Rajalakshmi Engineering College

Name: Keertana Anjimeeti Srihari
Email: 240701252@rajalakshmi.edu.in
Roll no: 240701252
Phone: 9884916024
Branch: REC
Department: I CSE AH
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
def calculate_gpa():
    total_grades = 0
    num_subjects = 0

    with open("magical_grades.txt", "w") as f:
        while True:
            student_name = input()
            if student_name.lower() == 'done':
                break

            f.write(f"Student: {student_name}\n")

            for i in range(2):
                subject = input()
                grade = float(input())

                f.write(f"  {subject}: {grade}\n")
            total_grades += grade
            num_subjects += 1

    if num_subjects > 0:
        gpa = total_grades / num_subjects
        print(f"{gpa:.2f}")
    else:
        print("No grades entered.")

calculate_gpa()
```

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

```
# You are using Python
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
    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 e:
    print(f"ValueError: {e}")
```

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

3. 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 !@#$%^&amp;* 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 validate_password(password):
    has_digit = False
    for char in password:
        if char.isdigit():
            has_digit = True
            break
    if not has_digit:
        raise ValueError("Should contain at least one digit")
    special_characters = "!@#$%^&*"
    has_special_char = False
    for char in password:
        if char in special_characters:
            has_special_char = True
            break
    if not has_special_char:
        raise ValueError("It should contain at least one special character")

    if not (10 <= len(password) <= 20):
        raise ValueError("Should be a minimum of 10 characters and a maximum of 
20 characters")

    return True

name = input()
mobile_number = input()
username = input()
password = input()

try:
    if validate_password(password):
        print("Valid Password")
except ValueError as e:
    print(e)
```

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

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 "

Input 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

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

def get_event_times():
   time_format = "%Y-%m-%d %H:%M:%S"

   try:
      start_time_str = input()
      start_time = datetime.strptime(start_time_str, time_format)

      end_time_str = input()
      end_time = datetime.strptime(end_time_str, time_format)

      print(start_time_str)
      print(end_time_str)

   except ValueError:
      print("Event time is not in the format")
```

```
get_event_times()
```

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