Rajalakshmi Engineering College

Name: Devadharshini K
Email: 240701626@rajalakshmi.edu.in
Roll no: 2116240701626
Phone: 8248408481
Branch: REC
Department: I CSE FF
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 4_PAH_Updated

Attempt : 1
Total Mark : 60
Marks Obtained : 60

Section 1 : Coding

1. Problem Statement

Hussain wants to create a program to calculate a person's BMI (Body Mass Index) based on their weight in kilograms and height in meters. The BMI is a measure of a person's body fat relative to their height.

Your program should take user input for weight and height, calculate the BMI, and display the result.

Function Signature: calculate_bmi(weight, height)

Formula: BMI = Weight/(Height)2

Input Format

The first line of input consists of a positive floating-point number, the person's

weight in kilograms.

The second line of input consists of a positive floating-point number, the person's height in meters.

Output Format

The output displays "Your BMI is: [BM] followed by a float value representing the calculated BMI, rounded off two decimal points.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 70.0
1.75
Output: Your BMI is: 22.86

Answer

```
weight = float(input())
height = float(input())

def calculate_bmi(weight,height):
    bmi=weight/pow(height,2)
    print(f"your BMI is: {bmi:.2f}")


calculate_bmi(weight, height)
```

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

2. Problem Statement

Ella is designing a messaging application that needs to handle long text messages efficiently. To optimize storage and transmission, she plans to implement a text compression feature that replaces consecutive repeated characters with the character followed by its count, while leaving non-repeated characters unchanged.

Help Ella create a recursive function to achieve this compression without

altering the original message's meaning.

Function Specification: def compress_string(*args)

Input Format

The input consists of a single line containing the string to be compressed.

Output Format

The output consists of a single line containing the compressed string.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: aaaBBBccc
Output: a3B3c3

Answer

```
def compress_string(s):
    # Helper function to compress the string recursively
    def helper(index, compressed):
        if index >= len(s):  # Base case: when we've processed all characters
            return compressed

        current_char = s[index]
        count = 1

        # Count how many times the current_char is repeated consecutively
        while index + 1 < len(s) and s[index + 1] == current_char:
            count += 1
            index += 1

        # Add the character and its count (if more than 1) to the result
        if count > 1:
            compressed += current_char + str(count)
        else:
            compressed += current_char

        return helper(index + 1, compressed)  # Recurse for the next part of the
string
```

```
    # Start recursion from index 0 with an empty result string
    return helper(0, '')

# Input and output handling
s = input().strip()  # Read input string
print(compress_string(s))  # Output the compressed string
```

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

3. Problem Statement

Create a Python program to monitor temperatures in a greenhouse using two sensors. Calculate and display the absolute temperature difference between the two sensor readings to ensure proper temperature control.

Note: Use the abs() built-in function.

Input Format

The first line consists of a floating-point number, representing the temperature reading from Sensor 1.

The second line consists of a floating-point number, representing the temperature reading from Sensor 2.

Output Format

The output displays the absolute temperature difference between Sensor 1 and Sensor 2, rounded to two decimal places.

Refer to the sample output for the exact format.

Sample Test Case

Input: 33.2
26.7
Output: Temperature difference: 6.50 °C

Answer

```
i=float(input())
j=float(input())
value=abs(i-j)
print(f"Temperature difference: {value:.2f} \u00b0C")
```

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

4. Problem Statement

Alice works at a digital marketing company, where she analyzes large datasets. One day, she's tasked with processing customer ID numbers, which are long numeric sequences.

To simplify her task, Alice needs to calculate the digital root of each ID. The digital root is obtained by repeatedly summing the digits of a number until a single digit remains.

Help Alice write a program that reads a customer ID number, calculates its digital root, and prints the result using a loop-based approach.

For example, the sum of the digits of 98675 is 9 + 8 + 6 + 7 + 5 = 35, then 3 + 5 = 8, which is the digital root.

Function prototype: def digital_root(num)

Input Format

The input consists of an integer num.

Output Format

The output prints an integer representing the sum of digits for a given number until a single digit is obtained.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 451110

Output: 3

Answer

```
num = int(input())

def digital_root(num):
    while num>=10:
        num=sum(int(digit)for digit in str(num))
    return num

print(digital_root(num))
```

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

5. Problem Statement

Sophia is developing a feature for her online banking application that calculates the total sum of digits in customers' account numbers. This sum is used to generate unique verification codes for secure transactions. She needs a program that takes an account number as input and outputs the sum of its digits.

Help Sophia to complete her task.

Function Specification: def sum_digits(num)

Input Format

The input consists of an integer, representing the customer's account number.

Output Format

The output prints an integer representing the sum of the digits of the account number.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 123245

Output: 17

Answer

```
num = int(input())

def sum_digits(num):
    return sum(int(digit) for digit in str(num))

sum = sum_digits(num)
print(sum)
```

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

6. Problem Statement

Ravi is working on analyzing a set of integers to determine how many of them are divisible by 3 and how many are divisible by 5. He decides to use lambda functions to filter and count the numbers based on their divisibility.

Write a program that takes a list of integers, calculates how many numbers are divisible by 3, and how many are divisible by 5, and then prints the results.

Additionally, the program should calculate the total sum of all numbers divisible by 3 and divisible by 5 separately.

Input Format

The first line contains an integer n, representing the number of integers in the list.

The second line contains n space-separated integers.

Output Format

The first line should print the count of numbers divisible by 3.

The second line should print the count of numbers divisible by 5.

The third line should print the sum of numbers divisible by 3.

The fourth line should print the sum of numbers divisible by 5.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 6
3 5 6 10 15 20

Output: 3
4
24
50

Answer

```
i=int(input())
list1=list(map(int,input().split()))
list2=list(filter(lambda x:x%3==0,list1))
list3=list(filter(lambda x:x%5==0,list1))
print(len(list2))
print(len(list3))
print(sum(list2))
print(sum(list3))
```

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