

## Lab assignment- 2.5

NAME: T.SRIHARSHITHA

HALL-TICKET :2303A51261

BATCH : 19

## 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 :

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows two files: "1.5 Assignment.py" and "2.5 Assignment.py".
- Code Editor:** Displays the content of "2.5 Assignment.py". The code uses list comprehension and slicing to calculate even and odd sums from a list of numbers.
- Bottom Bar:** Shows tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active.
- Terminal Output:** Shows the results of running the code in the terminal. It includes:
  - Test Case 2: Optimized Approach (Slicing)
  - Enter a string: manu
  - Reversed string: unam
  - Even Sum: 12
  - Odd Sum: 9
  - Even Sum: 12
  - Odd Sum: 9
- Status Bar:** Shows the current file path as "C:\Users\gunda\OneDrive\Documents\Desktop\AI\1.5 Assignment.py", the line number as "Line 16 Col 1", and the word count as "3142".

## 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 :

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows two files: "1.5 Assignment.py" and "2.5 Assignment.py".
- Code Editor:** Displays the content of "2.5 Assignment.py". The code uses list comprehension to calculate the sum of even and odd numbers from a list. It then defines a function to calculate the area of shapes based on their type and dimensions.
- Terminal:** Shows the output of running the script, which includes the calculation of even and odd sums for the first 6 numbers and the area of a circle with radius 2.
- Status Bar:** Shows the current file is "2.5 Assignment.py", the line number is 25, the column number is 9, and the status "Python 3.14.2 Go Live".

**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

The screenshot shows a Python script named `2.5 Assignment.py` open in the VS Code editor. The code defines a function `sum_even_odd` to calculate the sum of even and odd numbers from a list. It then prompts the user for input and prints the results. The terminal below shows the execution and output of the script.

```
1.5 Assignment.py U 2.5 Assignment.py U
2.5 Assignment.py > sum_even_odd
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 sum
23     even_sum, odd_sum = sum_even_odd(numbers)
24
25     # Display results
26     print(f"Sum of even numbers: {even_sum}")
27     print(f"Sum of odd numbers: {odd_sum}")
28
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

> > > TERMINAL

JavaSE-17 LTS  
Python

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

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

C:\Users\gunda\OneDrive\Documents\Desktop\AI\
```

Line 14, Col 29 Spaces: 4 UTF-8 LF {} Python 3.14.2 0-0 Go Live

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:

```

1.5 Assignment.py
2.5 Assignment.py >_ 
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)
14

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
> > > TERMINAL
0 q Enter a number: clear
Please enter a valid number.
Enter a number: 12
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/Desktop/AI/2.5 Assignment.py"
Sum of Even Numbers: 12
Sum of Odd Numbers: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>

```

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:

```

1.5 Assignment.py
2.5 Assignment.py >_ 
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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
> > > TERMINAL
0 q 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/Desktop/AI/2.5 Assignment.py"
Sum of Even Numbers: 12
Sum of Odd Numbers: 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/Desktop/AI/2.5 Assignment.py"
Even Sum: 12
Odd Sum: 9

C:\Users\gunda\OneDrive\Documents\Desktop\AI>

```

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 :

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows two files: "1.5 Assignment.py" and "2.5 Assignment.py".
- Code Editor:** Displays the content of "2.5 Assignment.py".

```
def calculate_even_odd_sum(numbers):
    even_sum = sum(n for n in numbers if n % 2 == 0)
    odd_sum = sum(n for n in numbers if n % 2 != 0)
    return even_sum, odd_sum

nums = [1, 2, 3, 4, 5, 6]
even, odd = calculate_even_odd_sum(nums)
print("Even Sum:", even)
print("Odd Sum:", odd)
```
- Terminal:** Shows three command-line executions of the script, each printing the even sum (12), odd sum (9), and the total sum (21).
- Status Bar:** Shows the file path "C:\Users\gunda\OneDrive\Documents\Desktop\AI", the line number "Ln 10, Col 1", and the Python version "3.14.2".

### 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.