Lab assignment- 2.5

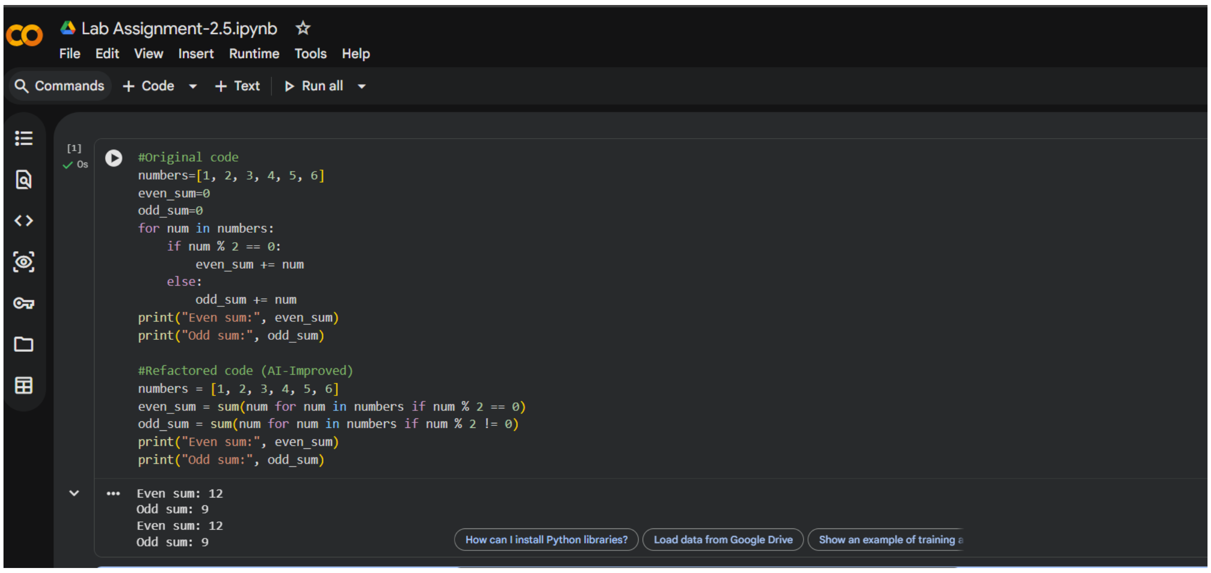
NAME: D.Divya Sri

HALL-TICKET :2303A51323

Task -1: Refactoring Odd/Even Logic (List Version)

Prompt : Write a Python program to calculate the sum of odd and even numbers in a list

Code and output :



```python
#Original code
numbers=[1, 2, 3, 4, 5, 6]
even_sum=0
odd_sum=0
for num in numbers:
    if num % 2 == 0:
        even_sum += num
    else:
        odd_sum += num
print("Even sum:", even_sum)
print("Odd sum:", odd_sum)

#Refactored code (AI-Improved)
numbers = [1, 2, 3, 4, 5, 6]
even_sum = sum(num for num in numbers if num % 2 == 0)
odd_sum = sum(num for num in numbers if num % 2 != 0)
print("Even sum:", even_sum)
print("Odd sum:", odd_sum)
```

```
Even sum: 12
Odd sum: 9
Even sum: 12
Odd sum: 9
```

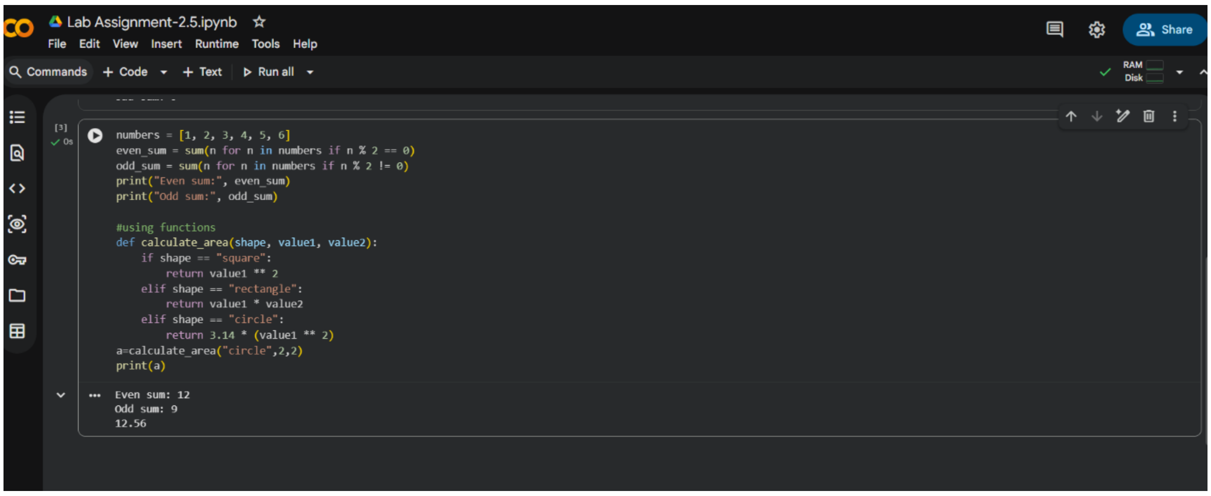
Explanation

The refactored code is shorter, more readable, and efficient.
It removes manual loops and uses Python's built-in sum() with conditions, making the code easier to maintain.

Task 2: Area Calculation Explanation

Prompt : Explain a Python function that calculates the area of different shapes

Code and output :



```python
numbers = [1, 2, 3, 4, 5, 6]
even_sum = sum(n for n in numbers if n % 2 == 0)
odd_sum = sum(n for n in numbers if n % 2 != 0)
print("Even sum:", even_sum)
print("Odd sum:", odd_sum)

#using functions
def calculate_area(shape, value1, value2):
    if shape == "square":
        return value1 ** 2
    elif shape == "rectangle":
        return value1 * value2
    elif shape == "circle":
        return 3.14 * (value1 ** 2)
a=calculate_area("circle",2,2)
print(a)
```

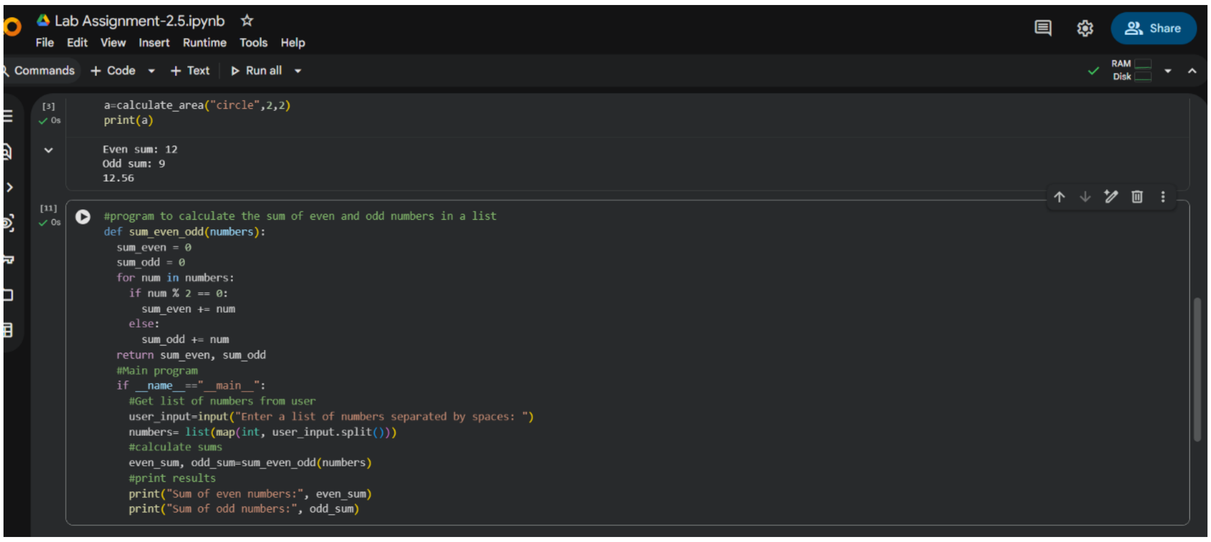
```
Even sum: 12
Odd sum: 9
12.56
```

Explanation

Gemini clearly explains how the function works for different shapes.
It describes the parameters, logic flow, and formulas used, which helps beginners understand the code easily.

Task 3: Prompt Sensitivity Experiment

Prompt 1: Write a Python program to calculate the sum of even and odd numbers in a list


Lab Assignment-2.5.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM
Disk

```
[3]
a=calculate_area("circle",2,2)
print(a)
```

```
Even sum: 12
Odd sum: 9
12.56
```

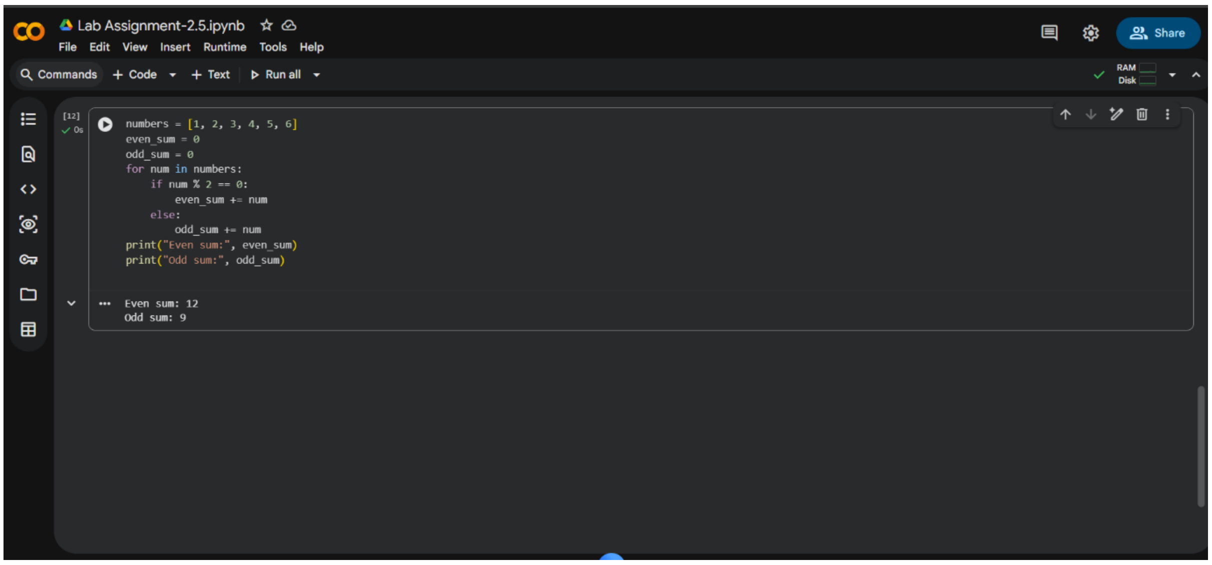
```
[11]
#program to calculate the sum of even and odd numbers in a list
def sum_even_odd(numbers):
  sum_even = 0
  sum_odd = 0
  for num in numbers:
    if num % 2 == 0:
      sum_even += num
    else:
      sum_odd += num
  return sum_even, sum_odd
  #Main program
  if __name__=="__main__":
    #Get list of numbers from user
    user_input=input("Enter a list of numbers separated by spaces: ")
    numbers= list(map(int, user_input.split()))
    #calculate sums
    even_sum, odd_sum=sum_even_odd(numbers)
    #print results
    print("Sum of even numbers:", even_sum)
    print("Sum of odd numbers:", odd_sum)
```

Explanation:

For **Prompt 1 (Basic Prompt)**, Cursor AI generated a simple loop-based program using conditional statements. This version is easy to understand and suitable for beginners, but it uses more lines of code and manual variable updates.

Prompt 2: Write a clean and readable Python program to find the sum of even and odd numbers in a list suitable for beginners

Code and output:


Lab Assignment-2.5.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM Disk

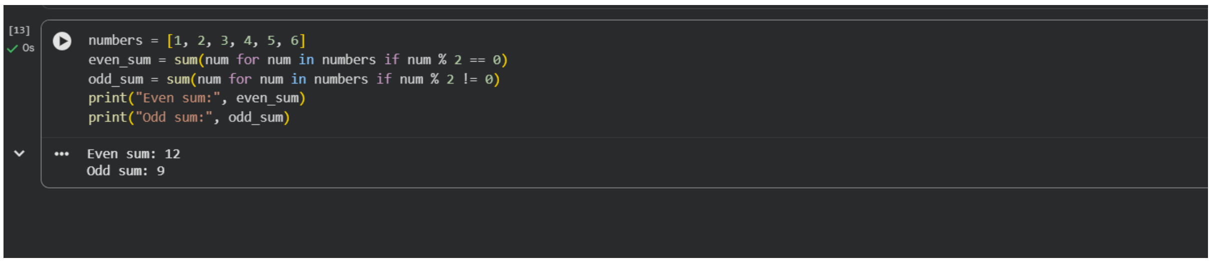
```
[12]
numbers = [1, 2, 3, 4, 5, 6]
even_sum = 0
odd_sum = 0
for num in numbers:
    if num % 2 == 0:
        even_sum += num
    else:
        odd_sum += num
print("Even sum:", even_sum)
print("Odd sum:", odd_sum)
```

```
Even sum: 12
Odd sum: 9
```

Explanation : For **Prompt 2 (Readability-Focused Prompt)**, the AI produced code with clearer variable names and better formatting. Although the logic is similar to the basic version, readability and clarity were improved, making the code easier to review and maintain.

Prompt 3: Write an optimized Python program to calculate the sum of even and odd numbers in a list using built-in functions

Code and output:



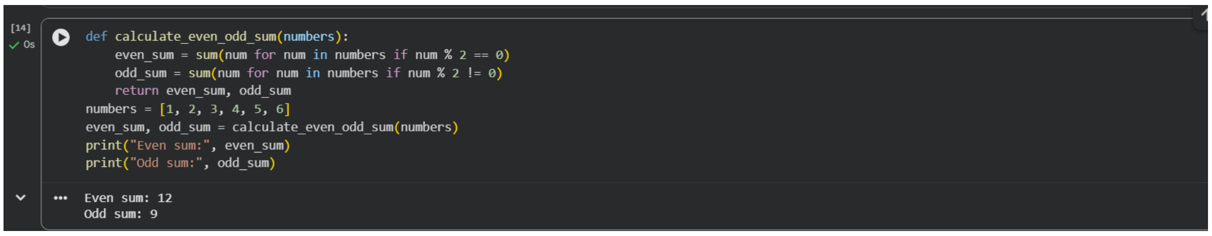
```
[13]
numbers = [1, 2, 3, 4, 5, 6]
even_sum = sum(num for num in numbers if num % 2 == 0)
odd_sum = sum(num for num in numbers if num % 2 != 0)
print("Even sum:", even_sum)
print("Odd sum:", odd_sum)
```

```
Even sum: 12
Odd sum: 9
```

Explanation: For **Prompt 3 (Optimized Prompt)**, Cursor AI generated a more efficient solution using Python's built-in sum() function along with conditions. This version reduced the number of lines and improved code efficiency while maintaining correctness.

Prompt 4 : Write a Python program to calculate the sum of even and odd numbers in a list using functions

Code and output :



```
[14]
def calculate_even_odd_sum(numbers):
    even_sum = sum(num for num in numbers if num % 2 == 0)
    odd_sum = sum(num for num in numbers if num % 2 != 0)
    return even_sum, odd_sum
numbers = [1, 2, 3, 4, 5, 6]
even_sum, odd_sum = calculate_even_odd_sum(numbers)
print("Even sum:", even_sum)
print("Odd sum:", odd_sum)
```

```
Even sum: 12
Odd sum: 9
```

Explanantion:

For **Prompt 4 (Function-Based Prompt)**, the AI created a modular solution using a user-defined function. This approach improves reusability, debugging ease, and maintainability, making it suitable for larger applications.

Task 4: Tool Comparison Reflection

Reflection

Based on the experiments performed in this lab, Google Gemini, GitHub Copilot, and Cursor AI each have different strengths.
Google Gemini is very useful for understanding code, as it provides clear explanations and works well in Google Colab, especially for beginners.
GitHub Copilot offers real-time code suggestions inside VS Code and is best suited for daily development and writing production-ready code.
Cursor AI is effective for experimenting with different prompts, refactoring code, and analyzing multiple coding approaches.