Rajalakshmi Engineering College

Name: Mohammed Rizwan
Email: 240701327@rajalakshmi.edu.in
Roll no: 240701327
Phone: 9944383207
Branch: REC
Department: I CSE FC
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

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
avg=sum(num)/len(num)
print(f"{avg:.2f}")
```

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

```
# You are using Python
class IllegalArgumentException(Exception):
   pass

class NumberFormatException(Exception):
   pass

class NoSuchElementException(Exception):
   pass

def validate_register_number(register_number):
   if len(register_number) != 9:
      raise IllegalArgumentException("Register Number should have exactly 9
characters.")
   if not (register_number[:2].isdigit() and register_number[2:5].isalpha() and
register_number[5:].isdigit()):
      raise IllegalArgumentException("Register Number should have the format: 2
numbers, 3 characters, and 4 numbers.")
   if not register_number.isalnum():
      raise NoSuchElementException("Register Number contains invalid
characters.")

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

def main():
   register_number = input()
   mobile_number = input()

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

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

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 Counter

def calculate_character_frequency(text):
    return Counter(text)
```

```
def save_to_file(frequencies):
    with open("char_frequency.txt", "w") as file:
        for char, count in frequencies.items():
            file.write(f"{char}: {count}\n")

def main():
    text = input()

    frequencies = calculate_character_frequency(text)

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

    save_to_file(frequencies)

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

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

```
import re

class PasswordValidationException(Exception):
   pass

def validate_password(password):
   if len(password) < 10 or len(password) > 20:
      raise PasswordValidationException("Should be a minimum of 10 characters
and a maximum of 20 characters")

   if not re.search(r'\d', password):
      raise PasswordValidationException("Should contain at least one digit")

   if not re.search(r'[!@#$%^&*]', password):
      raise PasswordValidationException("It should contain at least one special
character")

   return "Valid Password"
```

```
def main():
    name = input()
    mobile_number = input()
    username = input()
    password = input()

    try:
        result = validate_password(password)
        print(result)
    except PasswordValidationException as e:
        print(e)

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

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