Rajalakshmi Engineering College

Name: Dineshraj R
Email: 241501049@rajalakshmi.edu.in
Roll no: 241501049
Phone: 9363708090
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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

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 (a + c > b) and (b + c > a):
        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***

2. 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

```
import re
class IllegalArgumentException(Exception):
    pass
```

```
class NumberFormatException(Exception):
   pass
class NoSuchElementException(Exception):
   pass
def validate_register_number(reg_no):
   if len(reg_no) != 9:
      raise IllegalArgumentException("Register Number should have exactly 9
characters.")
   if not re.match(r"^\d{2}[A-Za-z]{3}\d{4}$", reg_no):
      if not re.match(r"^[A-Za-z0-9]+$", reg_no):
         raise NoSuchElementException("Register Number should only contain
digits and alphabets.")
      else:
         raise IllegalArgumentException("Register Number should have the format:
2 numbers, 3 characters, and 4 numbers.")
def validate_mobile_number(mob_no):
   if len(mob_no) != 10:
      raise IllegalArgumentException("Mobile Number should have exactly 10
characters.")
   if not mob_no.isdigit():
      raise NumberFormatException("Mobile Number should only contain digits.")
reg_no = input().strip()
mob_no = input().strip()

try:
   validate_register_number(reg_no)
   validate_mobile_number(mob_no)
   print("Valid")
except (IllegalArgumentException, NumberFormatException,
NoSuchElementException) as e:
   print(f"Invalid with exception message: {e}")
```

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

3. 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 OrderedDict
input_string = input()
char_freq = OrderedDict()
for char in input_string:
    if char in char_freq:
        char_freq[char] += 1
    else:
        char_freq[char] = 1
with open("char_frequency.txt", "w") as file:
    for char, count in char_freq.items():
        file.write(f"{char}: {count}\n")
print("Character Frequencies:")
for char, count in char_freq.items():
    print(f"{char}: {count}")
```

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_characters = "!@#$%^&*"
    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")

    has_special = False
    for char in password:
        if char in special_characters:
            has_special = True
            break
    if not has_special:
        raise ValueError("It should contain at least one special character")

    length = len(password)
    if length < 10 or length > 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***