

School of Computer Science and Artificial Intelligence

Lab Assignment # 4

Name of Student : G SAI TEJA
Enrollment No. : 2303A51526
Batch No. 22

Task Description-1

- **Zero-shot:** Prompt AI with only the instruction. Write a Python function to determine whether a given number is prime

Expected Output-1

- A basic Python function to check if a number is prime, demonstrating correct logical conditions without relying on examples or additional context

Prompt:-

Write a function to check if a number is prime using proper logical conditions, applicable to any user-defined range.

Code:-

```
ASSIGNMENT-4.2.py
C:\> AAC-LAB ASSIGNMENT-1 > ◆ ASSIGNMENT-4.2.py > ...
#task-1
#write a function to check if a number is prime using proper logical conditions, applicable to any user-defined range.
def is_prime(num):
    if num <= 1:
        return False
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return False
    return True
# Get user-defined range
start = int(input("Enter the start of the range: "))
end = int(input("Enter the end of the range: "))
print(f"Prime numbers between {start} and {end} are:")
for number in range(start, end + 1):
    if is_prime(number):
        print(number, end=" ")
print() # for a new line after listing primes
```

Output:-

```
Enter the start of the range: 4
Enter the end of the range: 5
Prime numbers between 4 and 5 are:
5
```

Justification:-

The function determines whether a number is prime by checking for divisibility from 2 up to the square root of the number. Since any composite number must have a factor less than or equal to its square root, this approach is both logically correct and computationally efficient.

Task Description-2

- **One-shot:** Provide one example: Input: [1, 2, 3, 4], Output: 10 to help AI generate a function that calculates the sum of elements in a list.

Expected Output-2

- A correct conversion function guided by the single example.

Prompt:-

Write a program to compute the sum of all elements in an array dynamically, without relying on fixed-size inputs.

Code:-

```
#task-2
#Write a program to compute the sum of all elements in an array dynamically, without relying on fixed-size inputs.
def sum_of_array(arr):
    total = 0
    for num in arr:
        total += num
    return total
# Get user-defined array size
size = int(input("Enter the number of elements in the array: "))
array = []
for i in range(size):
    element = int(input("Enter element " + str(i+1) + ": "))
    array.append(element)
total_sum = sum_of_array(array)
print(f"The sum of all elements in the array is: {total_sum}")
```

Output:-

```
PS C:\Users\SaiTeja> & C:/Users/SaiTeja/AppData/Local/Programs/Python/Python310/python.exe task-2.py
Enter the start of the range: 5
Enter the end of the range: 8
Prime numbers between 5 and 8 are:
5 7
Enter the number of elements in the array: 9
Enter element 1: 1
Enter element 2: 5
Enter element 3: 8
Enter element 4: 9
Enter element 5: 7
Enter element 6: 2
Enter element 7: 6
Enter element 8: 0
Enter element 9: 3
The sum of all elements in the array is: 41
PS C:\Users\SaiTeja>
```

Justification:-

The program accepts dynamic user input, enabling it to compute the sum of integers in an array of any length instead of being restricted to a predefined set of values.

Task Description-3

- **Few-shot:** Give 2–3 examples to create a function that extracts digits from an alphanumeric string.

Expected Output-3

- Accurate function that returns only the digits from alphanumeric string.

Prompt:-

Write a program that takes an alphanumeric string as input and extracts only the numeric digits using an accurate and well-defined function.

Code:-

```
#task-3
#Write a program that takes an alphanumeric string as input and extracts only the numeric digits using an accurate and well-de
def extract_digits(input_string):
    digits = ''.join([char for char in input_string if char.isdigit()])
    return digits
# get user input
user_input = input("Enter an alphanumeric string: ")
numeric_digits = extract_digits(user_input)
print(f"The numeric digits extracted from the string are: {numeric_digits}")
```

Output:-

```
PS C:\Users\SaiTeja> & C:/Users/SaiTeja/AppData/Local/Py
Enter an alphanumeric string: sai1526
The numeric digits extracted from the string are: 1526
PS C:\Users\SaiTeja> █
```

Justification:-

The function efficiently extracts numeric characters from the input string by using the built-in filter function in combination with str.isdigit. This concise approach leverages Python's built-in string processing capabilities for accurate and readable digit extraction.

Task Description-4

- Compare zero-shot vs few-shot prompting for generating a function that counts the number of vowels in a string.

Expected Output-4

- Output comparison + student explanation on how examples helped the model.

Prompt:-

write a program to function that counts the number of vowels in a given string.

Code:-

```
#task-4
#Write a program that defines a function to count the number of vowels in a given string.
def count_vowels(input_string):
    vowels = 'aeiouAEIOU'
    count = sum(1 for char in input_string if char in vowels)
    return count
# Get user input
user_input = input("Enter a string: ")
vowel_count = count_vowels(user_input)
print(f"The number of vowels in the string is: {vowel_count}")
```

Output:-

```
PS C:\Users\SaiTeja> & C:/Users/SaiTeja/AppD
Enter a string: sai teja
The number of vowels in the string is: 4
PS C:\Users\SaiTeja> █
```

Justification:-

The function iterates over each character in the input string and checks whether it belongs to a predefined set of vowels. A generator expression is used to count vowel occurrences efficiently, resulting in a concise and performance-friendly implementation.

Task Description-5

- Use few-shot prompting with 3 sample inputs to generate a function that determines the minimum of three numbers without using the built-in min() function.

Expected Output-5

- A function that handles all cases with correct logic based on example patterns.

Prompt:-

Write a program that defines a function to determine the minimum of three numbers without using the built-in min() function.

Code:-

```
#task-5
#write a program that defines a function to determine the minimum of three numbers without using the built-in min() function.
def find_minimum(a, b, c):
    if a <= b and a <= c:
        return a
    elif b <= a and b <= c:
        return b
    else:
        return c
# Get user input
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))
minimum = find_minimum(num1, num2, num3)
print(f"The minimum of the three numbers is: {minimum}")
```

Output:-

```
PS C:\Users\SaiTeja> C:\Users\SaiTeja\AppData\Local\Programs\Python\Python310\python.exe task-5.py
Enter the first number: 5
Enter the second number: 8
Enter the third number: 9
The minimum of the three numbers is: 5.0
PS C:\Users\SaiTeja> █
0 0 △ 0
```

Justification:-

The code snippets effectively fulfill each specified requirement using clear and logical implementations. Every function is designed to perform its intended task without relying on external libraries or built-in functions that would trivialize the logic, thereby demonstrating a strong understanding of fundamental programming concepts.