**实验3 文件操作与数据存取**

**姓名： 徐超信 学号：1935010102 班级：计科1902成绩：**

**1. 实验目标**

本次实验旨在了解Python的文件数据操作方法，理解和掌握数据存取的各种方式。需掌握的相关知识点主要包括：

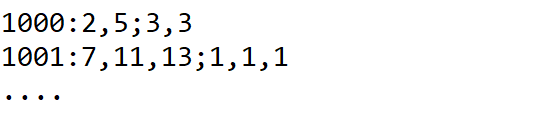
1) 文件数据读取：掌握文件的打开(open)以及打开模式（’r’,’w’和’a’）、读（read、readline、readlines）、写（write、writelines）方式，并学会with …. as句法，以及数据类型的转化、格式化处理和保存。

2) 文件管理：学会应用python的os包以及os.path模块管理文件系统。

3) 数据永久存储：pickle模块的dump和load方法。

**2. 实验练习题**

(1) 将1000~2000之间的正整数质因子分解结果保存到文本文件中。保存格式：每个整数的质因子分解结果占一行，该行开始是这个整数本身，占8个字符位，然后冒号，全部质因子（逗号隔开），然后分号，最后全部质因子的幂（逗号隔开）。文件格式可参考如下：



源代码：

""" you have to specify the """

path\_string="D:/OneDrive - pop.zjgsu.edu.cn/PythonPath/exp3/test.txt"

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):

    # 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

""" handle in range(1000,2000) """

string=""

f=open(path\_string,"w")

for i in range(1000,2000):

    # string=(str)(i)+":"

    f.write(str(i)+":")

    whether\_separator=True

    for list in decompose(i):

        # count\_separator=0

        string\_list=[str(i) for i in list]

        # print (string\_list)

        f.write(",".join(string\_list))

        if (whether\_separator):

            f.write(";")

            whether\_separator=False

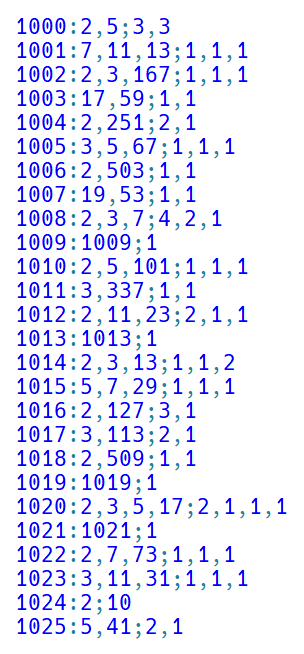
    f.write("\n")

f.close()

源代码截图：



文件内容截图：



(2) 本文文件“info\_stocks.txt”中每一行的name对应股票名称，shares对应多少股，price对应该只股票的单价。

(a) 请计算每只股票的总价，并按总价从大到小进行排序(程序中要求含lambda函数) ，将所有股票的名称、单价、总价依次打印输出。

(b) 要求使用Python内置函数filter过滤出单价大于80的股票有哪些? 并将符合条件的股票名称、单价、总价打印输出。

源代码：

f = open(path\_string\_fix+"info\_stocks.txt", "r")

ls = []

for line in f:

    ls.append(line.strip("\n").split(","))

f.close()

""" test the read result: """

# print(ls)

""" sum the price: """

def calculate\_index(list\_element):

    shares = int(list\_element[1].split(":")[1])

    price = float(list\_element[2].split(":")[1])

    sum\_price = shares\*price

    return sum\_price

ls.sort(key=lambda list\_element: calculate\_index(list\_element), reverse=True)

def print\_iterable(iterable):

     for list\_element in iterable:

        for i in [0, 2]:

            """ to make the print format tidily each column,we specify a width for the middle column: """

            print(list\_element[i].strip().ljust(20), end=" ")

        print("sum\_price="+str(calculate\_index(list\_element)))

# print(ls)

def print\_sort\_result(ls):

    print("\nthe sorted result:")

    print\_iterable(ls)

filter\_ls=filter(lambda list:float(list[2].split(":")[1])>80,ls)

def print\_filter\_result(filter\_ls):

    print("\nthe filtered result:")

    print\_iterable(filter\_ls)

    # for list\_element in filter\_ls:

    #     for i in [0,2]:

    #         print(list\_element[i].strip(),end=" ")

    #     print("sum\_price=%s"%calculate\_index(list\_element))

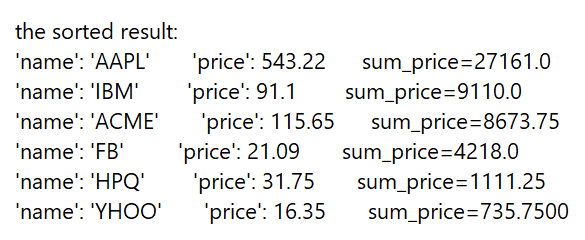
print\_sort\_result(ls)

print\_filter\_result(filter\_ls)

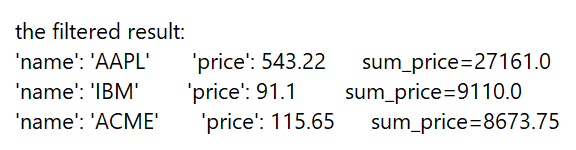
源代码截图：



(a)运行结果截图：



(b)运行结果截图：



# (3)从存储着某商场部分交易数据的文本文件“transactions.txt”中读取数据进行处理。编程实现：

(a)统计该数据文件中每行包含有多少个字符？请打印输出统计结果。

(b)可以观察到，若以逗号“,”为分隔符，该文件中每行含有9列数据。其中第1列表示某次交易事务编号，而第3至第7列为浮点型数据。那么以第4列数据为排序准则，将每行数据从小到大进行排列，保存到一个新的文件中去（文件取名可为’result3b.txt’）。

源代码：

ls = []

def open\_file\_exp3\_toList(fileName):

    with open(path\_string\_fix+fileName, "r") as file\_input\_stream:

        for line in file\_input\_stream:

            ls.append(line.strip("\n").split(","))

def calculate\_size(ls):

    size=0

    for item in ls:

        for word in item:

            size+=len(word)

    return size

""" out put the sorted result: """

open\_file\_exp3\_toList("transactions.txt")

ls.sort(key=lambda list\_element: float(list\_element[3]))

""" use the good structure to execute the i/o operations:with as file\_stream """

# print(ls[0])

with open(path\_string\_fix+"result3b.txt","w") as f\_out:

    print(ls[0])

    for list\_element in ls:

        string=""

        for word in list\_element:

            string+=(word+",")

        f\_out.write(string+"\n")

# print(ls)

""" open the stransactions.txt """

with open(path\_string\_fix+"transactions.txt","r") as file\_input\_stream:

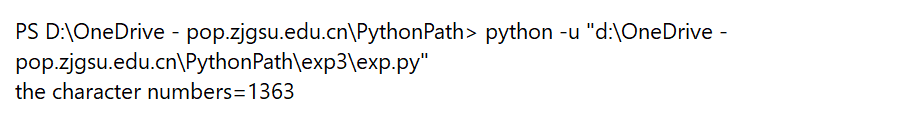
    ls=file\_input\_stream.readlines()

print("the character numbers=%d"%calculate\_size(ls))

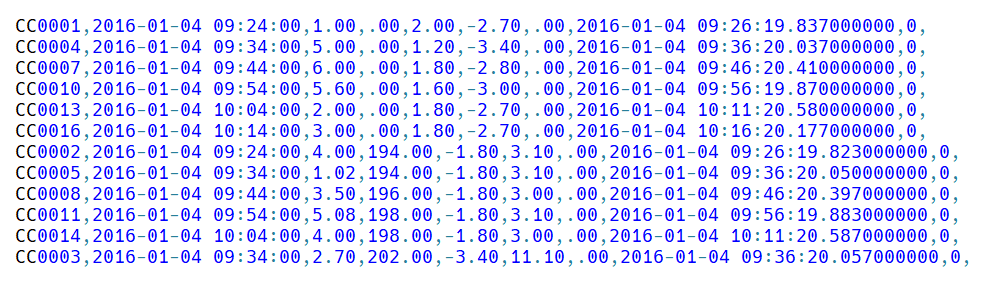
源代码截图：



(a)运行结果截图：

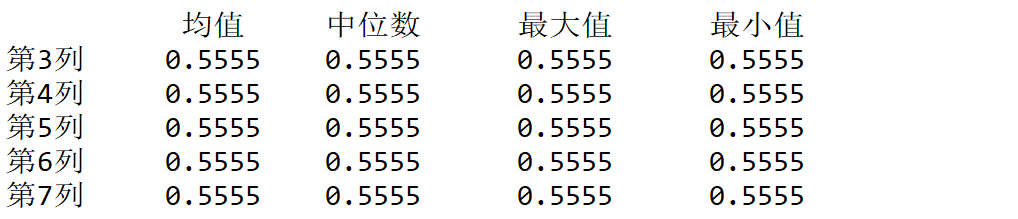


(b)文件内容截图：



(4) 数据来源同题(3), 从文件“transactions.txt”中读取数据进行处理。请计算第3至第7列中每列数据的均值、中位数、最大值和最小值，并将计算结果保留小数点后4位，保存到新的文件中（创建的新文件取名可为’result4.txt’）。

注：本题要求利用def自行定义计算均值、中位数的函数（形参为一个列表），不得调用任何python库函数进行直接计算，更不得使用numpy库中的函数方法。保存的文件格式参考如下：



源代码：

open\_file\_exp3\_toList("transactions.txt")

def get\_num\_list\_sorted(ls,i):

    """ i=2,3,4,5,6(i from 0) """

    get\_num\_list=[]

    for list\_element in ls:

        get\_num\_list.append(float(list\_element[i]))

    get\_num\_list.sort()

    return get\_num\_list

def get\_middle(list):

    return list[len(list)//2]

def get\_average(list):

    return sum(list)/len(list)

def calculate(i):

    # string=""

    num\_list=[]

    # for i in range(2,7):

    num\_list=get\_num\_list\_sorted(ls,i)

    calculated\_list=[]

    calculated\_list.append(get\_average(num\_list))

    calculated\_list.append(get\_middle(num\_list))

    calculated\_list.append(num\_list[-1])

    calculated\_list.append(num\_list[0])

    return calculated\_list

def get\_string(list):

    string=""

    for item in list:

        string+=str("%.4f"%item).ljust(20)

    return string

with open(path\_string\_fix+"result4.txt","w") as file\_output\_stream:

    file\_output\_stream.write("".ljust(20)+"均值".ljust(20)+"中位数".ljust(20)+"最大值".ljust(20)+"最小值".ljust(20)+"\n")

    i=3

    for i in range(2,7):

        num\_list=[]

        file\_output\_stream.write(("第%d列"%(i+1)).ljust(20)+get\_string(calculate(i))+"\n")

open\_file\_exp3\_toList("transactions.txt")

def get\_num\_list\_sorted(ls,i):

    """ i=2,3,4,5,6(i from 0) """

    get\_num\_list=[]

    for list\_element in ls:

        get\_num\_list.append(float(list\_element[i]))

    get\_num\_list.sort()

    return get\_num\_list

def get\_middle(list):

    return list[len(list)//2]

def get\_average(list):

    return sum(list)/len(list)

def calculate(i):

    # string=""

    num\_list=[]

    # for i in range(2,7):

    num\_list=get\_num\_list\_sorted(ls,i)

    calculated\_list=[]

    calculated\_list.append(get\_average(num\_list))

    calculated\_list.append(get\_middle(num\_list))

    calculated\_list.append(num\_list[-1])

    calculated\_list.append(num\_list[0])

    return calculated\_list

def get\_string(list):

    string=""

    for item in list:

        string+=str("%.4f"%item).ljust(20)

    return string

with open(path\_string\_fix+"result4.txt","w") as file\_output\_stream:

    file\_output\_stream.write("".ljust(20)+"均值".ljust(20)+"中位数".ljust(20)+"最大值".ljust(20)+"最小值".ljust(20)+"\n")

    i=3

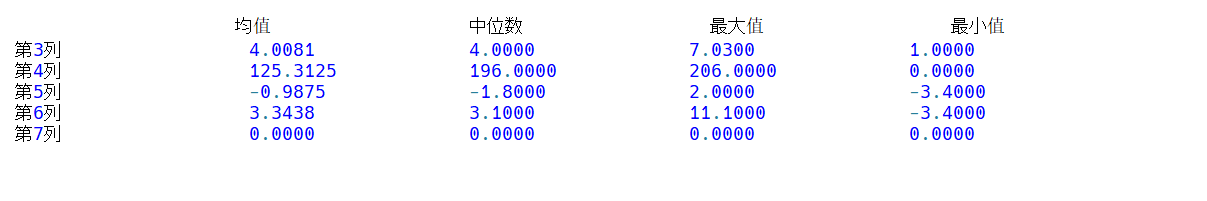
    for i in range(2,7):

        num\_list=[]

        file\_output\_stream.write(("第%d列"%(i+1)).ljust(20)+get\_string(calculate(i))+"\n")

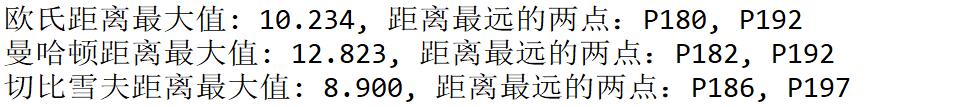
源代码截图：

文件内容截图：



(5) 文件“points.txt”保存了三维空间中100个点的坐标信息。若以逗号为分隔符，第1列为数据点编号，最后3列数据依次为x、y、z轴坐标。请计算任意两点之间的欧氏距离、曼哈顿距离和切比雪夫距离，分别找出这三种距离最大值，以及达到最大值时对应的两点，并将距离最大值相关数据保存至一个新的文件中去（创建的新文件取名可为’result5.txt’）。

注：以上三种距离的数学定义请自学(如：<https://blog.csdn.net/XZHSakura/article/details/107694724>)。编写Python代码时，要求利用def给出这三种距离的函数定义，且不能调用任何库函数进行直接计算。距离最大值保留小数点后3位，保存文件的格式参考如下：



源代码：

open\_file\_exp3\_toList("points.txt")

def Euclid\_distance(point\_list1, point\_list2):

    sum = 0

    for i in range(0, 3):

        sum += (point\_list1[i]-point\_list2[i])\*\*2

    return sum\*\*0.5

def Manhattan\_distance(point\_list1, point\_list2):

    sum = 0

    for i in range(0, 3):

        sum += abs(point\_list1[i]-point\_list2[i])

    return sum

def Chebyshev\_distance(point\_list1, point\_list2):

    list = []

    for i in range(0, 3):

        list.append(abs(point\_list1[i]-point\_list2[i]))

    return max(list)

""" fetch all points """

def get\_points\_list(ls):

    # num\_list=[]

    """ turn the digit strings to numbers """

    for list\_element in ls:

        for i in range(1, 4):

            list\_element[i] = float(list\_element[i])

    # return list\_element[1:4]

    """ append the three kinds of distances:"""

        # num\_list.append(float(list\_element[i]))

def max\_instance(point\_list1, point\_list2):

    list = []

    list.append(Euclid\_distance(point\_list1, point\_list2))

    list.append(Manhattan\_distance(point\_list1, point\_list2))

    list.append(Chebyshev\_distance(point\_list1, point\_list2))

    return max(list)

""" pairwise calculated points   """

get\_points\_list(ls)

Euclid\_distance\_list = []

Manhattan\_distance\_list = []

Chebyshev\_distance\_list = []

# point\_coordinate\_list1=[]

# point\_coordinate\_list2=[]

p1=[]

p2=[]

p\_No1=""

p\_No2=""

offset=0

for point\_list1 in ls:

    # for point\_list2 in ls:

    offset+=1

    for i in range(offset,len(ls)):

        p1,p2=point\_list1[1:4], ls[i][1:4]

        p\_No1,p\_No2=point\_list1[0],ls[i][0]

        Euclid\_distance\_list.append( [Euclid\_distance(p1,p2),(p\_No1,p\_No2)])

        Manhattan\_distance\_list.append( [Manhattan\_distance(p1,p2),(p\_No1,p\_No2)])

        Chebyshev\_distance\_list.append([Chebyshev\_distance(p1,p2),(p\_No1,p\_No2)])

Euclid\_distance\_list.sort(key=lambda list:list[0])

Manhattan\_distance\_list.sort(key=lambda list:list[0])

Chebyshev\_distance\_list.sort(key=lambda list:list[0])

# print(Euclid\_distance\_list)

with open(path\_string\_fix+"result5.txt","w") as file\_output\_stream:

    Euclid\_distance\_max=Euclid\_distance\_list[-1]

    Manhattan\_distance\_max=Manhattan\_distance\_list[-1]

    Chebyshev\_distance\_max=Chebyshev\_distance\_list[-1]

    file\_output\_stream.write("欧氏距离最大值:%.3f"%Euclid\_distance\_max[0]+", 距离最远的两点:%s,%s"%(Euclid\_distance\_max[1][0],Euclid\_distance\_max[1][1])+'\n')

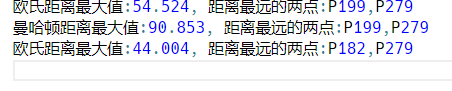
    file\_output\_stream.write("曼哈顿距离最大值:%.3f"%Manhattan\_distance\_max[0]+", 距离最远的两点:%s,%s"%(Manhattan\_distance\_max[1][0],Manhattan\_distance\_max[1][1])+'\n')

    file\_output\_stream.write("欧氏距离最大值:%.3f"%Chebyshev\_distance\_max[0]+", 距离最远的两点:%s,%s"%(Chebyshev\_distance\_max[1][0],Chebyshev\_distance\_max[1][1])+"\n")

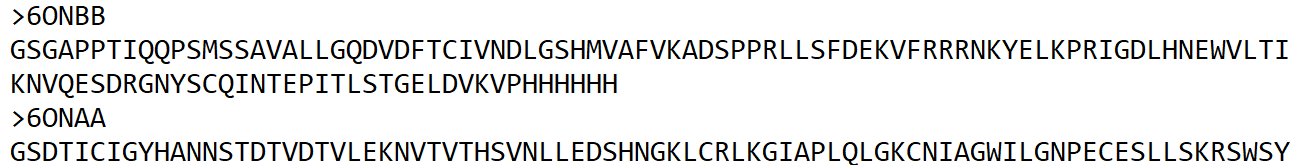
源代码截图：



文件内容截图：



# (6)文件” seqs\_fasta.txt”保存着具有fasta格式的100多个蛋白质氨基酸序列，截取部分如下图所示：

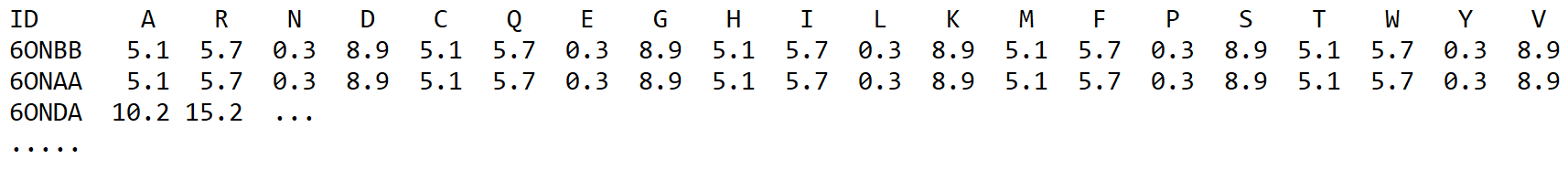


其中每条序列以“>”开头，紧接的“6ONBB”表示蛋白质序列ID，两行“>6ONBB”与“>6ONAA”之间展示的是ID为6ONBB的氨基酸序列。

自然界中，任何天然蛋白质主要由20种标准的氨基酸构成，可由相应的20个英文字母aa20={'A', 'R', 'N', 'D', 'C', 'Q', 'E', 'G', 'H', 'I', 'L', 'K', 'M', 'F', 'P', 'S', 'T', 'W', 'Y', 'V'}表示。同一种氨基酸在一条蛋白质序列中可能出现多次。

请计算文件“seqs\_fasta.txt”中每条蛋白质序列的氨基酸组分，即字母表aa20中的每个英文字母在该蛋白质序列中出现的频率（百分比%），并将统计结果保存至一个新的文件（创建的新文件可取名为’result6.txt’）。

注：要求氨基酸出现的频率值保留小数点后1位，且每个氨基酸所在列的频率值数据右对齐保存，保存文件的格式参考如下：



源代码：

count\_result\_string = "ID".rjust(6)

count\_dict = {'A': 0, 'R': 0, 'N': 0, 'D': 0, 'C': 0, 'Q': 0, 'E': 0, 'G': 0, 'H': 0,

              'I': 0, 'L': 0, 'K': 0, 'M': 0, 'F': 0, 'P': 0, 'S': 0, 'T': 0, 'W': 0, 'Y': 0, 'V': 0}

for character in count\_dict.keys():

    count\_result\_string += character.rjust(6)

count\_result\_string += "\n"

with open(path\_string\_fix+"seqs\_fasta.txt", "r") as file\_input\_stream:

    sequences\_list = []

    sequences\_list = file\_input\_stream.read().split(">")

    sequences\_list.pop(0)

    # sequence\_list=[sequence\_list for sequence in sequences\_list ]

    # print(string)

    result\_dict = {}

    for sequence in sequences\_list:

        count\_dict = {'A': 0, 'R': 0, 'N': 0, 'D': 0, 'C': 0, 'Q': 0, 'E': 0, 'G': 0, 'H': 0,

                      'I': 0, 'L': 0, 'K': 0, 'M': 0, 'F': 0, 'P': 0, 'S': 0, 'T': 0, 'W': 0, 'Y': 0, 'V': 0}

        sequence = sequence.split("\n")

        ID\_string = ""

        ID\_string = sequence.pop(0)

        size=0

        """ count a sequence: """

        result\_dict[ID\_string] = count\_dict

        for segment in sequence:

            size+=len(segment)

            for character in segment:

                count\_dict[character] += 1

        """ make up all counted result(accumulate) """

        count\_result\_string += ID\_string+" "

        for character in count\_dict.keys():

            count\_result\_string += str(round(count\_dict[character]/size\*100,1)).rjust(6)

            # print(count\_result\_string)

        count\_result\_string += "\n"

print(count\_result\_string)

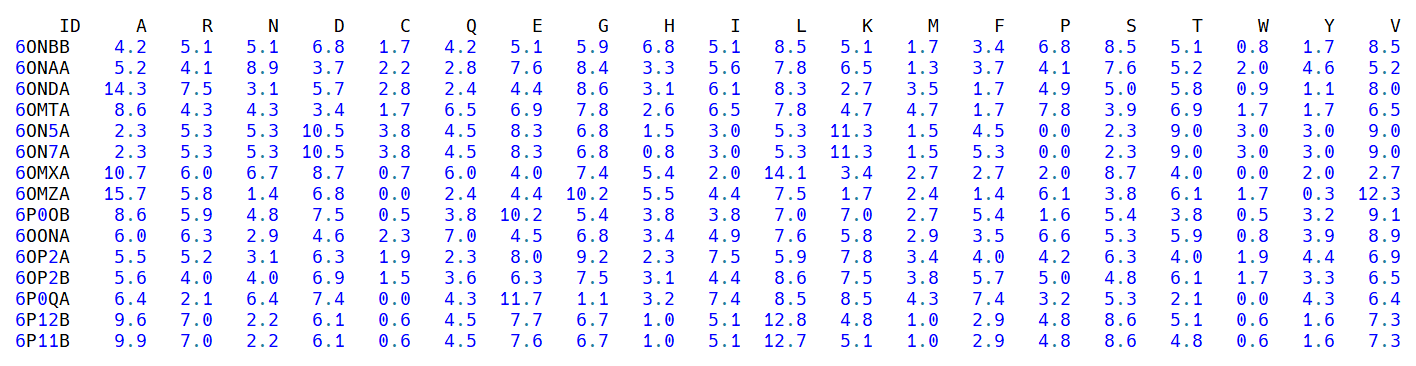
with open(path\_string\_fix+"result6.txt","w") as file\_output\_stream:

    file\_output\_stream.write(count\_result\_string)

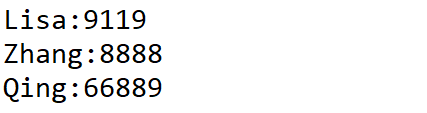
源代码截图：



文件内容截图：



(7)系统登录模拟。先手工制作一个文本文件‘account.txt’，内容包含以下3位用户的账号和密码：



然后，通过input()输入某位用户的账号和密码进行系统登录的模拟，要求将每次登录的信息记录在一个日志文件”log.txt”中，包含输入的账号与登录的具体时间，中间用逗号隔开。

另外，如果输入的账号和密码不与‘account.txt’文件中的任何一个相同，利用print()函数给出提示：“您的账号或密码有误！”；

如果连续五次登录失败，提示：“您的账号将被锁定！”。

源代码：

def get\_formatted\_time():

    return (time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))

account, pwd = "", ""

items\_list = []

def get\_accounts\_existed():

    with open(path\_string\_fix+"account.txt", "r") as file\_input\_stream:

        for item in file\_input\_stream:

            items\_list.append(item.strip().split(":"))

# print(items\_list)

def log\_record(account,pwd):

    with open(path\_string\_fix+"log.txt", "a") as file\_output\_stream:

        file\_output\_stream.write(account+","+pwd+",time:"+get\_formatted\_time()+"\n")

def login():

    for i in range(5):

        get\_accounts\_existed()

        account, pwd = (input("input account:")), (input("input password:"))

        log\_record(account,pwd)

        if [account, pwd] in items\_list:

            print("welcome! "+account)

            return

        else:

            print("您的账号或密码有误！")

        # if i==4:

        #     print("您的账号将被锁定！")

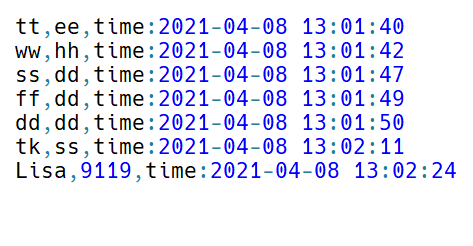
    print("您的账号将被锁定！")

login()

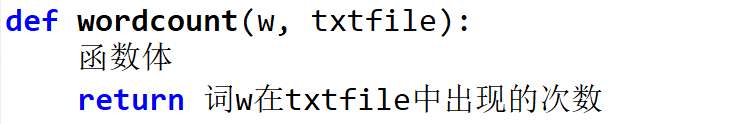
源代码截图：



运行结果截图：



(8) 首先，定义一个名为“wordcount”的函数，功能为统计中文文本中某个关键字出现的次数，函数原型如下：

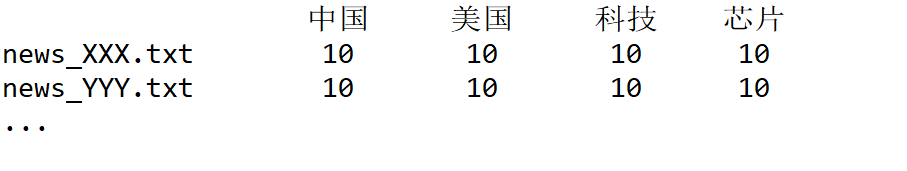


其中w和txtfile均为字符串。

其次，在存放本次实验材料的文件夹中，利用os.mkdir()创建一个新的文件夹，取名“mydir”；同时，自动识别出以“news\_”开头的所有文本文件，将其移动至新建的文件目录“mydir”中(注：需编程自动实现移动文件)。

进一步，利用pickle模块将函数wordcount以及识别出的以“news\_”开头的所有文本文件名组合成一个列表，永久保存至文件“wc.pkl”，并存储在文件夹“mydir”中。

最后，再次利用pickle模块将保存在“wc.pkl”中的列表数据载入，获得函数wordcount，并调用wordcount计算四个关键字“中国”、“美国”、“科技”和“芯片”在以“news\_”开头的所有文本文件中出现的次数，打印输出，格式参考如下：



源代码：

import os

import shutil

path\_src = path\_string\_fix

path\_des = path\_string\_fix+"mydir/"

""" create the dir mydir in the proper source path """

if not os.path.exists(path\_des):

    os.mkdir(path\_src+"mydir")

""" get the files in the path: """

files\_list = os.listdir(path\_src)

""" get the files start with news\_: """

file\_news\_list=os.listdir(path\_des)[:2]

""" move files from source path to destination path:"""

def move\_news():

    for file\_name in files\_list:

        if file\_name.startswith("news\_"):

            # print(file\_name)

            shutil.move(path\_src+file\_name,path\_des)

""" count the word in specified file """

def wordcount(w,txt\_file):

    """the frequency of appearance of word w in the file txt\_file(attention ,the txt\_file use the absolute path)

    !attention2:the function read files which is encode in gbk,so the open() use the encoding="gbk"(gb18030 is ok too) to read it correctly

    Args:

        w (str): [description]

        txt\_file (str): [absolute path]

    """

    # list=[]

    string=""

    with open(txt\_file,"r",encoding='gbk') as file\_input\_stream:

        string= file\_input\_stream.read()

        # print(string)

    return string.count(w)

# print(wordcount("t",path\_src+"log.txt"))

""" use(experience the serialize module pickle too store(dump) and use the object serialized:) """

def pickle\_deal():

    # obj\_list=obj\_list

    with open(path\_des+"wc.pkl","wb") as file\_output\_stream:

        pickle.dump((wordcount,file\_news\_list),file\_output\_stream)

    with open(path\_des+"wc.pkl","rb") as file\_input\_stream:

        return pickle.load(file\_input\_stream)

# print(obj\_list)

def print\_head(word\_list):

    for i in [""]+word\_list:

        print(i.center(20),end="")

    print()

def print\_result(word\_list):

    print\_head(word\_list)

    for file in obj\_list[1]:

        print(file.center(20),end="")

        file\_full\_path=path\_des+file

        for word in word\_list:

            frequency=0

            frequency=wordcount(word,file\_full\_path)

            frequency=str(frequency).center(20)

            print(frequency,end="")

        print()

word\_list=["中国","美国","科技","芯片"]

move\_news()

obj\_list=pickle\_deal()

"get the function from pickled file"

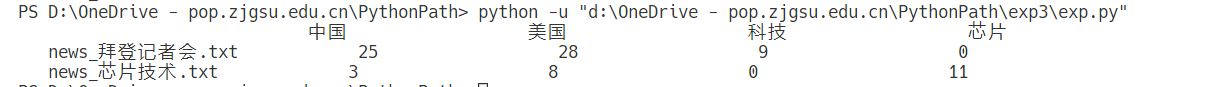
wordcount=obj\_list[0]

print\_result(word\_list)

源代码截图：



运行结果截图：



文件夹mydir截图：

