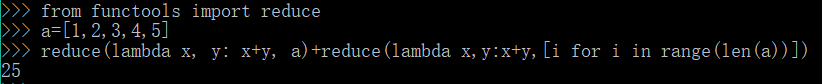
## 将一个列表的所有索引值和里面的所有数字求和

from functools import reduce

a=[1,2,3,4,5]

reduce(lambda x, y: x+y, a)+reduce(lambda x,y:x+y,[i for i in range(len(a))])

运行结果：

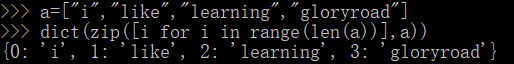


## 2、将一个列表中所有字符串的索引位置和字符串内容，放到一个字典中

a=["i","like","learning","gloryroad"]

dict(zip([i for i in range(len(a))],a))

运行结果：



## 3、写一个循环，不断的问用户想买什么，用户选择一个商品编号，就把对应的商品添加到购物车里， 最终用户输入q退出时，打印购物车里的商品列表。

#encoding=utf-8

def get\_shopping\_good():

print("""欢迎进入本商店

1001 代表白菜

1002 代表萝卜

1003 代表黄瓜

q 代表退出""")

goods=[{"1001":"白菜"},{"1002":"萝卜"},{"1003":"黄瓜"}]

shopping\_car=[]

while True:

choice=input("请输入商品编号：")

if choice=="q":

if len(shopping\_car)>0:

print("\*\*\*\*\*已购买的商品\*\*\*\*\*")

for i in shopping\_car:

print(" %s %s"%("".join(i.keys()),"".join(i.values())))

else:

print("购物车中无商品!")

break

else:

for good in goods:

if choice in good.keys():

shopping\_car.append(good)

print("%s已加入购物车"%(good))

break

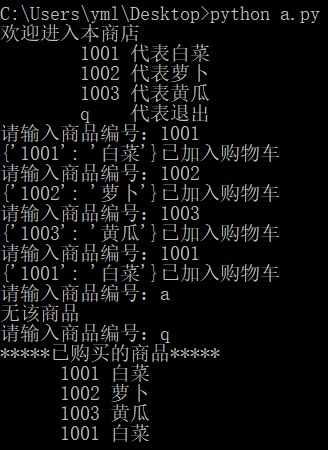
else:

print("无该商品")

if \_\_name\_\_=="\_\_main\_\_":

get\_shopping\_good()

运行结果：



## 4、一个字符串I am a boy，请依照如下规则转换为数字

abc–2, def–3, ghi–4, jkl–5, mno–6, pqrs–7, tuv–8 wxyz–9

def get\_num(num):

b=""

for i in num.lower():

if i.isspace():

b+=i

elif i in "yz":

b+="9"

else:

value=(ord(i)-97)//3

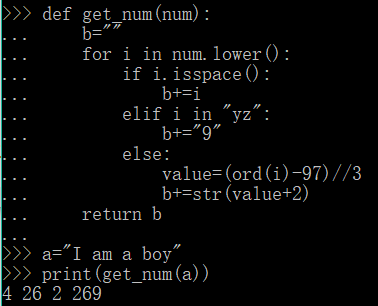
b+=str(value+2)

return b

a="I am a boy"

print(get\_num(a))

运行结果：



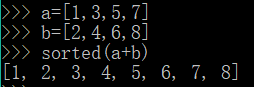
## 5、两个列表[1,3,5,7]和[2,4,6,8]合并为[1,2,3,4,5,6,7,8]

a=[1,3,5,7]

b=[2,4,6,8]

sorted(a+b)

运行结果：



## 6、[1,2,3,4,5,6,7,8],多次打印这个列表，每次打印前删除中间那个元素，直到最后剩一个元素

def print\_list(data):

while len(data)>=1:

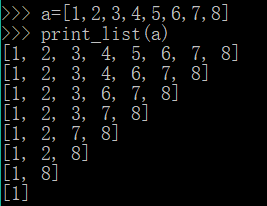
print(data)

data.pop(len(data)//2)

a=[1,2,3,4,5,6,7,8]

print\_list(a)

运行结果：



## 7、[6,2,11,1,4,9,5,7,8,20]将所有比6小的数字放到它的左边，大的放到6的右边。

def print\_data(num):

b=[]

for i in a:

if i<6:

b.insert(0,i)

else:

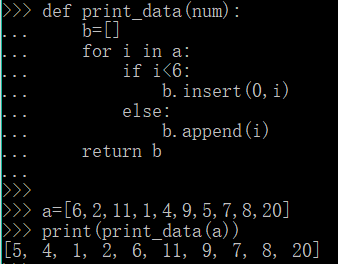
b.append(i)

return b

a=[6,2,11,1,4,9,5,7,8,20]

print(print\_data(a))

运行结果：



## 8、使用递归的方式倒序打印这个列表[1,2,3,4,5,6]

a=[]

def get\_num(num):

if len(num)==0:

print(a)

else:

a.append(num[-1])

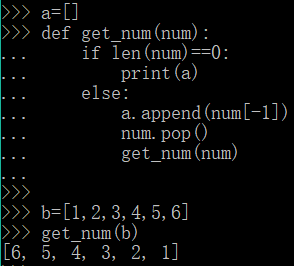
num.pop()

get\_num(num)

b=[1,2,3,4,5,6]

get\_num(b)

运行结果：



## 9、使用递归的方式正序打印这个列表[1,2,3,4,5,6]

a=[]

def get\_num(num):

if len(num)==0:

print(a)

else:

a.append(num[0])

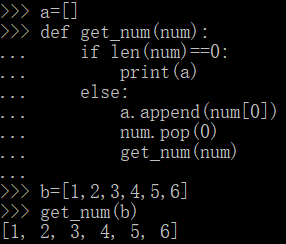
num.pop(0)

get\_num(num)

b=[1,2,3,4,5,6]

get\_num(b)

运行结果：



## 10、程序自动生成一个奇数且能被3和5整除的二维列表

import random

def get\_data(num):

result=[]

for i in range(num):

while True:

data = random.randint(1,100)

if data%3==0 and data%5==0:

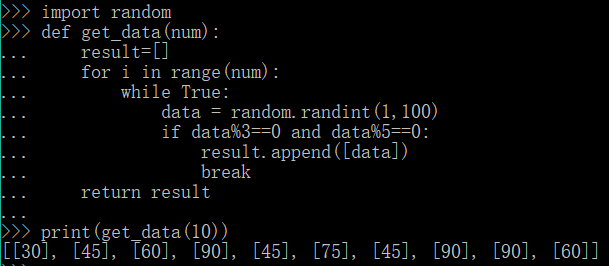
result.append([data])

break

return result

print(get\_data(10))

运行结果：



## 11、使用26个小写字母和26个小写字母对应的ascii生成一个字典

import string

def get\_dict():

b=[]

for i in string.ascii\_lowercase:

a=[i,ord(i)]

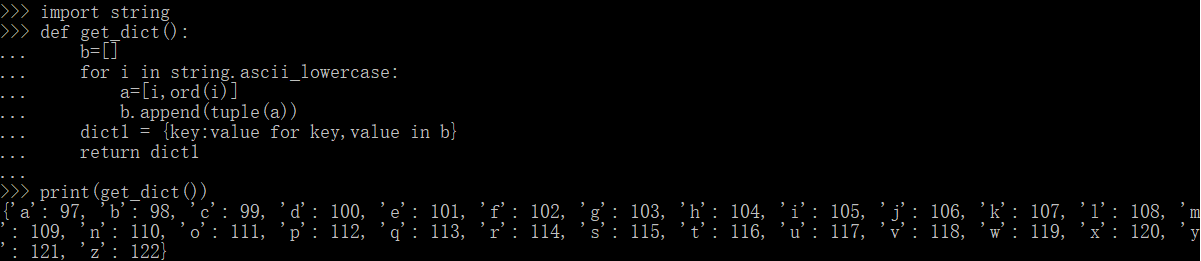
b.append(tuple(a))

dict1 = {key:value for key,value in b}

return dict1

print(get\_dict())

运行结果：



## 12、使用26个字母对应随机的asicii码（97-122之间），基于这个字典规则,将一个单词进行加密

#encoding=utf-8

import string

import random

def get\_dict():

b=[]

values=[]

for i in string.ascii\_lowercase:

#随机生成不重复的数字

while True:

num=random.randint(97,122)

if values.count(num)==0:

values.append(num)

break

#字母和对应随机的ascii码放到列表中

a=[i,num]

b.append(tuple(a))

dict1 = {key:value for key,value in b}

return dict1

'''按规则加密数据'''

def encrypt\_words(data,rule):

result=""

for i in data:

result+=str(rule.get(i))

return result

if \_\_name\_\_=="\_\_main\_\_":

#加密字符串

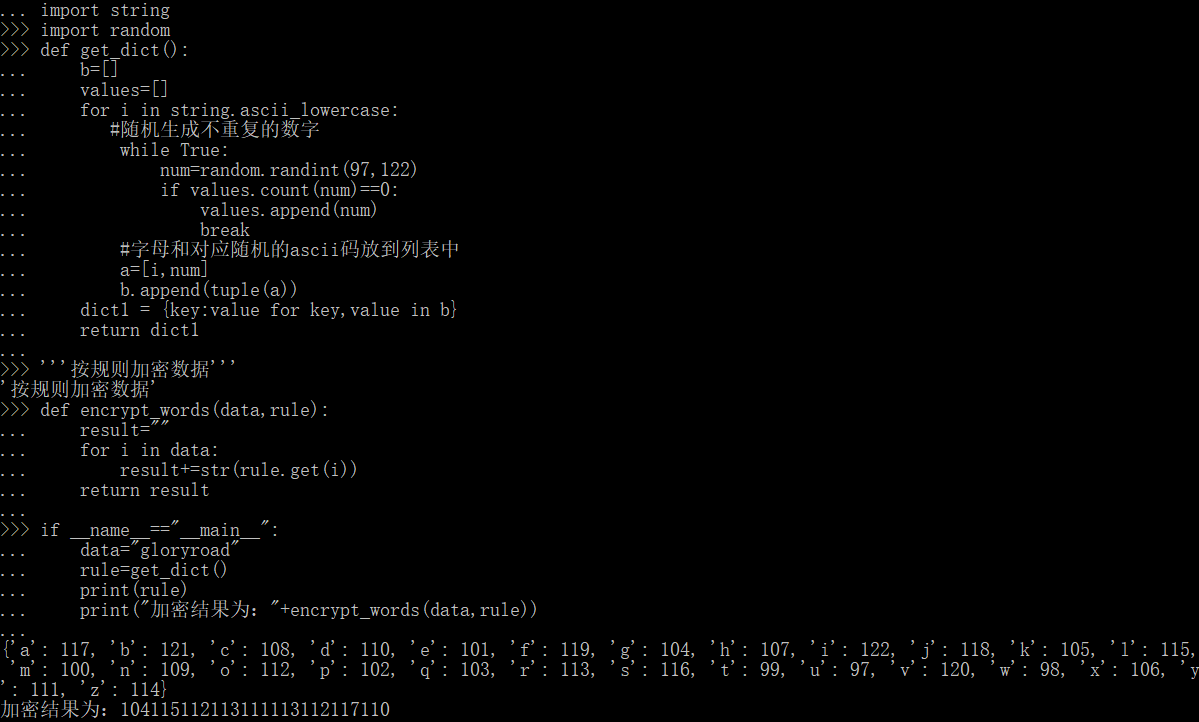
data="gloryroad"

rule=get\_dict()

print(rule)

print("加密结果为："+encrypt\_words(data,rule))

运行结果：



## 13、将12题的加密串进行解密

#encoding=utf-8

decrypt\_result=""

def get\_data(data,rule):

global decrypt\_result

if len(data)==0:

print("解密结果为："+decrypt\_result)

else:

#碰到数字是1的则截取3位解码，如果是其他数字则截取2位解码

if data[0]=="1":

for key,value in rule.items():

if value==int(data[:3]):

decrypt\_result+=key

get\_data(data[3:],rule)

else:

for key,value in rule.items():

if value==int(data[:2]):

decrypt\_result+=key

get\_data(data[2:],rule)

if \_\_name\_\_=="\_\_main\_\_":

#加密数据和解密规则

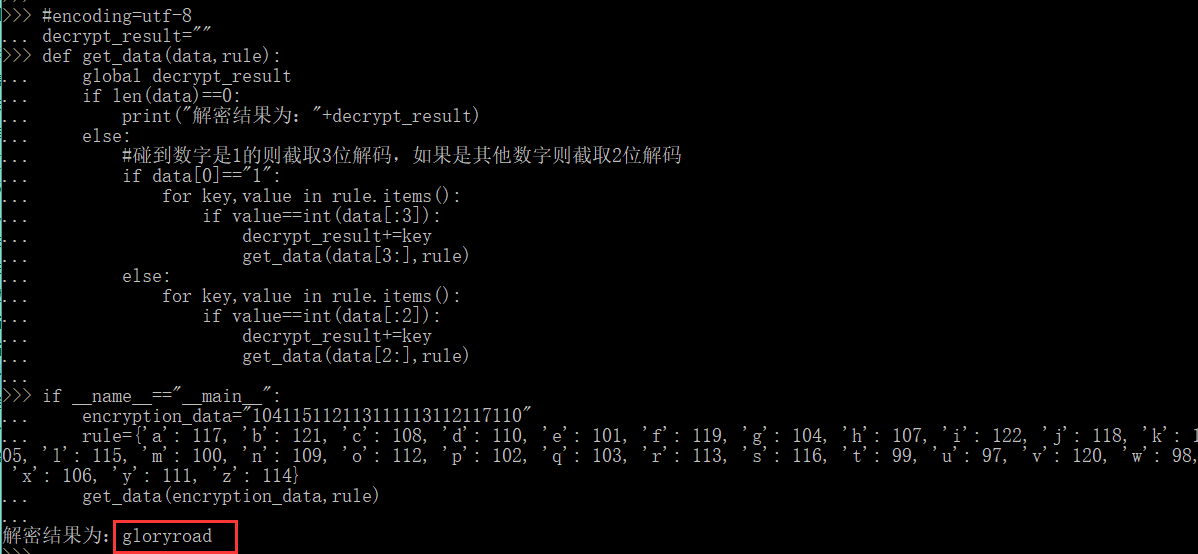
encryption\_data="104115112113111113112117110"

rule={'a': 117, 'b': 121, 'c': 108, 'd': 110, 'e': 101, 'f': 119, 'g': 104, 'h': 107, 'i': 122, 'j': 118, 'k': 105, 'l': 115, 'm': 100, 'n': 109, 'o': 112, 'p': 102, 'q': 103, 'r': 113, 's': 116, 't': 99, 'u': 97, 'v': 120, 'w': 98, 'x': 106, 'y': 111, 'z': 114}

#解密结果

get\_data(encryption\_data,rule)

运行结果：



## 14、写一个函数传入一个List列表，生成一个字典：所有元素作为字典的key（注意判断合法性，不合法的滤掉），value都为1

def get\_data(list\_data):

result=[]

for i in list\_data:

if isinstance(i,(str,int)):

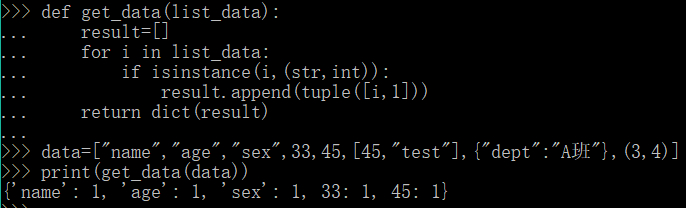
result.append(tuple([i,1]))

return dict(result)

data=["name","age","sex",33,45,[45,"test"],{"dept":"A班"},(3,4)]

print(get\_data(data))

运行结果：



## 15、生成一个n级深度的字典，例如：[1,2,3,4,5,6] 可以生成{1: {2: {3: {4: {6: 5}}}}}，写一个函数定义n级

def f(n):

#生成n级字典，如n为5，则需要数字到7才能生成5级

a=[x for x in range(1,n+2)]

data=a.pop(-2)

a.reverse()

#从最内层字典开始生成起，然后将内层字典作为下个key的value

for i in a:

result=dict()

result[i]=data

data=result

return result

if \_\_name\_\_=="\_\_main\_\_":

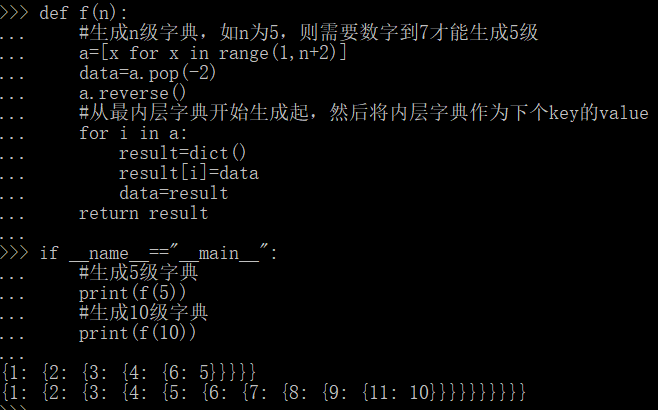
#生成5级字典

print(f(5))

#生成10级字典

print(f(10))

运行结果：



## 16、{1: {2: {3: {4: {6: 5}}}}} 将所有key和value的数字求和，且层数可能不一定

b=[]

def get\_sum(a):

for key,value in a.items():

b.append(key)

if isinstance(value,dict):

get\_sum(value)

else:

b.append(value)

return sum(b)

a={1:{2:{3:{4:{6:{66:55,55:{67:{2:3,4:5}}}}}}}}

print(get\_sum(a))

运行结果：

