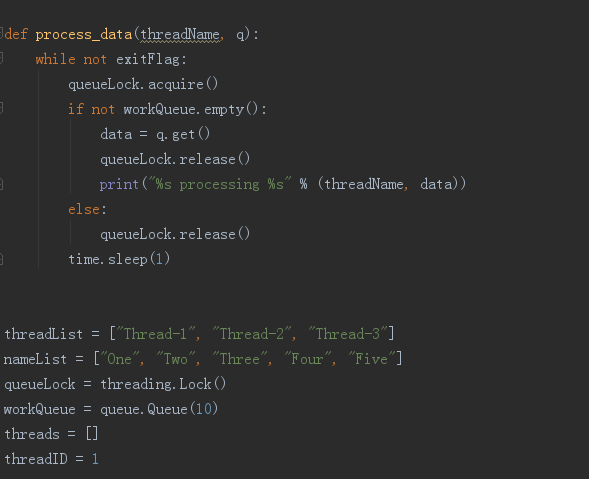


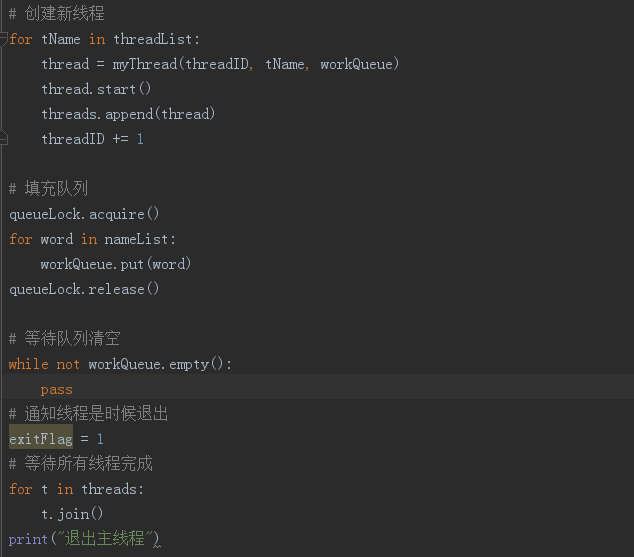






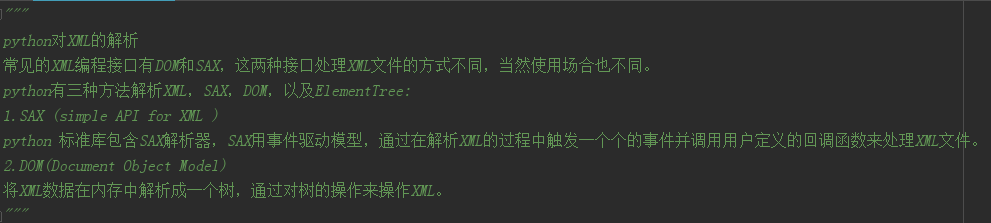


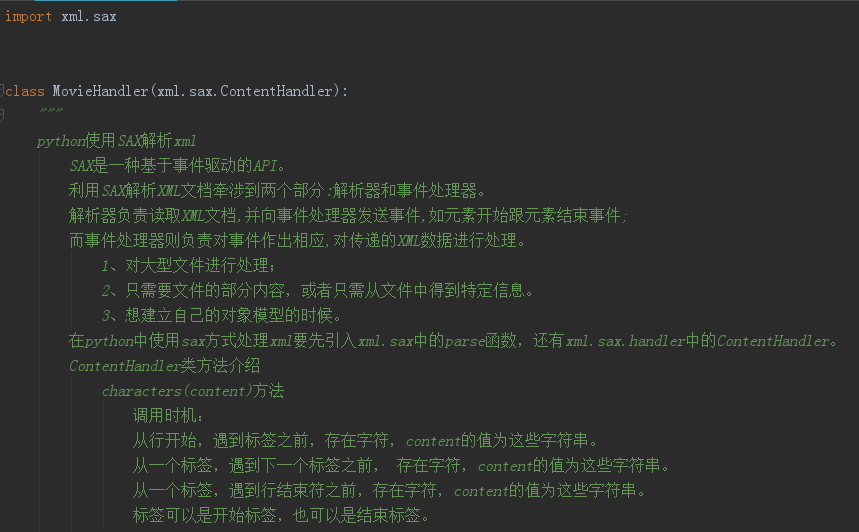


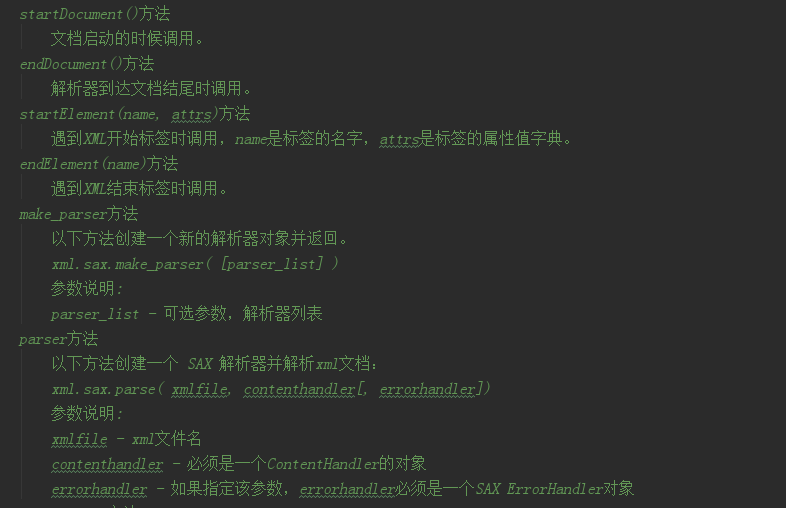


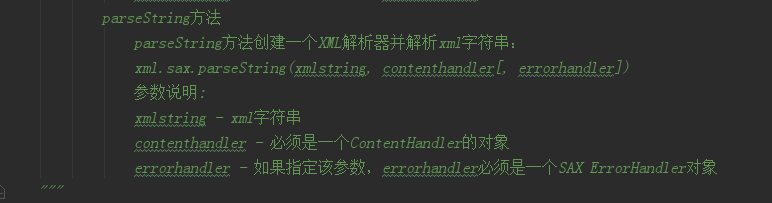
**xml解析:**

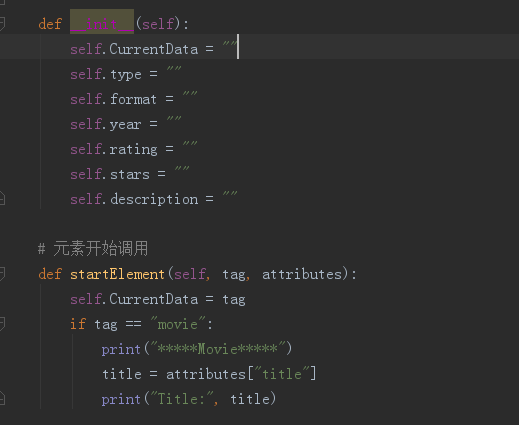
**1.sax:**

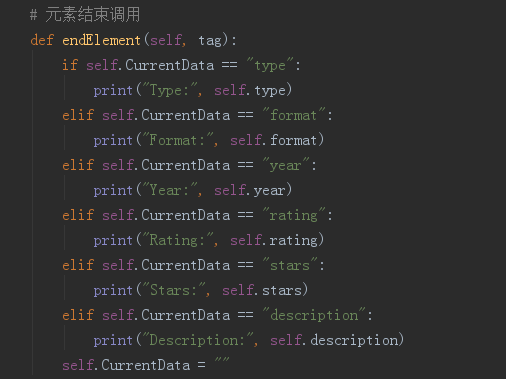








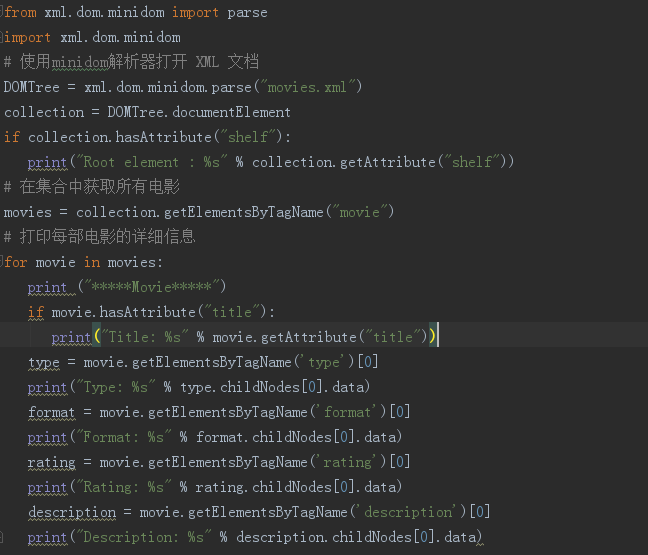






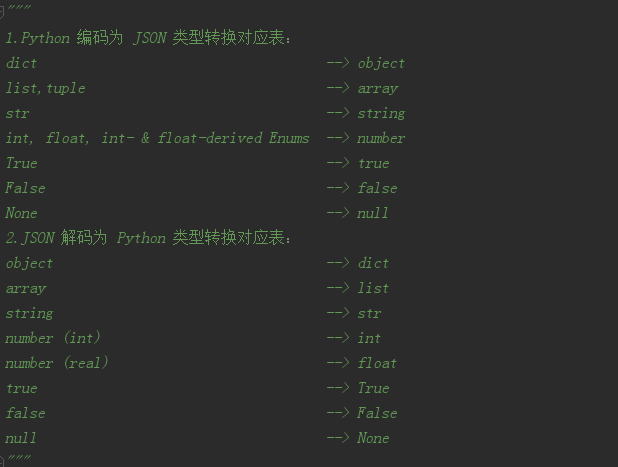


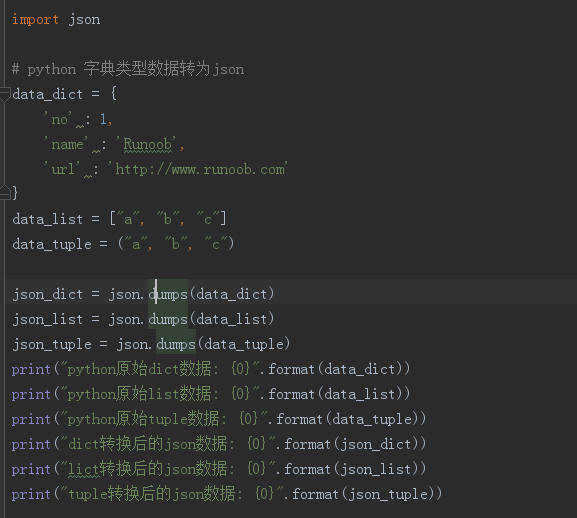
**2.dom:**

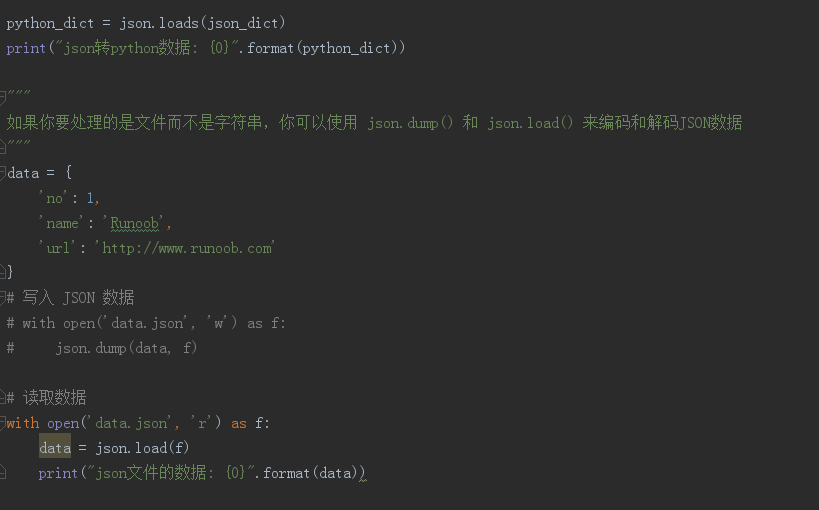


**DAY 14**

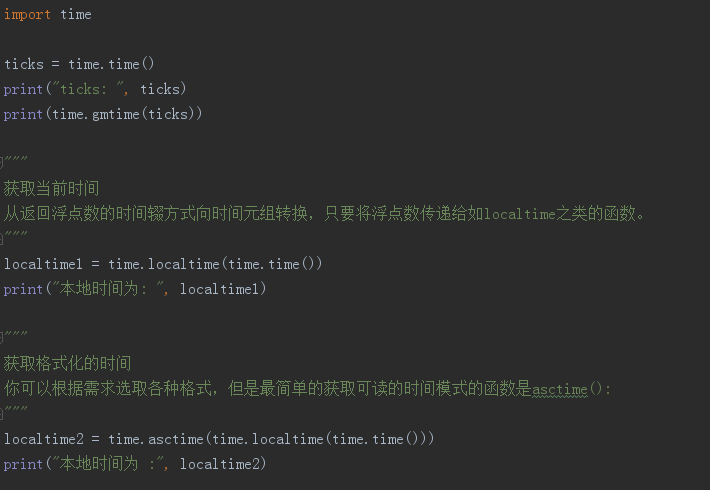
**JSON:**

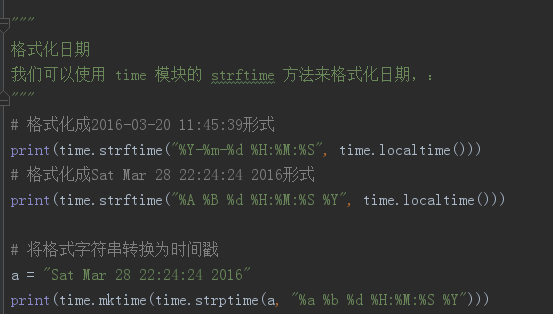


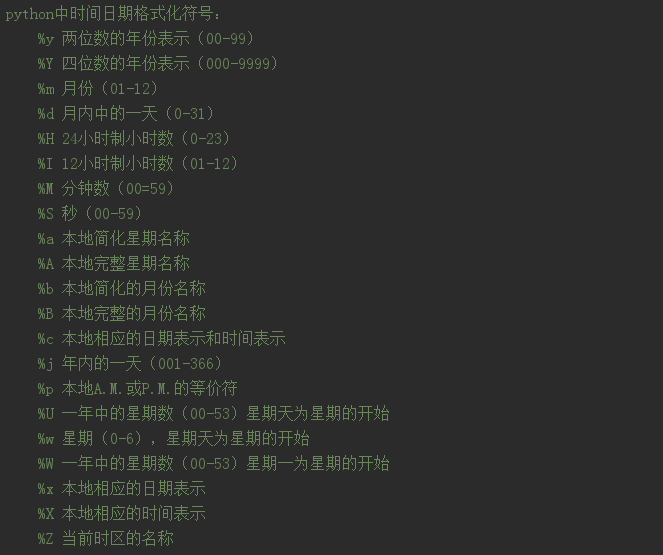


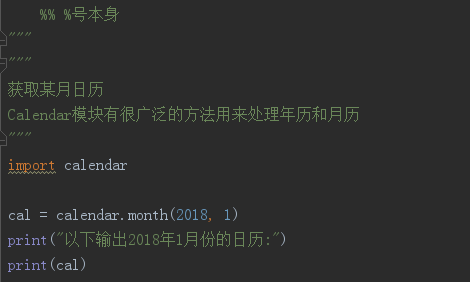


**时间日期:**

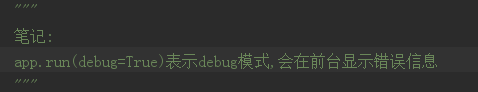


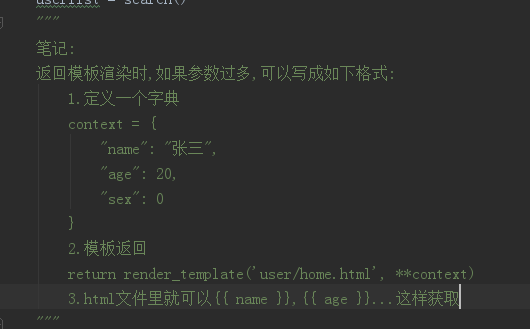






**DAY 15**





**DAY 16**

