AI ASSISTANT CODING
ASSIGNMENT-2.5

Name: A.Sathwik

Enrollment no: 2303A52158

Batch: 34

Semester: VI

Branch: CSE(AIML)

Lab 2: Exploring Additional AI Coding Tools beyond Copilot – Gemini (Colab) and Cursor AI

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

❖ **Scenario:**

You are improving legacy code.

❖ **Task:**

Write a program to calculate the sum of odd and even numbers in a list, then refactor it using AI.

❖ **Expected Output:**

❖ **Original and improved code:**

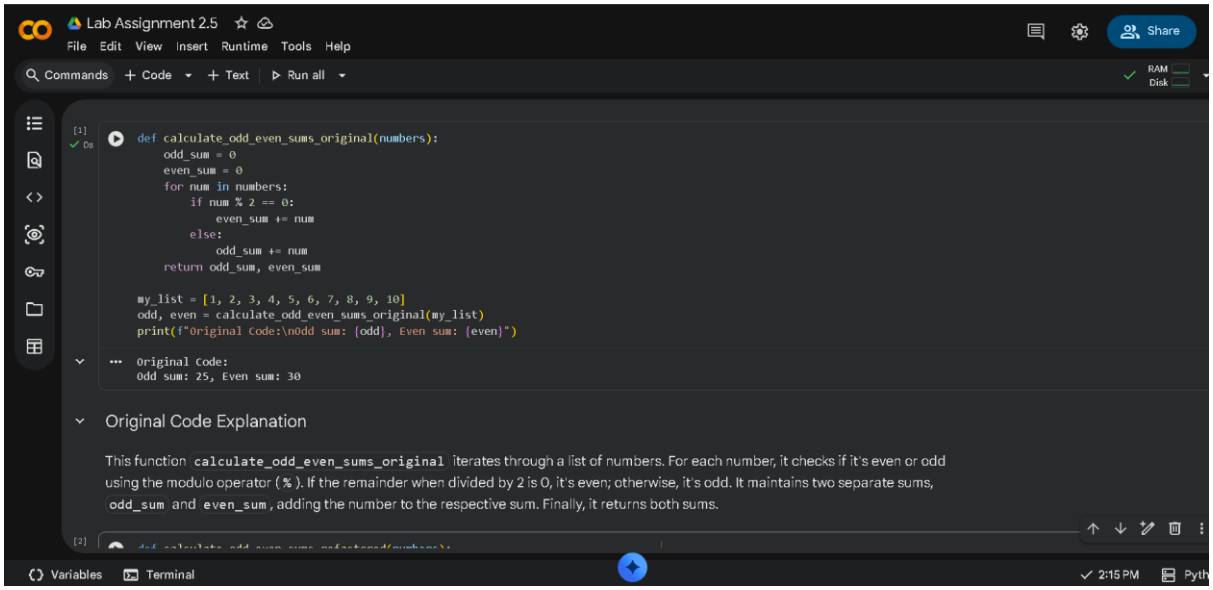
Original Code:

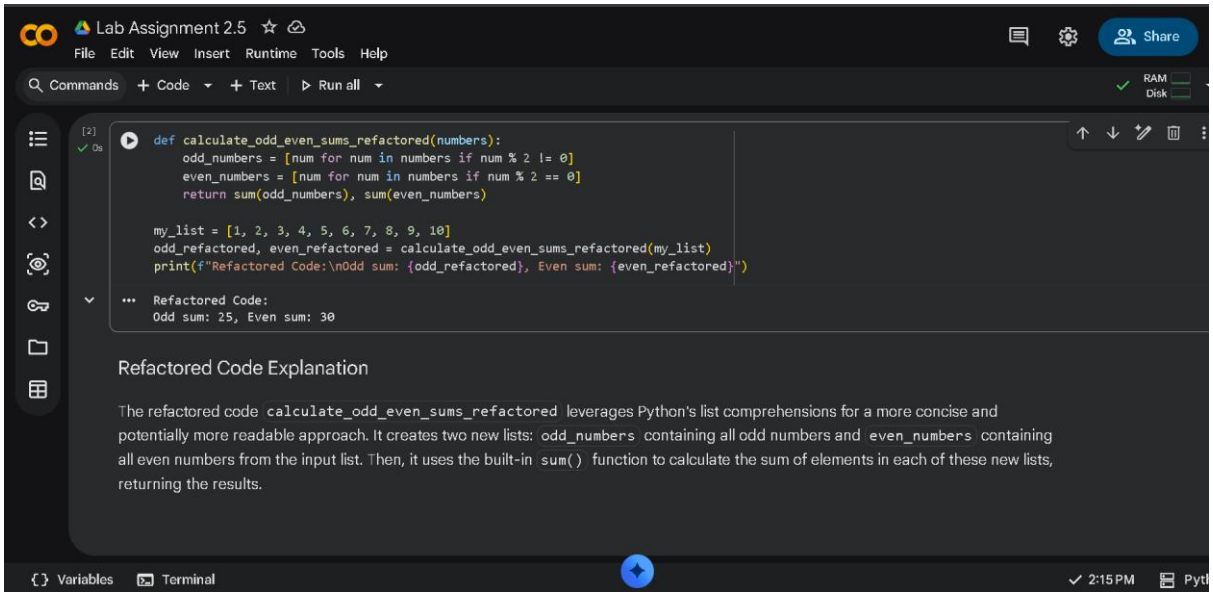
```
def calculate_odd_even_sums_original(numbers):
    odd_sum = 0
```

```
    even_sum = 0
    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num
    return odd_sum, even_sum

my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
odd, even = calculate_odd_even_sums_original(my_list)
print(f"Original Code:\nOdd sum: {odd}, Even sum: {even}")
```

Refactored Code:

```
def calculate_odd_even_sums_refactored(numbers):
    odd_numbers = [num for num in numbers if num % 2 != 0]
    even_numbers = [num for num in numbers if num % 2 == 0]
    return sum(odd_numbers), sum(even_numbers)

my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
odd_refactored, even_refactored = calculate_odd_even_sums_refactored(my_list)
print(f"Refactored Code:\nOdd sum: {odd_refactored}, Even sum: {even_refactored}")
```





Task 2: Area Calculation Explanation

❖ Scenario:

You are onboarding a junior developer.

❖ Task:

Ask Gemini to explain a function that calculates the area of different shapes.

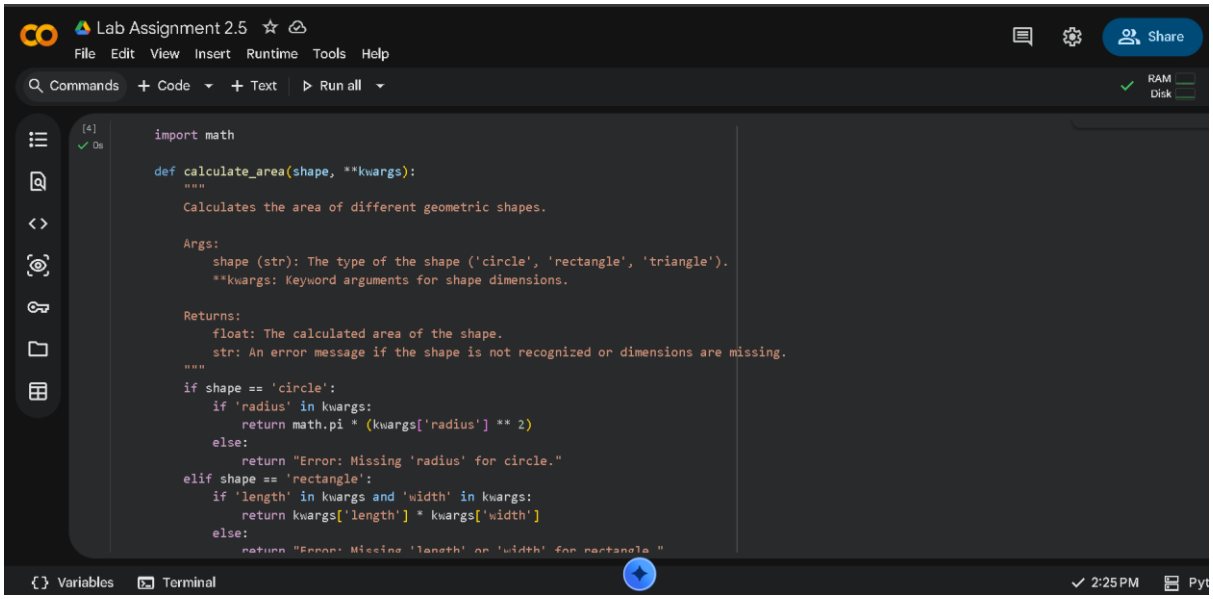
❖ Expected Output:

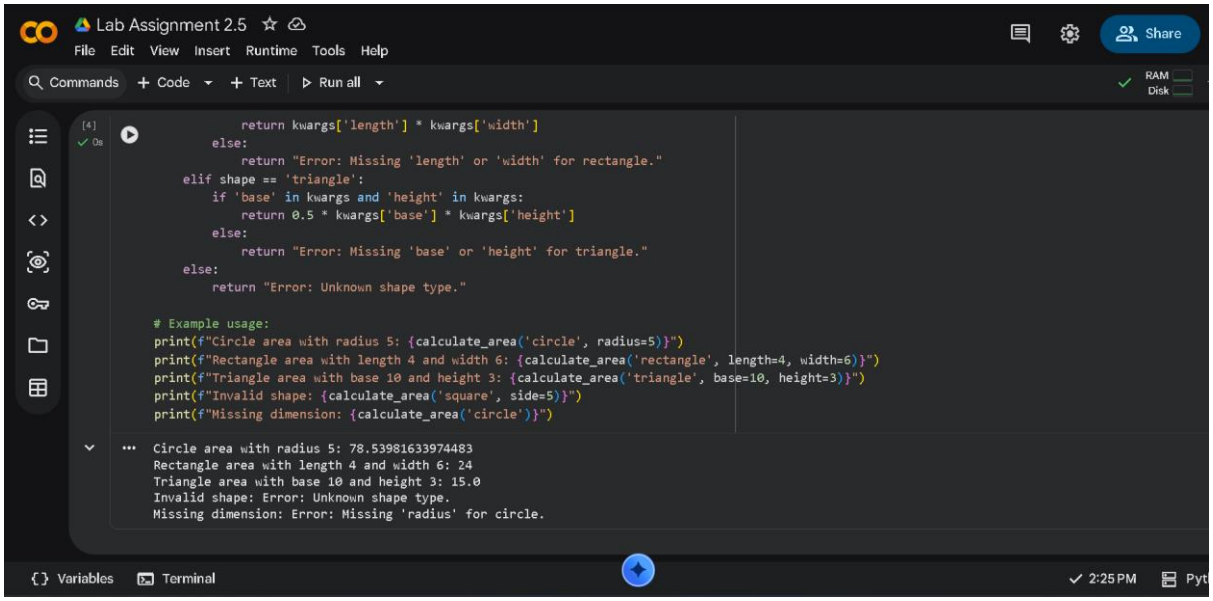
➢ Code & Explanation

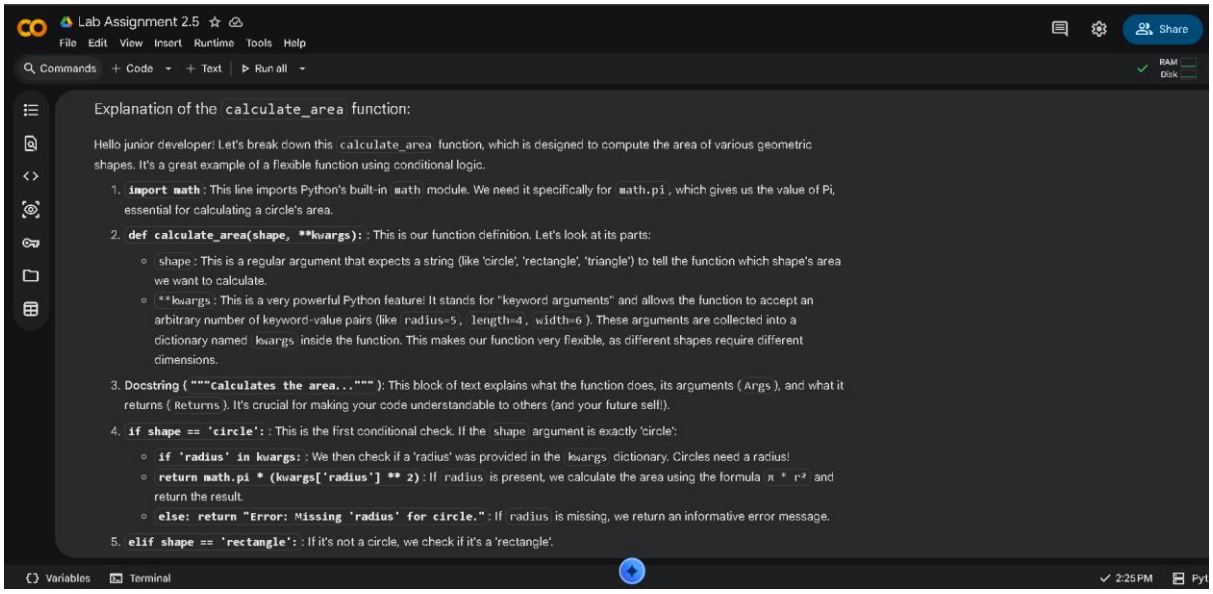
```
import math
def calculate_area(shape, **kwargs):
    """
    Calculates the area of different geometric shapes.
    Args:
        shape (str): The type of the shape ('circle', 'rectangle', 'triangle').
        **kwargs: Keyword arguments for shape dimensions.
    Returns:
        float: The calculated area of the shape.
        str: An error message if the shape is not recognized or dimensions are missing.
    """
    if shape == 'circle':
        if 'radius' in kwargs:
            return math.pi * (kwargs['radius'] ** 2)
        else:
            return "Error: Missing 'radius' for circle."
    elif shape == 'rectangle':
        if 'length' in kwargs and 'width' in kwargs:
            return kwargs['length'] * kwargs['width']
        else:
            return "Error: Missing 'length' or 'width' for rectangle."
    elif shape == 'triangle':
        if 'base' in kwargs and 'height' in kwargs:
            return 0.5 * kwargs['base'] * kwargs['height']
        else:
            return "Error: Missing 'base' or 'height' for triangle."
    else:
        return "Error: Unknown shape type."

# Example usage:
print(f"Circle area with radius 5: {calculate_area('circle', radius=5)}")
print(f"Rectangle area with length 4 and width 6: {calculate_area('rectangle', length=4, width=6)}")
print(f"Triangle area with base 10 and height 3: {calculate_area('triangle', base=10, height=3)}")
```

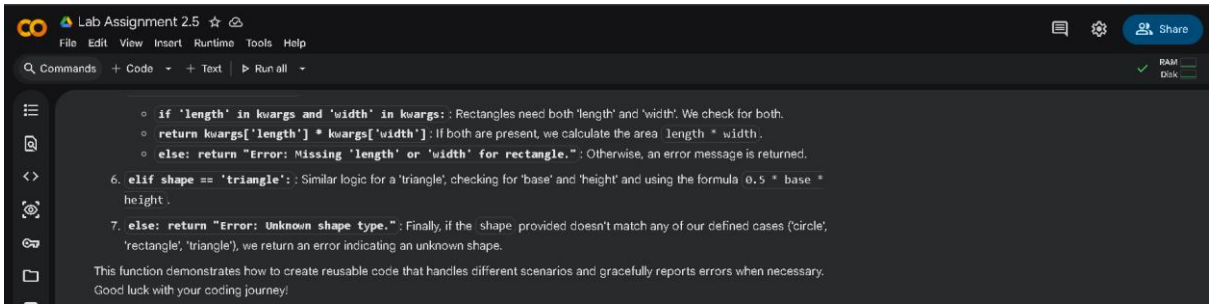
print(f"Invalid shape: {calculate_area('square', side=5)}")

print(f"Missing dimension: {calculate_area('circle')}")









Explanation:

The function accepts the shape name and required dimensions as input. Conditional statements decide which formula to apply. For a circle, radius is used, while rectangle and triangle require two values. The function returns the calculated area based on the selected shape.

Task 3: Prompt Sensitivity Experiment

❖ Scenario:

You are testing how AI responds to different prompts.

❖ Task:

Use Cursor AI with different prompts for the same problem and observe code changes.

❖ Expected Output:

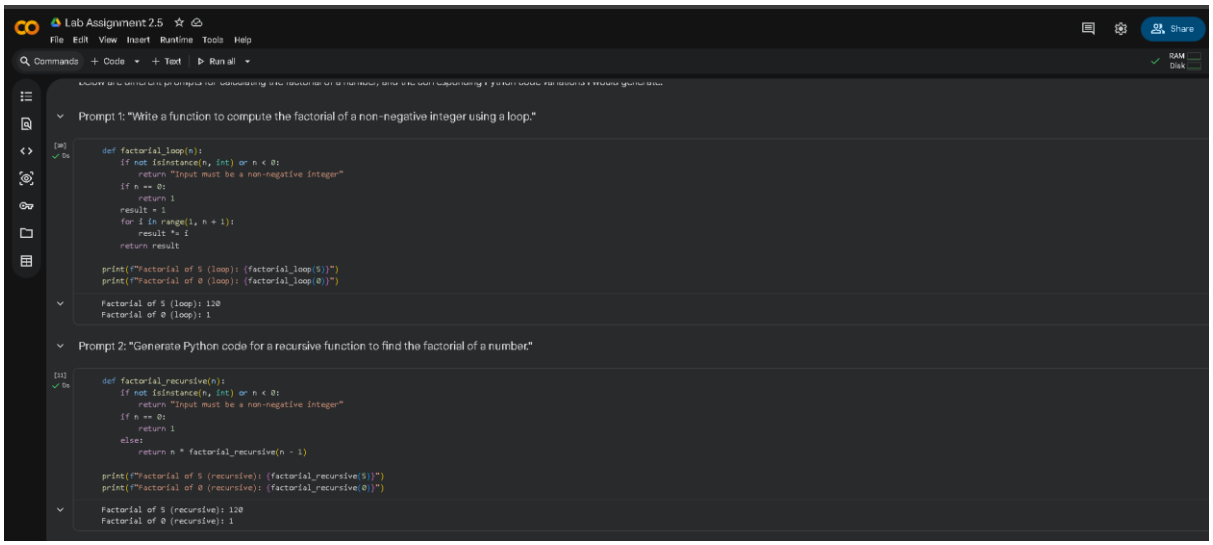
➢ Prompt list:

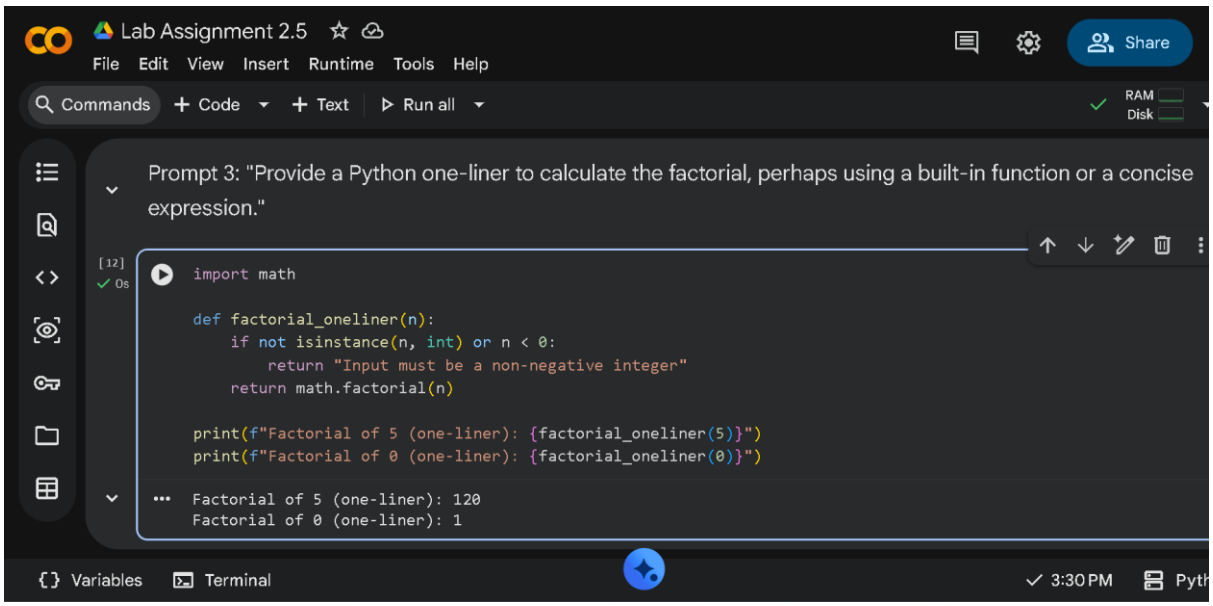
1) Write a function to compute the factorial of a non-negative integer using a loop.

2) Generate Python code for a recursive function to find the factorial of number.

3) Provide a Python one-liner to calculate the factorial, perhaps using a built-in function or a concise expression.

➢ Code variation:





Observation:

The AI produced different code styles depending on the prompt. Optimized prompts improved performance and readability, while beginner-friendly prompts added validation and safety checks.

Task 4: Tool Comparison Reflection

❖ Scenario:

You must recommend an AI coding tool.

❖ Task:

Based on your work in this topic, compare Gemini, Copilot, and Cursor AI for usability and code quality.

❖ Expected Output:

Short written reflection:

Cursor AI:

Cursor AI is a prompt-driven AI coding tool that enables code generation, refactoring, and explanation within a single environment. It is highly sensitive to prompt wording, producing noticeable changes in code structure, optimization, and documentation. Cursor AI excels in refactoring legacy code and improving readability and maintainability. Its interactive editor integration reduces context switching and improves productivity.