Rajalakshmi Engineering College

Name: Samyuktha S
Email: 240701701@rajalakshmi.edu.in
Roll no: 240701701
Phone: 6380226314
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

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

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

2. 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(a,b,c):
   if a<=0 or b<=0 or c<=0:
      raise ValueError("Side lengths must be positive")
   if a+b>c and b+c>a and a+c>b:
      return True
   else:
      return False
a=int(input())
b=int(input())
c=int(input())
try:
   if is_valid_triangle(a,b,c):
        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***

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():
    student_grades={}
    while True:
        name=input()
        if name.lower()=='done':
            break
        try:
            subject1=input()
            grade1=float(input())
            subject2=input()
            grade2=float(input())
            if not(0<=grade1<=100 and 0<=grade2<=100):
                print("Grades should be between 0 and 100.")
                continue
            student_grades[name]=(grade1,grade2)
            with open("magical_grades.txt","a")as f:
                f.write(f"{name},{subject1},{grade1},{subject2},{grade2}\n")
        except ValueError:
            print("Invalid input. Please enter numeric grades.")
    if student_grades:
        total_grades_sum=0
        total_students=0
        for name,grades in student_grades.items():
            average_grade=sum(grades)/len(grades)
            total_grades_sum+=average_grade
            total_students+=1
        if total_students>0:
            gpa=total_grades_sum/total_students
            print(f"{gpa:.2f}")
        else:
            print("No student grades entered.")
    else:
        print("No student grades entered.")
if __name__=="__main__":
```

```
calculate_gpa()
```

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

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

```
def track_daily_sales():
    try:
        n=int(input())
        if n>30:
            print("Exceeding limit!")
            return
        sales_counts=list(map(int, input().split()))
        price=int(input())
        daily_earnings=[]
        with open("sales.txt","w") as f:
            for count in sales_counts:
                earnings=count*price
                daily_earnings.append(earnings)
                f.write(str(earnings)+"\n")
        with open("sales.txt","r") as f:
            for line in f:
                print(line.strip())
    except ValueError:
        print("Invalid input. Please enter numbers only.")
if __name__=="__main__":
    track_daily_sales()
```

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