ASSIGNMENT-2.5

Name-Arsha vardhini

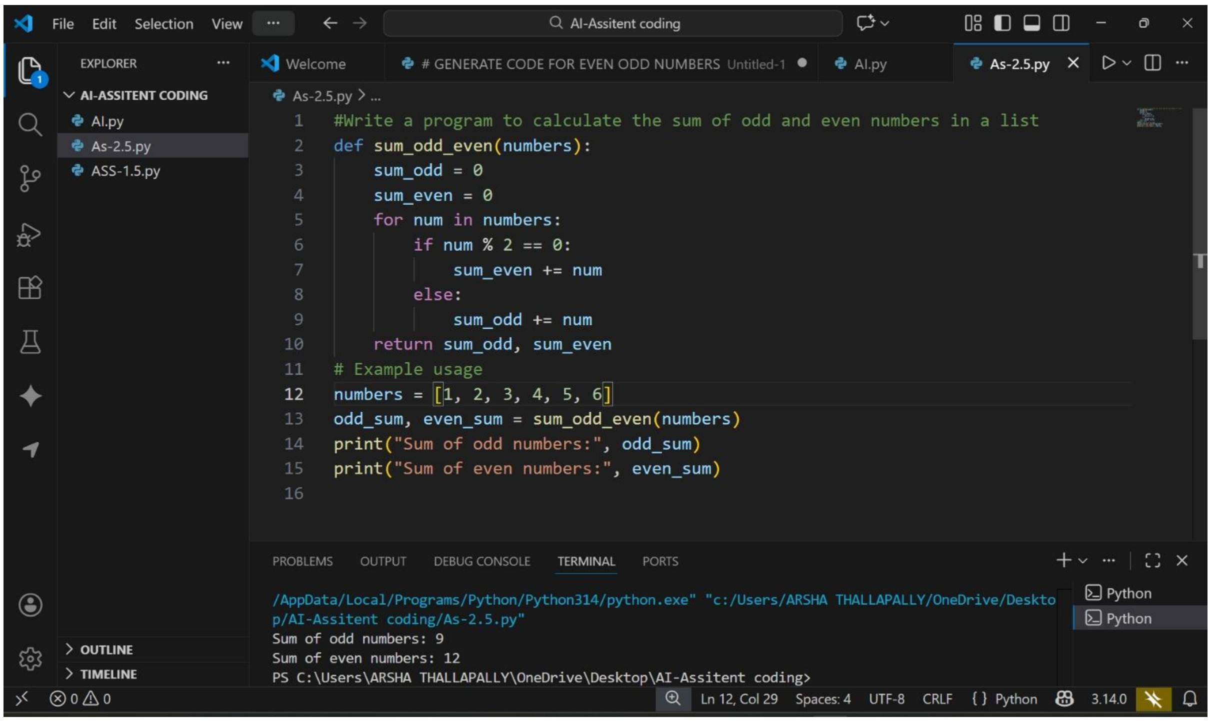
Roll no – 2303A51600

Batch-29

Task-1:

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

Code:



```python
#Write a program to calculate the sum of odd and even numbers in a list
def sum_odd_even(numbers):
    sum_odd = 0
    sum_even = 0
    for num in numbers:
        if num % 2 == 0:
            sum_even += num
        else:
            sum_odd += num
    return sum_odd, sum_even
# Example usage
numbers = [1, 2, 3, 4, 5, 6]
odd_sum, even_sum = sum_odd_even(numbers)
print("Sum of odd numbers:", odd_sum)
print("Sum of even numbers:", even_sum)
```

```
/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/ARSHA THALLAPALLY/OneDrive/Desktop/AI-Assitent coding/As-2.5.py"
Sum of odd numbers: 9
Sum of even numbers: 12
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

Observation:

The **original code** works correctly but is written as a single block, making it harder to reuse and test.

The **refactored (AI-improved) code** separates logic into a function, improving:

- **Readability**
- **Reusability**
- **Maintainability**

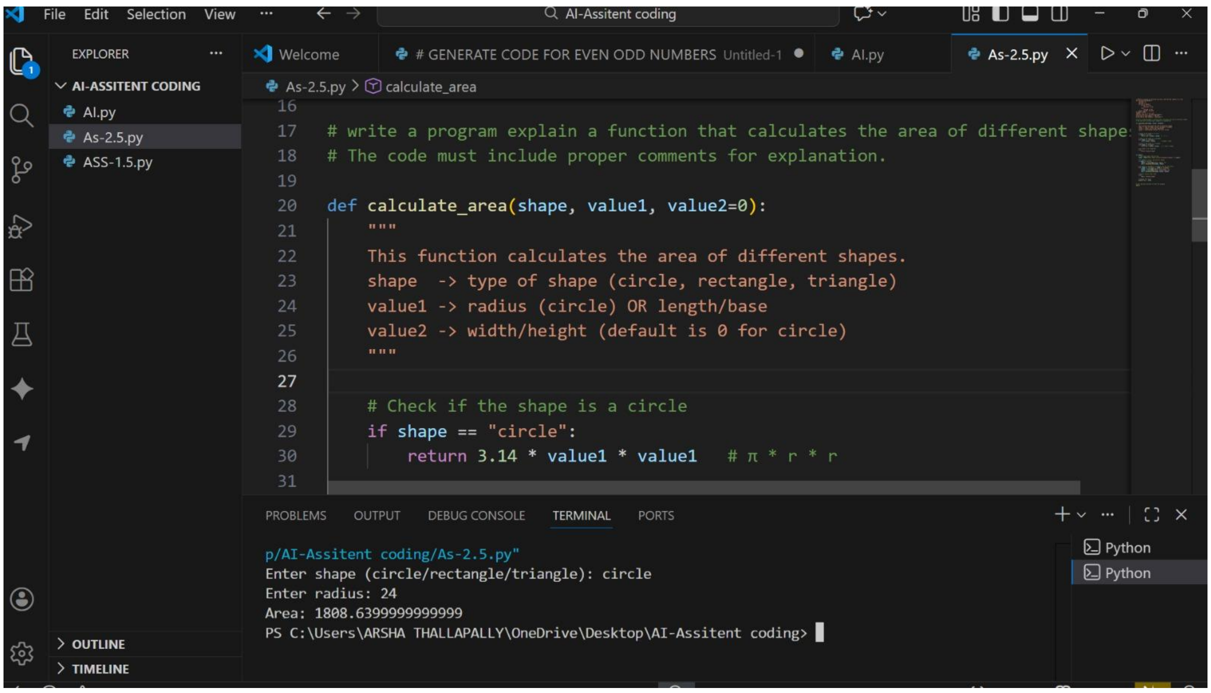
Using a function allows the same logic to be reused with different lists without rewriting code.

Task-2:

Prompt: write a program explain a function that calculates the area of different shapes.

The code must include proper comments for explanation.

Code:



```python

# write a program explain a function that calculates the area of different shape
# The code must include proper comments for explanation.

def calculate_area(shape, value1, value2=0):
    """
    This function calculates the area of different shapes.
    shape  -> type of shape (circle, rectangle, triangle)
    value1 -> radius (circle) OR length/base
    value2 -> width/height (default is 0 for circle)
    """

    # Check if the shape is a circle
    if shape == "circle":
        return 3.14 * value1 * value1   # π * r * r

```

```
p/AI-Assitent coding/As-2.5.py"
Enter shape (circle/rectangle/triangle): circle
Enter radius: 24
Area: 1808.6399999999999
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

Observation:

This program uses **one function** to calculate the area of **multiple shapes**, which avoids code duplication.

The shape parameter decides **which formula** to apply.

The function uses **conditional statements** (if / elif) to select the correct formula.

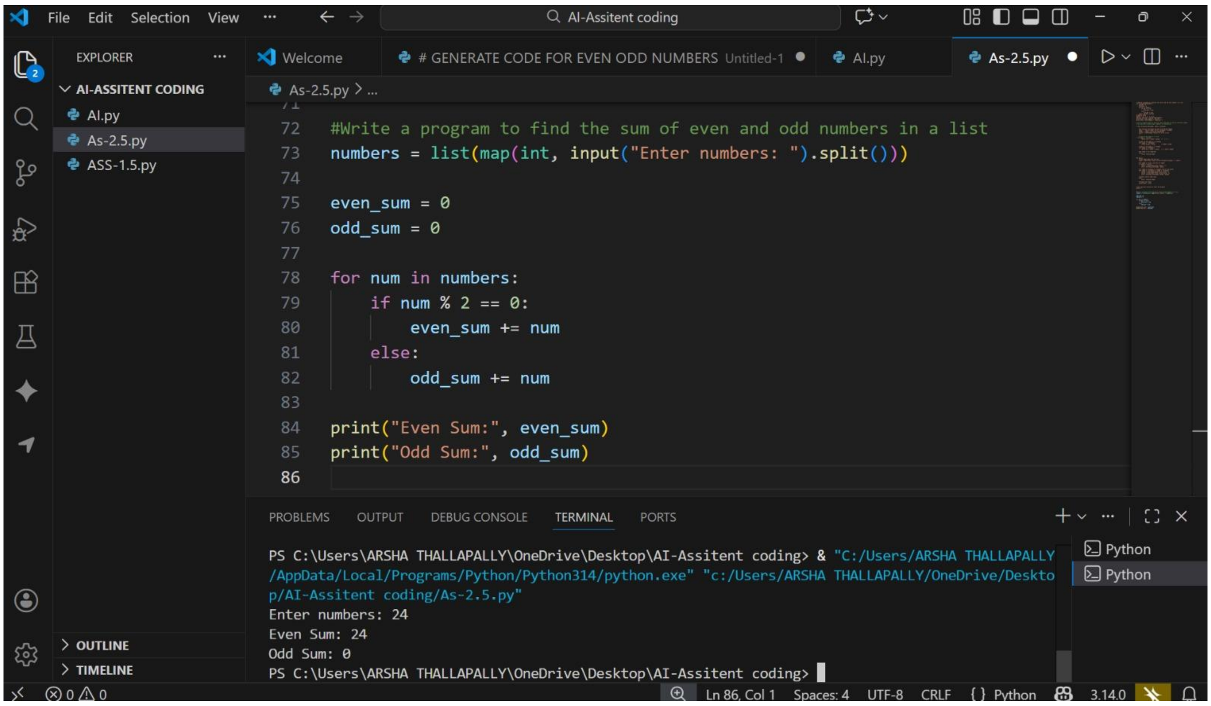
It improves **code clarity**, making onboarding easier and faster.

Task:3

Prompt: explain a function that calculates the area of different shapes (curser used)

Shapes.   Write a program to find the sum of even and odd numbers in a list

Code:



```python
#Write a program to find the sum of even and odd numbers in a list
numbers = list(map(int, input("Enter numbers: ").split()))

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)
```

```
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding> & "C:/Users/ARSHA THALLAPALLY
/AppData/Local/Programs/Python/Python314/python.exe" "c:/Users/ARSHA THALLAPALLY/OneDrive/Deskto
p/AI-Assitent coding/As-2.5.py"
Enter numbers: 24
Even Sum: 24
Odd Sum: 0
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

Observation:

The program demonstrates **how one function can handle multiple use cases**.

Comments clearly explain:

What the function does

Why each condition exists

What each parameter represents

Using comments makes the code **junior-developer friendly**, which is ideal for onboarding.

The main () function separates **user interaction** from **business logic**, improving structure.

This style is considered **clean, readable, and professional** in real-world projects.

Task-4:

Prompt: Based on practical usage and experimentation, compare **Gemini**, **GitHub Copilot**, and **Cursor AI** in terms of **usability** and **code quality**.

Observation:

Gemini is best suited for **explanations and learning support**. It produces readable, beginner-friendly code and clear step-by-step reasoning, making it ideal for onboarding juniors and understanding concepts.

GitHub Copilot excels in **real-time coding assistance** inside IDEs. It is fast, context-aware, and highly productive for experienced developers, but its code may lack explanations.

Cursor AI stands out for **prompt sensitivity and refactoring quality**. It responds strongly to detailed prompts, generating cleaner, more structured, and optimized code, making it suitable for improving legacy codebases.

usability, Copilot integrates seamlessly into workflows, Gemini is conversational and educational, and Cursor AI offers powerful prompt-driven refactoring.

code quality, Cursor AI and Copilot generally produce more professional, production-ready code, while Gemini focuses on clarity over optimization