**实验2 Python基础语法实践**

**姓名：宁浩 学号：2312190617 班级：软件2301 成绩：**

**1. 实验目的**

本次实验旨在了解Python基础语法知识，掌握Python3的基本数据结构、控制结构、函数等，具备数学计算和字符串/文本处理的能力。相关知识点包括：

1) 数据结构：列表、元组、字符串、字典、集合;

2) 控制结构：条件语句以及条件表达式、while/for循环、break和continue语句、try-except语句

3) 函数：函数的定义、函数的参数形式、函数的调用、lambda函数，以及map、filter等内置函数的用法

请完成下面的实验练习题，报告书写要求同第1次实验作业。注意，请在本报告中将自己的姓名、学号、班级书写正确。最后，在规定时间内完成本次实验，并将报告的word文档(非pdf版本)以附件形式提交至学习通平台。

**2. 实验练习题**

**答题要求**：将正确运行的**源代码书写在题目下方**，并紧跟着贴上一份**源代码的截图**和运行结果的**截图**。答题格式同第1次实验作业。

(1)编写函数，接收一个字符串，分别统计大写字母、小写字母、数字、其他字符的个数，并以元组的形式返回结果。字符串通过input输入，然后调用该函数，并打印出函数的返回值。

源代码：

String=input("输入一个字符串:")

upper\_num=0

lower\_num=0

number\_num=0

other\_num=0

for char in String:

if char.isupper():

upper\_num+=1

elif char.islower():

lower\_num+=1

elif char.isdigit():

number\_num+=1

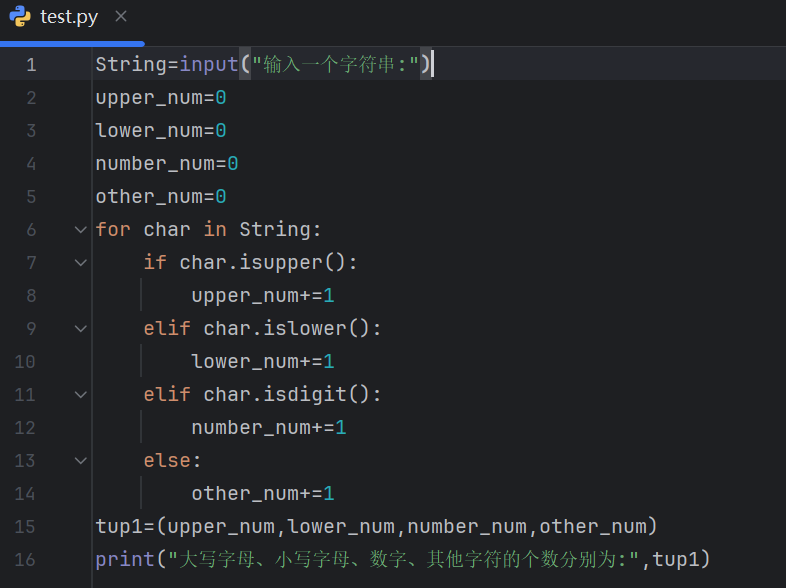
else:

other\_num+=1

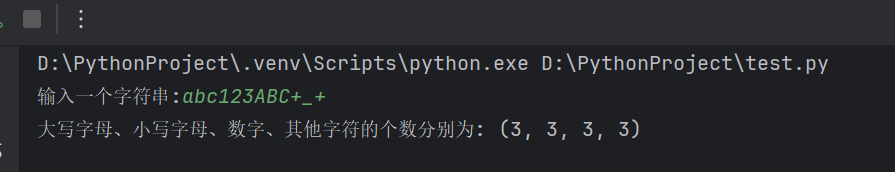
tup1=(upper\_num,lower\_num,number\_num,other\_num)

print("大写字母、小写字母、数字、其他字符的个数分别为:",tup1)

源代码截图：



运行结果截图：



(2)输入三个非负整数、和，输出的进制数。  
源代码：

A=int(input("输入一个非负整数A(A<=2\*\*30-1):"))

B=int(input("输入一个非负整数B(B<=2\*\*30-1):"))

D=int(input("输入一个非负整数D(1<D<=10):"))

C=A+B

result=""

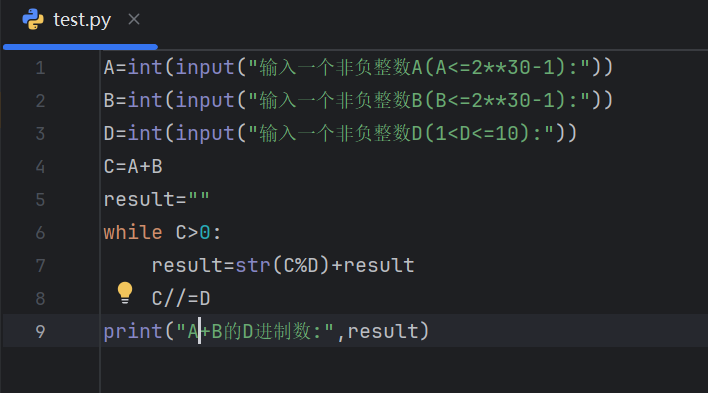
while C>0:

result=str(C%D)+result

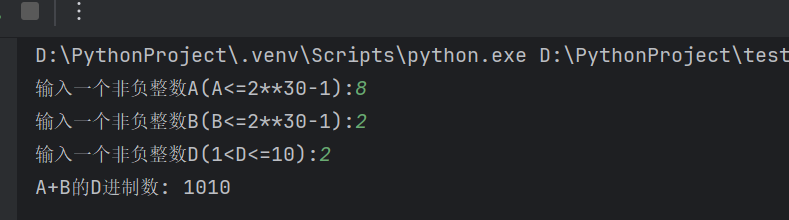
C//=D

print("A+B的D进制数:",result)

源代码截图：



运行结果截图：



(3)正整数的质因子分解。要求编写自定义函数实现该功能。例如，给定10，打印出结果：[[2,5],[1,1]]，表示10=21\*51；给定200，打印出结果：[[2,5],[3,2]]，表示200=23\*52.

源代码：

import math

N=int(input("请输入一个正整数:"))

def isprime(n): #除了1和它本身外，不能被其他自然数整除(1不是质数)

if n==1:

return False

if n==2:

return True

for i in range(2,int(math.sqrt(n))+1):

if n%i==0:

return False

return True

def divide(a):

array=[0]\*N

j=2

A=a

while j<=A:

if a==1:

return array

if a%j==0 and isprime(j):

a/=j

array[j]+=1

j-=1

j+=1

array=divide(N)

list1=[]

list2=[]

for i in range(0,N):

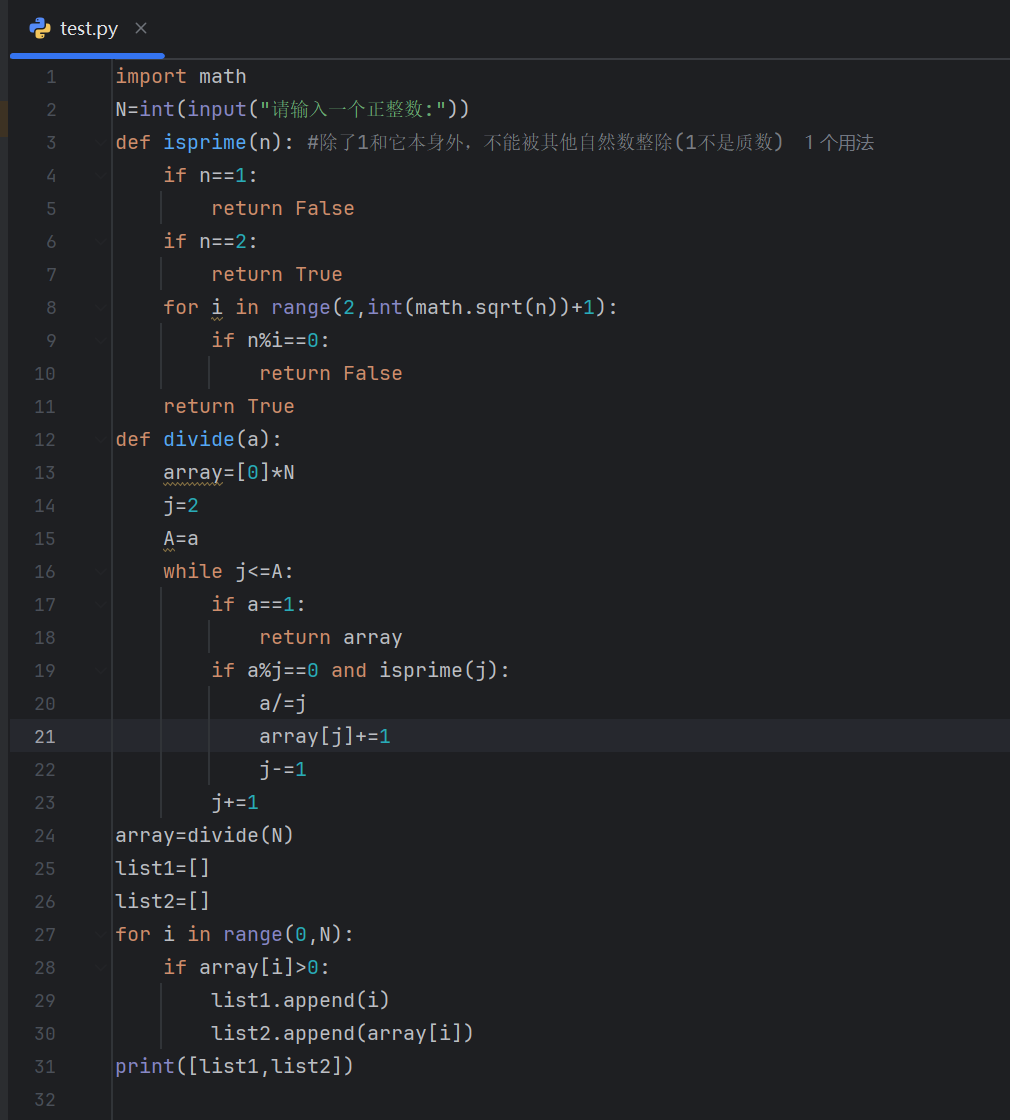
if array[i]>0:

list1.append(i)

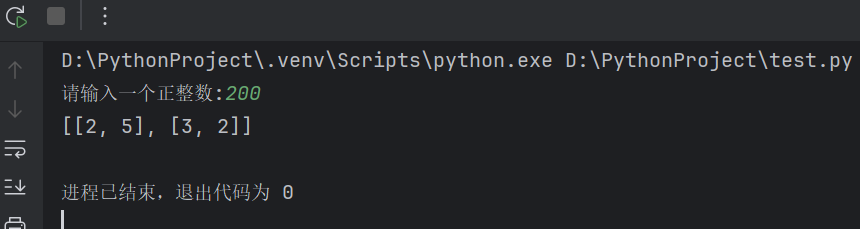
list2.append(array[i])

print([list1,list2])

源代码截图：



运行结果截图：



(4)现有如下的字典数据列表: dict=[{'id':'4','name':'b'}, {'id':'6','name':'c'}, {'id':'3','name':'a'}, {'id':'1','name':'g'}, {'id':'8','name':'f'}]. 请根据列表中元素的关键字’id’的大小进行排序，要求从小到大进行排序，输出一个与dict相似的新列表。

源代码：

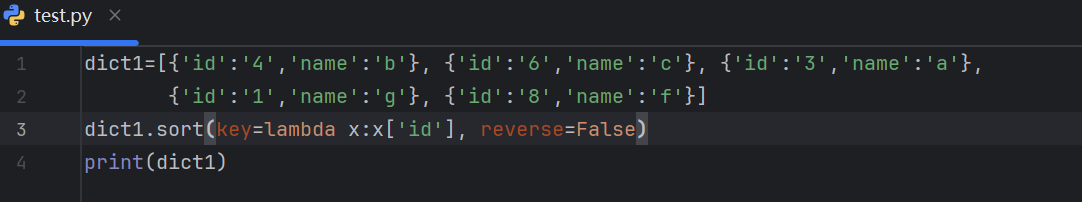
dict1=[{'id':'4','name':'b'}, {'id':'6','name':'c'}, {'id':'3','name':'a'},

{'id':'1','name':'g'}, {'id':'8','name':'f'}]

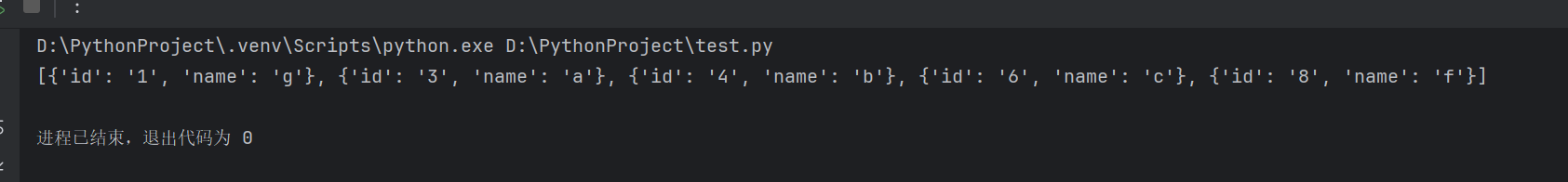
dict1.sort(key=lambda x:x['id'], reverse=False)

print(dict1)

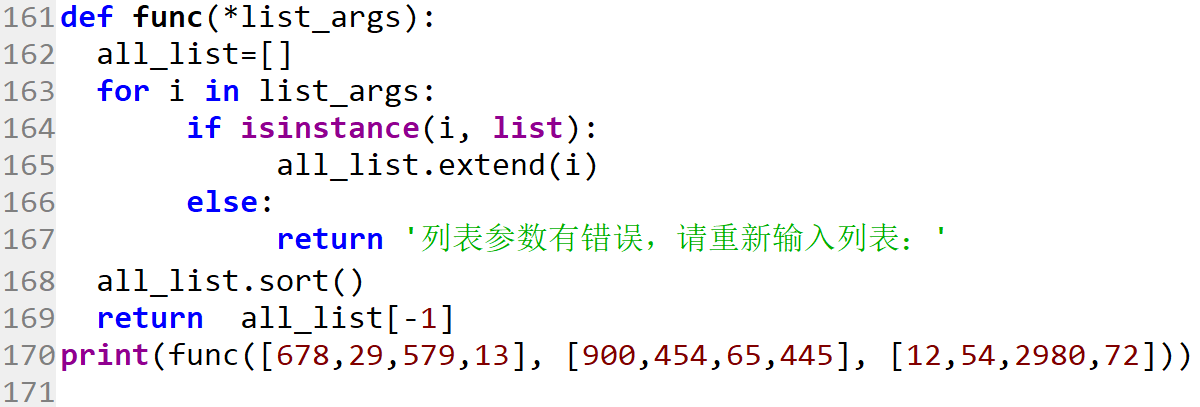
源代码截图：



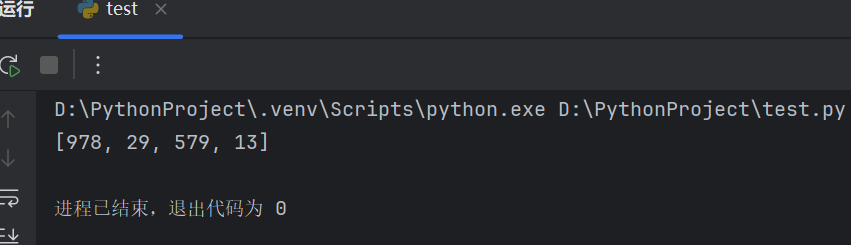
运行结果截图：



(5)运行以下代码，截图粘贴运行结果，并说明函数func的功能：



运行结果截图：



函数功能说明：

函数 func 的功能是：合并所有传入的列表参数中的元素，返回合并后排序的最大值。若参数中存在非列表类型，则立即返回错误提示。

(6)已知学生信息列表 students = [ ('陈小明', '计算机学院', 20, 85.5), ('李思雨', '文学院', 19, 92.3), ('王浩然', '计算机学院', 21, 78.9), ('张雨欣', '医学院', 20, 85.5), ('刘子轩', '文学院', 18, 96.0) ]，其中每个元素依次表示：姓名、学院、年龄、平均绩点。请使用 sorted() 函数结合 lambda 表达式，对该列表进行排序。

排序规则如下：

主排序条件：按平均绩点从高到低排列

次排序条件：当平均绩点相同时，按学院名称的字典序升序排列

最终排序条件：若前两项都相同，则按年龄从小到大排列

将排序结果转换为只包含「姓名」和「学院」的新列表

输出结果时，姓名需用中文姓氏大写格式显示（如：CHEN Xiaoming）

源代码：

students = [ ('陈小明', '计算机学院', 20, 85.5), ('李思雨', '文学院', 19, 92.3),

('王浩然', '计算机学院', 21, 78.9), ('张雨欣', '医学院', 20, 85.5), ('刘子轩', '文学院', 18, 96.0) ]

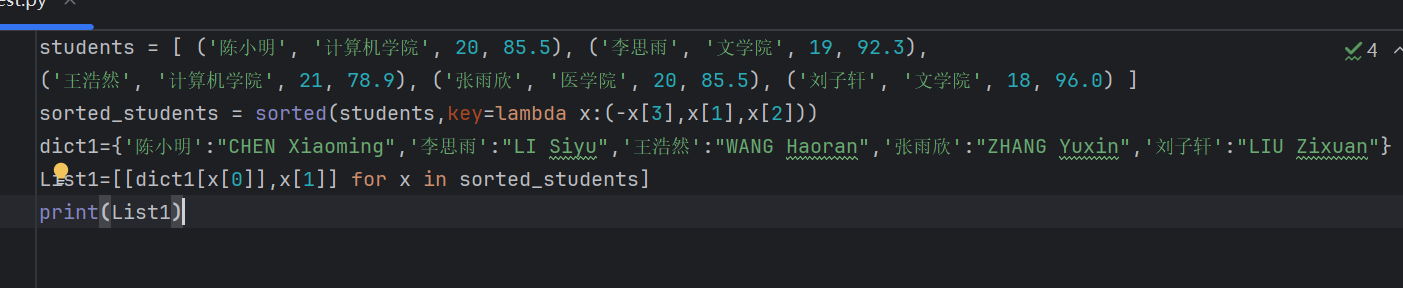
sorted\_students = sorted(students,key=lambda x:(-x[3],x[1],x[2]))

dict1={'陈小明':"CHEN Xiaoming",'李思雨':"LI Siyu",'王浩然':"WANG Haoran",'张雨欣':"ZHANG Yuxin",'刘子轩':"LIU Zixuan"}

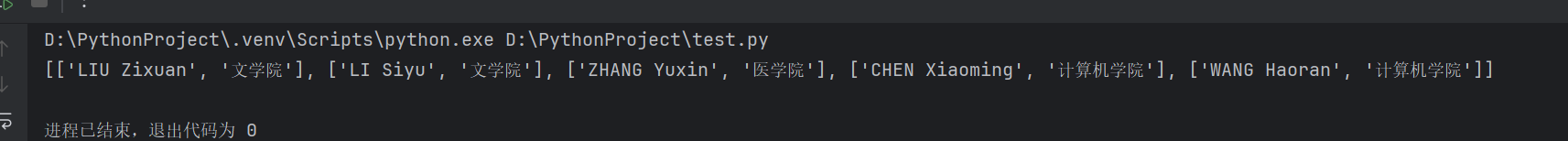
List1=[[dict1[x[0]],x[1]] for x in sorted\_students]

print(List1)

源代码截图：



运行结果截图：



函数功能说明：

sorted(Iterable,Key,reverse) 函数

Iterable:迭代对象

Key:主要是用来进行比较的元素，只有一个参数，具体的函数的参数就是取自于可迭代对象中，指定可迭代对象中的一个元素来进行排序。

reverse :排序规则，reverse = True 降序 ， reverse = False 升序（默认）

(7)回文数：反向排列与原来一样的数就叫做回文数。

编写一个python程序，实现按回文数的规则，判断输入例子（123321、12321、12345321）的数是否为回文数，可输入多组数据，输入0则结束程序。

源代码：

while True:

str=input("请输入一串数字:")

if str=="0":

print("程序结束")

break

else:

str1=str[::-1]

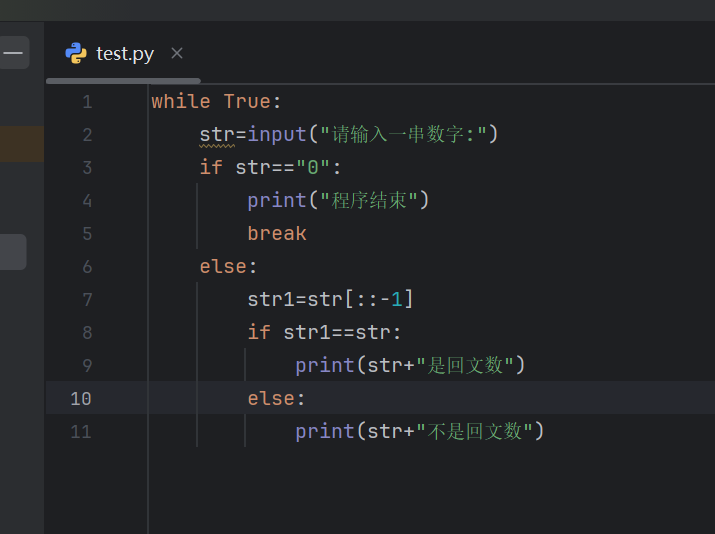
if str1==str:

print(str+"是回文数")

else:

print(str+"不是回文数")

源代码截图：



运行结果截图：



(8)编写Python程序实现以下功能：  
使用map和reduce函数将罗马数字字符串转换为整数，规则如下：

I = 1

V = 5

X = 10

L = 50

C = 100

D = 500

M = 1000

特殊规则：当左符号小于右符号时，左符号的值需被减去（如 IV=4, IX=9）。

源代码：

from functools import reduce

def change(a):

dict1 = {"I": 1, "V": 5, "X": 10, "L": 50, "C": 100, "D": 500, "M": 1000}

return dict1[a]

string=input("请输入罗马数字字符串:")

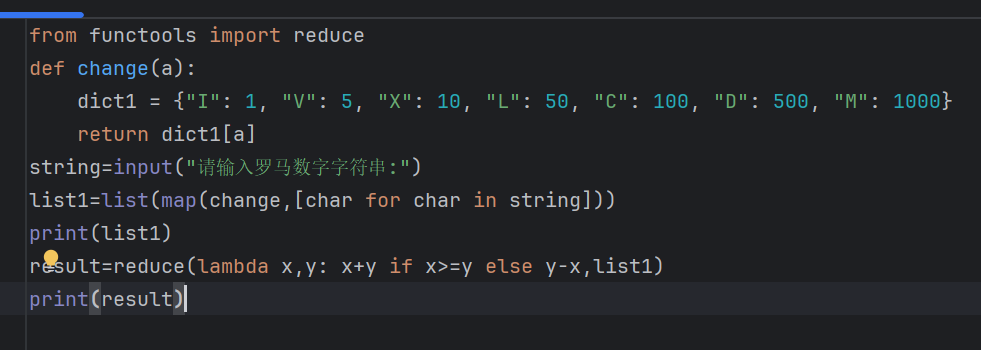
list1=list(map(change,[char for char in string]))

print(list1)

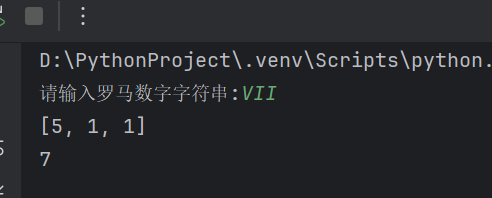
result=reduce(lambda x,y: x+y if x>=y else y-x,list1)

print(result)

源代码截图：



运行结果截图：



(9)请编写程序，找出一段给定文字中出现最频繁的那个英文字母。输入一个长度不超过1000的字符串，字符串由ASCII码表中任意可见字符及空格组成，至少包含1个英文字母，以回车结束（回车不算在内）。在一行中输出出现频率最高的那个英文字母及其出现次数，其间以空格分隔。如果有并列，则输出按字母序最小的那个字母。统计时不区分大小写，输出小写字母。

源代码：

string=input("输入长度不超过1000的字符串:")

list1=[char for char in string.lower()]

list1.sort()

a=[0]\*260

max=0

maxset=""

for i in list1:

if "a"<=i<="z":

a[ord(i)]+=1

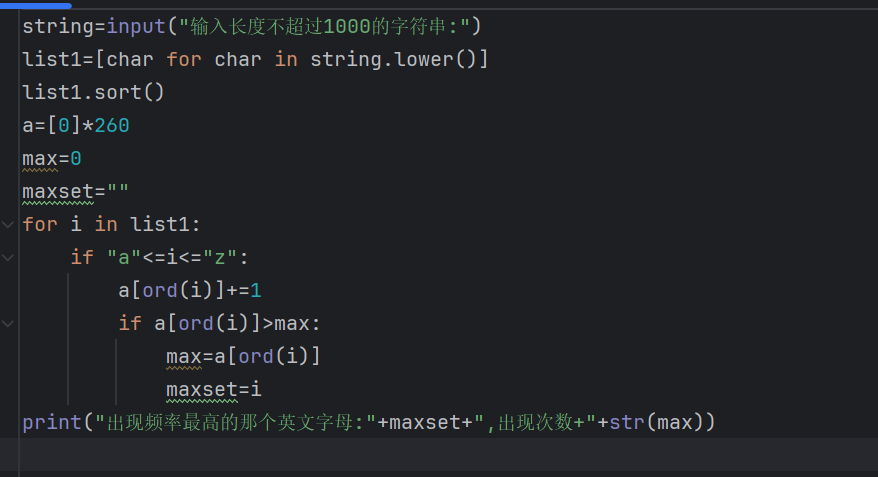
if a[ord(i)]>max:

max=a[ord(i)]

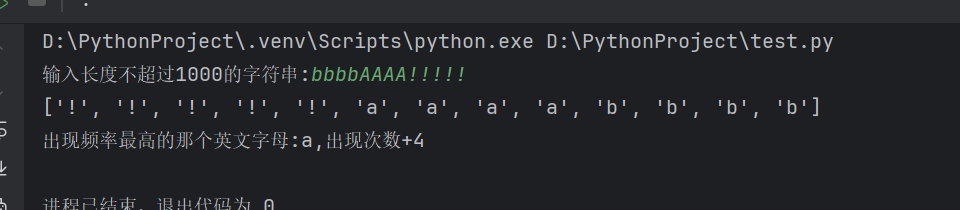
maxset=i

print("出现频率最高的那个英文字母:"+maxset+",出现次数+"+str(max))

源代码截图：



运行结果截图：



(10)编写函数实现[pow(x,n)](https://www.cplusplus.com/reference/valarray/pow/)，即计算x的整数n次幂函数（即）。注意：不得直接调用python的x\*\*n计算方式或其它库函数。

源代码：

def pow(x,n):

sum=1

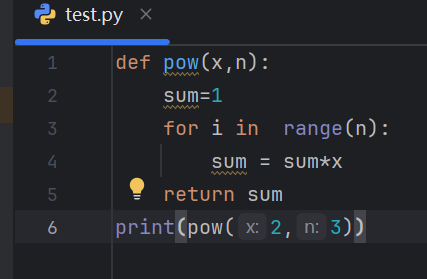
for i in range(n):

sum = sum\*x

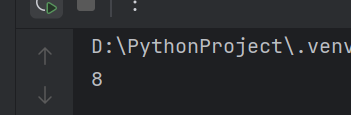
return sum

print(pow(2,3))

源代码截图：



运行结果截图：



(11)现声明三人的字典结构信息为：

people=[

{"name":"xiaoming","age":25},

{"name":"xiaohong","age":10},

{"name":"xiaohua","age":18}

]

请使用filter以及lambda函数将年龄小于20的人名输出。

源代码：

people=[

{"name":"xiaoming","age":25},

{"name":"xiaohong","age":10},

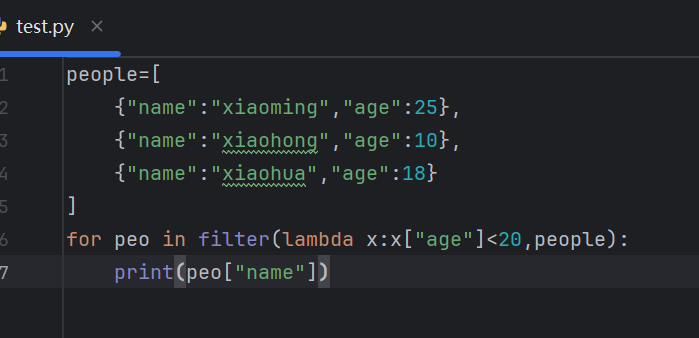
{"name":"xiaohua","age":18}

]

for peo in filter(lambda x:x["age"]<20,people):

print(peo["name"])

源代码截图：



运行结果截图：



(12)已知商品价格列表，其中每个元素是一个字典，包含'product'和'price'键：

products = [

{'product': '苹果', 'price': 10.5},

{'product': '香蕉', 'price': 5.2},

{'product': '橘子', 'price': 7.8},

{'product': '苹果', 'price': 12.0},

{'product': '葡萄', 'price': 15.0}

]

编写程序完成以下任务：

提取价格列表：使用 map() 函数提取所有商品价格

计算平均价格：使用 reduce() 函数计算平均价格（保留两位小数）

筛选高价商品：使用 filter() 函数筛选出价格高于平均价格的商品

格式化输出：

平均价格输出为 ¥12.34 格式

高价商品列表显示为 [["苹果", 12.0], ["葡萄", 15.0]] 格式

源代码：

from functools import reduce

products = [

{'product': '苹果', 'price': 10.5},

{'product': '香蕉', 'price': 5.2},

{'product': '橘子', 'price': 7.8},

{'product': '苹果', 'price': 12.0},

{'product': '葡萄', 'price': 15.0}

]

price=list(map(lambda x: x['price'], products))

average\_price=reduce(lambda x, y: x+y, price)/len(price)

high\_price\_product=filter(lambda x:x['price']>average\_price,products)

list1=[]

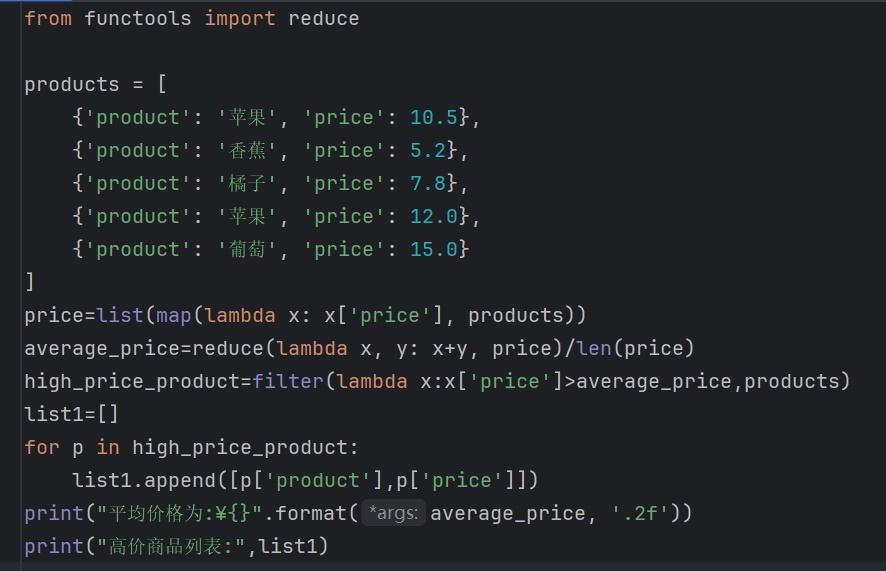
for p in high\_price\_product:

list1.append([p['product'],p['price']])

print("平均价格为:¥{}".format(average\_price, '.2f'))

print("高价商品列表:",list1)

源代码截图：



运行结果截图：

