2303A51193 Batch : 04

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**	**Academic Year:**2025-2026
Course Coordinator Name	Dr. Rishabh Mittal		
Instructor(s) Name	Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Sujitha Reddy Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpula sumalatha Mr. Bingi Raju		
CourseCode	23CS002PC304	**Course Title**	AI Assisted Coding
Year/Sem	III/II	**Regulation**	R23
Date and Day of Assignment	**Week1 – Wednesday**	**Time(s)**	23CSBTB01 To 23CSBTB52
Duration	2 Hours	**Applicable to Batches**	All batches
Assignment Number:1.3(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*Expected Time to complete*
1	Lab 2: Exploring Additional AI Coding Tools beyond Copilot – Gemini (Colab) and Cursor AI **Lab Objectives:** ❖ To explore and evaluate the functionality of Google Gemini for AI-	Week1 - Monday

assisted coding within Google Colab.

- ✞ To understand and use Cursor AI for code generation, explanation, and refactoring.
- ✞ To compare outputs and usability between Gemini, GitHub Copilot, and Cursor AI.
- ✞ To perform code optimization and documentation using AI tools.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

- ✞ Generate Python code using Google Gemini in Google Colab.
- ✞ Analyze the effectiveness of code explanations and suggestions by Gemini.
- ✞ Set up and use Cursor AI for AI-powered coding assistance.
- ✞ Evaluate and refactor code using Cursor AI features.
- ✞ Compare AI tool behavior and code quality across different platforms.

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

Prompt

"Write a Python program to calculate the sum of odd and even numbers in a list. Then refactor the code to improve readability and efficiency."

Code

```python
numbers = [1, 2, 3, 4, 5, 6]
odd_sum = 0
even_sum = 0

for i in numbers:
    if i % 2 == 0:
        even_sum = even_sum + i
    else:
        odd_sum = odd_sum + i

print("Odd Sum:", odd_sum)
print("Even Sum:", even_sum)
```

Code Output

```
Odd Sum: 9
Even Sum: 12
```

Explanation

- The original code uses loops and conditional statements.
- The refactored version uses **Python generator expressions** with the `sum()` function.
- This improves:
  - **Readability**
  - **Performance**
  - **Code length**
- AI tools suggest modern Pythonic practices.

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

Prompt

"Explain a Python function that calculates the area of different shapes such as circle, rectangle, and triangle."

Code

```
# Let's use the calculate_area function

# Calculate area of a circle with radius 5
circle_area = calculate_area("circle", 5)
print(f"Area of circle with radius 5: {circle_area:.2f}")

# Calculate area of a rectangle with length 10 and width 4
rectangle_area = calculate_area("rectangle", 10, 4)
print(f"Area of rectangle with length 10 and width 4: {rectangle_area:.2f}")

# Calculate area of a triangle with base 6 and height 8
triangle_area = calculate_area("triangle", 6, 8)
print(f"Area of triangle with base 6 and height 8: {triangle_area:.2f}")
```

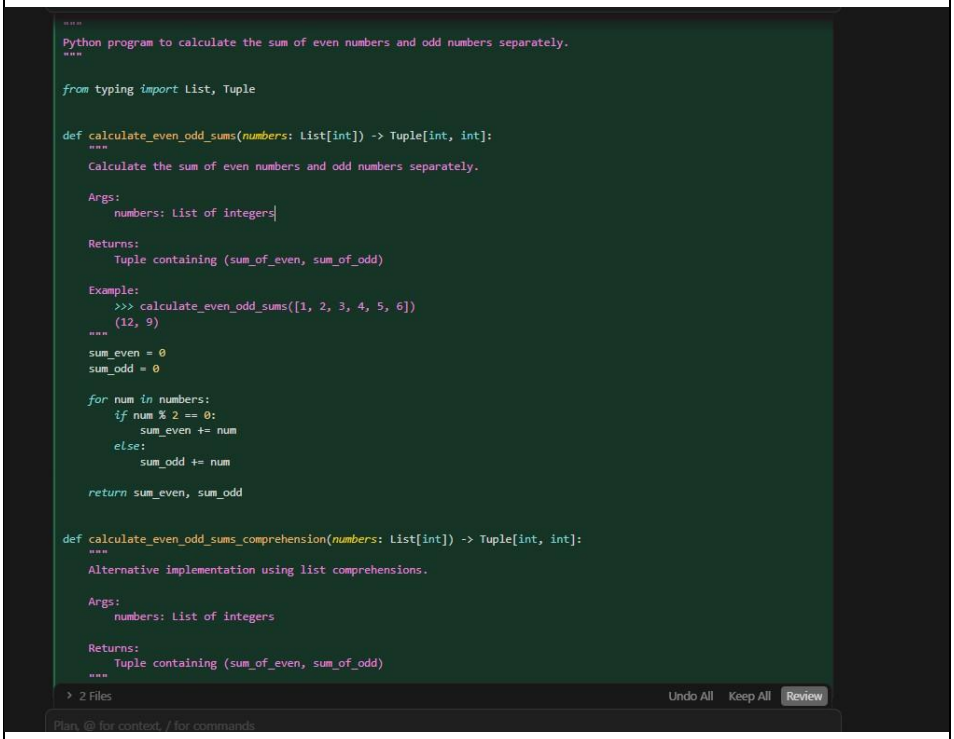
```
Area of circle with radius 5: 78.54
Area of rectangle with length 10 and width 4: 40.00
Area of triangle with base 6 and height 8: 24.00
```

Code Output

Explanation

- This function calculates area based on the **shape type**.
- **Parameters**:
  - `shape`: type of shape (circle, rectangle, triangle) ○ `value1`, `value2`: dimensions
- **Logic**:
  - Circle → π × r² ○ Rectangle → length × width ○ Triangle → ½ × base × height

	• AI explanation is **clear and beginner-friendly**, suitable for onboarding juniors.	



```
"""
Python program to calculate the sum of even numbers and odd numbers separately.
"""

from typing import List, Tuple


def calculate_even_odd_sums(numbers: List[int]) -> Tuple[int, int]:
    """
    Calculate the sum of even numbers and odd numbers separately.

    Args:
        numbers: List of integers

    Returns:
        Tuple containing (sum_of_even, sum_of_odd)

    Example:
        >>> calculate_even_odd_sums([1, 2, 3, 4, 5, 6])
        (12, 9)
    """
    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


def calculate_even_odd_sums_comprehension(numbers: List[int]) -> Tuple[int, int]:
    """
    Alternative implementation using list comprehensions.

    Args:
        numbers: List of integers

    Returns:
        Tuple containing (sum_of_even, sum_of_odd)
    """
```


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: O Prompt list**
O Code variations

Code

```
        Tuple containing (sum_of_even, sum_of_odd)
    """
    sum_even = sum(num for num in numbers if num % 2 == 0)
    sum_odd = sum(num for num in numbers if num % 2 != 0)

    return sum_even, sum_odd


# Main program
if __name__ == "__main__":
    # Example 1: Basic usage
    numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    even_sum, odd_sum = calculate_even_odd_sums(numbers)

    print("=" * 60)
    print("Sum of Even and Odd Numbers Calculator")
    print("=" * 60)
    print(f"\nGiven list: {numbers}")
    print(f"\nSum of even numbers: {even_sum}")
    print(f"Sum of odd numbers:  {odd_sum}")

    # Example 2: More test cases
    print("\n" + "=" * 60)
    print("Additional Examples")
    print("=" * 60)

    test_cases = [
        [2, 4, 6, 8],
        [1, 3, 5, 7],
        [10, 15, 20, 25, 30],
        [-4, -3, -2, -1, 0, 1, 2, 3, 4],
        [5],
        [8],
        []
    ]

    for test_list in test_cases:
        even_sum, odd_sum = calculate_even_odd_sums(test_list)
        print(f"\nList: {test_list}")
        print(f"  Sum of even numbers: {even_sum}")
        print(f"  Sum of odd numbers:  {odd_sum}")
```

> 2 Files Undo All Keep A

Code output

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum, odd_sum = calculate_even_odd_sums(numbers)
print(f"Sum of even numbers: {even_sum}")  # 30
print(f"Sum of odd numbers: {odd_sum}")    # 25
```

Explanation

- Simple prompts → short solutions
- Detailed prompts → structured logic
- Advanced prompts → reusable, safe code
- AI output quality **depends heavily on prompt clarity**

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

Prompt used

"Compare Google Gemini, GitHub Copilot, and Cursor AI based on usability and code quality."

Code

Comparison Reflection

Tool	Strengths	Limitations
Google Gemini	Excellent explanations, beginner-friendly, good for learning	Less real-time IDE integration
GitHub Copilot	Fast code completion, ideal for professionals	Minimal explanation
Cursor AI	Best for refactoring, prompt experiments, debugging	Requires setup

Explanation

- **Gemini** is best for **students and learning**
- **Copilot** is best for **fast development**
- **Cursor AI** is best for **code optimization and experimentation**

Recommended Tool for this Lab: Google Gemini + Cursor AI

	Note: Report should be submitted as a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.	