Rajalakshmi Engineering College

Name: swetha veeramani
Email: 241501261@rajalakshmi.edu.in
Roll no: 241501261
Phone: 9790907713
Branch: REC
Department: I AI & ML FC
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 : 36.5

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

class IllegalArgumentException(Exception):
    pass

class NumberFormatException(Exception):
    pass

class NoSuchElementException(Exception):
    pass

def validate_student_details():
    register_number = input()
    mobile_number = input()
    try:
        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.")
        if len(register_number) != 9:
            raise IllegalArgumentException("Register Number should have exactly 9
characters.")
        if not re.fullmatch(r'\d{2}[a-zA-Z]{3}\d{4}', register_number):
            raise NoSuchElementException("Register Number should have the format:
2 numbers, 3 characters, and 4 numbers.")
        print("Valid")

    except (IllegalArgumentException, NumberFormatException,
NoSuchElementException) as e:
        print(f"Invalid with exception message: {e}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

validate_student_details()
```

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 !@#$%^&* 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

def validate_password(password):
    has_digit = False
    for char in password:
        if char.isdigit():
            has_digit = True
            break
    if not has_digit:
        raise Exception("Should contain at least one digit")
    special_characters = "!@#$%^&*"
    has_special_char = False
    for char in password:
        if char in special_characters:
            has_special_char = True
            break
    if not has_special_char:
        raise Exception("It should contain at least one special character")
    if not (10 <= len(password) <= 20):
        raise Exception("Should be a minimum of 10 characters and a maximum of
20 characters")
    return True

def main():
```

```
    name = input()
    mobile_number = input()
    username = input()
    password = input()

    try:
        validate_password(password)
        print("Valid Password")
    except Exception as e:
        print(e)

main()
```

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

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

```
# You are using Python
def academic_alchemist():
    file_name = "magical_grades.txt"
    last_gpa = 0.0
    try:
        with open(file_name, 'a') as file:
            while True:
                student_name = input()
                if student_name.lower() == 'done':
                    break

                subject1 = input()
                grade1 = int(input())

                subject2 = input()
                grade2 = int(input())

                current_gpa = (grade1 + grade2) / 2.0
                last_gpa = current_gpa
                file.write(f"{student_name},{subject1},{grade1},{subject2},{grade2}\n")

        print(f"{last_gpa:.2f}")

    except ValueError:
        print("Invalid input: Grades must be numbers.")
    except IOError:
        print(f"Error: Could not access file {file_name}.")
    except Exception as e:
```

```
        print(f"An unexpected error occurred: {e}")

academic_alchemist()
```

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

```
# You are using Python
def calculate_and_display_sales():
    MAX_DAYS = 30
    file_name = "sales.txt"
    N = int(input())
    if N > MAX_DAYS:
        print("Exceeding limit!")
        return

    items_sold_str = input().split()
    items_sold_per_day = [int(item) for item in items_sold_str]

    M = int(input())

    try:
        with open(file_name, 'w') as file:
            for items_today in items_sold_per_day:
                total_earnings_today = items_today * M
                file.write(str(total_earnings_today) + '\n')
    except IOError:
        print(f"Error: Could not write to file {file_name}.")
        return

    try:
        with open(file_name, 'r') as file:
            for line in file:
                print(line.strip())
    except FileNotFoundError:
```

```
        print(f"Error: The file '{file_name}' was not found.")
    except IOError:
        print(f"Error: Could not read from file {file_name}.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

calculate_and_display_sales()
```

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