jupyter H.W. 3 Mohammed Al Salem Last Checkpoint: 04/15/2022 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Edit Metadata

ICS 104 Homework #3 (Due Saturday April 9, 2022 at midnight)

- Please note the following regarding homework submissions:
  - No email submissions will be accepted under any circumstances.
  - The deadline for all homework assignments will be at midnight.
  - The submission after the deadline will be opened for 6 hours (i.e. until 6:00am next morning) and submissions will be marked as Late. However, you will not lose any marks because of that.
  - After 6:00am, submission will be closed. Failure to submit before that time results in an automatic zero.

Instructions:

- Solve Questions 1 and 2 in the respective cells below. **Make sure you fill in any place that says `YOUR CODE HERE`.**
- **Make sure that your program output matches the sample runs in all Questions.**
- Also, make sure that you use the **same strings in the print statements** of all questions.
- After you are done writing your code, save your jupyter notebook and close it. Then submit it from its folder to Blackboard
- The grade distribution is as follows:
  - Question 1:
    - Correct Code: 55 points.
    - Programming style (comments and variable names): 5 points
  - Question 2:
    - Correct Code: 30 points.
    - Programming style (comments and variable names): 5 points
    - Proper use of functions: 5 points.

Note:

- You should write comments to help other programmers understand your code.
- Do not use magic numbers and define the constants value as a constant variable.
- Name your variables based on Variable Naming Conventions rules.

Edit Metadata

Question 1 (60 points)

Write a program that read a real blood donation database `donations.txt` that has been downloaded from Kaggle data science community website.The `donations.txt` file contains blood donation on individuals who donate blood.This data is used by the hospital to communicate with donors, record their donation and test result details.

```
Donor_ID,Donor_Age,Donation_Date,Gender,Blood_Group,
Donor_Weight,Donor_Hemoglobin,Donor_Blood_Pressure
102493,38,2/22/2016,M,B-,80,13,120/80
122646,25,8/29/2016,M,A+,60,13,120/80
131869,44,11/13/2016,M,A+,80,13,120/80
137470,50,1/6/2017,M,B+,75,14,120/80
182717,21,5/8/2018,M,A-,65,14,120/80
126199,32,9/28/2016,M,A+,85,13,120/80
```

First column is Donor ID, and the remaining columns are Donor Age,Donation Date, Gender, Blood Group, Donor Weight, Donor Hemoglobin, and Donor Blood Pressure. Your program needs to read all those data from the file and store it into a dictionary whose keys are Donor ID and whose values are a list of donation details. This data is to be processed to produce an output file with the name `donations_report.txt`. This file should format the donations as follows:

```
------------------------------------------------------------------------------------------------------------------------
Donor ID    Donor Age    Donation Date    Gender    Blood Group    Donor Weight    Donor Hemoglobin    Donor Blood Pressure
------------------------------------------------------------------------------------------------------------------------
102493         38          2/22/2016        M            B-             80                13                 120/80
122646         25          8/29/2016        M            A+             60                13                 120/80
131869         44          11/13/2016       M            A+             80                13                 120/80
.
.
.
170844         32          12/20/2017       M            B+             70                13                 120/80
155622         30          7/13/2017        M            A+             70                13                 120/80

########## Blood Donations statistics ##########

Blood Type  Number of donation

    B-            40
    A+           105
    B+           116
    A-           121
    O-             6
    AB-            4
    AB+            9
    O+             9

##########    End of report    ##########
```

Note:

- You must read all of the donation records before the report can be generated.
- Handles all exceptions when dealing with files.

Sample Run

```
Enter file name : donations.txt
Generating the Blood donations report donations_report.txt
```

In [1]:

Edit Metadata

```python
##
# This program reads and processes a collection of blood donations and prints the blood donations report
# in the tabular format
#
# asking the user for the input file name
inputFileName = input("Enter file name: ")
# Validating file name
found = False
while not found:
    if("donations.txt" in inputFileName):
        print("Generating the Blood donations report donations_report.txt")
        found = True
    else:
        inputFileName = input("Invalid file extension. Please re-enter the input filename: ")
try:
    # we open the file containing the data for reading and output file for writing
    infile = open(inputFileName, 'r')
    outfile = open("donations_report.txt", 'w')
    # Reading first line that contains the 8 catigories
    line1 = infile.readline()
    # Split funtion will change each "," into a list staring from index 0 which is the ID to 7
    firstLine = line1.strip().split(",")
    # to remove all _ to spaces
    for i in range(len(firstLine)):
        firstLine[i] = firstLine[i].replace('_'," ")
    # Writing file
    outfile.write(("-"*125)+"\n")
    outfile.write("{:<13} {:<13} {:<20} {:<13} {:<13} {:<13} {:<17} {:<17}\n".format(firstLine[0],firstLine[1]\
                  ,firstLine[2],firstLine[3],firstLine[4],firstLine[5],firstLine[6],firstLine[7]))
    outfile.write(("-"*125)+"\n")
    data = dict()
    # to iterate the lines
    for lines in infile:
        line = lines.strip().split(",")
        outfile.write("{:<15} {:<15} {:<20} {:<15} {:<15} {:<15} {:<20} {:<20}\n".format(line[0],line[1],line[2],line[3]\
                      ,line[4],line[5],line[6],line[7]))
        # the key is the ID
        data[line[0]] = [line[1],line[2],line[3],line[4],line[5],line[6],line[7]]

    # writing data analysis into file
    outfile.write("\n########## Blood Donations statistics ##########\n")
    # Now we define a function to get all the statistics of the input file
    def countBlood(file):
        #small a,b, and ab indicate -
        #capital A,B, and AB indicate +
        counta = 0 ;countA = 0; countb = 0; countB = 0; counto =0;countO=0;countab=0;countAB=0



        for line in file:
            data = line.split(",")
            if data[4] == "B-":
                countb += 1
            if data[4] == "B+":
                countB += 1
            if data[4] == "A-":
                counta += 1
            if data[4] == "A+":
                countA += 1
            if data[4] == "O-":
                counto += 1
            if data[4] == "O+":
                countO += 1
            if data[4] == "AB+":
                countAB += 1
            if data[4] == "AB-":
                countab += 1

        return [str(counta),str(countA), str(countb), str(countB), str(counto),str(countO),str(countab),str(countAB)]
    # we reopen the file to start reading from the beggining again...
    newReadFile = open("donations.txt", "r")
    g = countBlood(newReadFile)
    outfile.write("\nBlood Type          Number of Donations\n")
    outfile.write("B-     %28s\n"%g[2])
    outfile.write("A+     %28s\n"%g[1])
    outfile.write("B+     %28s\n"%g[3])
    outfile.write("A-     %28s\n"% g[0])
    outfile.write("O-     %27s\n"%g[4])
    outfile.write("AB-     %26s\n"%g[6])
    outfile.write("AB+     %26s\n"%g[7])
    outfile.write("O+     %27s\n"%g[5])
    outfile.write("\n##########        End of report        ##########\n")




except Exception as error:
    print(error)
finally:
    infile.close()
    outfile.close()

```

```
Enter file name: donations.txt
Generating the Blood donations report donations_report.txt
```

Edit Metadata

Question 2 (40 points)

In statistics, Pearson correlation coefficient is a measure of the linear correlation between two sets of data. Write a python program that calculates the Pearson's correlation coefficient.

Given two sequences of numbers $x = [x_0, x_1, x_2, \ldots, x_{n-1}]$ and $y = [y_0, y_1, y_2, \ldots, y_{n-1}]$, it can be defined mathematically as:

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

where:

- n is sample size.
- x_i, y_i are the individual sample points indexed with i.
- $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ (the sample mean); and analogously for $\bar{y}$

The program should have the following user-defined functions.

- Write the `main` function that asks the user for file names that contain the two sequences of real values. The function call `readInput` and `correlation` functions to read data from files and calculates the Pearson correlation coefficient.
- Write a function `readData(filename)` that take as input a string representing a `file name`. The function opens the file (its name is given as a parameter) and reads all values in the file into two `lists`. Then, the function returns the `lists`. If the reading is not successful, it prints an error message and terminates.
- Write a function `correlation(list1, list2)` that receives two lists of real values, then it computes correlation between two values as defined by the above formula.

Note:

- Your functions must be general and not specific to the given example.
- It is not allowed to use **global variables** and any **external library** to calculate the correlation.
- Handles all exceptions when dealing with files

Sample Run

- Sample input file: `data.txt`

```
20.5   30.5
24.6   20.7
17.4   27.3
```

- **Sample Run#1** (`data1.txt` file does not exist)

```
Enter the input file name: data1.txt
Error reading data: the input file data1.txt is not found
```

- **Sample Run#2**

```
Enter the input file name: data.txt
Pearsons correlation: -0.718
```

In [1]:

Edit Metadata

```python
##
# This program  calculates the Pearson's correlation coefficient between two sets of data.
#
##
# This program  calculates the Pearson's correlation coefficient between two sets of data.
#we import square root since we need it in the formula
from math import sqrt
def main():
    try:
        inputFileName = input("Enter the name of the file: ")
        firstlist, secondlist = readInput(inputFileName) # a function that give us two lists
        coeff = correlation(firstlist,secondlist)
        print("Pearsons correlation:%.3f" % coeff)
    except IOError: #input error entered by the user
        print("Error reading data: the input file", inputFileName, "is not found")
def readInput(file):
    infile = open(file,"r")
    list1 = []
    list2 = []
    for line in infile:
        data1,data2 = line.strip().split()
        data1 = float(data1)
        data2 = float(data2)
        list1.append(data1) # to add the values at the end of the list
        list2.append(data2)
    infile.close()
    return list1,list2
def correlation(list1,list2):
    n = len(list1)
    z = len(list2)
    xtotal = 0
    ytotal = 0
    for i in range(n):
        xtotal = xtotal + list1[i]
    xavg = (1/n) * xtotal
    for j in range(z):
        ytotal = list2[j] + ytotal
    yavg=(1/n)*ytotal
    rnewmonator = 0
    rdenominator = 0
    xdenominator = 0
    ydenominator = 0
    for al in range(n):
        rnewmonator = (list1[al] - xavg) *(list2[al] - yavg) + rnewmonator
        xdenominator = ((list1[al] - xavg) **2) + xdenominator
        ydenominator = ((list2[al] - yavg) **2) + ydenominator
        rdenominator = sqrt(xdenominator*ydenominator)
    finalAnswer = rnewmonator/rdenominator # final answer = r
    return finalAnswer

main()
```

```
Enter the name of the file: temp.txt
Pearsons correlation:-0.718
```

In [ ]:

Edit Metadata

```python
#NOTE: I got 95.5 in this HW so there could be better ways to solve this HW
# GOOD LUCK
```