

# Rajalakshmi Engineering College

Name: DEVISHREE J  
Email: 240701702@rajalakshmi.edu.in  
Roll no: 240701702  
Phone: 7397766309  
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\_COD\_Updated

Attempt : 1  
Total Mark : 50  
Marks Obtained : 50

#### Section 1 : Coding

##### 1. Problem Statement

Imagine you are developing a text analysis tool for a cybersecurity company. Your task is to create a function that analyzes input strings to categorize and count the characters into four categories: uppercase letters, lowercase letters, digits, and special characters. The company needs this tool to process log files and identify potential security threats.

Function Signature: `analyze_string(input_string)`

##### ***Input Format***

The input consists of a single string (without space), which may include uppercase letters, lowercase letters, digits, and special characters.

##### ***Output Format***

The first line contains an integer representing the count of uppercase letters in the format "Uppercase letters: [count]".

The second line contains an integer representing the count of lowercase letters in the format "Lowercase letters: [count]".

The third line contains an integer representing the count of digits in the format "Digits: [count]".

The fourth line contains an integer representing the count of special characters in the format "Special characters: [count]".

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: Hello123

Output: Uppercase letters: 1

Lowercase letters: 4

Digits: 3

Special characters: 0

### ***Answer***

```
def analyze_string(input_string):
    uppercase_count=0
    lowercase_count=0
    digit_count=0
    special_count=0
    for char in input_string:
        if char.isupper():
            uppercase_count+=1
        elif char.islower():
            lowercase_count+=1
        elif char.isdigit():
            digit_count+=1
        else:
            special_count+=1
    return uppercase_count,lowercase_count,digit_count,special_count
```

```
input_string = input()
uppercase_count, lowercase_count, digit_count, special_count =
analyze_string(input_string)
```

```
print("Uppercase letters:", uppercase_count)
print("Lowercase letters:", lowercase_count)
print("Digits:", digit_count)
print("Special characters:", special_count)
```

**Status :** Correct

**Marks : 10/10**

## 2. Problem Statement

Sara is developing a text-processing tool that checks if a given string starts with a specific character or substring. She needs to implement a function that accepts a string and a character (or substring), and returns True if the string starts with the provided character/substring, or False otherwise.

Write a program that uses a lambda function to help Sara perform this check.

### **Input Format**

The first line contains a string `str` representing the main string to be checked.

The second line contains a string `n`, which is the character or substring to check if the main string starts with it.

### **Output Format**

The first line of output prints "True" if the string starts with the given character/substring, otherwise prints "False".

Refer to the sample for the formatting specifications.

### **Sample Test Case**

Input: Examly

e

Output: False

**Answer**

```
m=input()
n=input()
check=lambda s,sub:s.startswith(sub)
print(check(m,n))
```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Sneha is building a more advanced exponential calculator. She wants to implement a program that does the following:

Calculates the result of raising a given base to a specific exponent using Python's built-in `pow()` function. Displays all intermediate powers from  $\text{base}^1$  to  $\text{base}^{\text{exponent}}$  as a list. Calculates and displays the sum of these intermediate powers.

Help her build this program to automate her calculations.

**Input Format**

The input consists of line-separated two integer values representing base and exponent.

**Output Format**

The first line of the output prints the calculated result of raising the base to the exponent.

The second line prints a list of all powers from  $\text{base}^1$  to  $\text{base}^{\text{exponent}}$ .

The third line prints the sum of all these powers.

Refer to the sample output for formatting specifications.

**Sample Test Case**

Input: 2  
3  
Output: 8  
[2, 4, 8]  
14

**Answer**

```
base=int(input())
exponent=int(input())
result=pow(base,exponent)
print(result)
powers=[pow(base,i) for i in range(1,exponent+1)]
print(powers)
print(sum(powers))
```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Imagine you are building a messaging application, and you want to know the length of the messages sent by the users. You need to create a program that calculates the length of a message using the built-in function `len()`.

**Input Format**

The input consists of a string representing the message.

**Output Format**

The output prints an integer representing the length of the entered message.

Refer to the sample output for formatting specifications.

**Sample Test Case**

Input: hello!!  
Output: 7

**Answer**

```
def myfun(n):  
    n1=len(n)  
    return n1  
n=input()  
print(myfun(n))
```

**Status :** Correct

**Marks :** 10/10

## 5. Problem Statement

Implement a program that needs to identify Armstrong numbers. Armstrong numbers are special numbers that are equal to the sum of their digits, each raised to the power of the number of digits in the number.

Write a function `is_armstrong_number(number)` that checks if a given number is an Armstrong number or not.

Function Signature: `armstrong_number(number)`

### ***Input Format***

The first line of the input consists of a single integer, `n`, representing the number to be checked.

### ***Output Format***

The output should consist of a single line that displays a message indicating whether the input number is an Armstrong number or not.

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: 153

Output: 153 is an Armstrong number.

### ***Answer***

```
def armstrong_number(number):  
    num_str=str(number)
```

```
power=len(num_str)
sumofpowers=0
for digit in num_str:
    sumofpowers+=int(digit)**power
if sumofpowers==number:
    print(f"{number} is an Armstrong number.")
else:
    print(f"{number} is not an Armstrong number.")

n=int(input())
armstrong_number(n)
```

**Status :** Correct

**Marks :** 10/10