Rajalakshmi Engineering College

Name: TARUN ANTONY M
Email: 240701556@rajalakshmi.edu.in
Roll no:
Phone: 7338796644
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 : 20

Section 1 : Coding

1. Problem Statement

Implement a program that checks whether a set of three input values can form the sides of a valid triangle. The program defines a function is_valid_triangle that takes three side lengths as arguments and raises a ValueError if any side length is not a positive value. It then checks whether the sum of any two sides is greater than the third side to determine the validity of the triangle.

Input Format

The first line of input consists of an integer A, representing side1.

The second line of input consists of an integer B, representing side2.

The third line of input consists of an integer C, representing side3.

Output Format

The output prints either "It's a valid triangle" if the input side lengths form a valid triangle,

or "It's not a valid triangle" if they do not.

If there is a ValueError, it should print "ValueError: <error_message>".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
4
5
Output: It's a valid triangle

Answer

```
def is_valid_triangle(side1, side2, side3):
    if side1 <= 0 or side2 <= 0 or side3 <= 0:
        raise ValueError("Side lengths must be positive")
    if side1 + side2 > side3 and side1 + side3 > side2 and side2 + side3 > side1:
        return True
    return False

try:
    side1 = int(input())
    side2 = int(input())
    side3 = int(input())
    if is_valid_triangle(side1, side2, side3):
        print("It's a valid triangle")
    else:
        print("It's not a valid triangle")
except ValueError as e:
    print(f"ValueError: {e}")
```

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

2. 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 Counter

def count_character_frequencies(input_string):
    return Counter(input_string)

def save_frequencies_to_file(frequencies, filename="char_frequency.txt"):
    with open(filename, "w") as file:
        file.write("Character Frequencies:\n")
        for char, count in frequencies.items():
            file.write(f"{char}: {count}\n")
```

```
def display_frequencies(frequencies):
    print("Character Frequencies:")
    for char, count in frequencies.items():
        print(f"{char}: {count}")

if __name__ == "__main__":
    input_string = input()
    frequencies = count_character_frequencies(input_string)
    save_frequencies_to_file(frequencies)
    display_frequencies(frequencies)
```

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

3. 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

```
def calculate_gpa(grades):
    return sum(grades) / len(grades)

def save_grades_to_file(student_name, subject1, grade1, subject2, grade2, gpa):
    with open("magical_grades.txt", "a") as file:
        file.write(f"{student_name} - {subject1}: {grade1}, {subject2}: {grade2}, GPA:
{gpa:.2f}\n")

def main():
    while True:
        student_name = input("Enter student's name (or 'done' to finish): ")
        if student_name.lower() == 'done':
            break
        subject1 = input("Enter first subject: ")
        grade1 = float(input(f"Enter grade for {subject1}: "))
        subject2 = input("Enter second subject: ")
        grade2 = float(input(f"Enter grade for {subject2}: "))

        grades = [grade1, grade2]
        gpa = calculate_gpa(grades)
        print(f"GPA: {gpa:.2f}")

        save_grades_to_file(student_name, subject1, grade1, subject2, grade2, gpa)

if __name__ == "__main__":
    main()
```

Status : Wrong **Marks : 0/10**

4. 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

-

Status : *-* ***Marks : 0/10***