**实验2 Python基础语法实践**

**姓名： 徐超信 学号：1935010102 班级：计科1902 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

**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：

from typing import overload

def count(string):

    count\_lowercase=0

    count\_uppercase=0

    count\_digit=0

    count\_others=0

    for i in string:

        if i.islower():

            count\_lowercase+=1

        elif i.isupper():

            count\_uppercase+=1

        elif i.isdigit():

            count\_digit+=1

        else:

            count\_others+=1

    return count\_uppercase,count\_lowercase,count\_digit,count\_others

string= string=input("input a string:")

"""

23u0jnd;gjaSSS

"""

result\_tuple=count(string)

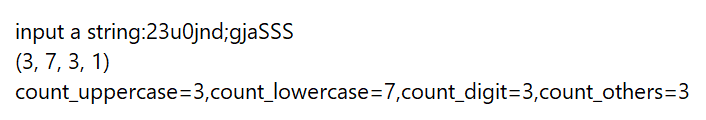
print(result\_tuple)

print(f"count\_uppercase={result\_tuple[0]},count\_lowercase={result\_tuple[1]},count\_digit={result\_tuple[0]},count\_others={result\_tuple[2]}")

源代码截图：



运行结果截图：



# (2) 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：

def isprime(n):

    if n==1 or n==0:

        return False

    for i in range(2,n):

        if n%i==0:

            return False

    return n

def decompose():

    n=(int)(input("input a positive integer:"))

    list\_factor=[]

    list\_power=[]

    list=[list\_factor,list\_power]

    """ python3:the dict is ordered! """

    # dict={}

    """ use the index variable to guide the power update operation """

    j=-1

    i=2

    # for i in range(2,n):

    while n>1:

        # if n==1:

        #     break

        if n%i==0:

            if isprime(i):

                """ iterate n: """

                n=n//i

                """ update the prime factor and its correspondent power  """

                if(i not in list\_factor):

                    list\_factor.append(i)

                    list\_power.append(1)

                    j+=1

                else:

                    list\_power[j]+=1

            else:

                i+=1

        else:

            i+=1

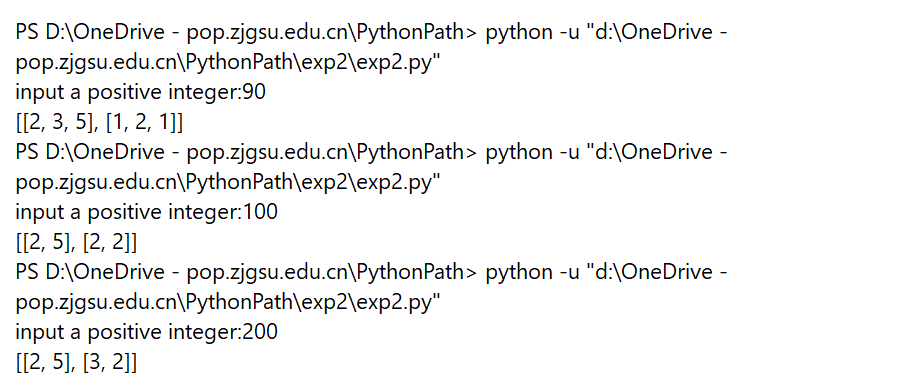
    return list

print(decompose())

源代码截图：



运行结果截图：



# (3) 有一分数序列：2/1，3/2，5/3，8/5，13/8，21/13..., 分别求出这个数列的前10项之和以及前20项之和，并打印输出，输出格式要求小数点后保留4位。

# 源代码：

""" iterator """

def fib\_variant(generate\_max\_times):

    count = 0

    a, b = 2, 3

    while count < generate\_max\_times:

        yield a

        a, b = b, a+b

        count += 1

    return "end"

def fib\_variant2(generate\_max\_times):

    count = 0

    a, b = 1, 2

    while count < generate\_max\_times:

        yield a

        a, b = b, a+b

        count += 1

""" test the iterator: """

# for i in fib\_variant(10):

#     print(i)

# for i in fib\_variant2(10):

#     print(i)

""" calculate:sum: 2/1，3/2，5/3，8/5，13/8，21/13."""

def sum\_fib\_variant(bound):

    count = 0

    f1 = fib\_variant(bound)

    f2 = fib\_variant2(bound)

    """ range from 0(not 1) """

    for i in range(0, bound):

        count += next(f1)/next(f2)

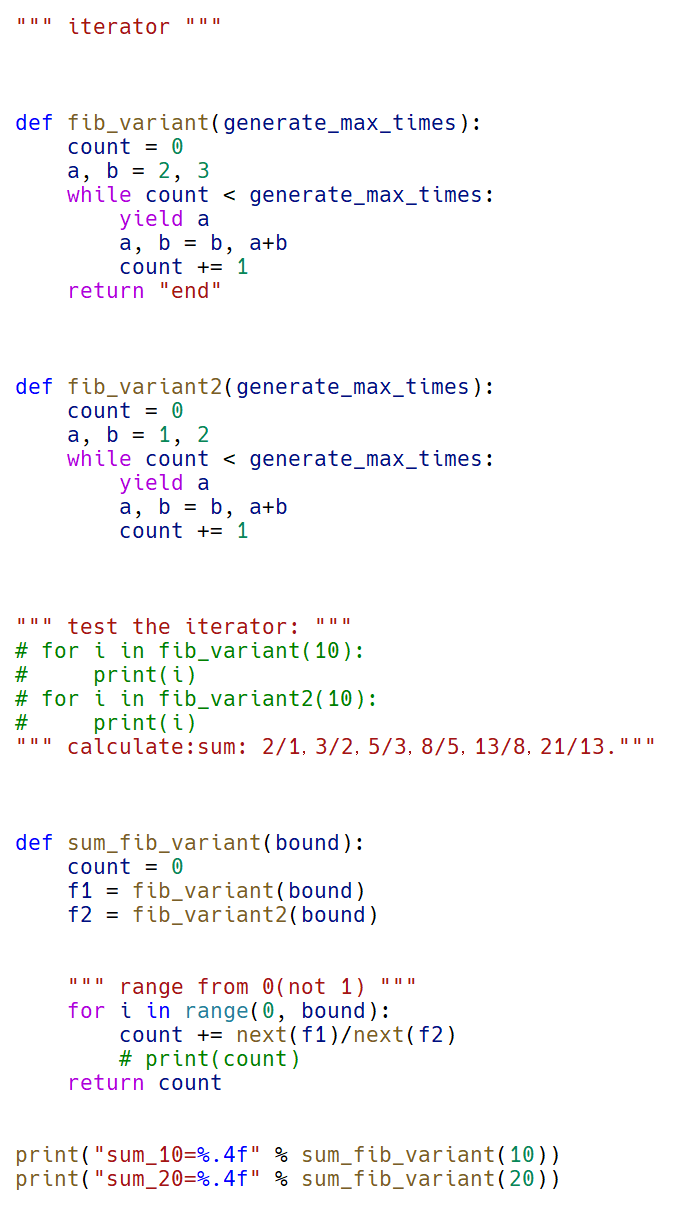
        # print(count)

    return count

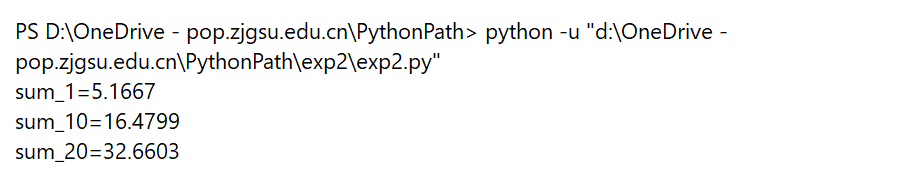
print("sum\_10=%.4f" % sum\_fib\_variant(10))

print("sum\_20=%.4f" % sum\_fib\_variant(20))

源代码截图：



运行结果截图：



# (4) 任意输入一串字符，请编程实现: (a) 统计子串’ab’出现的次数，(b) ‘ab’出现的地方全部替换为’AB’，(c) 以’ab’拆分原字符串，打印出结果。

源代码：

def count(string):

    count=0

    # count=string.replace('ab','AB');

    while 'ab' in string:

        string=string.replace('ab','AB',1)

        count+=1

    return count

string=input("input a string contains several 'ab':")

print("the string contains:%d'ab'\n"%count(string))

string=string.split(sep='ab')

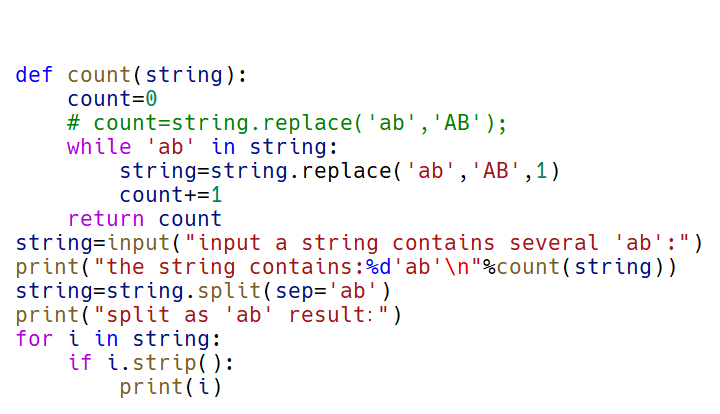
print("split as 'ab' result：")

for i in string:

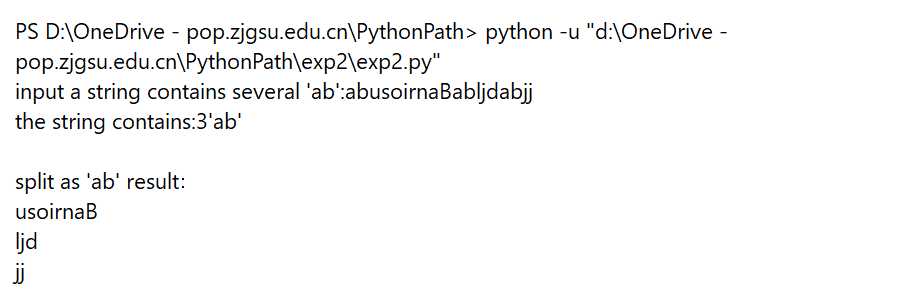
    if i.strip():

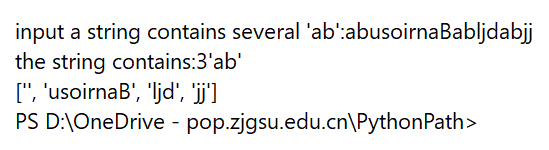
        print(i)

源代码截图：



运行结果截图：





# (5) 编写一个函数，参数为a、b、c，要求计算如下数学公式的值。

源代码：

import math

from math import e

def calculate(a,b,c):

    return e\*\*b\*\*((math.pi/2)\*\*0.5)+(-b+(b\*\*2-4\*a\*c)\*\*0.5)/2\*a+(math.log10(abs(a+b))+a/b)/math.log(a\*\*b+100,e)

print(calculate(1,2,3))

""" test: """

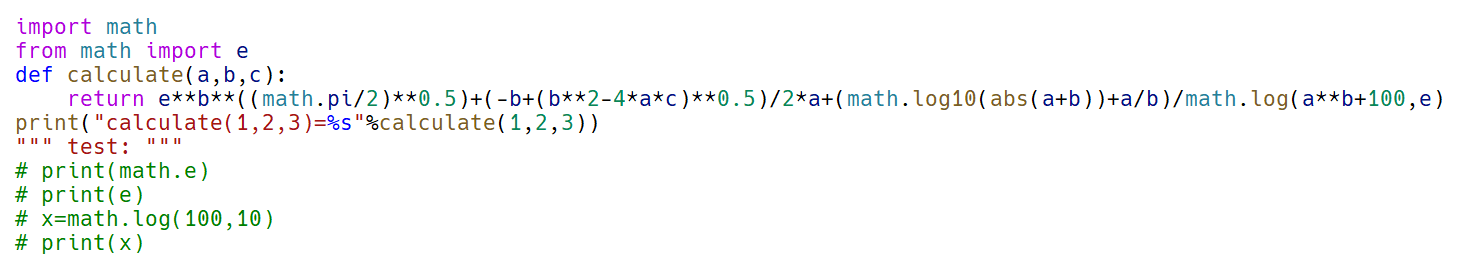
# print(math.e)

# print(e)

# x=math.log(100,10)

# print(x)

源代码截图：



运行结果截图：



# (6)编写一个函数，形式参数为整数n，要求打印出形如下列的图案（菱形），最中间一行的’\*’ 符号个数为2n+1. 然后，任意输入一个n值（通过input函数实现），打印出对应的菱形图案。

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*

源代码：

def print\_up\_triangle(n):

    j=n-1

    for i in range(0,n):

        print(j\*' '+(i\*2+1)\*'\*')

        j-=1

def print\_down\_triangle(n):

    j=1

    for i in range(n-2,-1,-1):

        print(j\*' '+(2\*i+1)\*'\*')

        j+=1

def print\_diamond(n):

    print\_up\_triangle(n)

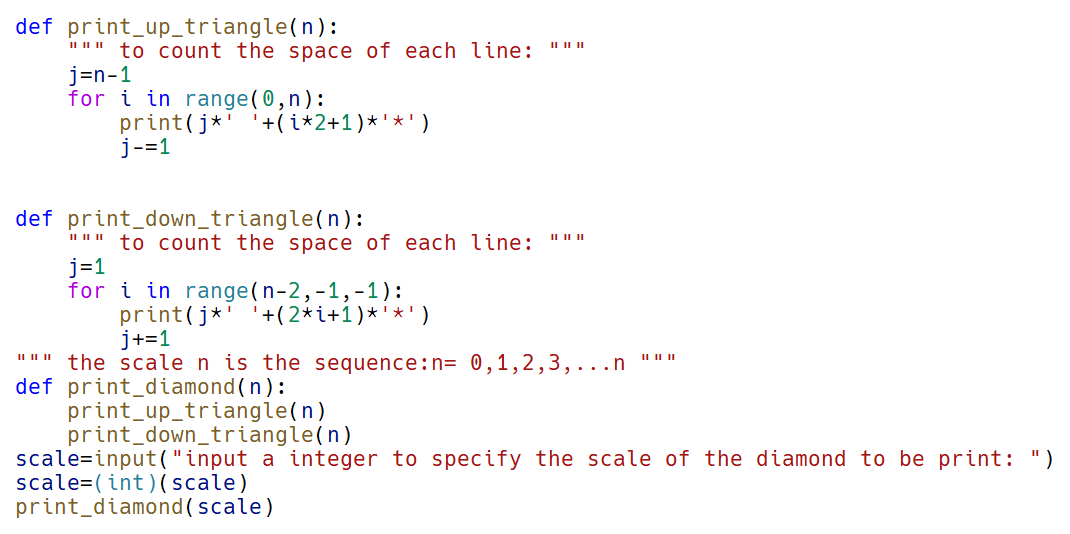
    print\_down\_triangle(n)

scale=input("input a integer to specify the scale of the diamond to be print: ")

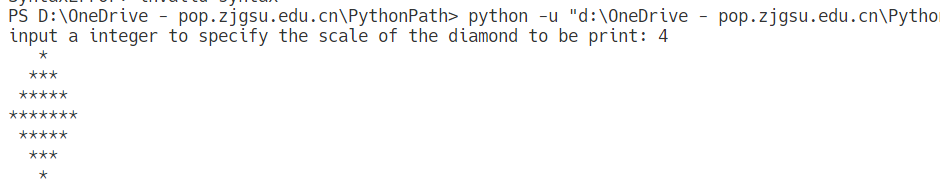
scale=(int)(scale)

print\_diamond(scale)

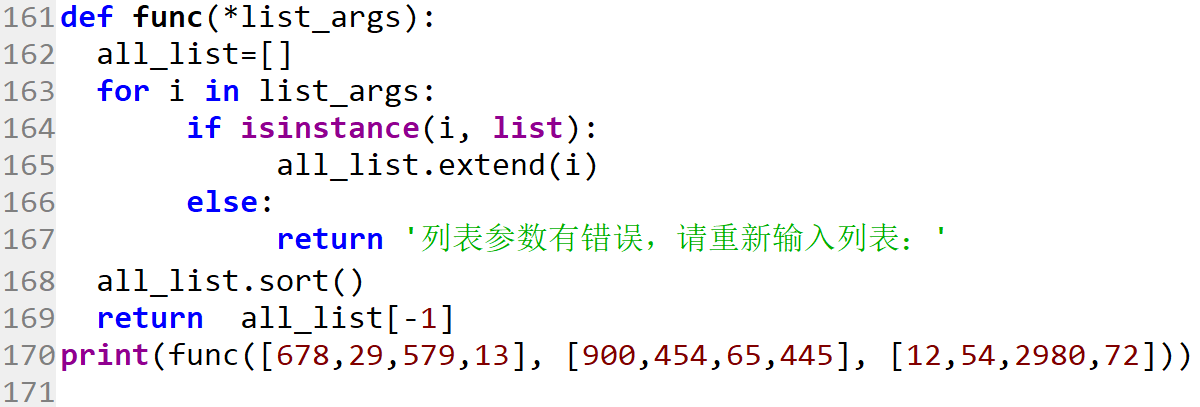
源代码截图：



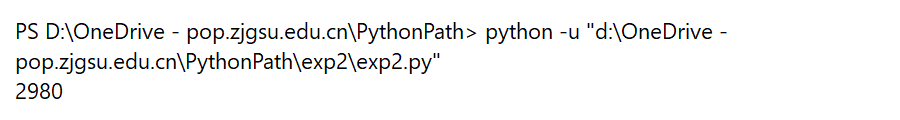
运行结果截图：



# (7) 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()可以接受不定数量的各种参数,但是如果传入了非list类型的实参,那么就会返回:"列表参数有错误，请重新输入列表：";func内部将对输入的参数中list类型的参数进行处理,利用isinstance()来判断各个参数是否为list类型,如果是list类型,那么对list实参中的数字元素通过extend()方法批量的添加到all\_list列表中,然后对all\_list中的元素进行排序,左后使用all\_list[-1]将其中的最大元素打印出来(即2980) """

# (8) 现有一个整数列表ls=[11,22,33,44,55,66,77,88,99,100,110,200,230,330], 将ls中所有大于66的元素保存至字典的第一个key ’k1’，将小于等于66的元素保存至字典的第二个key ’k2’, 以及将ls中的全部奇数保存至字典的第三个key ’k3’. 打印出这个字典，并通过key访问字典元素，计算ls中所有奇数的平方和。

源代码：

def classifier(ls):

    list\_small = []

    list\_big = []

    list\_odd = []

    dict = {'k1': list\_small, 'k2': list\_big, 'k3': list\_odd}

    for i in ls:

        if i <= 66:

            list\_small.append(i)

        elif i > 66:

            list\_big.append(i)

        if i % 2 == 1:

            list\_odd.append(i)

    return dict

ls = [11, 22, 33, 44, 55, 66, 77, 88, 99, 100, 110, 200, 230, 330]

def sum\_of\_squares(odd\_dict):

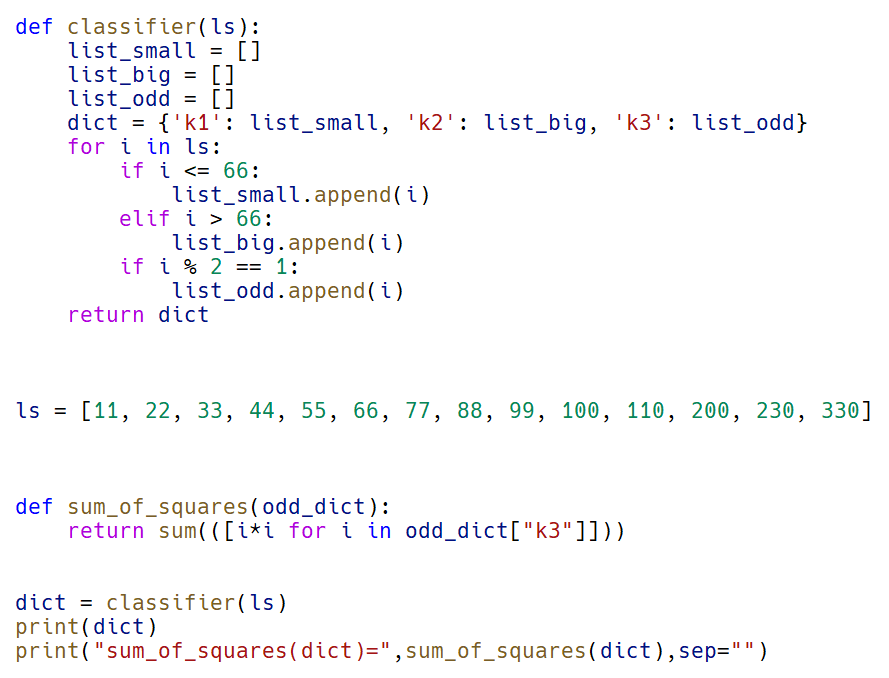
    return sum(([i\*i for i in odd\_dict["k3"]]))

dict = classifier(ls)

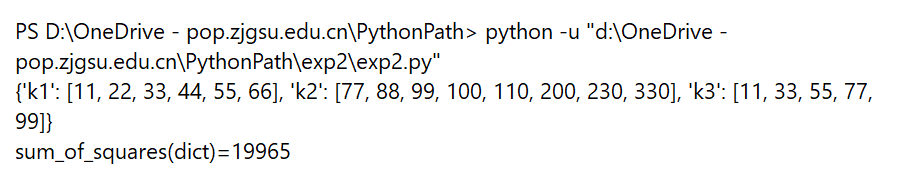
print(dict)

print("sum\_of\_squares(dict)=",sum\_of\_squares(dict),sep="")

源代码截图：



运行结果截图：



# (9) 写函数，检查传入字典（元素数目不确定）的每一个value的长度,如果大于2，那么仅保留前两个长度的内容，并将新内容返回给调用者，同时也请给出一个测试样例。

源代码：

def deal\_dict(dict):

    """ result\_dict: """

    ret = {}

    for key,value in dict.items():

        if len(value) > 2:

            """ use section """

            ret[key] = value[0:2]

        else:

            """ the length doesn't longer than 2,return directly: """

            ret[key] = value

    """ return the finally result\_dict """

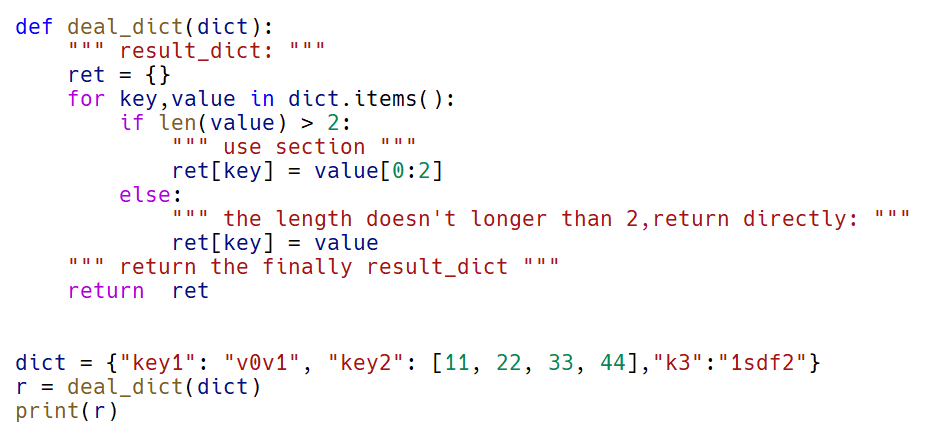
    return  ret

dict = {"key1": "v0v1", "key2": [11, 22, 33, 44],"k3":"1sdf2"}

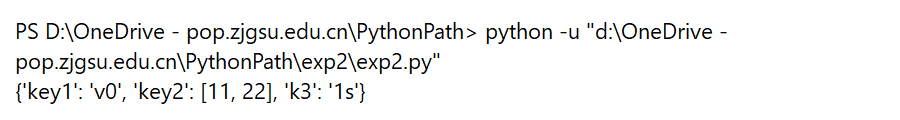
r = deal\_dict(dict)

print(r)

源代码截图：



运行结果截图：



# (10) 利用一个简单的lambda函数，对列表people = [('牛牛', 'F', 15), ('道长', 'M', 12), ('大师兄','M', 10), ('小师妹','F', 13)]按照年龄进行排序。这里的年龄指列表中元组的第3个分量。

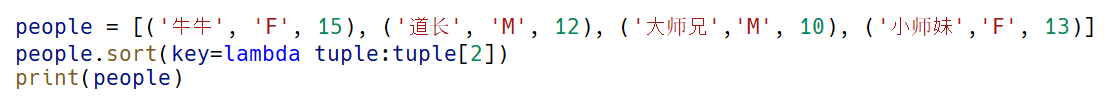
源代码：

people = [('牛牛', 'F', 15), ('道长', 'M', 12), ('大师兄','M', 10), ('小师妹','F', 13)]

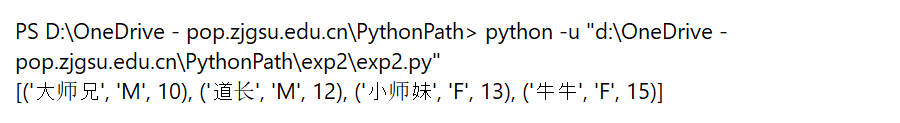
people.sort(key=lambda tuple:tuple[2])

print(people)

源代码截图：



运行结果截图：



# (11) 编写函数，传入n个数，返回字典{‘max’:最大值, ‘min’:最小值}. 给出一个实例进行测试。

源代码：

def extrem\_value\_dict(list):

    dict={}

    dict['min']=min(list)

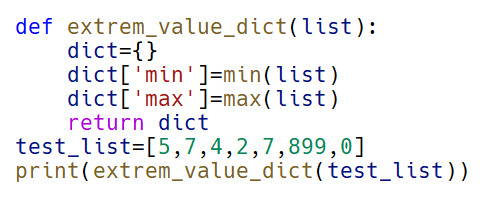
    dict['max']=max(list)

    return dict

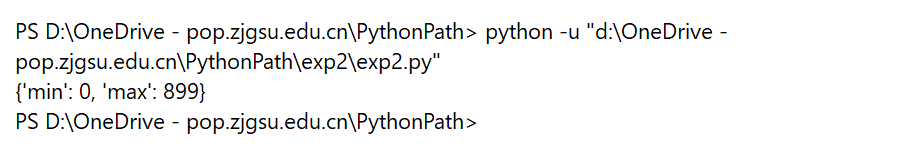
test\_list=[5,7,4,2,7,899,0]

print(extrem\_value\_dict(test\_list))

源代码截图：



运行结果截图：



# (12)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。

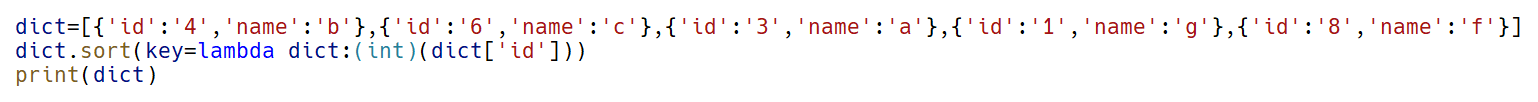
源代码：

dict=[{'id':'4','name':'b'},{'id':'6','name':'c'},{'id':'3','name':'a'},{'id':'1','name':'g'},{'id':'8','name':'f'}]

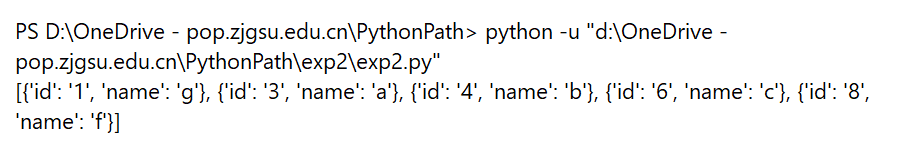
dict.sort(key=lambda dict:(int)(dict['id']))

print(dict)

源代码截图：



运行结果截图：



# (13)请利用map和reduce函数编写一个str2float函数, 把字符串’123.456’转换成浮点数123.456. 此题不允许直接使用float()对字符串进行直接转换。

源代码：

from functools import reduce

""" the function just to ignore the float point to turn to a integer number:  """

def strToInteger(s):

    def fn(x, y):

        return x \* 10 + y

    def char2num(s):

        """ create a built-in dict to deal with map references """

        return {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9}[s]

    """ use map and reduce: """

    """ use: map(char2num, s.replace(".","") to transfor the digit char to int digit

    use: the reduce to iterate(reduce) the scale of the input """

    return reduce(fn, map(char2num, s.replace(".", "")))

def str2float(s):

    """ if you meet the float ,turn make a divide by a proper number to convert it to a float correctly"""

    if s.find(".") != -1:

        print('str2float(\'%s\') =' % s, strToInteger(

            s)/pow(10, (len(s)-s.find(".")-1)))

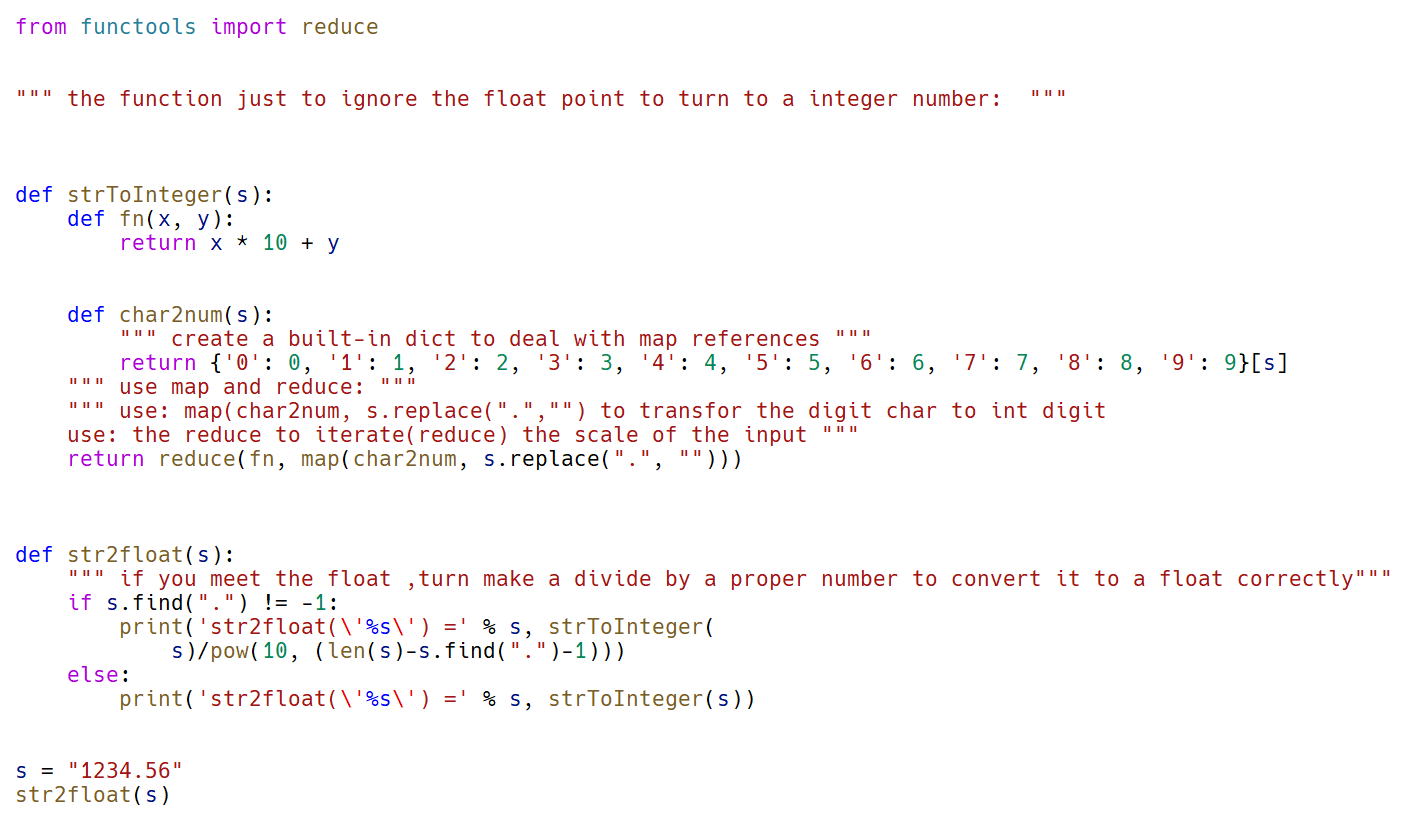
    else:

        print('str2float(\'%s\') =' % s, strToInteger(s))

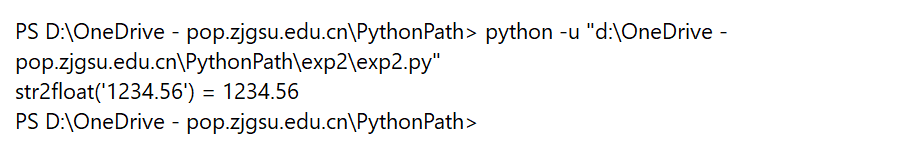
s = "1234.56"

str2float(s)

源代码截图：



运行结果截图：



# (14) 随意给出一个字符串列表，请在这个列表中的所有字符串尾部添加“\_good”，比方原先字符串为“alex”，则变为“alex\_good”。要求用map和lambda函数来处理该问题，并给出一个实例进行测试。

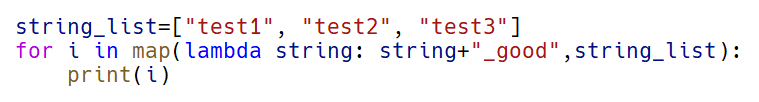
源代码：

string\_list=["test1", "test2", "test3"]

for i in map(lambda string: string+"\_good",string\_list):

    print(i)

源代码截图：



运行结果截图：

