**School of Computer Science and Artificial Intelligence**

---

**Lab Assignment # 2.5**

---

**Program** : B. Tech (CSE)

**Specialization** :AIML

**Course Title** : AI Assisted Coding

**Course Code** : 23CS002PC304

**Semester** : VI

**Academic Session** : 2025-2026

**Name of Student** : k.sathwika

**Enrollment No.** : 2303A52096

**Batch No.** : 33

**Date** :09/01/26

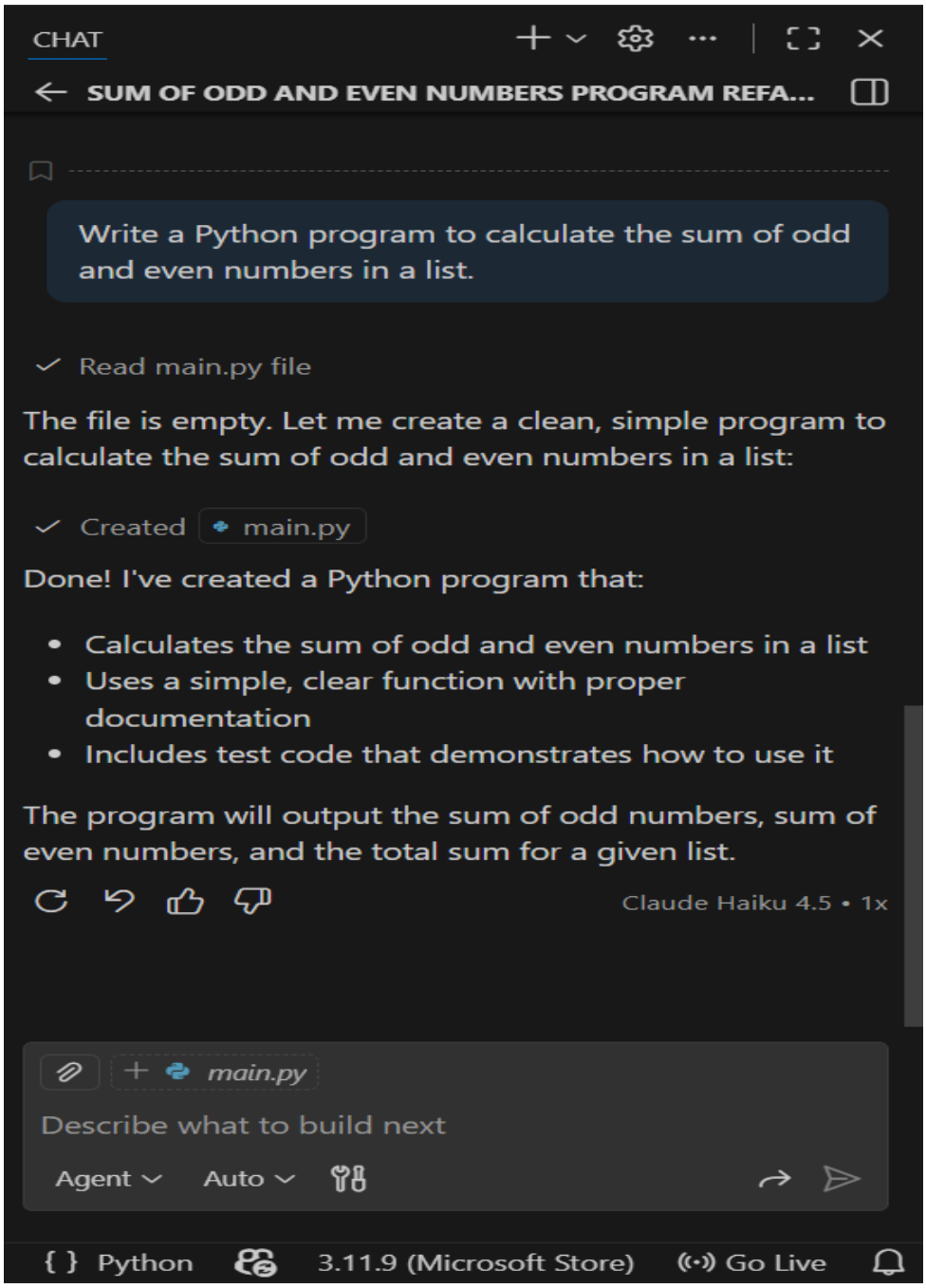
# **TASK_01:** Refactoring Odd/Even Logic (List Version)

**Scenario:**

You are improving legacy code.

**Task Description:**

Write a program to calculate the sum of odd and even numbers in a list, then refactor it using AI.



main.py

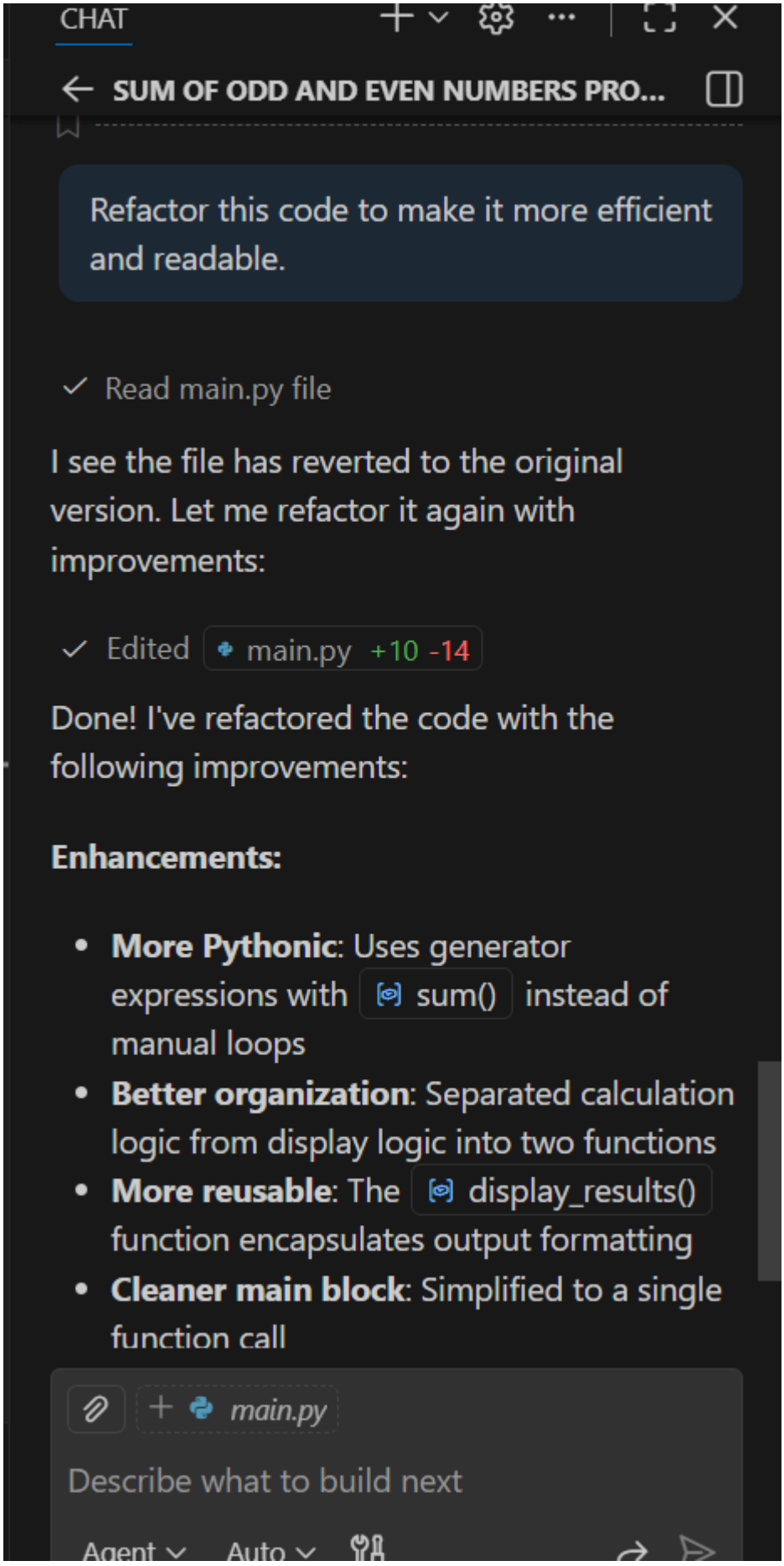
Assignment 1.2 > Assignemnt 2.5 > main.py > ...

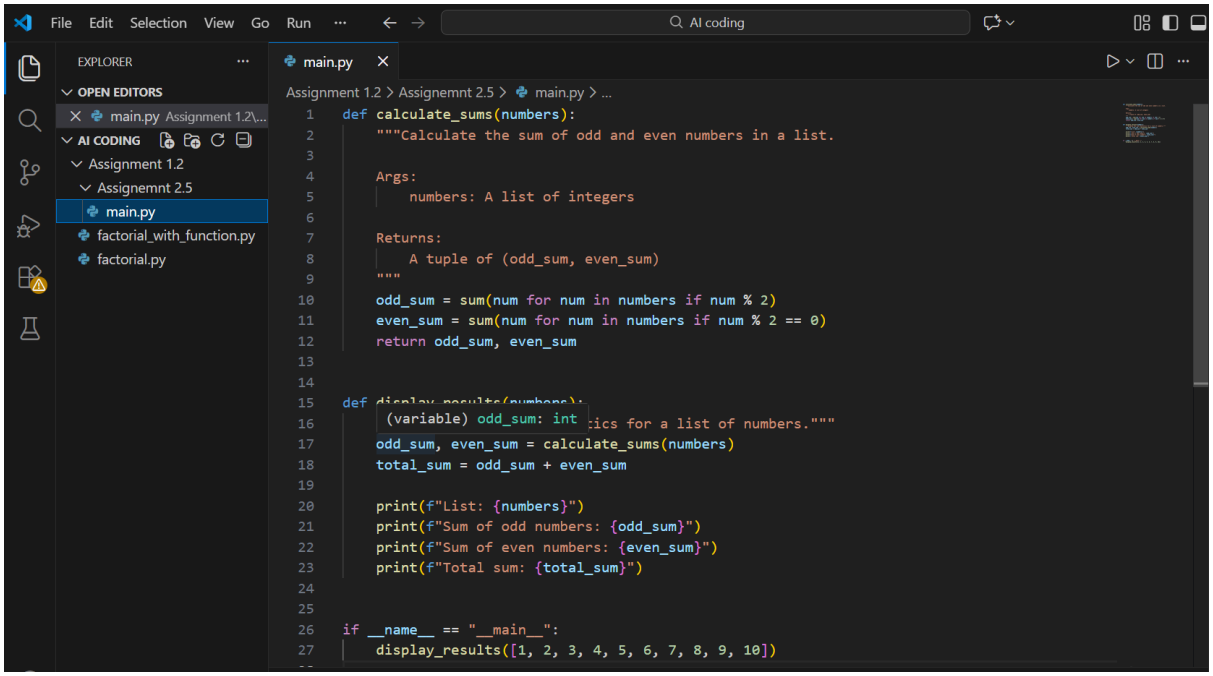
```python
def calculate_sums(numbers):
    """Calculate the sum of odd and even numbers in a list.

    Args:
        numbers: A list of integers

    Returns:
        A tuple of (odd_sum, even_sum)
    """
    odd_sum = 0
    even_sum = 0

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

    return odd_sum, even_sum


if __name__ == "__main__":
    # Test the function
    numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

    odd_sum, even_sum = calculate_sums(numbers)

    print(f"List: {numbers}")
    print(f"Sum of odd numbers: {odd_sum}")
    print(f"Sum of even numbers: {even_sum}")
    print(f"Total sum: {odd_sum + even_sum}")
```

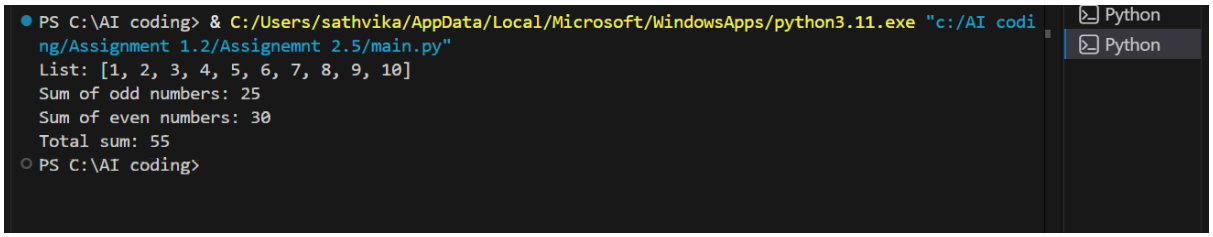
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

powersh...
Python
Python

```
PS C:\AI coding> & C:/Users/sathvika/AppData/Local/Microsoft/WindowsA
pps/python3.11.exe "c:/AI coding/Assignment 1.2/Assignemnt 2.5/main.p
y"
List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
y"
List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Sum of odd numbers: 25
Sum of odd numbers: 25
Sum of even numbers: 30
Sum of even numbers: 30
Total sum: 55
PS C:\AI coding>
```


CHAT
SUM OF ODD AND EVEN NUMBERS PRO...
Refactor this code to make it more efficient and readable.
Read main.py file
I see the file has reverted to the original version. Let me refactor it again with improvements:
Edited main.py +10 -14
Done! I've refactored the code with the following improvements:
Enhancements:
More Pythonic: Uses generator expressions with sum() instead of manual loops
Better organization: Separated calculation logic from display logic into two functions
More reusable: The display_results() function encapsulates output formatting
Cleaner main block: Simplified to a single function call
main.py
Describe what to build next
Agent
Auto




# EXPLANATION:

The refactored code uses Python's built-in sum() function with conditions.

This reduces unnecessary variables and improves readability.

The logic is clearer and more efficient than the original version.

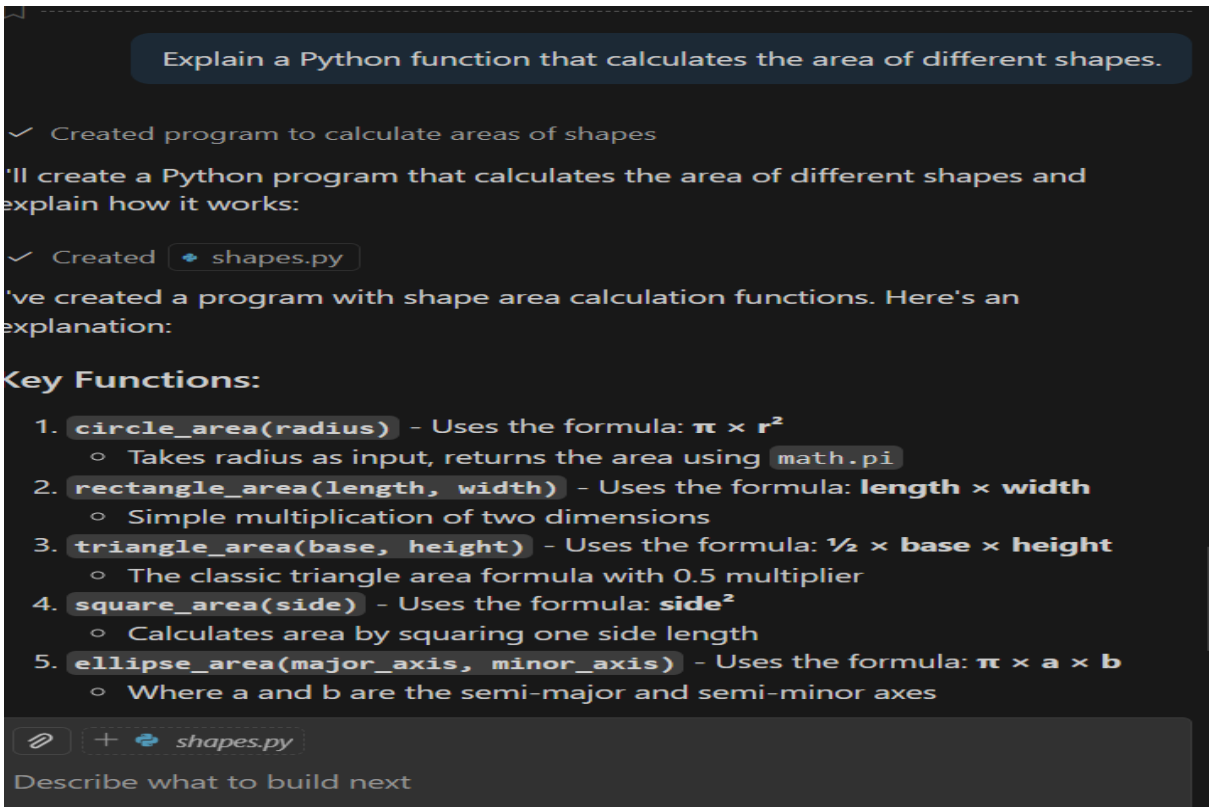
# TASK_02:

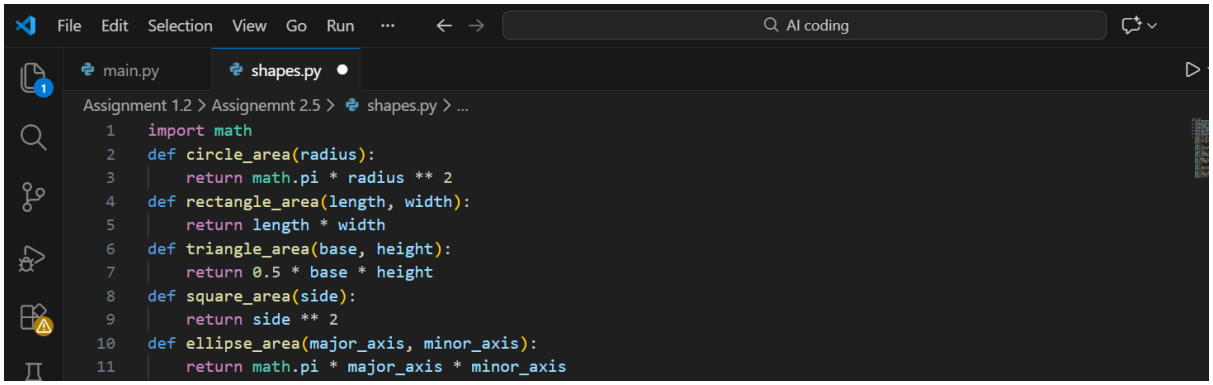
## Scenario

You are onboarding a junior developer.

## Task Description

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



```python
if __name__ == "__main__":
    print("=" * 50)
    print("AREA CALCULATOR FOR DIFFERENT SHAPES")
    print("=" * 50)
    # Circle
    radius = 5
    area = circle_area(radius)
    print(f"\nCircle with radius {radius}: Area = {area:.2f}")
    # Rectangle
    length, width = 10, 6
    area = rectangle_area(length, width)
    print(f"Rectangle ({length} x {width}): Area = {area:.2f}")
    # Triangle
    base, height = 8, 5
    area = triangle_area(base, height)
    print(f"Triangle (base={base}, height={height}): Area = {area:.2f}")
    # Square
    side = 7
    area = square_area(side)
    print(f"Square with side {side}: Area = {area:.2f}")
    # Ellipse
    major, minor = 10, 5
    area = ellipse_area(major, minor)
    print(f"Ellipse (major={major}, minor={minor}): Area = {area:.2f}")
    print("\n" + "=" * 50)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\AI coding> & C:/Users/sathvika/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/AI coding/Assignment 1.2/Assignemnt 2.5/shapes.py"
==================================================
AREA CALCULATOR FOR DIFFERENT SHAPES
==================================================

Circle with radius 5: Area = 78.54
Rectangle (10 x 6): Area = 60.00
Triangle (base=8, height=5): Area = 20.00
Square with side 7: Area = 49.00
Ellipse (major=10, minor=5): Area = 157.08

Ellipse (major=10, minor=5): Area = 157.08

==================================================
PS C:\AI coding>
```

powersh...
Python
Python

## AI Explanation

This function calculates the area of different shapes.

- For a circle, it uses the formula $\pi \times r^2$
- For a rectangle, it uses length × width
- For a triangle, it uses ½ × base × height…The function selects the correct formula using conditional statements

# TASK_03:

## Scenario

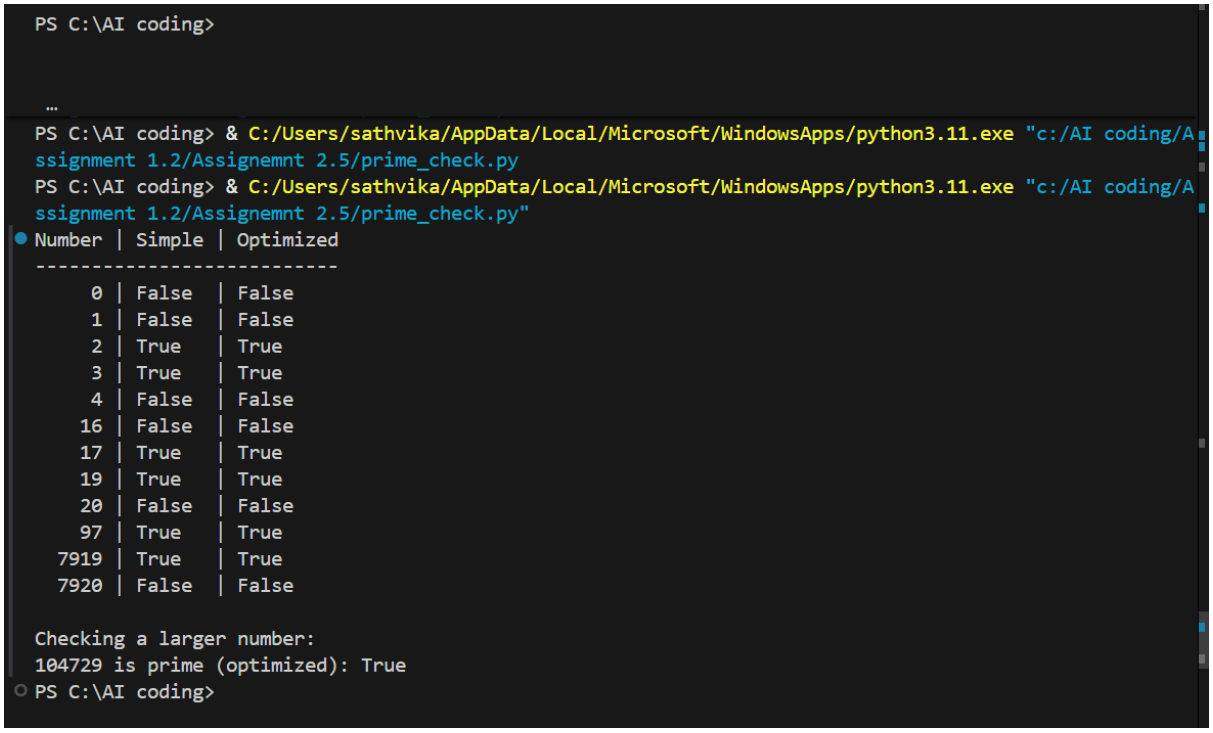
You are testing how AI responds to different prompts.

## Task Description

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

```python
import math
def is_prime_simple(n):
    if n <= 1:
        return False
    for i in range(2, n):
        if n % i == 0:
            return False
    return True
def is_prime_optimized(n):
   if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True
if __name__ == "__main__":
    test_numbers = [0, 1, 2, 3, 4, 16, 17, 19, 20, 97, 7919, 7920]

    print("Number | Simple | Optimized")
    print("--------------------------")
    for num in test_numbers:
        simple = is_prime_simple(num)
        opt = is_prime_optimized(num)
        print(f"{num:6} | {str(simple):6} | {str(opt):9}")
    large = 104729  # 10000th prime
    print("\nChecking a larger number:")
```

Ln 15, Col 21  Spaces: 4  UTF-8




**Observation**

Different prompts produce different styles of code:

- Simple
- Optimized
- Commented

Prompt wording strongly affects AI output quality.

**TASK_04:**

**Scenario**

You must recommend an AI coding tool**.**

**Task Description**

Compare Gemini, Copilot, and Cursor AI for usability and code quality.

GitHub Copilot, Google Gemini, and Cursor AI are all useful AI coding tools, but each has different strengths.

GitHub Copilot works directly inside VS Code and provides real-time code suggestions while typing. It is best for fast development and daily programming tasks.

Google Gemini in Google Colab is more useful for explanations and understanding concepts. It helps beginners learn coding logic clearly.

Cursor AI is good for testing different prompts, refactoring code, and improving code quality. It allows users to experiment with multiple coding styles.

conclusion:

- **Copilot** is best for speed
- **Gemini** is best for learning
- **Cursor AI** is best for experimentation