ASSIGNMENT -2.5

NAME : SYED SUFIYAN

2303A51980

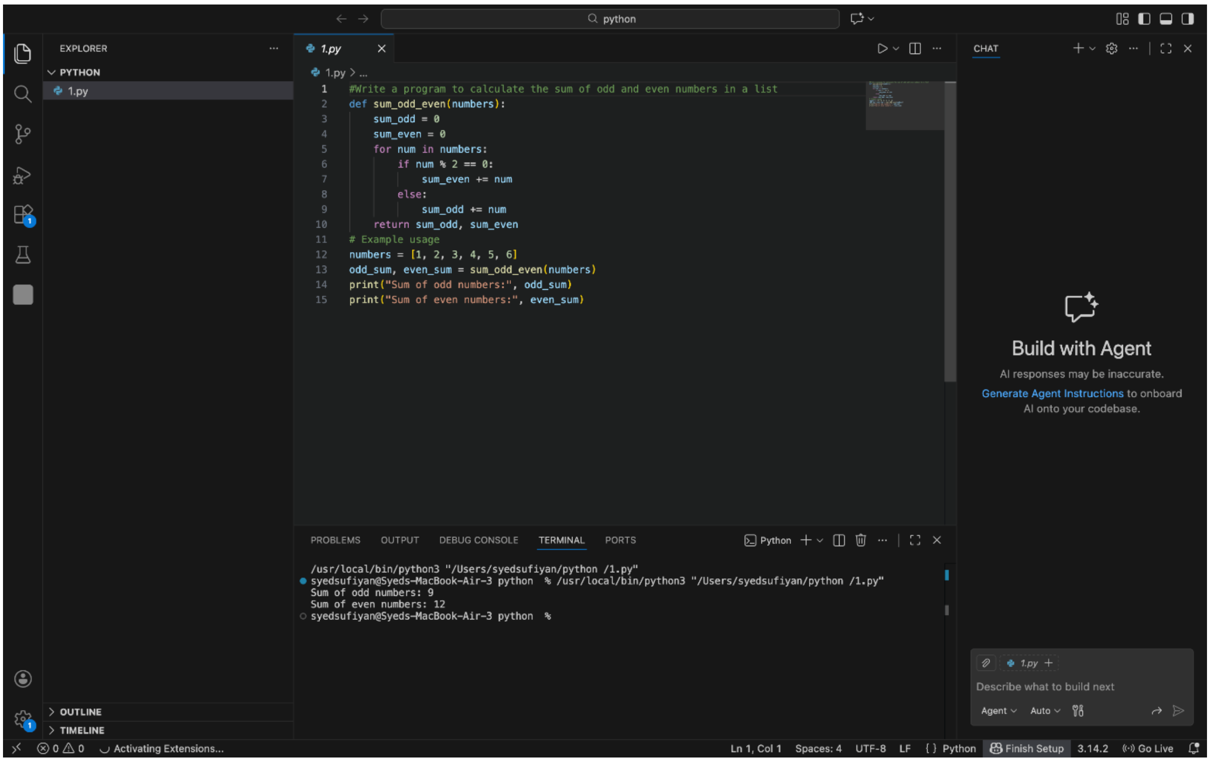
BATCH:30

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)
```

```
/usr/local/bin/python3 "/Users/syedsufiyan/python /1.py"
syedsufiyan@Syeds-MacBook-Air-3 python  % /usr/local/bin/python3 "/Users/syedsufiyan/python /1.py"
Sum of odd numbers: 9
Sum of even numbers: 12
syedsufiyan@Syeds-MacBook-Air-3 python  %
```

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

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

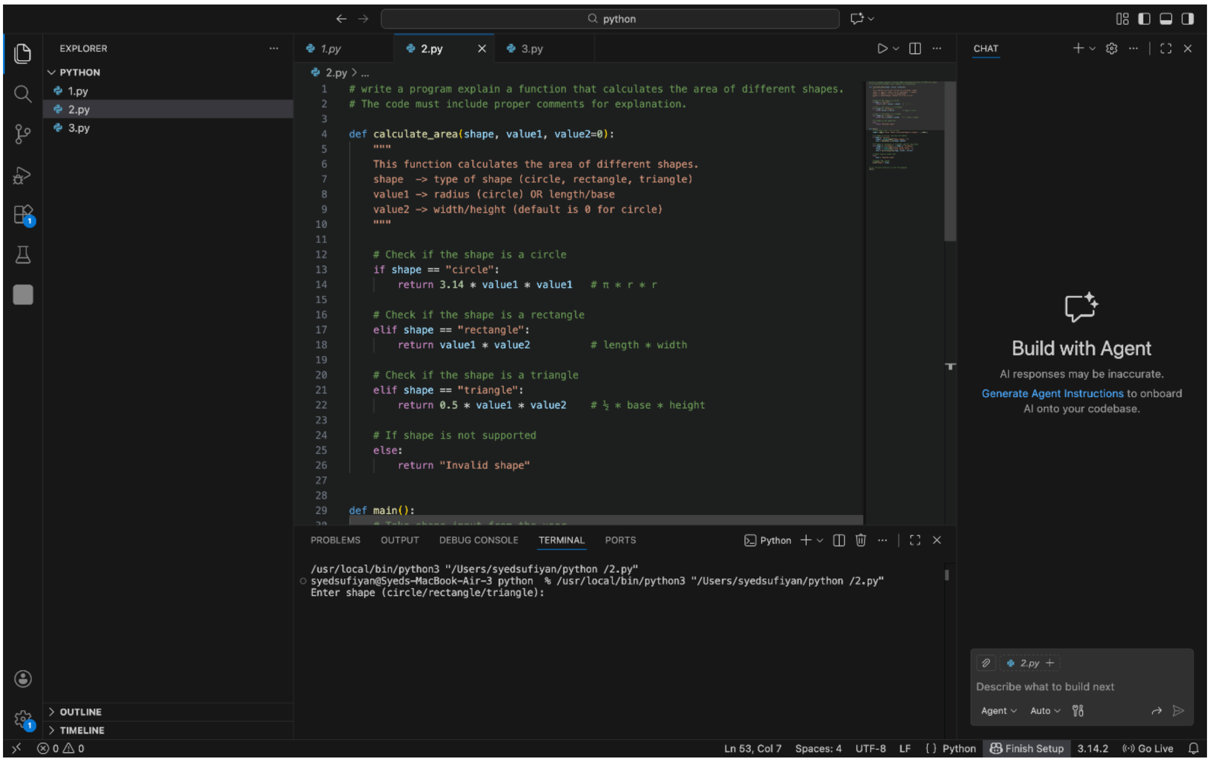
Using a function allows the same logic to be reused with different lists without rewriting code.

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 shapes.
# 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

    # Check if the shape is a rectangle
    elif shape == "rectangle":
        return value1 * value2          # length * width

    # Check if the shape is a triangle
    elif shape == "triangle":
        return 0.5 * value1 * value2    # ½ * base * height

    # If shape is not supported
    else:
        return "Invalid shape"


def main():
```

```
/usr/local/bin/python3 "/Users/syedsufiyan/python /2.py"
syedsufiyan@Syeds-MacBook-Air-3 python  % /usr/local/bin/python3 "/Users/syedsufiyan/python /2.py"
Enter shape (circle/rectangle/triangle):
```

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.

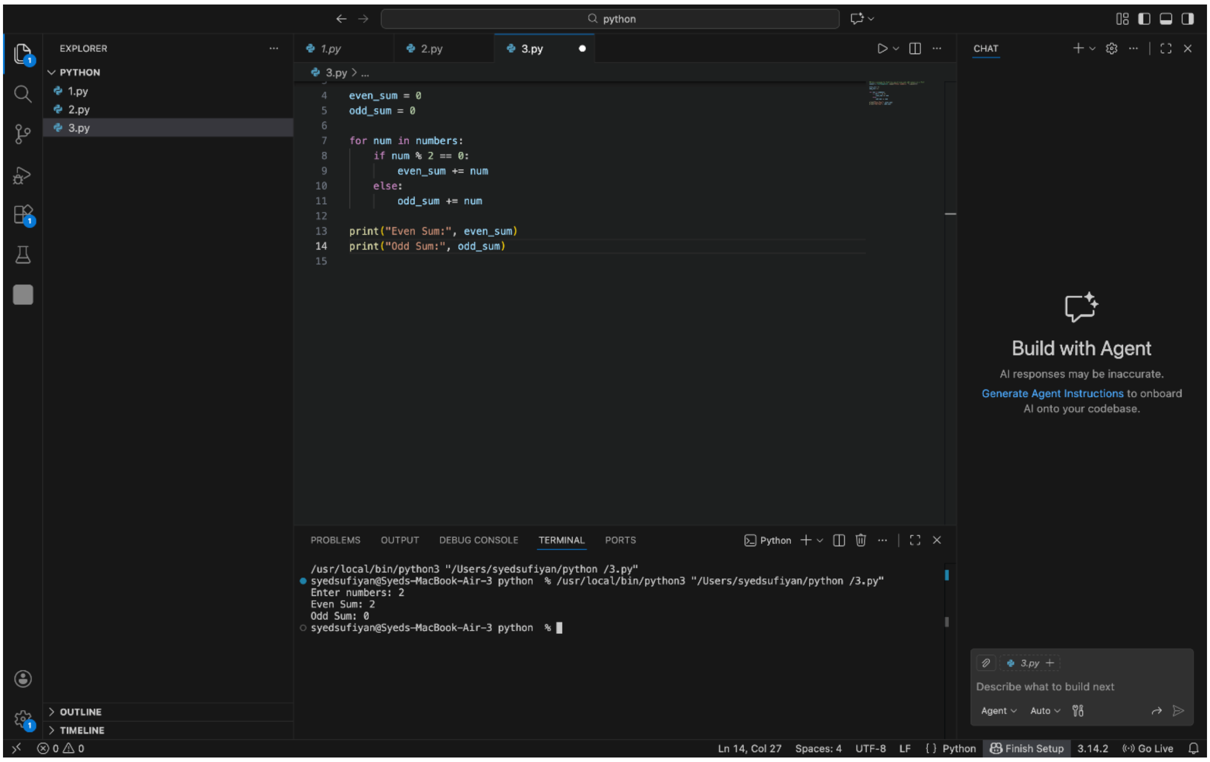
TASK:3

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
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)
```

```
/usr/local/bin/python3 "/Users/syedsufiyan/python /3.py"
syedsufiyan@Syeds-MacBook-Air-3 python  % /usr/local/bin/python3 "/Users/syedsufiyan/python /3.py"
Enter numbers: 2
Even Sum: 2
Odd Sum: 0
syedsufiyan@Syeds-MacBook-Air-3 python  %
```

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.

GitHubT Copilo 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