

Lab assignment- 2.5

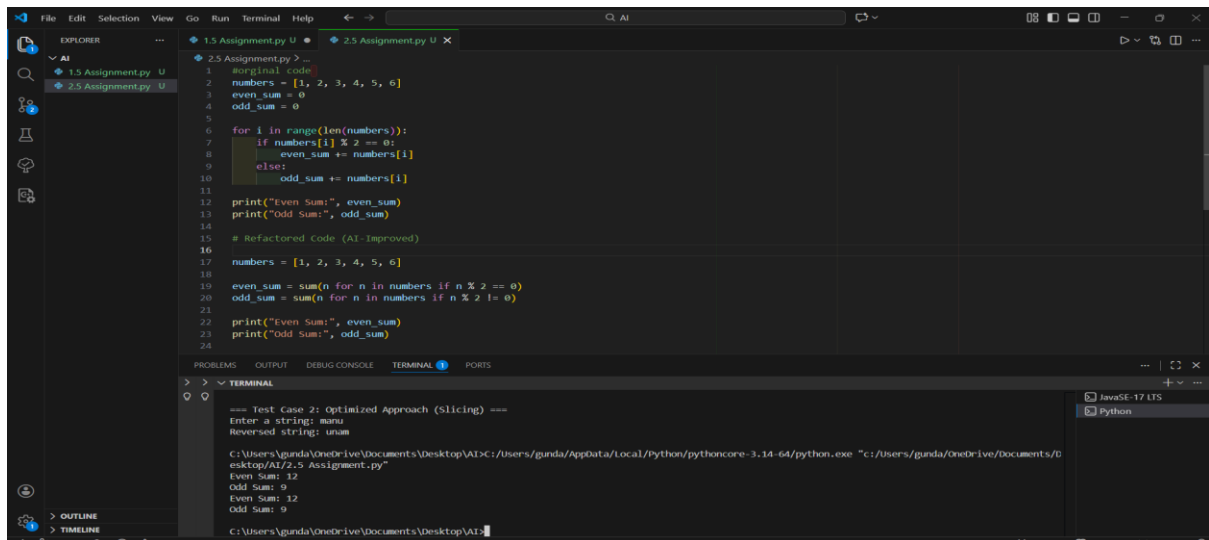
NAME: A.Abhilash Goud

HALL-TICKET :2303A51359

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

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

Code and output :



```
File Edit Selection View Go Run Terminal Help
2.5 Assignment.py U 2.5 Assignment.py U X
EXPLORER
AI
1.5 Assignment.py U
2.5 Assignment.py U
1
#original code
2
numbers = [1, 2, 3, 4, 5, 6]
3
even_sum = 0
4
odd_sum = 0
5
6
for i in range(len(numbers)):
7
    if numbers[i] % 2 == 0:
8
        even_sum += numbers[i]
9
    else:
10
        odd_sum += numbers[i]
11
12
print("Even Sum:", even_sum)
13
print("Odd Sum:", odd_sum)
14
15
# Refactored Code (AI-Improved)
16
numbers = [1, 2, 3, 4, 5, 6]
17
18
even_sum = sum(n for n in numbers if n % 2 == 0)
19
odd_sum = sum(n for n in numbers if n % 2 != 0)
20
21
print("Even Sum:", even_sum)
22
print("Odd Sum:", odd_sum)
23
24
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
> >
> >
=== Test Case 2: Optimized Approach (Slicing) ===
Enter a string: manu
Reversed string: unam
C:\Users\gunda\OneDrive\Documents\Desktop\AI> c:/Users/gunda/Appdata/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Even Sum: 12
Odd Sum: 9
Even Sum: 12
Odd Sum: 9
C:\Users\gunda\OneDrive\Documents\Desktop\AI>
```

Explanation

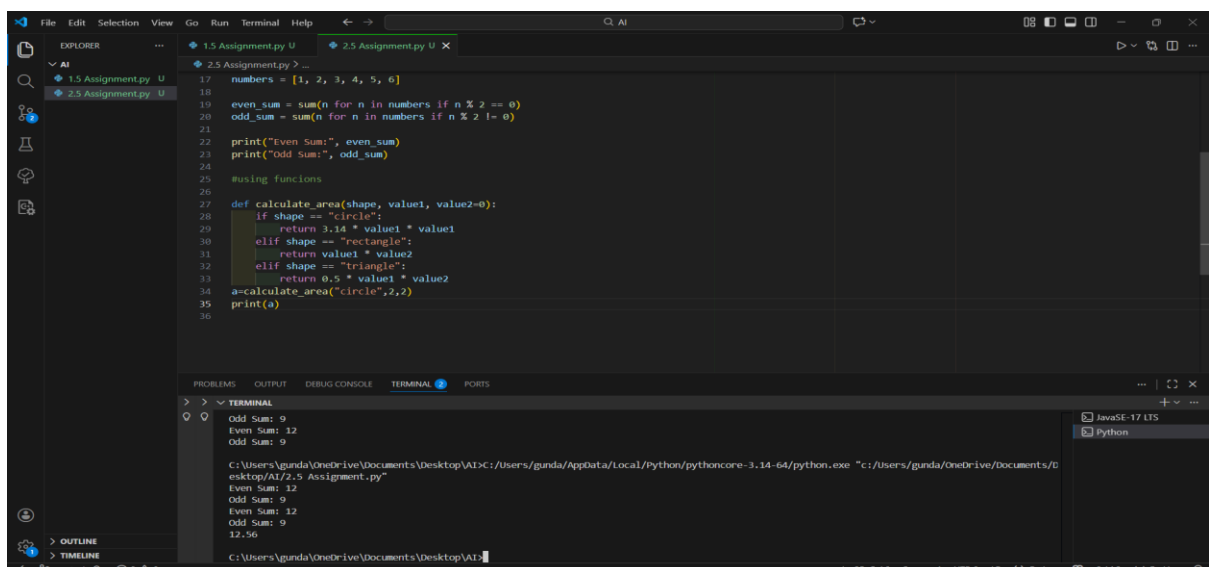
The refactored code is shorter, more readable, and efficient.

It removes manual loops and uses Python's built-in `sum()` with conditions, making the code easier to maintain.

Task 2: Area Calculation Explanation

Prompt : Explain a Python function that calculates the area of different shapes

Code and output :



```
File Edit Selection View Go Run Terminal Help
2.5 Assignment.py U 2.5 Assignment.py U X
EXPLORER
AI
1.5 Assignment.py U
2.5 Assignment.py U
17
numbers = [1, 2, 3, 4, 5, 6]
18
19
even_sum = sum(n for n in numbers if n % 2 == 0)
20
odd_sum = sum(n for n in numbers if n % 2 != 0)
21
22
print("Even Sum:", even_sum)
23
print("Odd Sum:", odd_sum)
24
25
#using functions
26
27
def calculate_area(shape, value1, value2=0):
28
    if shape == "circle":
29
        return 3.14 * value1 * value1
30
    elif shape == "rectangle":
31
        return value1 * value2
32
    elif shape == "triangle":
33
        return 0.5 * value1 * value2
34
a=calculate_area("circle",2,2)
35
print(a)
36
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
> >
> >
Odd Sum: 9
Even Sum: 12
Odd Sum: 9
C:\Users\gunda\OneDrive\Documents\Desktop\AI> c:/Users/gunda/Appdata/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Even Sum: 12
Odd Sum: 9
Even Sum: 12
Odd Sum: 9
12.56
C:\Users\gunda\OneDrive\Documents\Desktop\AI>
```

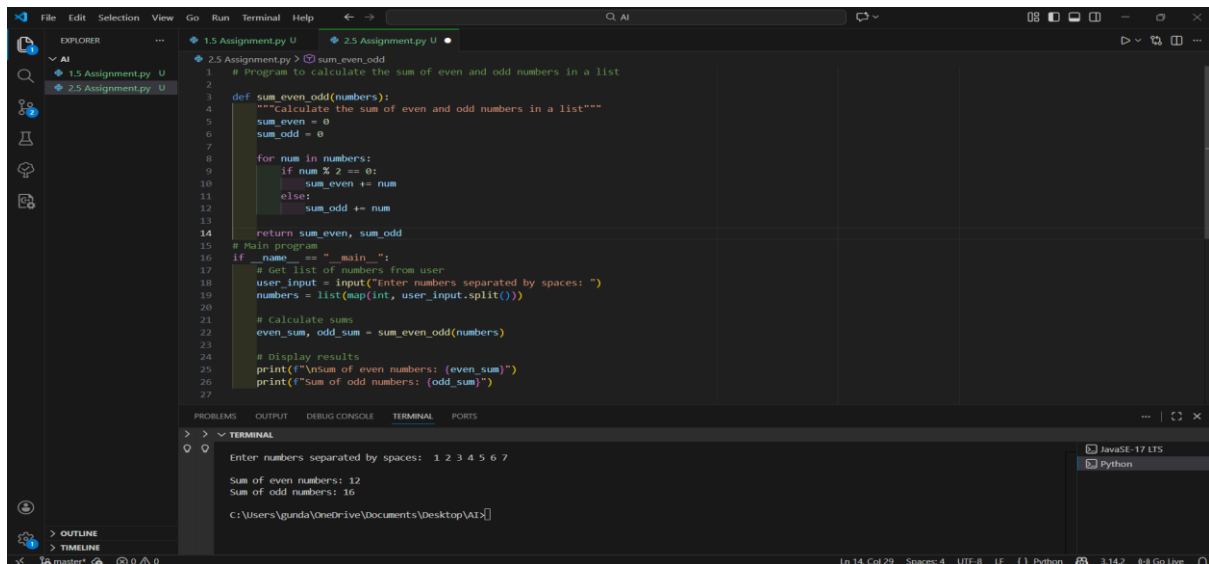
Explanation

Gemini clearly explains how the function works for different shapes.

It describes the parameters, logic flow, and formulas used, which helps beginners understand the code easily.

Task 3: Prompt Sensitivity Experiment

Prompt 1: Write a Python program to calculate the sum of even and odd numbers in a list



```
1 # Program to calculate the sum of even and odd numbers in a list
2
3 def sum_even_odd(numbers):
4     """Calculate the sum of even and odd numbers in a list"""
5     sum_even = 0
6     sum_odd = 0
7
8     for num in numbers:
9         if num % 2 == 0:
10             sum_even += num
11         else:
12             sum_odd += num
13
14     return sum_even, sum_odd
15
16 # Main program
17 if __name__ == "__main__":
18     # Get list of numbers from user
19     user_input = input("Enter numbers separated by spaces: ")
20     numbers = list(map(int, user_input.split()))
21
22     # Calculate sums
23     even_sum, odd_sum = sum_even_odd(numbers)
24
25     # Display results
26     print(f"\nSum of even numbers: {even_sum}")
27     print(f"Sum of odd numbers: {odd_sum}")
```

Enter numbers separated by spaces: 1 2 3 4 5 6 7

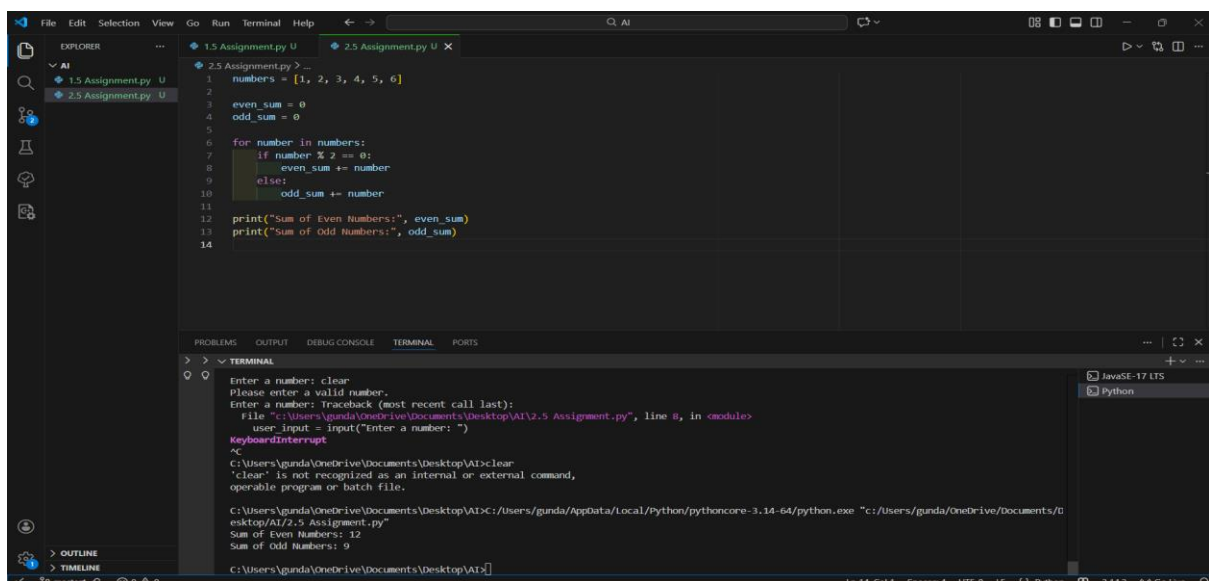
Sum of even numbers: 12
Sum of odd numbers: 16

Explanation:

For **Prompt 1 (Basic Prompt)**, Cursor AI generated a simple loop-based program using conditional statements. This version is easy to understand and suitable for beginners, but it uses more lines of code and manual variable updates.

Prompt 2: Write a clean and readable Python program to find the sum of even and odd numbers in a list suitable for beginners

Code and output:



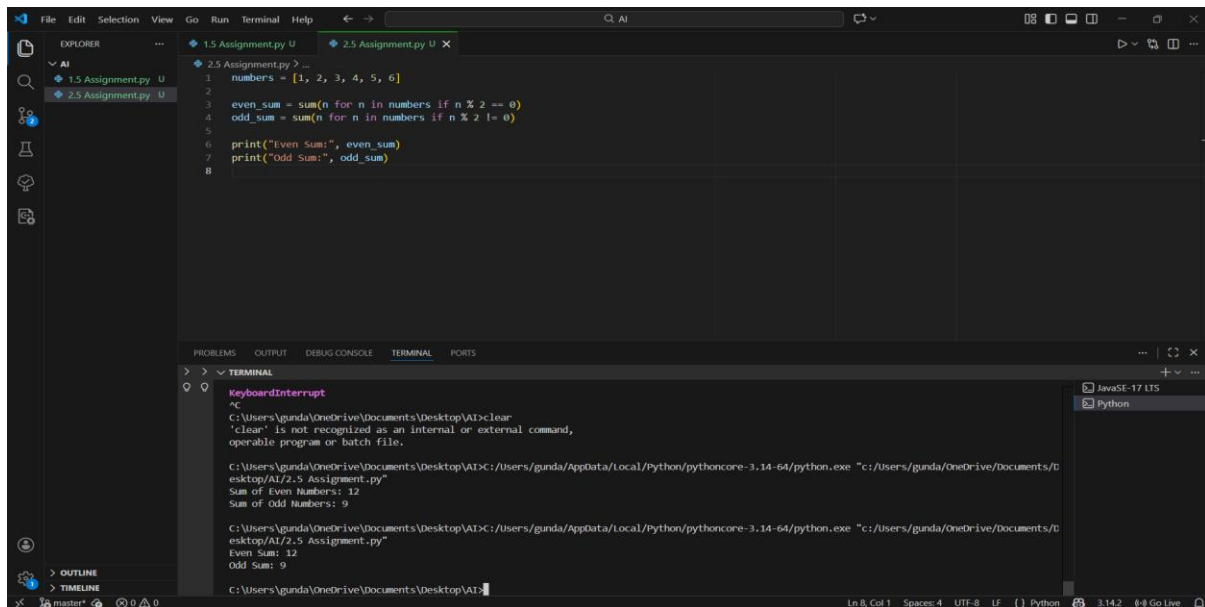
```
1 numbers = [1, 2, 3, 4, 5, 6]
2
3 even_sum = 0
4 odd_sum = 0
5
6 for number in numbers:
7     if number % 2 == 0:
8         even_sum += number
9     else:
10        odd_sum += number
11
12 print("Sum of Even Numbers:", even_sum)
13 print("Sum of Odd Numbers:", odd_sum)
```

Enter a number: clear
Please enter a valid number.
Enter a number: Traceback (most recent call last):
File "c:/Users/gunda/OneDrive/Documents/Desktop/AI/2.5 Assignment.py", line 8, in <module>
user_input = input("Enter a number: ")
KeyboardInterrupt
<C
C:/Users/gunda/OneDrive/Documents/Desktop/AI>clear
'clear' is not recognized as an internal or external command,
operable program or batch file.
C:/Users/gunda/OneDrive/Documents/Desktop/AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Sum of Even Numbers: 12
Sum of Odd Numbers: 9

Explanation : For **Prompt 2 (Readability-Focused Prompt)**, the AI produced code with clearer variable names and better formatting. Although the logic is similar to the basic version, readability and clarity were improved, making the code easier to review and maintain.

Prompt 3: Write an optimized Python program to calculate the sum of even and odd numbers in a list using built-in functions

Code and output:



```
1 numbers = [1, 2, 3, 4, 5, 6]
2
3 even_sum = sum(n for n in numbers if n % 2 == 0)
4 odd_sum = sum(n for n in numbers if n % 2 != 0)
5
6 print("Even Sum:", even_sum)
7 print("Odd Sum:", odd_sum)
8
```

Terminal Output:

```
KeyboardInterrupt
^C
C:\Users\gunda\OneDrive\Documents\Desktop\AI>clear
'clear' is not recognized as an internal or external command,
operable program or batch file.

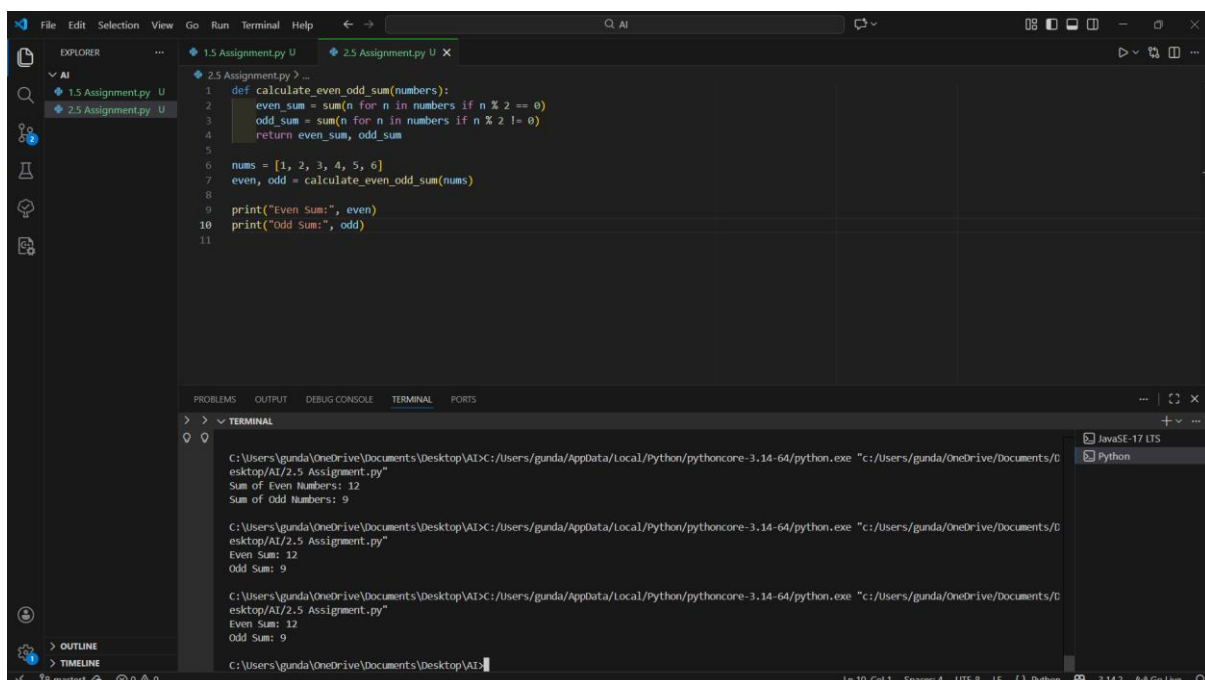
C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Sum of Even Numbers: 12
Even Sum: 12
Sum of Odd Numbers: 9
Odd Sum: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Even Sum: 12
Odd Sum: 9
```

Explanation: For **Prompt 3 (Optimized Prompt)**, Cursor AI generated a more efficient solution using Python's built-in sum() function along with conditions. This version reduced the number of lines and improved code efficiency while maintaining correctness.

Prompt 4 : Write a Python program to calculate the sum of even and odd numbers in a list using functions

Code and output :



```
1 def calculate_even_odd_sum(numbers):
2     even_sum = sum(n for n in numbers if n % 2 == 0)
3     odd_sum = sum(n for n in numbers if n % 2 != 0)
4     return even_sum, odd_sum
5
6 nums = [1, 2, 3, 4, 5, 6]
7 even, odd = calculate_even_odd_sum(nums)
8
9 print("Even Sum:", even)
10 print("Odd Sum:", odd)
11
```

Terminal Output:

```
C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Sum of Even Numbers: 12
Even Sum: 12
Sum of Odd Numbers: 9
Odd Sum: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Even Sum: 12
Odd Sum: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>C:/Users/gunda/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/gunda/OneDrive/Documents/D
esktop/AI/2.5 Assignment.py"
Even Sum: 12
Odd Sum: 9
```

Explanation:

For **Prompt 4 (Function-Based Prompt)**, the AI created a modular solution using a user-defined function. This approach improves reusability, debugging ease, and maintainability, making it suitable for larger applications.

Task 4: Tool Comparison Reflection

Reflection

Based on the experiments performed in this lab, Google Gemini, GitHub Copilot, and Cursor AI each have different strengths.

Google Gemini is very useful for understanding code, as it provides clear explanations and works well in Google Colab, especially for beginners.

GitHub Copilot offers real-time code suggestions inside VS Code and is best suited for daily development and writing production-ready code.

Cursor AI is effective for experimenting with different prompts, refactoring code, and analyzing multiple coding approaches.