Rajalakshmi Engineering College

Name: Mathu Methaa K
Email: 240701306@rajalakshmi.edu.in
Roll no: 2116240701306
Phone: 6381399807
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 6_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1.   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

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

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

```python
earnings = [item * M for item in items_sold]


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


with open("sales.txt", "r") as f:
    lines = f.readlines()
    earnings_from_file = [line.strip() for line in lines]

print(" ".join(earnings_from_file))
```

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

2. 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

```python
# You are using Python
def store_grades_and_calculate_gpa():
    filename = "magical_grades.txt"

    with open(filename, 'w') 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())


            if not (0 <= grade1 <= 100 and 0 <= grade2 <= 100):
                print("Invalid grade input. Grades must be between 0 and 100.")
                return


            gpa = (grade1 + grade2) / 2


            file.write(f"{student_name},{subject1},{grade1},{subject2},{grade2}\n")
```

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

store_grades_and_calculate_gpa()
```

Status : Correct   *Marks : 10/10*

3. 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

```python
# You are using Python
def is_valid_triangle(a, b, c):
    if a <= 0 or b <= 0 or c <= 0:
        raise ValueError("Side lengths must be positive")
    # Check triangle inequality
    if a + b > c and a + c > b and b + c > a:
        return True
    else:
        return False

try:
    A = int(input())
    B = int(input())
    C = int(input())

    if is_valid_triangle(A, B, C):
        print("It's a valid triangle")
    else:
        print("It's not a valid triangle")

except ValueError as ve:
    print(f"ValueError: {ve}")
```

Status : Correct   *Marks : 10/10*

4. Problem Statement

Write a program to obtain the start time and end time for the stage event show. If the user enters a different format other than specified, an exception occurs and the program is interrupted. To avoid that, handle the exception and prompt the user to enter the right format as specified.

Start time and end time should be in the format 'YYYY-MM-DD HH:MM:SS'If the input is in the above format, print the start time and end time.If the input does not follow the above format, print "Event time is not in the format "

The first line of input consists of the start time of the event.

The second line of the input consists of the end time of the event.

Output Format

If the input is in the given format, print the start time and end time.

If the input does not follow the given format, print "Event time is not in the format".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2022-01-12 06:10:00
2022-02-12 10:10:12

Output: 2022-01-12 06:10:00
2022-02-12 10:10:12

Answer

```python
# You are using Python
from datetime import datetime

def validate_event_time(time_str):
    try:
        # Try to parse the input string into a datetime object
        datetime.strptime(time_str, "%Y-%m-%d %H:%M:%S")
        return True
    except ValueError:
        return False


# Read input start and end times
start_time = input().strip()
end_time = input().strip()

# Validate input format
if validate_event_time(start_time) and validate_event_time(end_time):
    print(start_time)
```

```
    print(end_time)
else:
    print("Event time is not in the format")
```

Status : Correct   *Marks : 10/10*