Rajalakshmi Engineering College

Name: Kaushika T
Email: 240701239@rajalakshmi.edu.in
Roll no: 240701239
Phone: 6381820352
Branch: REC
Department: I CSE FC
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

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

```
# You are using Pyt
s1=input()
s2=input()
l=len(s2)
x=lambda s1,s2:True if s1[0:l]==s2 else False
print(x(s1,s2))
```

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

2. 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 base¹ to base^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 base^1 to base^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

```
# You are using Python
base=int(input())
expo=int(input())
y=[]
x=pow(base,expo)
print(x)
for i in range(1,expo+1):

    y.append(pow(base,i))
print(y)
print(sum(y))
```

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

3. 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):

# You are using Python
    #Type your code here
```

```
    uppercase_count=0
    lowercase_count=0
    digit_count=0
    special_count=0
    for i in input_string:
        if i.isupper():
            uppercase_count+=1
        elif i.islower():
            lowercase_count+=1
        elif i.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***

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

```
# You are using Python
a=input()
b=len(a)
print(b)
```

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

```
# You are using Python
def armstrong_number(number,digit):
    cube=0
    while number!=0:

        num=number%10
        cube+=num**digit
        number//=10
    return cube
n=int(input())
s=str(n)
l=len(s)
cub=armstrong_number(n,l)
if cub==n:
    print(f"{n} is an Armstrong number.")
else:
    print(f"{n} is not an Armstrong number.")
```

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