

# AI-Assisted Coding

## Assignment 2.5

HTNo:2303A52299

Name:A.Sai Shrehan

Batch:45

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

❖ Scenario:

You are improving legacy code.

❖ Task:

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

then refactor it using AI.

❖ Expected Output:

❖ improved code

Output:

The screenshot shows a code editor interface with two main panes. The left pane displays the Python script `assignment_2.5.py`. The right pane shows the terminal output and a chat window.

**Code Content (`assignment_2.5.py`):**

```
File Edit Selection View Go Run Terminal Help < > Q AI assisted coding
assignment_2.5.py U x
assignment_2.5.py > ...
23 def main():
    # Convert input to list of integers
    user_input = input("Enter numbers separated by spaces: ")
    numbers = [int(x) for x in user_input.split()]
    # Calculate sums
    sum_odd, sum_even = calculate_odd_even_sums(numbers)
    # Display results
    print("\n" + "=" * 50)
    print(f"Numbers entered: {numbers}")
    print("=" * 50)
    print(f"Sum of odd numbers: {sum_odd}")
    print(f"Sum of even numbers: {sum_even}")
    print(f"Total sum: {sum_odd + sum_even}")
    print("=" * 50)

except ValueError:
    print("Error: Please enter valid integers only.")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
```

**Terminal Output:**

```
PS D:\AI assisted coding & C:/Users/vaish/AppData/Local/Programs/Python/Python314/python.exe "d:/AI assisted coding/assignment_2.5.py"
=====
Sum of odd and Even Numbers Calculator
=====

Enter numbers separated by spaces: 1 2 3 4 5 6 7 8
=====
Numbers entered: [1, 2, 3, 4, 5, 6, 7, 8]
Sum of odd numbers: 16
Sum of even numbers: 20
Total sum: 36
=====
```

**Chat Window:**

- cd "d:/AI assisted coding"; python assignment\_2.5.py <<< "1 2 3 4 5 6 7 8 9 10"
- Used echo to pipe data instead of here-string syntax
- cd "d:/AI assisted coding"; echo "1 2 3 4 5 6 7 8 9 10" | python assignment\_2.5.py
- Confirmed program functionality and tested results

Perfect! I've created a complete program in `assignment_2.5.py` that calculates the sum of odd and even numbers in a list.

**Key features:**

- `calculate_odd_even_sums()` function: Separates and sums odd and even numbers
- Input validation: Handles invalid input gracefully
- Clear output: Displays the numbers, odd sum, even sum, and total
- Well-documented: Includes docstrings and comments

**Test result:** With input 1 2 3 4 5 6 7 8 9 10:

- Odd sum: 25 (1+3+5+7+9)
- Even sum: 30 (2+4+6+8+10)
- Total: 55 ✓

The program is ready to use!

## Task 2: Area Calculation Explanation

### ❖ Scenario:

You are onboarding a junior developer.

### ❖ Task:

Ask Gemini to explain a function that calculates the area of different shapes.

### ❖ Expected Output:

### ➤ Code

### Output:

The screenshot shows a code editor interface with two tabs: 'assignment\_2.5.py' and 'area\_calculator.py'. The 'area\_calculator.py' tab is active and displays the following Python code:

```
def calculate_area(shape: str, *args) -> float:
    """
    Calculates the area of a given shape.

    Supported shapes and required arguments:
    - 'circle': radius
    - 'rectangle': length, width
    - 'triangle': base, height

    Args:
        shape (str): The name of the shape (case-insensitive).
        *args: Variable length argument list for dimensions.

    Returns:
        float: The calculated area.

    Raises:
        ValueError: If the shape is unsupported or dimensions are invalid.
    """
    shape = shape.lower().strip()

    if shape == 'circle':
        if len(args) != 1:
            raise ValueError("Circle requires exactly 1 argument: radius.")
        radius = args[0]
        if radius < 0:
```

The code editor has a sidebar with AI-assisted documentation for the function. It includes sections for Type Hinting, Variable Arguments (\*args), and Docstrings. The sidebar also provides prompts for refactoring and adding unit tests. At the bottom, there is a terminal window showing the execution of the script and its output for three different shapes: Circle, Rectangle, and Triangle.

## Task 3: Prompt Sensitivity Experiment

### ❖ Scenario:

You are testing how AI responds to different prompts.

### ❖ Task:

Use Cursor AI with different prompts for the same problem and observe code changes.

### ❖ Expected Output:

### ➤ Prompt list

## ➤ Code variations

Prompt:-

=>Calculate the factorial of a number

The screenshot shows the AI assisted coding interface. On the left, there's a file explorer with various files listed under 'AI ASSISTED CODING'. In the center, a code editor window titled 'Factorial calculation' displays Python code for calculating factorial using a loop:

```
def factorial(n):
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result
```

Below this, another section shows a recursive version:

```
def factorial_recursive(n):
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    if n == 0 or n == 1:
        return 1
    return n * factorial_recursive(n - 1)
```

On the right, an 'Agents' panel shows a single agent named 'Factorial calculation' with the status 'Now'. A message box says 'Factorial of a number in ...'.

=> write factorial function in python using recursion

The screenshot shows the AI assisted coding interface. The file explorer on the left is identical to the previous one. The central code editor window is titled 'Factorial calculation' and contains Python code for a recursive factorial function:

```
def factorial(n):
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    if n == 0 or n == 1:
        return 1
    return n * factorial(n - 1)
```

Below the code, a message box says 'Thought 2s'. On the right, the 'Agents' panel shows the same agent with the status 'Now'. A message box says 'Recursive factorial funct...'. A tooltip at the bottom left suggests adding 'cursor' or 'code' commands.