Rajalakshmi Engineering College

Name: Afrin N
Email: 240701022@rajalakshmi.edu.in
Roll no:
Phone: null
Branch: REC
Department: I CSE FA
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 : 37.5

Section 1 : Coding

1. Problem Statement

Write a program to read the Register Number and Mobile Number of a student. Create user-defined exception and handle the following:

If the Register Number does not contain exactly 9 characters in the specified format(2 numbers followed by 3 characters followed by 4 numbers) or if the Mobile Number does not contain exactly 10 characters, throw an IllegalArgumentException. If the Mobile Number contains any character other than a digit, raise a NumberFormatException.If the Register Number contains any character other than digits and alphabets, throw a NoSuchElementException.If they are valid, print the message 'valid' or else print an Invalid message.

Input Format

The first line of the input consists of a string representing the Register number.

The second line of the input consists of a string representing the Mobile number.

Output Format

The output should display any one of the following messages:

If both numbers are valid, print "Valid".

If an exception is raised, print "Invalid with exception message: ", followed by the specific exception message.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 19ABC1001
9949596920

Output: Valid

Answer

```
# You are using Python
import re

class InvalidRegisterNumberException(ValueError):
    pass

class InvalidMobileNumberException(ValueError):
    pass

class InvalidCharacterException(Exception):
    pass

def validate_register_number(register_number):
    if len(register_number) != 9:
        raise InvalidRegisterNumberException("Register Number should have
exactly 9 characters.")

    if not re.match(r'^\d{2}[A-Za-z]{3}\d{4}$', register_number):
        raise InvalidRegisterNumberException("Register Number should have the
format: 2 numbers, 3 characters, and 4 numbers.")
```

```
    if not register_number.isalnum():
        raise InvalidCharacterException("Register Number contains invalid
characters.")

def validate_mobile_number(mobile_number):
    if len(mobile_number) != 10:
        raise InvalidMobileNumberException("Mobile Number should have exactly
10 characters.")
    if not mobile_number.isdigit():
        raise NumberFormatException("Mobile Number should contain only digits.")

class NumberFormatException(Exception):
    pass

def main():
    register_number = input().strip()
    mobile_number = input().strip()
    try:
        validate_register_number(register_number)
        validate_mobile_number(mobile_number)
        print("Valid")
    except (InvalidRegisterNumberException, InvalidMobileNumberException,
InvalidCharacterException, NumberFormatException) as e:
        print("Invalid with exception message: " + str(e))

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

Status : Partially correct **_Marks : 7.5/10_**

2. 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

```
# You are using Python
def main():
    N = int(input())

    if N > 30:
        print("Exceeding limit!")
        return

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

    total_earnings = [items * M for items in items_sold]

    with open('sales.txt', 'w') as file:
        for earnings in total_earnings:
            file.write(f"{earnings}\n")

    with open('sales.txt', 'r') as file:
        for line in file:
            print(line.strip())

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

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

```
# 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")

    return (side1 + side2 > side3) and (side1 + side3 > side2) and (side2 + side3 > 
side1)

def main():
    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}")

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

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

```
# You are using Python
```

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

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

    return frequency

def main():
    input_string = input()
    frequency = analyze_character_frequency(input_string)

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

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

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

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