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

NAME:SUMANTH AKARAPU

BATCH:34

SUBJECT:ASSISTANT CODING

SEMESTER-VI

Tools Used: Google Gemini (Google Colab), Cursor AI, GitHub Copilot
Language: Python

Lab Objectives

- To explore and evaluate Google Gemini for AI-assisted coding in 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

After completing this lab, the student is able to:

- Generate Python code using Google Gemini
- Analyze AI-generated explanations
- Use Cursor AI for refactoring and prompt-based coding
- Compare AI tools based on usability and code quality

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

Scenario

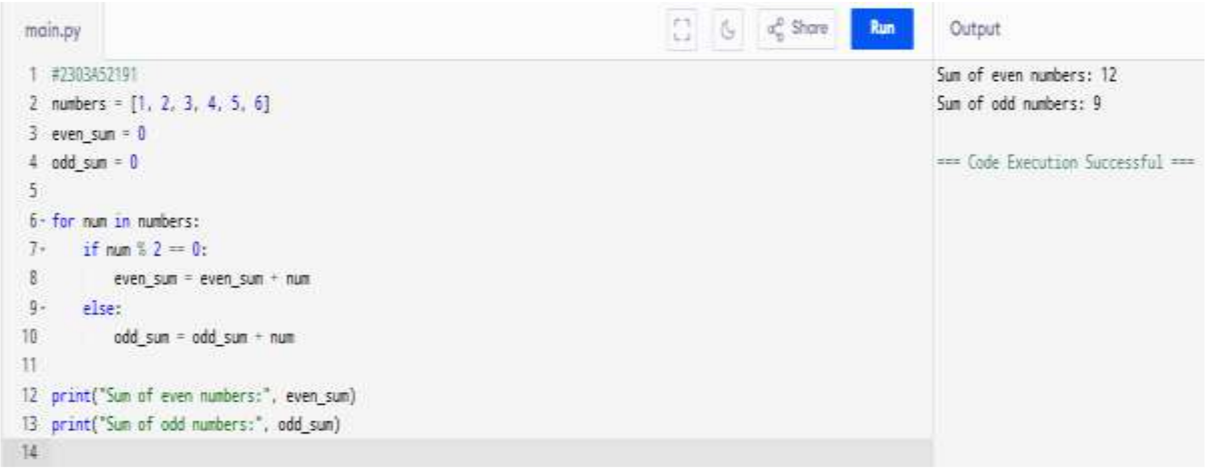
You are improving legacy code.

Task

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

Original Code (Before AI Refactoring)

```
numbers = [1, 2, 3, 4, 5, 6]

even_sum = 0

odd_sum = 0


for num in numbers:

    if num % 2 == 0:

        even_sum = even_sum + num

    else:

        odd_sum = odd_sum + num


print("Sum of even numbers:", even_sum)

print("Sum of odd numbers:", odd_sum)
```

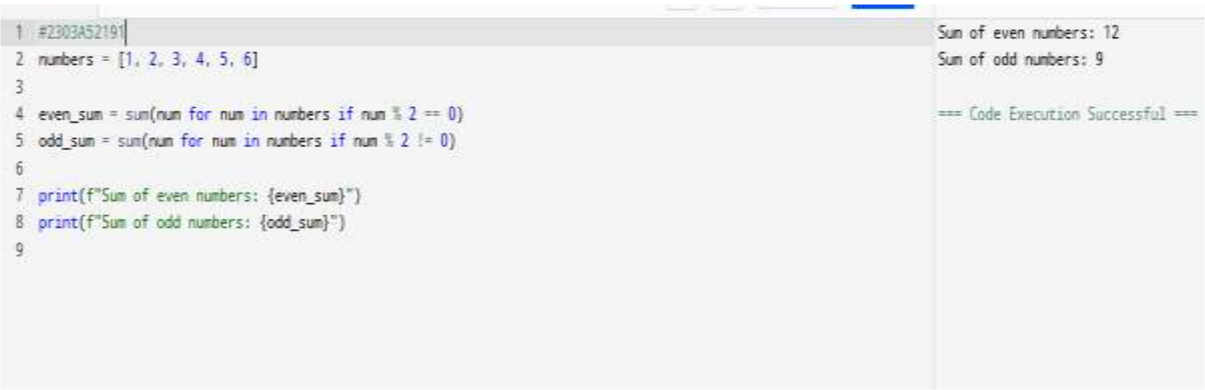


```
#2303A52191
numbers = [1, 2, 3, 4, 5, 6]
even_sum = 0
odd_sum = 0

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

print("Sum of even numbers:", even_sum)
print("Sum of odd numbers:", odd_sum)
```

```
Sum of even numbers: 12
Sum of odd numbers: 9

=== Code Execution Successful ===
```

AI-Refactored Code (Using Gemini / Cursor AI)

```
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(f"Sum of even numbers: {even_sum}")

print(f"Sum of odd numbers: {odd_sum}")
```



```python
#2303A52191
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(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")

```

```
Sum of even numbers: 12
Sum of odd numbers: 9

=== Code Execution Successful ===
```

Improvement Explanation

- Reduced number of variables
- Cleaner and more readable logic
- Uses Pythonic constructs (sum() and comprehensions)
- Easier to maintain and extend

Task 2: Area Calculation Explanation (Using Gemini)

Scenario

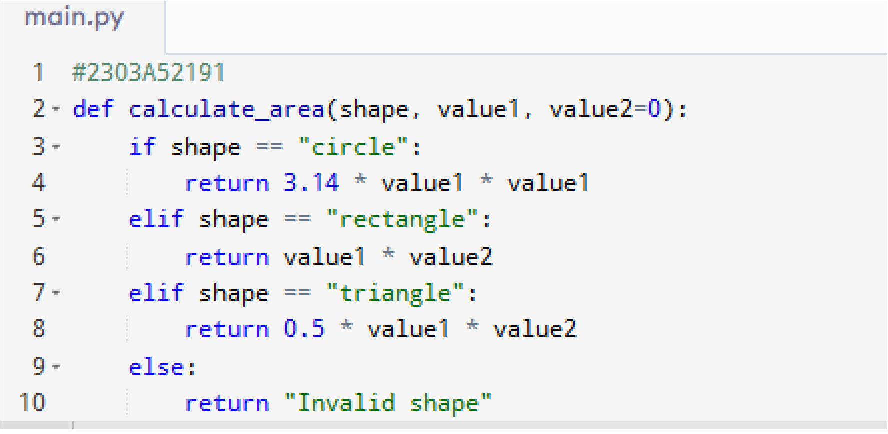
You are onboarding a junior developer.

Task

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

Prompt Given to Gemini

"Explain the Python function that calculates the area of a circle, rectangle, and triangle with examples."



main.py

```python
#2303A52191
def calculate_area(shape, value1, value2=0):
    if shape == "circle":
        return 3.14 * value1 * value1
    elif shape == "rectangle":
        return value1 * value2
    elif shape == "triangle":
        return 0.5 * value1 * value2
    else:
        return "Invalid shape"
```

Code Generated by Gemini

```
def calculate_area(shape, value1, value2=0):
    if shape == "circle":
        return 3.14 * value1 * value1
    elif shape == "rectangle":
        return value1 * value2
    elif shape == "triangle":
        return 0.5 * value1 * value2
    else:
        return "Invalid shape"
```

Explanation (Gemini Output Summary)

- The function uses conditional statements to identify the shape
- For a circle, value1 represents the radius
- For rectangle and triangle, value1 and value2 represent dimensions
- The function returns the calculated area based on the shape

---

Task 3: Prompt Sensitivity Experiment (Using Cursor AI)

Scenario

You are testing how AI responds to different prompts.

Task

Use Cursor AI with different prompts for the same problem.

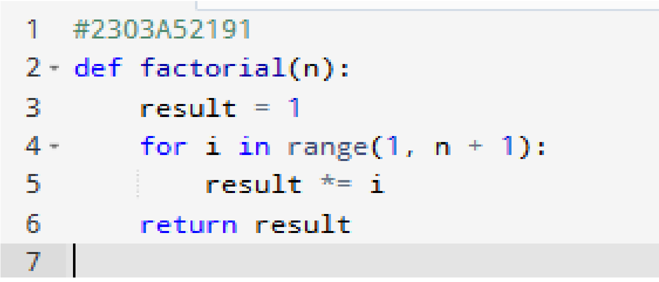
Problem Statement

Calculate factorial of a number.

Prompt 1

“Write a simple Python program to calculate factorial.”

```
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i

    return result
```

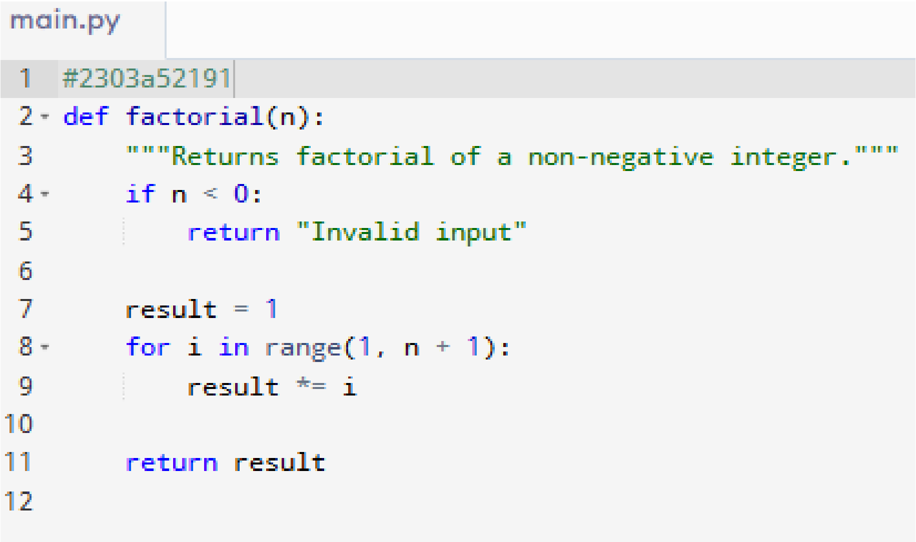


```
#2303A52191
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

```

Prompt 2

“Write an optimized and readable factorial program with comments.”

```
def factorial(n):

    """Returns factorial of a non-negative integer."""

    if n < 0:

        return "Invalid input"


    result = 1

    for i in range(1, n + 1):

        result *= i


    return result
```



```
#2303a52191
def factorial(n):
    """Returns factorial of a non-negative integer."""
    if n < 0:
        return "Invalid input"

    result = 1
    for i in range(1, n + 1):
        result *= i

    return result

```

Observation

- More detailed prompts produced cleaner, documented code
- Cursor AI adapts code quality based on prompt clarity

Task 4: Tool Comparison Reflection

Scenario

You must recommend an AI coding tool.

Comparison: Gemini vs Copilot vs Cursor AI

Google Gemini

- Excellent for explanations and beginner understanding
- Strong natural-language explanations
- Best suited for learning and onboarding

GitHub Copilot

- Fast inline suggestions
- Best for experienced developers
- Requires careful review

Cursor AI

- Strong refactoring and prompt sensitivity
- Best balance between explanation and code quality
- Excellent for code cleanup and optimization

Recommendation

For students and learning environments, **Gemini** is ideal.
For professional development and refactoring, **Cursor AI** is the best choice.
For real-time coding speed, **GitHub Copilot** is most effective.