AI ASSISTED CODING

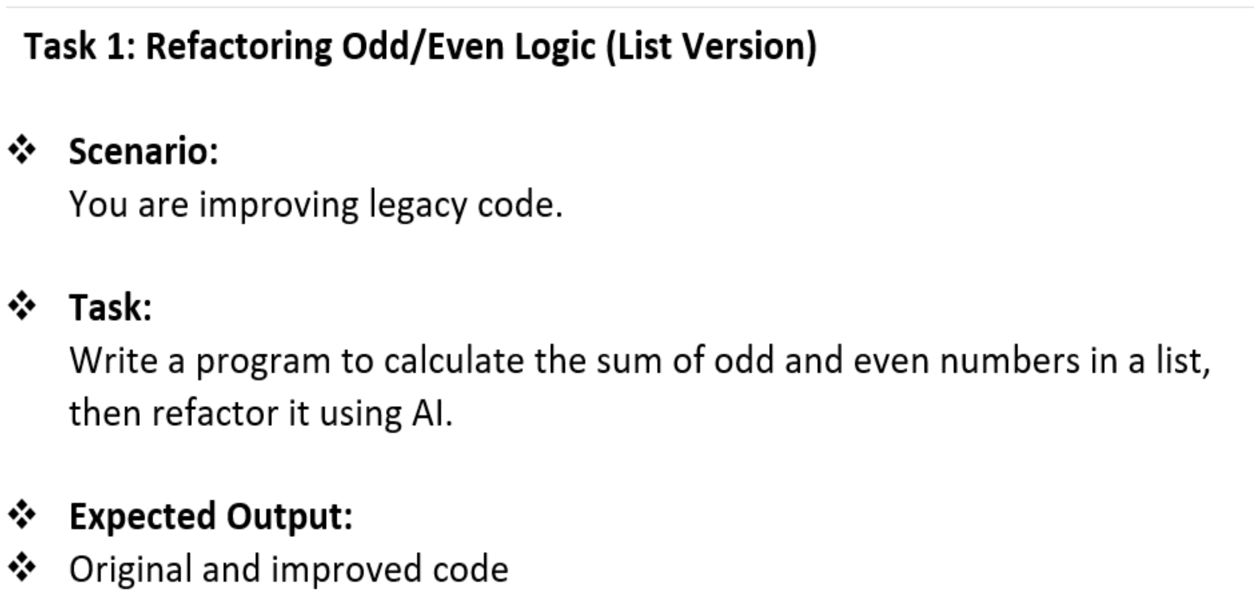
ASSIGNMENT - 2.5

NAME-S.SAI

H:NO-2303A52492

BATCH-50

TASK-1



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.

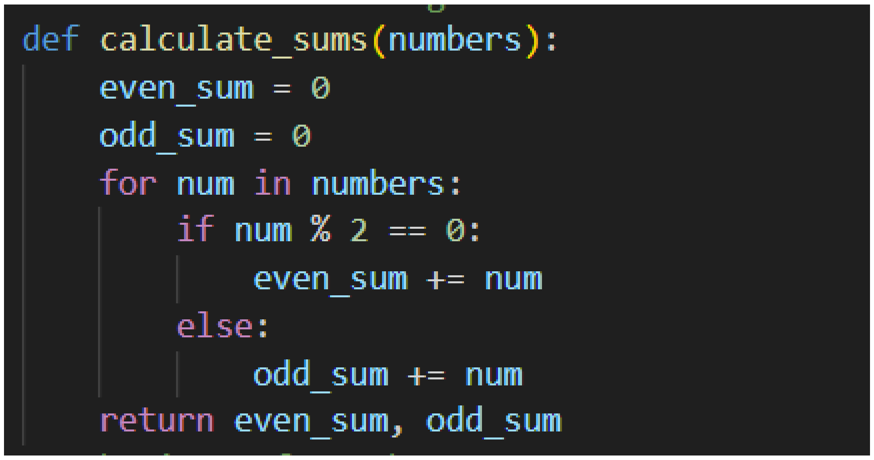
❖ **Expected Output:**

❖ Original and improved code

PROMPT:

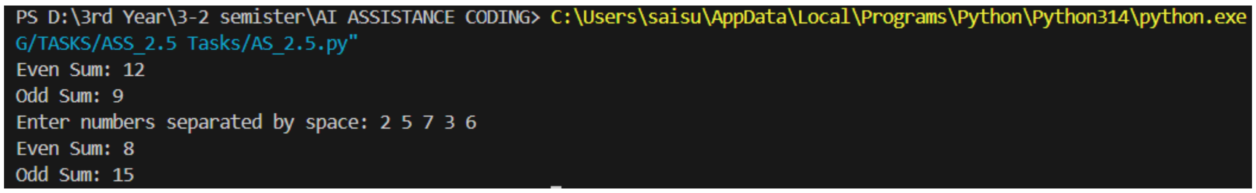
"Hey, I have this old Python script that uses a basic for-loop and `range(len())` to find the sum of odd and even numbers in a list. It feels a bit clunky. Can you refactor this to be more 'Pythonic' using list comprehensions or built-in functions? Show me the original vs the new version."

CODE LOGIC:



```python
def calculate_sums(numbers):
    even_sum = 0
    odd_sum = 0
    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num
    return even_sum, odd_sum
```

OUTPUT:



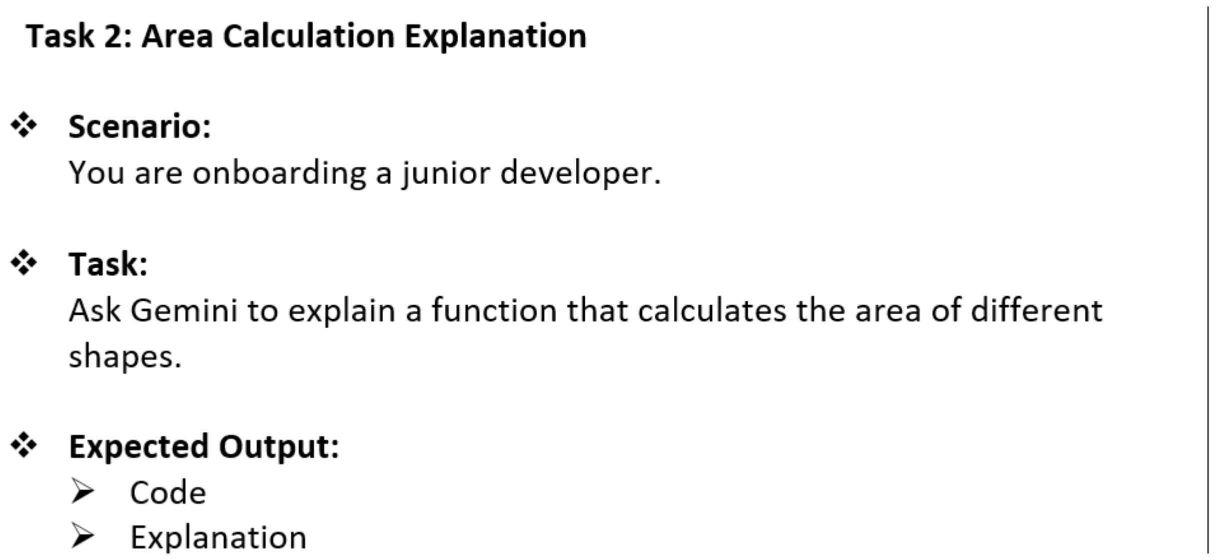
```
PS D:\3rd Year\3-2 semister\AI ASSISTANCE CODING> C:\Users\saisu\AppData\Local\Programs\Python\Python314\python.exe
G/TASKS/ASS_2.5 Tasks/AS_2.5.py"
Even Sum: 12
Odd Sum: 9
Enter numbers separated by space: 2 5 7 3 6
Even Sum: 8
Odd Sum: 15
```

JUSTIFICATION:

This works because it gives the AI a **role** (cleaner of legacy code) and specifies a **target style** (Pythonic/List Comprehension). This prevents the AI from just giving you the same code back and forces it to optimize for readability.

TASK-2

QUESTION:



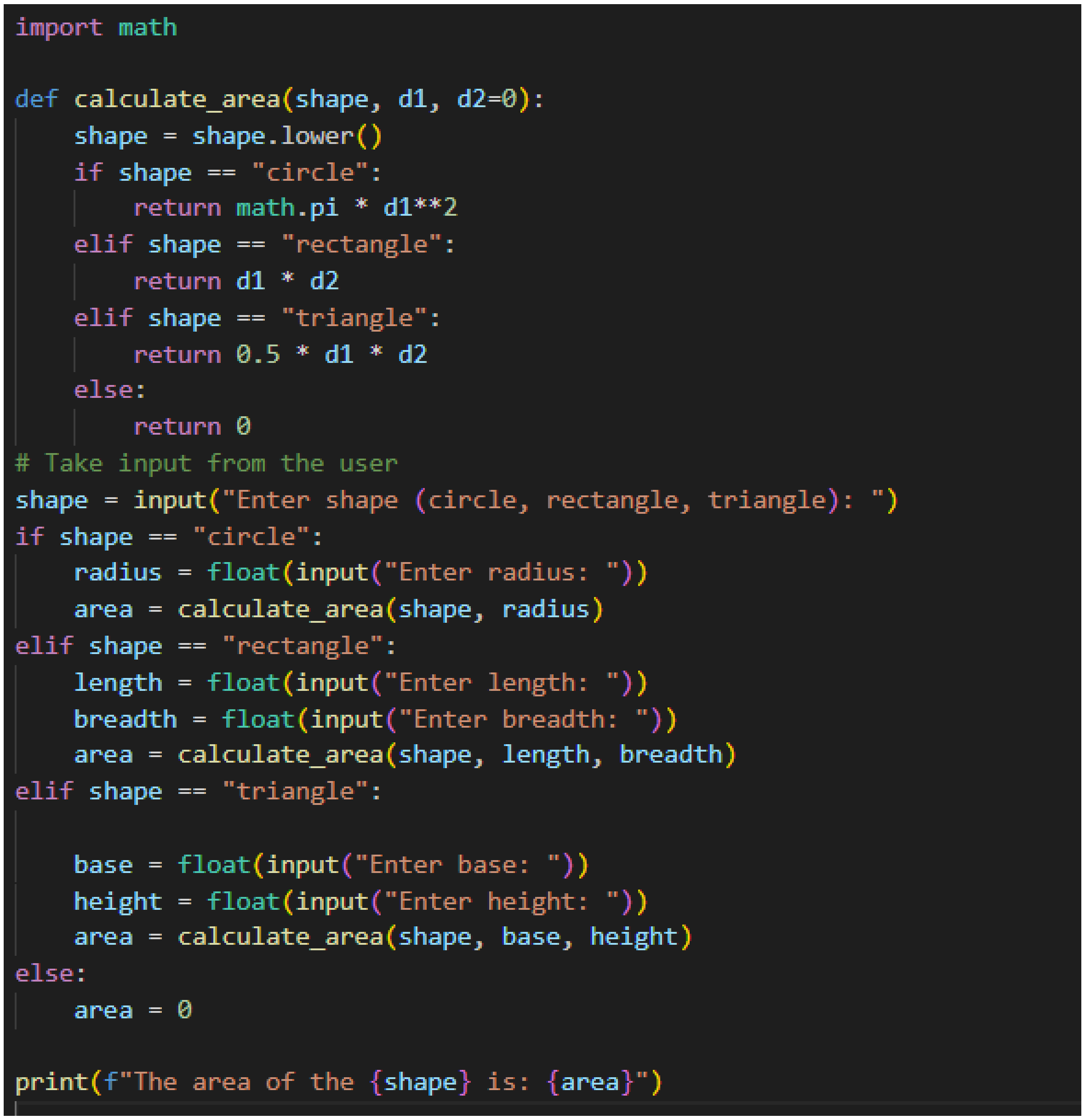
Task 2: Area Calculation Explanation

- ❖ **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:

"I'm onboarding a junior developer today. Can you write a Python function that calculates the area for a circle, rectangle, and triangle? After the code, give a simple, friendly explanation of how it works—especially why we use `math.pi` and how we handle different inputs—so the junior can follow along easily."

CODE:



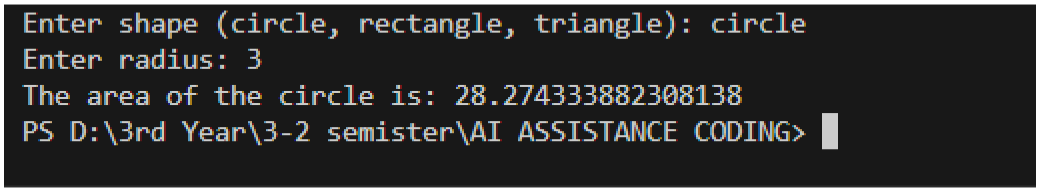
```python
import math

def calculate_area(shape, d1, d2=0):
    shape = shape.lower()
    if shape == "circle":
        return math.pi * d1**2
    elif shape == "rectangle":
        return d1 * d2
    elif shape == "triangle":
        return 0.5 * d1 * d2
    else:
        return 0
# Take input from the user
shape = input("Enter shape (circle, rectangle, triangle): ")
if shape == "circle":
    radius = float(input("Enter radius: "))
    area = calculate_area(shape, radius)
elif shape == "rectangle":
    length = float(input("Enter length: "))
    breadth = float(input("Enter breadth: "))
    area = calculate_area(shape, length, breadth)
elif shape == "triangle":

    base = float(input("Enter base: "))
    height = float(input("Enter height: "))
    area = calculate_area(shape, base, height)
else:
    area = 0

print(f"The area of the {shape} is: {area}")
```

OUTPUT:

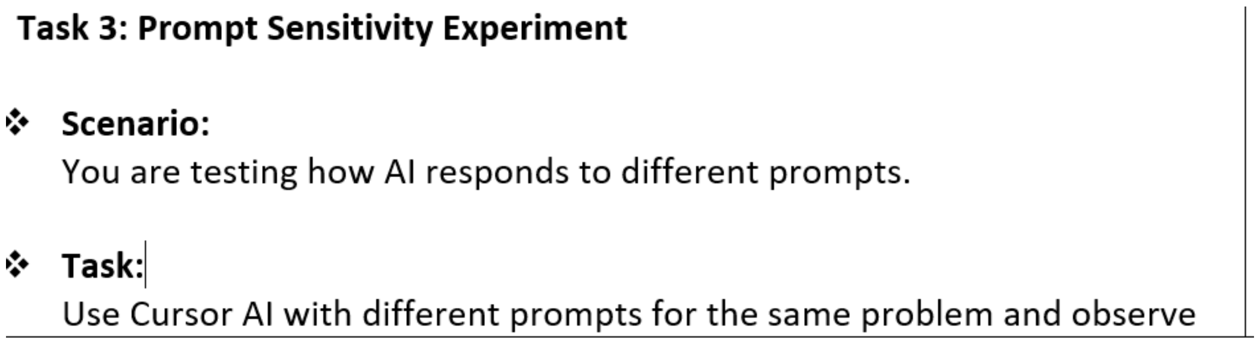


```
Enter shape (circle, rectangle, triangle): circle
Enter radius: 3
The area of the circle is: 28.274333882308138
PS D:\3rd Year\3-2 semister\AI ASSISTANCE CODING>
```

JUSTIFICATION:

By setting a **persona** (onboarding a junior), you ensure the AI doesn't just give a technical summary. It will use simpler language and focus on the "why" behind the code, which is exactly what your lab task requires.

TASK 3



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

PROMPT:

Prompt A: "Write a function to find the max value in a list."
Prompt B: "Write a function to find the max value in a list, but do it manually using a for-loop and don't use any built-in functions like `max()`."

CODE:

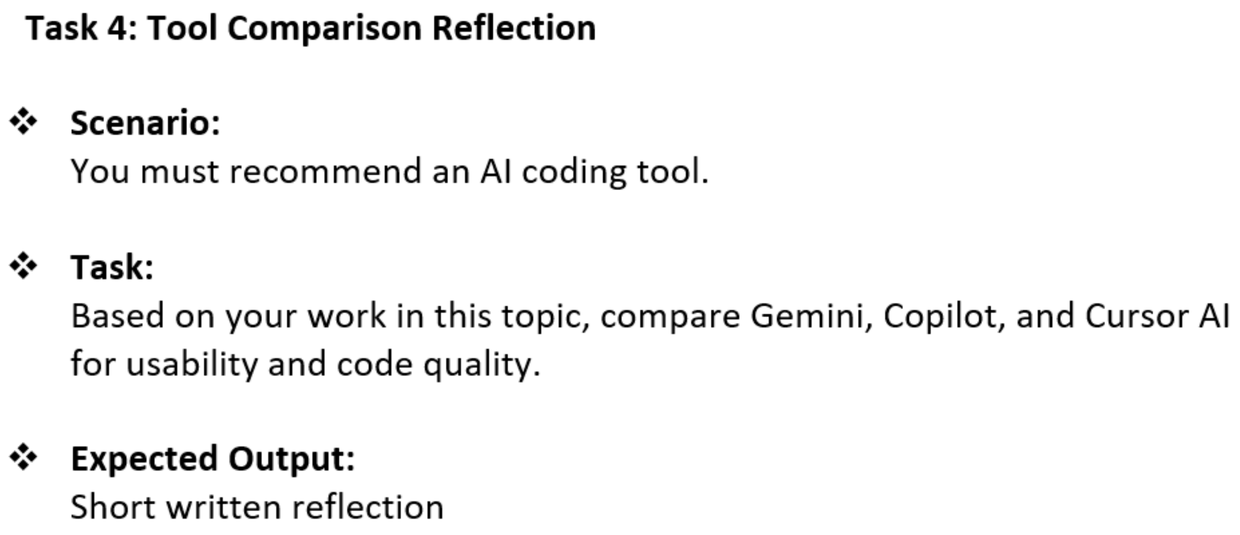
```
def get_max(lst):
    return max(lst)
```

JUSTIFICATION:

This is a **constraint-based experiment**. Prompt A lets the AI be "lazy" and efficient, while Prompt B forces it to show its algorithmic logic. Comparing these shows you how much "control" you have over the AI's output by just changing a few words.

TASK 4

QUESTION:



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:

"I've been using Gemini in Colab, GitHub Copilot, and Cursor AI for these lab tasks. Can you help me summarize the differences? Focus on which one felt best for explaining code to a human and which one was better at refactoring my files. Give me a short paragraph I can use in my lab report."

Justification:

This is a synthesis prompt. It asks the AI to look back at the work you've done and help you verbalize your thoughts into a professional reflection. It's perfect for the "Conclusion" or "Reflection" section of your SRU lab report.