Rajalakshmi Engineering College

Name: Anirudh Sathishkumar
Email: 240701618@rajalakshmi.edu.in
Roll no: 240701618
Phone: 6385589962
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 : 36.5

Section 1 : Coding

1. 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

```
def name_sorter():
   names = []

   print()

   while True:
      name = input()
      if name.lower() == 'q':
         break
      names.append(name)

   names.sort()

   with open('sorted_names.txt', 'w') as file:
      for name in names:
         file.write(name + '\n')

   for name in names:
      print(name)

name_sorter()
```

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

2. 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(name, mobile, username, password):
    has_digit = any(char.isdigit() for char in password)
    if not has_digit:
        return "Should contain at least one digit"

    special_chars = "!@#$%^&*"
    has_special = any(char in special_chars for char in password)
    if not has_special:
        return "It should contain at least one special character"

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

    return "Valid Password"

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

print(validate_password(name, mobile, username, password))
```

Status : Partially correct ***Marks : 6.5/10***

3. 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

```
def validate_reg_num(reg_num):
    if len(reg_num) != 9:
        raise ValueError("Register Number should have exactly 9 characters.")

    if not (reg_num[:2].isdigit() and
            reg_num[2:5].isalpha() and
            reg_num[5:].isdigit()):
        raise ValueError("Register Number should have the format: 2 numbers, 3
characters, and 4 numbers.")

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

def validate_mob_num(mob_num):
    if len(mob_num) != 10:
        raise ValueError("Mobile Number should have exactly 10 characters.")

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

```
reg_num = input()
mob_num = input()
try:
   validate_reg_num(reg_num)
   validate_mob_num(mob_num)

   print("Valid")

except ValueError as ve:
   print(f"Invalid with exception message: {ve}")
except Exception as e:
   print(f"Invalid with exception message: {e}")
```

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

```
n = int(input())

if n > 30:
   print("Exceeding limit!")
else:
   items = input().split()
   price = int(input())

   earnings = []
   for i in range(n):
      count = int(items[i])
      total = count * price
      earnings.append(total)

   f = open('sales.txt', 'w')
   for e in earnings:
      f.write(str(e) + '\n')
   f.close()
```

```
f = open('sales.txt', 'r')
lines = f.readlines()
f.close()

for line in lines:
    print(line.strip())
```

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