

Lab assignment- 2.5

HT :2303A51589

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 <- > AI
EXPLORER ... 1.5 Assignment.py U 2.5 Assignment.py X
2.5 Assignment.py >-
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 (At-Improved)
16
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
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1 PORTS
> < V TERMINAL
JavaSE-17 LTS Python
--- 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/Desktop/AI>2.5 Assignment.py"
Even Sum: 12
Odd Sum: 9
Even Sum: 12
Odd Sum: 9
C:\Users\gunda\OneDrive\Documents\Desktop\AI>
In 16, Col 1  Spaces: 4  UFT-8  LF  { } Python 3.14.2  6.0 Go Live
```

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 <- > AI
EXPLORER ... 1.5 Assignment.py U 2.5 Assignment.py X
2.5 Assignment.py >-
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
26 using functions
27
28 def calculate_area(shape, value1, value2=0):
29     if shape == "circle":
30         return 3.14 * value1 * value1
31     elif shape == "rectangle":
32         return value1 * value2
33     elif shape == "triangle":
34         return 0.5 * value1 * value2
35     print(a)
36
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1 PORTS
> < V TERMINAL
JavaSE-17 LTS Python
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/Desktop/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>
In 35, Col 9  Spaces: 4  UFT-8  LF  { } Python 3.14.2  6.0 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 the VS Code interface with two files open: 1.5 Assignment.py and 2.5 Assignment.py. The 2.5 Assignment.py file contains the following code:

```
# Program to calculate the sum of even and odd numbers in a list
def sum_even_odd(numbers):
    """Calculate the sum of even and odd numbers in a list"""
    sum_even = 0
    sum_odd = 0

    for num in numbers:
        if num % 2 == 0:
            sum_even += num
        else:
            sum_odd += num

    return sum_even, sum_odd

# Main program
if __name__ == "__main__":
    # Get list of numbers from user
    user_input = input("Enter numbers separated by spaces: ")
    numbers = list(map(int, user_input.split()))

    # Calculate sums
    even_sum, odd_sum = sum_even_odd(numbers)

    # Display results
    print(f"\nsum of even numbers: {even_sum}")
    print(f"sum of odd numbers: {odd_sum}")

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

The terminal window shows the output of running the program with input "1 2 3 4 5 6 7".

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:

The screenshot shows the VS Code interface with two files open: 1.5 Assignment.py and 2.5 Assignment.py. The 2.5 Assignment.py file contains the following code:

```
numbers = [1, 2, 3, 4, 5, 6]
even_sum = 0
odd_sum = 0

for number in numbers:
    if number % 2 == 0:
        even_sum += number
    else:
        odd_sum += number

print("Sum of Even Numbers:", even_sum)
print("Sum of Odd Numbers:", odd_sum)

C:\Users\gunda\OneDrive\Documents\Desktop\AI> Enter a number: clear
Please enter a valid number.
Enter a number: Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
    user_input = input("Enter a number: ")
KeyboardInterrupt
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>python 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:

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

- File Explorer:** Shows files 1.5 Assignment.py and 2.5 Assignment.py.
- Code Editor:** Displays the content of 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
```
- Terminal:** Shows the output of running the script:

```
> > > TERMINAL
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>
```
- Status Bar:** Shows the current file is master, with 0 changes, and the status bar indicates Line 8, Column 1, Spaces: 4, UTF-8, LF, Python 3.14.2, and Go Live.

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 a Visual Studio Code (VS Code) interface. The left sidebar has icons for Explorer, Search, and Issues. The main area displays two Python files: '1.5 Assignment.py' and '2.5 Assignment.py'. The '2.5 Assignment.py' file is open and contains the following code:

```
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
```

The 'TERMINAL' tab at the bottom shows the output of running the script:

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

Explanantion:

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.