Rajalakshmi Engineering College

Name: Pavithra J
Email: 240701381@rajalakshmi.edu.in
Roll no: 240701381
Phone: 9363364978
Branch: REC
Department: I CSE FD
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

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

def main():
    while True:
        student_name = input()
        if student_name.lower() == 'done':
            break

        subjects = []
        grades = []

        for _ in range(2):
            subject = input()
            subjects.append(subject)
            while True:
                try:
                    grade = float(input())
```

```
          if 0 <= grade <= 100:
            grades.append(grade)
            break
          else:
            print("Grade must be between 0 and 100. Please try again.")
        except ValueError:
          print("Invalid input. Please enter a numeric value for the grade.")

    gpa = calculate_gpa(grades)


    with open("magical_grades.txt", "a") as file:
      file.write(f"{student_name}: {subjects[0]} - {grades[0]}, {subjects[1]} -
{grades[1]}\n")


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

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

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(a, b, c):
   # Check for positive side lengths
   if a <= 0 or b <= 0 or c <= 0:
      raise ValueError("Side lengths must be positive")

   # Triangle inequality check
   if (a + b > c) and (a + c > b) and (b + c > a):
      return True
   else:
      return False

def main():
   try:
      # Reading inputs
      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}")

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

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
def main():
    names = []

    while True:
        name = input().strip()
        if name.lower() == 'q':
            break
        names.append(name)

    # Write names to the file
    with open("sorted_names.txt", "w") as file:
        for name in names:
            file.write(name + "\n")

    # Sort the names alphabetically
    names.sort()

    # Print the sorted names separated by space
    print(*names)

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

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 !@#$%^&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):
    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 not (10 <= len(password) <= 20):
        raise Exception("Should be a minimum of 10 characters and a maximum of
20 characters")

def main():
    name = input().strip()
    mobile = input().strip()
    username = input().strip()
    password = input().strip()

    try:
        validate_password(password)
        print("Valid Password")
    except Exception as e:
        print(e)

main()
```

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