Rajalakshmi Engineering College

Name: Sudharsan R
Email: 240701543@rajalakshmi.edu.in
Roll no: 240701543
Phone: 9176060959
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

```
# You are using Python
import re

# Custom exception classes
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'^[A-Za-z0-9]+$', reg_no):
        raise NoSuchElementException("Register Number should contain only digits and alphabets.")
```

```
    if not re.match(r'^\d{2}[A-Za-z]{3}\d{4}$', reg_no):
        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.")

# Read inputs
reg_no = input().strip()
mob_no = input().strip()

# Validate and handle exceptions
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**

2. 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

```
# You are using Python
# File to store the names
filename = "sorted_names.txt"

# Collecting names until 'q' is entered
names = []
while True:
    name = input().strip()
    if name.lower() == 'q':
        break
    names.append(name)

# Sort the names alphabetically
names.sort()

# Write sorted names to file
with open(filename, 'w') as f:
    for name in names:
        f.write(name + '\n')

# Read and display sorted names from file
with open(filename, 'r') as f:
```

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

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

3. 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

```
# You are using Python
# Read input
N = int(input())

# Check if N exceeds 30
if N > 30:
    print("Exceeding limit!")
else:
    items_sold = list(map(int, input().split()))
    M = int(input())

    # Calculate total earnings per day and write to file
    with open("sales.txt", "w") as file:
        for sold in items_sold:
            total = sold * M
            file.write(str(total) + "\n")

    # Read from file and display each day's total earnings
    with open("sales.txt", "r") as file:
        for line in file:
            print(line.strip())
```

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

4. 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

```
# You are using Python
# Open the mystical file to write magical grades
with open("magical_grades.txt", "w") as file:
    while True:
        name = input()
```

```
        if name == "done":
            break
        subject1 = input()
        grade1 = int(input())
        subject2 = input()
        grade2 = int(input())

        # Write to the magical file
        file.write(f"{name},{subject1},{grade1},{subject2},{grade2}\n")

# Now read from the mystical file and reveal the GPA
with open("magical_grades.txt", "r") as file:
    for line in file:
        parts = line.strip().split(",")
        grade1 = int(parts[2])
        grade2 = int(parts[4])
        gpa = (grade1 + grade2) / 2
        print(f"{gpa:.2f}")
```

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