Rajalakshmi Engineering College

Name: NISHANTH ELANGO RAJAN
Email: 241901075@rajalakshmi.edu.in
Roll no: 241901075
Phone: 9444909050
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

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

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 (10 <= len(password) <= 20):
      raise ValueError("Should be a minimum of 10 characters and a maximum of
20 characters")

   if not any(char.isdigit() for char in password):
      raise ValueError("Should contain at least one digit")

   if not any(char in special_chars for char in password):
      raise ValueError("It should contain at least one special character")

   return "Valid Password"

name = input().strip()
mobile_number = input().strip()
username = input().strip()
password = input().strip()
```

```
try:
   print(validate_password(password))
except ValueError as e:
   print(e)
```

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 C + A > B:
        return True
    else:
        return False

def main():
    try:
        A = int(input().strip())
        B = int(input().strip())
        C = int(input().strip())
        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}")
main()
```

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

3. 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 calculate_total_sales():
    N = int(input().strip())
    if N > 30:
        print("Exceeding limit!")
        return
    items_sold = list(map(int, input().strip().split()))
    M = int(input().strip())

    with open('sales.txt', 'w') as file:
        for items in items_sold:
            total_earnings = items * M
            file.write(f"{total_earnings}\n")
    with open('sales.txt', 'r') as file:
        for line in file:
            print(line.strip())
calculate_total_sales()
```

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

4. 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 analyze_character_frequency():
    input_string = input().strip()
    frequency = Counter(input_string)
    with open('char_frequency.txt', 'w') as file:
        file.write("Character Frequencies:\n")
        print("Character Frequencies:")
        for char, count in frequency.items():
            output_line = f"{char}: {count}"
            print(output_line)
            file.write(output_line + "\n")

analyze_character_frequency()
```

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