Rajalakshmi Engineering College

Name: Mahalakshmi S
Email: 240701301@rajalakshmi.edu.in
Roll no:
Phone: 9566463810
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 : 40

Section 1 : Coding

1. 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(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

def main():
    try:
        side1 = int(input().strip())
        side2 = int(input().strip())
        side3 = int(input().strip())

        if is_valid_triangle(side1, side2, side3):
            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***

2. 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

```
def calculate_character_frequencies(input_string):
    frequency = {}

    for char in input_string:
        if char in frequency:
            frequency[char] += 1
        else:
            frequency[char] = 1

    return frequency

def save_frequencies_to_file(frequency, filename="char_frequency.txt"):
    with open(filename, 'w') as file:
        for char, count in frequency.items():
            file.write(f"{char}: {count}\n")

def main():
    input_string = input().strip()

    frequencies = calculate_character_frequencies(input_string)

    print("Character Frequencies:")
    for char, count in frequencies.items():
        print(f"{char}: {count}")

    save_frequencies_to_file(frequencies)

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

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):
    if len(password) < 10 or 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")

    special_characters = "!@#$%^&*"
    if not any(char in special_characters for char in password):
        raise ValueError("It should contain at least one special character")
    return True

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

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

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

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

4. 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

```
def main():
   names = []

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

   names.sort()

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

   for name in names:
      print(name)

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

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