Rajalakshmi Engineering College

Name: visweswaran v
Email: 240701607@rajalakshmi.edu.in
Roll no:
Phone: null
Branch: REC
Department: I CSE FF
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

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

def validate_register_number(reg_num):
    if len(reg_num) != 9:
        raise IllegalArgumentException("Register Number should have exactly 9 
characters.")
    if not re.match(r'^\d{2}[A-Za-z]{3}\d{4}$', reg_num):
        raise IllegalArgumentException("Register Number should have the format: 2 
numbers, 3 characters, and 4 numbers.")
    if not reg_num.isalnum():
        raise NoSuchElementException("Register Number should contain only digits 
and alphabets.")

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

```
# Custom Exceptions
class IllegalArgumentException(Exception):
    pass

class NoSuchElementException(Exception):
    pass

class NumberFormatException(Exception):
    pass

# Main Execution
try:
    reg_num = input().strip()
    mobile = input().strip()

    validate_register_number(reg_num)
    validate_mobile_number(mobile)

    print("Valid")

except (IllegalArgumentException, NoSuchElementException,
NumberFormatException) as e:
    print(f"Invalid with exception message: {e}")
```

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

```
from collections import Counter

s = input()
f = {}
f = Counter(s)

with open("char_frequency.txt", "w") as file:
   for c, v in f.items():
      file.write(c + ":" + str(v) + "\n")

with open("char_frequency.txt", "r") as file:
   print("Character Frequencies:")
   for ln in file:
      print(ln, end="")
```

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

3. 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

```
with open("sorted_names.txt", "w") as f:
    while True:
        i = input()
        if i == 'q':
            break
        f.write(i.strip() + "\n")

with open("sorted_names.txt", "r") as f:
    l = [line for line in f]
    l.sort()
    for j in l:
        print(j, end="")
```

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

4. 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

```
class LimitError(Exception):
    pass

try:
    n = int(input())
    if n > 30:
        raise LimitError
except LimitError:
    print("Exceeding limit!")
else:
    l = list(map(int, input().split()))
    p = int(input())
    a = []
    for i in l:
        a.append(str(i * p) + "\n")
    with open("sales.txt", "w") as f:
        f.writelines(a)
    with open("sales.txt", "r") as f:
        for line in f:
            print(line, end="")
```

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