

# Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.  
Email: 241501020@rajalakshmi.edu.in  
Roll no: 241501020  
Phone: 9025837734  
Branch: REC  
Department: I AI & ML FA  
Batch: 2028  
Degree: B.E - AI & ML

Scan to verify results



## NeoColab\_REC\_CS23221\_Python Programming

### REC\_Python\_Week 4\_MCQ

Attempt : 1  
Total Mark : 15  
Marks Obtained : 14

#### Section 1 : MCQ

1. What will be the output of the following Python code?

```
multiply = lambda x, y: x * y  
print(multiply(2, 'Hello'))
```

**Answer**

HelloHello

**Status : Correct**

**Marks : 1/1**

2. What is the output of the following code?

```
x=12  
def f1(a,b=x):  
    print(a,b)
```

x=15  
f1(4)

**Answer**

4 12

**Status :** Correct

**Marks :** 1/1

3. What will be the output of the following code?

```
num1 = 10  
num2 = -10  
result = abs(num1) + abs(num2)  
print(result)
```

**Answer**

20

**Status :** Correct

**Marks :** 1/1

4. What is the output of the following code snippet?

```
def my_function(x):  
    x += 5  
    return x  
  
a = 10  
result = my_function(a)  
print(a, result)
```

**Answer**

10 15

**Status :** Correct

**Marks :** 1/1

5. What will be the output of the following code?

```
value = 42  
result = abs(value) + len(str(value))
```

```
print(result)
```

**Answer**

44

**Status : Correct**

**Marks : 1/1**

6. What is the output of the following code snippet?

```
def square(x):  
    return x ** 2
```

```
result = square(4)  
print(result)
```

**Answer**

16

**Status : Correct**

**Marks : 1/1**

7. What is the output of the code shown?

```
def f():  
    global a  
    print(a)  
    a = "hello"  
    print(a)  
    a = "world"  
    f()  
    print(a)
```

**Answer**

hellohelloworld

**Status : Wrong**

**Marks : 0/1**

8. What is the output of the following code snippet?

```
def fun(x, y=2, z=3):
```

```
    return x + y + z
result = fun(1, z=4)
print(result)
```

**Answer**

7

**Status :** Correct

**Marks :** 1/1

9. What will be the output of the following Python code?

```
def cube(x):
    return x * x * x
x = cube(3)
print(x)
```

**Answer**

27

**Status :** Correct

**Marks :** 1/1

10. What will be the output of the following Python code?

```
def is_even(number):
    if number % 2 == 0:
        return True
```

```
result = is_even(6)
print(result)
```

**Answer**

True

**Status :** Correct

**Marks :** 1/1

11. What will be the output of the following Python code?

```
def absolute_value(x):
```

```
if x < 0:  
    return -x  
return x
```

```
result = absolute_value(-9)  
print(result, absolute_value(5))
```

**Answer**

9 5

**Status :** Correct

**Marks :** 1/1

12. What will be the output of the following code?

```
num = -5  
result = abs(num)  
print(result)
```

**Answer**

5

**Status :** Correct

**Marks :** 1/1

13. What is the main advantage of using lambda functions in Python?

**Answer**

They allow you to write shorter code than regular functions

**Status :** Correct

**Marks :** 1/1

14. What is the output of the code shown below?

```
def f1(x):  
    x += 1  
    print(x)
```

```
global_variable = 15  
f1(global_variable)
```

```
print("hello")
```

**Answer**

16hello

**Status :** Correct

**Marks :** 1/1

15. What will be the output of the following code?

```
def display(*args):  
    for arg in args:  
        print(arg)
```

```
display(10, 20, 30)
```

**Answer**

102030

**Status :** Correct

**Marks :** 1/1

# Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.  
Email: 241501020@rajalakshmi.edu.in  
Roll no: 241501020  
Phone: 9025837734  
Branch: REC  
Department: I AI & ML FA  
Batch: 2028  
Degree: B.E - AI & ML

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

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 base<sup>1</sup> to base<sup>exponent</sup> 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**

```
# You are using Python
base=int(input())
exponent=int(input())
result=base**exponent
print(result)
power_list=[base**i for i in range(1,exponent+1)]
print(power_list)
sum_of_powers=sum(power_list)
print(sum_of_powers)
```

**Status :** Correct

**Marks : 10/10**

## **2. 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
message=input()
result=len(message)
print(result)
```

**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 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

#### 4. 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 Python
main_string=input()
substring=input()
check_starts_with=lambda s,sub:s.startswith(sub)
print(check_starts_with(main_string,substring))
```

**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 is_armstrong(n):
    str_n=str(n)
    num_digits=len(str_n)
    armstrong_sum=sum(int(digit)**num_digits for digit in str_n)
    return armstrong_sum==n
n=int(input())
if is_armstrong(n):
    print(f"{n} is an armstrong number.")
else:
    print(f"{n} is not an armstrong number.")
```

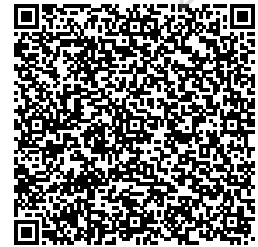
**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.  
Email: 241501020@rajalakshmi.edu.in  
Roll no: 241501020  
Phone: 9025837734  
Branch: REC  
Department: I AI & ML FA  
Batch: 2028  
Degree: B.E - AI & ML

Scan to verify results



## NeoColab\_REC\_CS23221\_Python Programming

### REC\_Python\_Week 4\_CY

Attempt : 1  
Total Mark : 40  
Marks Obtained : 40

### Section 1 : Coding

#### 1. Problem Statement

Implement a program for a retail store that needs to find the highest even price in a list of product prices. Your goal is to efficiently determine the maximum even price from a series of product prices. Utilize the max() inbuilt function in the program.

For example, if the prices are 10 15 24 8 37 16, the even prices are 10 24 8 16. So, the maximum even price is 24.

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

The input consists of a series of product prices separated by a space.

The prices should be entered as a space-separated string of numbers.

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

If there are even prices in the input, the output prints "The maximum even price is:" followed by the maximum even price.

If there are no even prices in the input, the output prints "No even prices were found".

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 10 15 24 8 37 16

Output: The maximum even price is: 24

### **Answer**

```
# You are using Python
```

```
prices = input()
```

```
prices_list = list(map(int, prices.split()))
```

```
highest_even = None
```

```
for price in prices_list:
```

```
    if price % 2 == 0:
```

```
        if highest_even is None or price > highest_even:
```

```
            highest_even = price
```

```
if highest_even is not None:
```

```
    print(f"The maximum even price is: {highest_even}")
```

```
else:
```

```
    print("No even prices were found")
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Imagine you are tasked with developing a function for calculating the total cost of an item after applying a sales tax. The sales tax rate is equal to 0.08 and it is defined as a global variable.

The function should accept the cost of the item as a parameter, calculate the tax amount, and return the total cost.

Additionally, the program should display the item cost, sales tax rate, and total cost to the user.

Function Signature: `total_cost(item_cost)`

### ***Input Format***

The input consists of a single line containing a positive floating-point number representing the cost of the item.

### ***Output Format***

The output consists of three lines:

"Item Cost:" followed by the cost of the item formatted to two decimal places.

"Sales Tax Rate:" followed by the sales tax rate in percentage.

"Total Cost:" followed by the calculated total cost after applying the sales tax, formatted to two decimal places.

Refer to the sample output for formatting specifications.

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

Input: 50.00

Output: Item Cost: \$50.00

Sales Tax Rate: 8.0%

Total Cost: \$54.00

### ***Answer***

#



```
SALES_TAX_RATE = 0.08
```

```
def total_cost(item_cost):
```

```
    tax_amount = item_cost * SALES_TAX_RATE
```

```
    total = item_cost + tax_amount
```

```
    return total
```

```
item_cost = float(input())
```

```
total_cost = total_cost(item_cost)
```

```
print(f"Item Cost: ${item_cost:.2f}")
```

```
print(f"Sales Tax Rate: {SALES_TAX_RATE * 100}%")
```

```
print(f"Total Cost: ${total_cost:.2f}")
```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Develop a text analysis tool that needs to count the occurrences of a specific substring within a given text string.

Write a function `count_substrings(text, substring)` that takes two inputs: the text string and the substring to be counted. The function should count how many times the substring appears in the text string and return the count.

Function Signature: `count_substrings(text, substring)`

#### **Input Format**

The first line of the input consists of a string representing the text.

The second line consists of a string representing the substring.

#### **Output Format**

The output should display a single line of output containing the count of occurrences of the substring in the text string.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: programming is fun and programming is cool  
programming

Output: The substring 'programming' appears 2 times in the text.

### **Answer**

```
# You are using Python
def count_substrings(text, substring):
```

```
    count = text.count(substring)
    return count
```

```
text = input().strip()
```

```
substring = input().strip()
```

```
count = count_substrings(text, substring)
```

```
print(f"The substring '{substring}' appears {count} times in the text.")
```

**Status :** Correct

**Marks : 10/10**

## **4. Problem Statement**

You are tasked with designing a shipping cost calculator program that calculates the shipping cost for packages based on their weight and destination. The program utilizes different shipping rates for domestic, international, and remote destinations. The rates for each destination type are provided as global constants.

Constant Values:

DOMESTIC\_RATE = 5.0

INTERNATIONAL\_RATE = 10.0

REMOTE\_RATE = 15.0

Function Signature: calculate\_shipping(weight, destination)

Formula: shipping cost = weight \* destination rate

### ***Input Format***

The first line of the input consists of a float representing the weight of the package.

The second line consists of a string representing the destinations(Domestic or International or Remote).

### ***Output Format***

The program outputs any one of the following:

1. If the input is valid and the destination is recognized, the output should consist of a single line stating the calculated shipping cost for the given weight and destination in the format: "Shipping cost to [destination] for a [weight] kg package: \$[calculated cost]" with two decimal places.
2. If the input weight is not a positive float, print "Invalid weight. Weight must be greater than 0."
3. If the input destination is not one of the valid options, print "Invalid destination."

Refer to the sample output for the formatting specifications.

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

Input: 5.5

Domestic

Output: Shipping cost to Domestic for a 5.5 kg package: \$27.50

### ***Answer***

#

```
def calculate_shipping(weight, destination):
    rates={
        "Domestic": 5.0,
        "International":10.0,
        "Remote":15.0
    }
    if weight<=0:
        print("Invalid weight. Weight must be greater than 0.")
    elif destination not in rates:
        print("Invalid destination.")
    else:
        return weight*rates[destination]
weight=float(input())
destination=input()

shipping_cost=calculate_shipping(weight, destination)
if shipping_cost is not None:
    print(f"Shipping cost to {destination} for a {weight} kg package:
    ${shipping_cost:.2f}")
```

**Status :** Correct

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Anirudh Kashyab.L.M.  
Email: 241501020@rajalakshmi.edu.in  
Roll no: 241501020  
Phone: 9025837734  
Branch: REC  
Department: I AI & ML FA  
Batch: 2028  
Degree: B.E - AI & ML

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

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(num) for num in str(num))
sum = sum_digits(num)
print(sum)
```

**Status :** Correct

**Marks :** 10/10

## 2. 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:  $BMI = \text{Weight} / (\text{Height})^2$

**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())

# You are using Python
def calculate_bmi(weight,height):
    bmi=weight/((height)**2)
    print(f"Your BMI is: {bmi:.2f}")

calculate_bmi(weight, height)
```

**Status :** Correct

**Marks :** 10/10

## **3. 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**

# You are using Python

```
def compress_string(*args):
    s=args[0]
    if not s:
        return ""
    result=""
    count=1
    for i in range(1,len(s)):
        if s[i]==s[i-1]:
            count+=1
        else:
            result+=s[i-1]+(str(count) if count>1 else "")
            count=1
    result+=s[-1]+(str(count)if count>1 else "")
    return result
s=input()
print(compress_string(s))
```

**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())
```

```
# You are using Python
```

```
def digital_root(num):
```

```
    while num >= 10:
```

```
sum_digits=0
while num>0:
    sum_digits+=num%10
    num//=10
    num=sum_digits
return num
print(digital_root(num))
```

**Status :** Correct

**Marks :** 10/10

## 5. 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**

```
# You are using Python
n=int(input())
nums=list(map(int, input().split()))
div_by_3=list(filter(lambda x:x%3==0,nums))
div_by_5=list(filter(lambda x:x%5==0,nums))
print(len(div_by_3))
print(len(div_by_5))
print(sum(div_by_3))
print(sum(div_by_5))
```

**Status :** Correct

**Marks :** 10/10

**6. 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**

```
# You are using Python
a=float(input())
b=float(input())
c=abs(a-b)
print(f"Temperature difference: {c:.2f} °C")
```

**Status :** Correct

**Marks :** 10/10