Rajalakshmi Engineering College

Name: Sujit S
Email: 240701545@rajalakshmi.edu.in
Roll no: 240701545
Phone: 9444081254
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 : 36.5

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

```
# You are using Python
grades = []

with open("magical_grades.txt", "w") as file:
    while True:
        name = input().strip()
        if name.lower() == "done":
            break

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

        try:
            g1 = float(grade1)
            g2 = float(grade2)
            if not (0 <= g1 <= 100 and 0 <= g2 <= 100):
```

```
            continue
        except:
            continue

        file.write(f"{name},{subject1},{g1},{subject2},{g2}\n")
        grades.append(g1)
        grades.append(g2)

if grades:
    gpa = sum(grades) / len(grades)
    print(f"{gpa:.2f}")
else:
    print("0.00")
```

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

2. 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

```
# You are using Python
N = int(input())
if N > 30:
    print("Exceeding limit!")
    exit()

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

with open("sales.txt", "w") as f:
    for i in range(N):
        total = items_sold[i] * M
        f.write(str(total) + "\n")

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

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

3. 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

```
# You are using Python
names = []
```

```
while True:
    name = input().strip()
    if name.lower() == 'q':
        break
    if 3 <= len(name) <= 30:
        names.append(name)

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

names.sort()

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

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

4. Problem Statement

Alex is creating an account and needs to set up a password. The program prompts Alex to enter their name, mobile number, chosen username, and desired password. Password validation criteria include:

Length between 10 and 20 characters.At least one digit.At least one special character from !@#$%^&* set. Display "Valid Password" if criteria are met; otherwise, raise an exception with an appropriate error message.

Input Format

The first line of the input consists of the name as a string.

The second line of the input consists of the mobile number as a string.

The third line of the input consists of the username as a string.

The fourth line of the input consists of the password as a string.

Output Format

If the password is valid (meets all the criteria), it will print "Valid Password"

If the password is weak (fails any one or more criteria), it will print an error message accordingly.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: John
9874563210
john
john1#nhoj
Output: Valid Password

Answer

```
def check_password_strength(password):
    special_chars = "!@#$%^&*"
    if not any(char.isdigit() for char in password):
        raise Exception("Should contain at least one digit")
    if not any(char in special_chars for char in password):
        raise Exception("It should contain at least one special character")
    if len(password) < 10 or len(password) > 20:
        raise Exception("Should be a minimum of 10 characters and a maximum of
20 characters")
    return "Valid Password"
name = input()
mobile_number = input()
username = input()
password = input()
try:
    print(check_password_strength(password))
except Exception as e:
    print(e)
```

Status : Partially correct ***Marks : 6.5/10***