Rajalakshmi Engineering College

Name: Rittvik S
Email: 240701616@rajalakshmi.edu.in
Roll no: 240701616
Phone: null
Branch: REC
Department: I CSE FF
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 6_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

In the enchanted realm of Academia, you, the Academic Alchemist, are bestowed with a magical quill and a parchment to weave the grades of aspiring students into a tapestry of academic brilliance.

The mission is to craft a Python program that empowers faculty members to enter student grades for any two subjects, stores these magical grades in a mystical file, and then, with a wave of your virtual wand, calculates the GPA to unveil the true essence of academic achievement.

Input Format

The input format is a string representing the student's name, any two subjects, and corresponding grades.

After entering grades, they can type 'done' when prompted for the student's name.

Output Format

The output should display the (average of grades) calculated GPA with a precision of two decimal places.

The magical grades will be saved in a mystical file named "magical_grades.txt".

Refer to the sample output for format specifications.

Sample Test Case

Input: Alice
Math
95
English
88
done

Output: 91.50

Answer

```
with open("magical_grades.txt", "a") as file:
   while True:

      student_name = input().strip()

      if student_name.lower() == "done":
         break

      subject1 = input().strip()
      grade1 = float(input().strip())
      subject2 = input().strip()
      grade2 = float(input().strip())

      gpa = (grade1 + grade2) / 2

      file.write(f"{student_name}\n")
      file.write(f"{subject1}: {grade1}\n")
```

```
    file.write(f"{subject2}: {grade2}\n")
    file.write(f"GPA: {gpa:.2f}\n\n")

    print(f"{gpa:.2f}")
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Alice is developing a program called "Name Sorter" that helps users organize and sort names alphabetically.

The program takes names as input from the user, saves them in a file, and then displays the names in sorted order.

File Name: sorted_names.txt.

Input Format

The input consists of multiple lines, each containing a name represented as a string.

To end the input and proceed with sorting, the user can enter 'q'.

Output Format

The output displays the names in alphabetical order, each name on a new line.

Refer to the sample output for the formatting specifications.

Sample Test Case

```
Input: Alice Smith
John Doe
Emma Johnson
q
Output: Alice Smith
Emma Johnson
John Doe
```

Answer

```
names = []

while True:
   name = input().strip()
   if name.lower() == "q":
      break
   names.append(name)

names.sort()

with open("sorted_names.txt", "w") as file:
   for name in names:
      file.write(name + "\n")

for name in names:
   print(name)
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

A shopkeeper is recording the daily sales of an item for N days, where the price of the item remains the same for all days. Write a program to calculate the total sales for each day and save them in a file named sales.txt that can store the data for a maximum of 30 days. Then, read the file and display the total earnings for each day.

Note: Total Earnings for each day = Number of Items sold in that day × Price of the item.

Input Format

The first line of input consists of an integer N, representing the number of days.

The second line of input consists of N space-separated integers representing the number of items sold each day.

The third line of input consists of an integer M, representing the price of the item that is common for all N days.

Output Format

If the number of days entered exceeds 30 (N > 30), the output prints "Exceeding limit!" and terminates.

Otherwise, the code reads the contents of the file and displays the total earnings for each day on separate lines.

Contents of the file: The total earnings for N days, with each day's earnings appearing on a separate line.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
5 10 5 0
20

Output: 100
200
100
0

Answer

```
N = int(input().strip())

if N > 30:
   print("Exceeding limit!")
else:

   items_sold = list(map(int, input().strip().split()))

   M = int(input().strip())

   earnings = [items * M for items in items_sold]
```

```
with open("sales.txt", "w") as file:
    for earning in earnings:
        file.write(str(earning) + "\n")

with open("sales.txt", "r") as file:
    for line in file:
        print(line.strip())
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

Bob, a data analyst, requires a program to automate the process of analyzing character frequency in a given text. This program should allow the user to input a string, calculate the frequency of each character within the text, save these character frequencies to a file named "char_frequency.txt," and display the results.

Input Format

The input consists of the string.

Output Format

The first line prints "Character Frequencies:".

The following lines print the character frequency in the format: "X: Y" where X is the character and Y is the count.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: aaabbbccc

Output: Character Frequencies:
```
a: 3
b: 3
c: 3
```

Answer

```
from collections import OrderedDict

def char_frequency_analyzer():
    text = input()

    freq = OrderedDict()
    for char in text:
        freq[char] = freq.get(char, 0) + 1

    with open("char_frequency.txt", "w") as file:
        file.write("Character Frequencies:\n")
        print("Character Frequencies:")
        for char, count in freq.items():
            line = f"{char}: {count}"
            print(line)
            file.write(line + "\n")

char_frequency_analyzer()
```

Status : Correct ***Marks : 10/10***