Rajalakshmi Engineering College

Name: Aishwarya R
Email: 241501011@rajalakshmi.edu.in
Roll no: 241501011
Phone: null
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 : 40

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

```
import re

register_number = input()
mobile_number = input()

try:
    if len(register_number) != 9:
        raise ValueError("Register Number should have exactly 9 characters.")

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

    if not register_number.isalnum():
        raise StopIteration("Register Number should only contain alphabets and
digits.")

    if len(mobile_number) != 10:
        raise ValueError("Mobile Number should have exactly 10 characters.")
```

```
    if not mobile_number.isdigit():
        raise TypeError("Mobile Number should only contain digits.")

    print("Valid")

except ValueError as ve:
    print(f"Invalid with exception message: {ve}")
except TypeError as te:
    print(f"Invalid with exception message: {te}")
except StopIteration as ne:
    print(f"Invalid with exception message: {ne}")
```

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 (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***

3. 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

```
text = []
flag = True

while flag:
    str1 = input()
    if str1 == "done":
        flag = False
    else:
```

```
    text.append(str1)

num = []
for i in range(len(text)):
    try:
        num.append(int(text[i]))
    except ValueError:
        pass

if len(num) > 0:
    avg = sum(num) / len(num)
    print(f"{avg:.2f}")
else:
    print("No valid numbers entered.")
```

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 !@#$%^&* 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:
        return "Should be a minimum of 10 characters and a maximum of 20 characters"
    if not any(char.isdigit() for char in password):
        return "Should contain at least one digit"
    special_characters = {'!', '@', '#', '$', '%', '^', '&', '*'}
    if not any(char in special_characters for char in password):
        return "It should contain at least one special character"
    return "Valid Password"

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

result = validate_password(password)
print(result)
```

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