Rajalakshmi Engineering College

Name: Divya darshini S
Email: 241501051@rajalakshmi.edu.in
Roll no: 241501051
Phone: 6383045036
Branch: REC
Department: I AIML FA
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 : 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

```
with open("magical_grades.txt", "w") as file:
    while True:

        student_name = input().strip()
        if student_name.lower() == 'done':
            break


        subject1 = input().strip()
        grade1 = float(input().strip())

        subject2 = input().strip()
        grade2 = float(input().strip())


        if not (0 <= grade1 <= 100 and 0 <= grade2 <= 100):
            print("Grades must be between 0 and 100.")
```

```
        continue

    file.write(f"{student_name} {subject1} {grade1} {subject2} {grade2}\n")

    gpa = (grade1 + grade2) / 2
    print(f"{gpa:.2f}")
```

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 OrderedDict


text = input()


char_freq = OrderedDict()


for char in text:
    if char in char_freq:
        char_freq[char] += 1
    else:
        char_freq[char] = 1


with open("char_frequency.txt", "w") as file:
    for char, freq in char_freq.items():
        file.write(f"{char}: {freq}\n")


print("Character Frequencies:")
for char, freq in char_freq.items():
    print(f"{char}: {freq}", end=" ")
print()
```

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

3. Problem Statement

Implement a program that checks whether a set of three input values can form the sides of a valid triangle. The program defines a function is_valid_triangle that takes three side lengths as arguments and raises a ValueError if any side length is not a positive value. It then checks whether the sum of any two sides is greater than the third side to determine the validity of the triangle.

Input Format

The first line of input consists of an integer A, representing side1.

The second line of input consists of an integer B, representing side2.

The third line of input consists of an integer C, representing side3.

Output Format

The output prints either "It's a valid triangle" if the input side lengths form a valid triangle,

or "It's not a valid triangle" if they do not.

If there is a ValueError, it should print "ValueError: <error_message>".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
4
5
Output: It's a valid triangle

Answer

```
def is_valid_triangle(a, b, c):

    if a <= 0 or b <= 0 or c <= 0:
        raise ValueError("Side lengths must be positive")


    if (a + b > c) and (a + c > b) and (b + c > a):
        return True
    else:
        return False

try:

    a = int(input())
```

```
    b = int(input())
    c = int(input())


    if is_valid_triangle(a, b, c):
        print("It's a valid triangle")
    else:
        print("It's not a valid triangle")

except ValueError as ve:
    print(f"ValueError: {ve}")
```

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

4. 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


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'^[0-9]{2}[A-Za-z]{3}[0-9]{4}$', reg_no):
        raise IllegalArgumentException("Register Number should have the format: 2 numbers, 3 characters, and 4 numbers.")

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

def validate_mobile_number(mobile_no):
    if len(mobile_no) != 10:
        raise IllegalArgumentException("Mobile Number should have exactly 10 characters.")
```

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

try:
    reg_no = input().strip()
    mobile_no = input().strip()

    validate_register_number(reg_no)
    validate_mobile_number(mobile_no)

    print("Valid")

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

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