

ASSIGNMENT-2.5

Name-A. Sravani

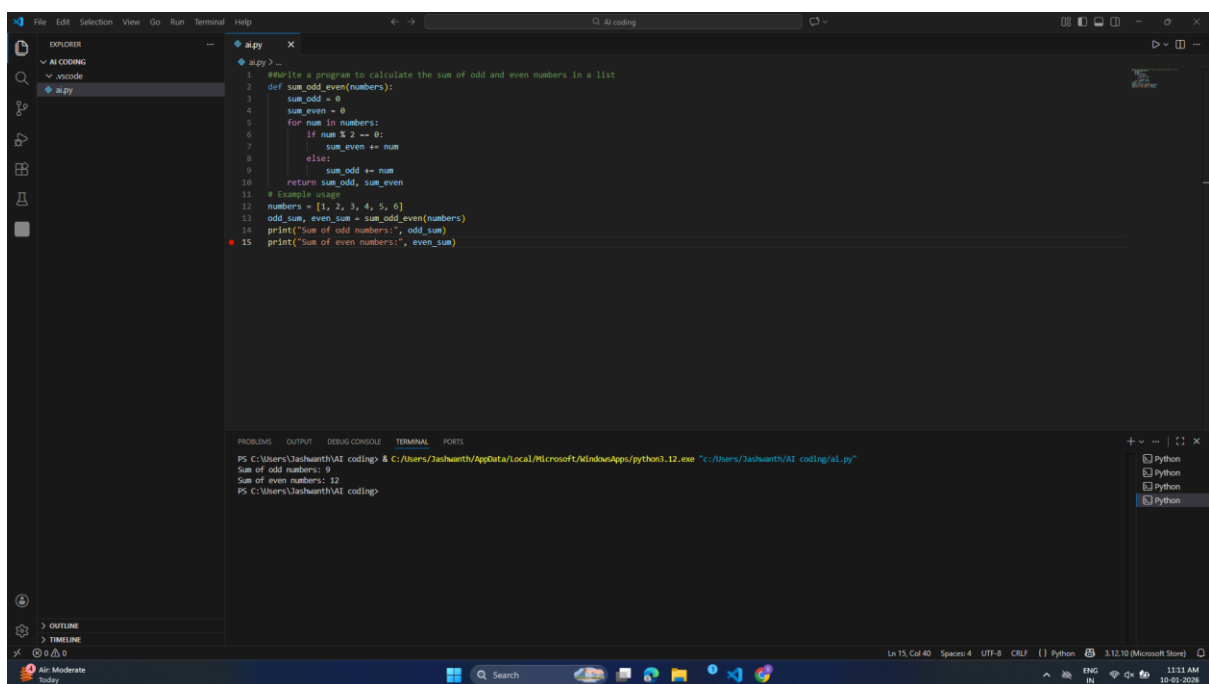
Roll no – 2303A510G7

Batch-30

Task-1:

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

Code:



The screenshot shows a Visual Studio Code editor window with a Python file named 'ai.py'. The code defines a function 'sum_odd_even' that takes a list of numbers and returns the sum of odd and even numbers. It then uses this function to calculate the sum for a specific list of numbers. The terminal output shows the execution of the script, displaying the sum of odd numbers (9) and the sum of even numbers (12).

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Jashwanth\AI coding> C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python3.12.exe "C:\Users\Jashwanth\AI coding\ai.py"

Sum of odd numbers: 9
Sum of even numbers: 12
PS C:\Users\Jashwanth\AI 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

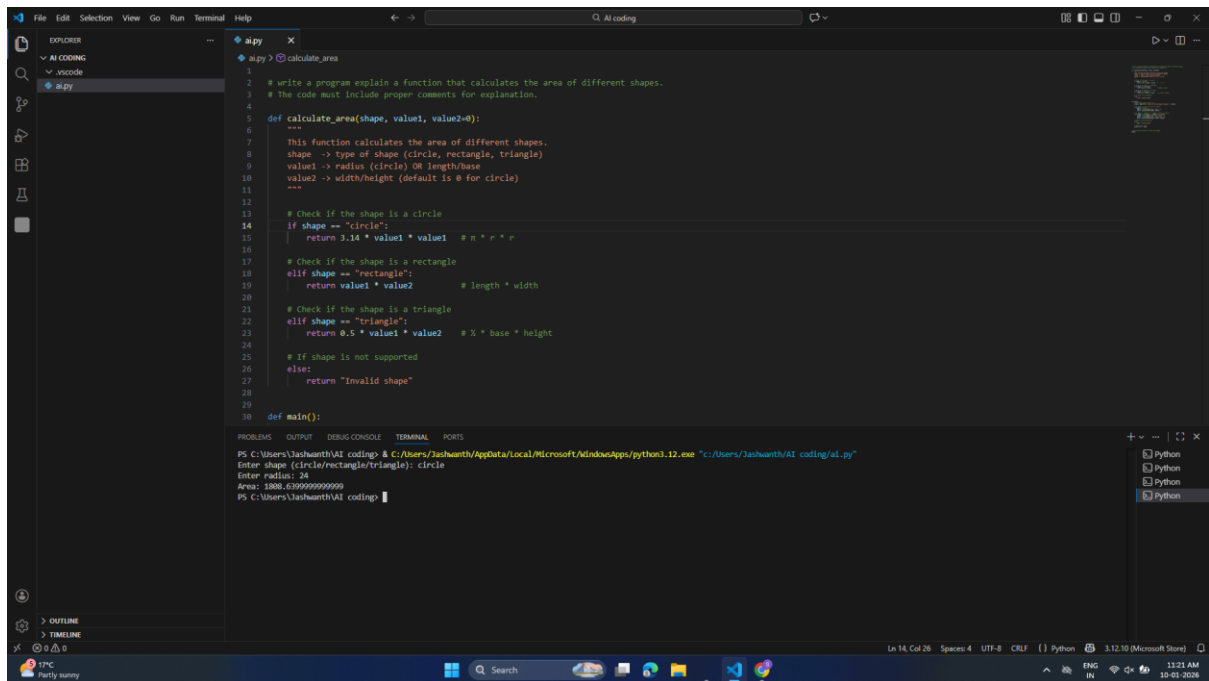
Using a function allows the same logic to be reused with different lists without rewriting code.

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:



```
1 # write a program explain a function that calculates the area of different shapes.
2 # The code must include proper comments for explanation.
3
4
5 def calculate_area(shape, value1, value2=0):
6     """
7     This function calculates the area of different shapes.
8     shape -> type of shape (circle, rectangle, triangle)
9     value1 -> radius (circle) OR length/base
10    value2 -> width/height (default is 0 for circle)
11    """
12
13    # Check if the shape is a circle
14    if shape == "circle":
15        return 3.14 * value1 * value1 # π * r * r
16
17    # Check if the shape is a rectangle
18    elif shape == "rectangle":
19        return value1 * value2 # length * width
20
21    # Check if the shape is a triangle
22    elif shape == "triangle":
23        return 0.5 * value1 * value2 # 1/2 * base * height
24
25    # If shape is not supported
26    else:
27        return "Invalid shape"
28
29
30 def main():
31
32    PS C:\Users\Jashwanth\AI codings> & C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python3.12.exe "C:\Users\Jashwanth\AI codings\ai.py"
33    Enter shape (circle/rectangle/triangle): circle
34    Enter radius: 24
35    Area: 1808.6399999999999
36    PS C:\Users\Jashwanth\AI codings>
```

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.

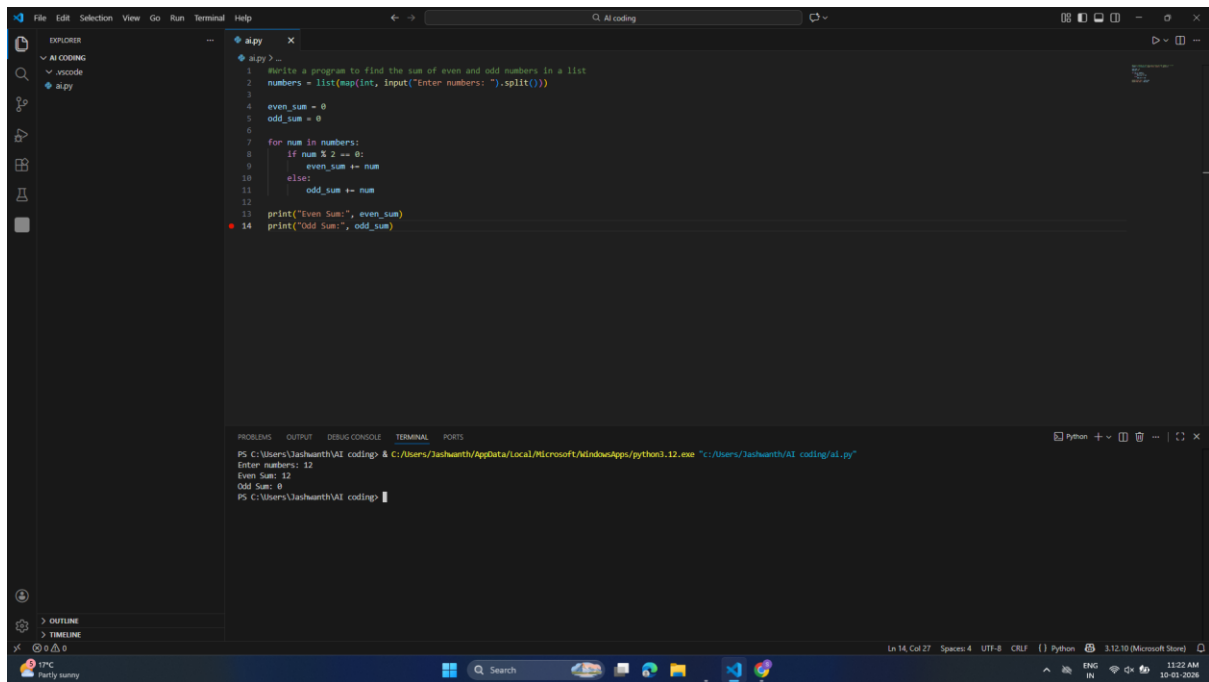
It improves **code clarity**, making onboarding easier and faster.

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:



```
1 # Write a program to find the sum of even and odd numbers in a list
2 numbers = list(map(int, input("Enter numbers: ").split()))
3
4 even_sum = 0
5 odd_sum = 0
6
7 for num in numbers:
8     if num % 2 == 0:
9         even_sum += num
10    else:
11        odd_sum += num
12
13 print("Even Sum:", even_sum)
14 print("Odd Sum:", odd_sum)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python3.12.exe "C:\Users\Jashwanth\AI coding\al.py"

Enter numbers: 12
Even Sum: 12
Odd Sum: 0
PS C:\Users\Jashwanth\AI 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