AI ASSISTANT CODING
LAB-02

Name : M.Harish

Roll No : 2303A5387

Batch No : 34

Lab 2: Exploring Additional AI Coding Tools beyond Copilot – Google Gemini (Colab) and Cursor AI

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

Scenario: You are improving legacy code.

The given legacy program calculates the sum of odd and even numbers in a list. The code needs to be improved for better readability and efficiency using AI tools.

Prompt Used :

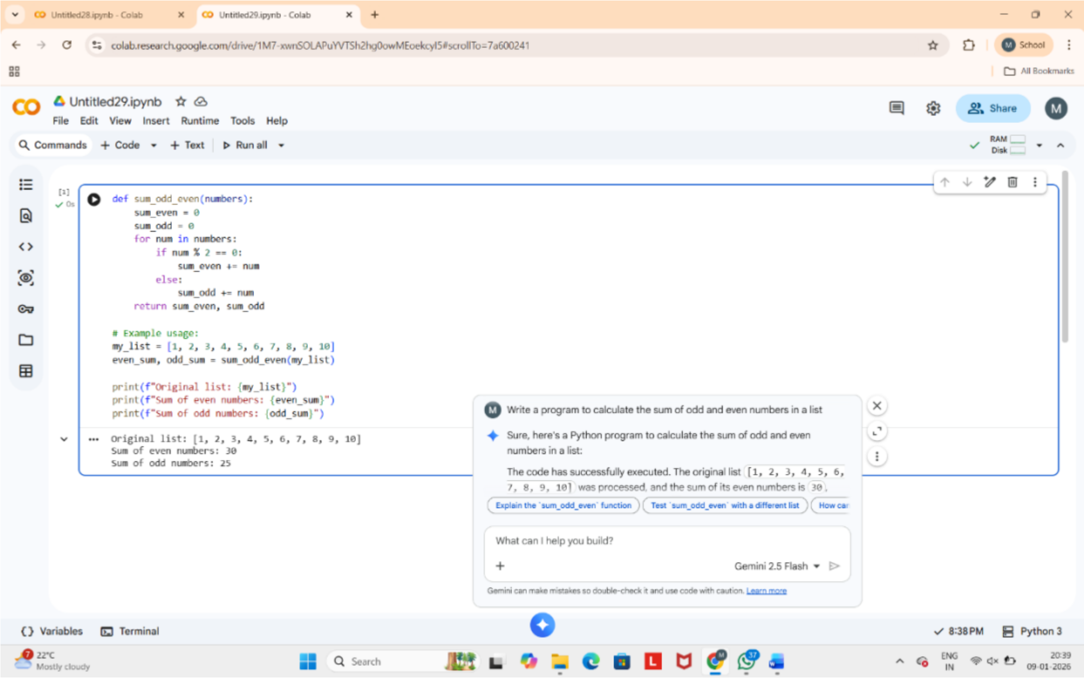
Write a program to calculate the sum of odd and even numbers in a list,
Refactor this Python code to improve readability and efficiency.

Original Code :

```
def sum_odd_even(numbers: list[int]) -> tuple[int, int]:
    even_sum = 0
    odd_sum = 0
    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num
    return even_sum, odd_sum
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum, odd_sum = sum_odd_even(my_list)
print(f"List: {my_list}")
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
```

AI-Refactored Code :

```
def sum_odd_even(numbers: list[int]) -> tuple[int, int]:
    """
    Calculates the sum of even and odd numbers in a list.

    Args:
        numbers: A list of integers.

    Returns:
        A tuple containing (even_sum, odd_sum).
    """
    even_sum = sum(num for num in numbers if num % 2 == 0)
    odd_sum = sum(num for num in numbers if num % 2 != 0)
    return even_sum, odd_sum

# Example usage:
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum, odd_sum = sum_odd_even(my_list)

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


Untitled29.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Explain the `sum_odd_even` function
Test `sum_odd_even` with a different list
What can I help you build?
Gemini 2.5 Flash
Variables
Terminal
Python 3

```
def sum_odd_even(numbers):
    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

# Example usage:
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum, odd_sum = sum_odd_even(my_list)

print(f"Original list: {my_list}")
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
```

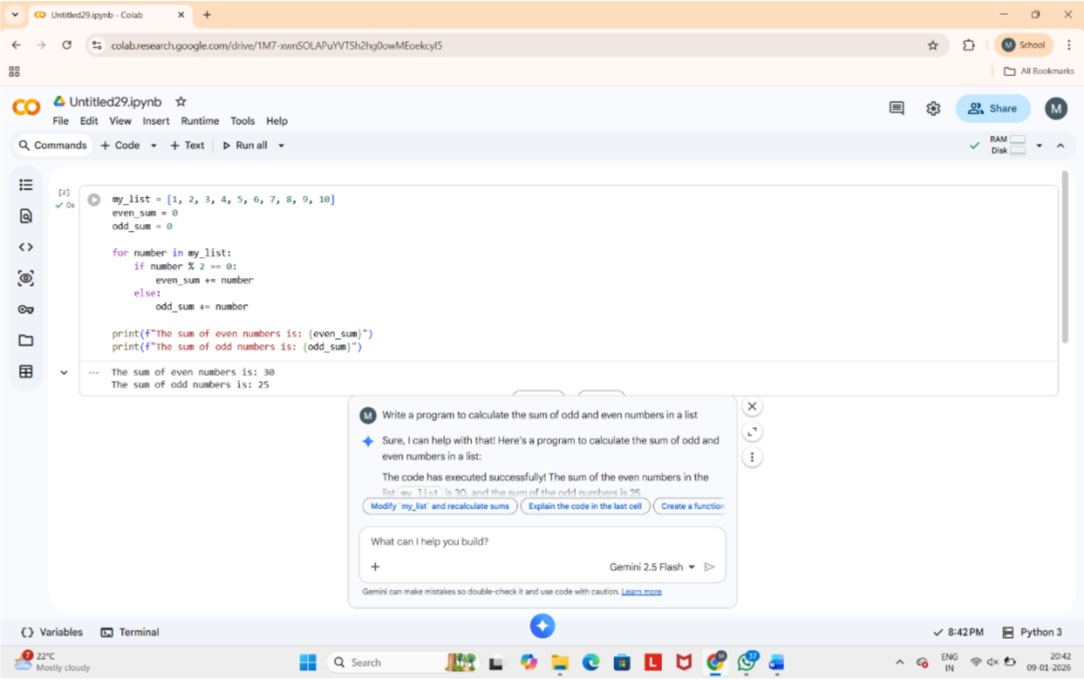
```
Original list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Sum of even numbers: 30
Sum of odd numbers: 25
```

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

Sure, here's a Python program to calculate the sum of odd and even numbers in a list:

The code has successfully executed. The original list `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]` was processed, and the sum of its even numbers is `30`.

Gemini can make mistakes so double-check it and use code with caution. Learn more


Untitled29.ipynb
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Modify `my_list` and recalculate sums
Explain the code in the last cell
Create a function
What can I help you build?
Gemini 2.5 Flash
Variables
Terminal
Python 3

```
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum = 0
odd_sum = 0

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

print(f"The sum of even numbers is: {even_sum}")
print(f"The sum of odd numbers is: {odd_sum}")
```

```
The sum of even numbers is: 30
The sum of odd numbers is: 25
```

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

Sure, I can help with that! Here's a program to calculate the sum of odd and even numbers in a list:

The code has executed successfully! The sum of the even numbers in the list `my_list` is 30, and the sum of the odd numbers is 25.

Gemini can make mistakes so double-check it and use code with caution. Learn more

Explanation :

The refactored code reduces the number of variables and uses Python’s built-in sum() function. It improves readability, reduces code length, and enhances maintainability. The logic is clearer and easier to modify.

Task 2: Area Calculation Explanation (Gemini)

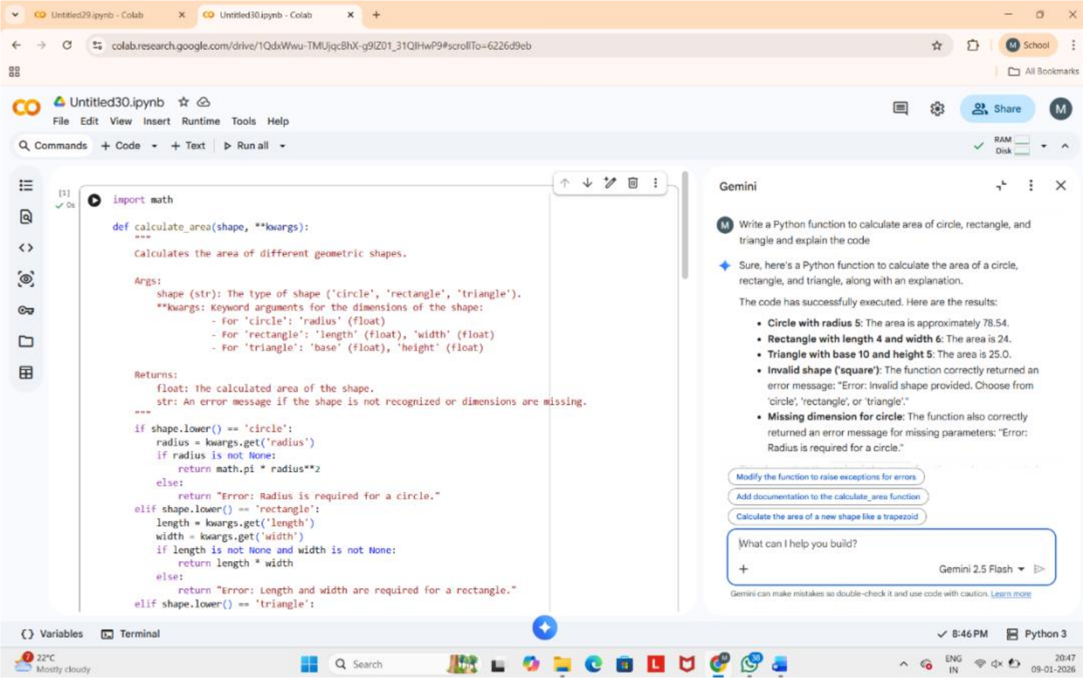
Scenario

This task helps onboard a junior developer by explaining a function that calculates areas of different shapes.

Prompt Used :

Write a Python function to calculate area of circle, rectangle, and triangle and explain the code

```
import math


def calculate_area(shape: str, **kwargs) -> float:
    """
    Calculates the area of different geometric shapes.


    Args:
        shape (str): The type of shape ('circle', 'rectangle', 'triangle').
        **kwargs: Keyword arguments for the dimensions of the shape:
            - For 'circle': 'radius' (float)
            - For 'rectangle': 'length' (float), 'width' (float)
            - For 'triangle': 'base' (float), 'height' (float)


    Returns:
        float: The calculated area of the shape.


    Raises:
        ValueError: If an unsupported shape is provided or required dimensions are missing.
    """
    if shape.lower() == 'circle':
        if 'radius' in kwargs:
            radius = kwargs['radius']
            return math.pi * radius**2
        else:
            raise ValueError("For 'circle', 'radius' is required.")
    elif shape.lower() == 'rectangle':
        if 'length' in kwargs and 'width' in kwargs:
            length = kwargs['length']
            width = kwargs['width']
            return length * width
        else:
            raise ValueError("For 'rectangle', 'length' and 'width' are required.")
    elif shape.lower() == 'triangle':
        if 'base' in kwargs and 'height' in kwargs:
            base = kwargs['base']
            height = kwargs['height']
            return 0.5 * base * height
        else:
            raise ValueError("For 'triangle', 'base' and 'height' are required.")
    else:
        raise ValueError("Unsupported shape. Choose from 'circle', 'rectangle', or 'triangle'.")

# Example usage:
print(f"Area of circle (radius=5): {calculate_area('circle', radius=5):.2f}")
print(f"Area of rectangle (length=4, width=6): {calculate_area('rectangle', length=4, width=6):.2f}")
print(f"Area of triangle (base=7, height=3): {calculate_area('triangle', base=7, height=3):.2f}")

# Example of handling an error (uncomment to test):
# try:
#     print(calculate_area('square', side=5))
# except ValueError as e:
#     print(f"Error: {e}")
```



```
import math

def calculate_area(shape, **kwargs):
    """
    Calculates the area of different geometric shapes.

    Args:
        shape (str): The type of shape ('circle', 'rectangle', 'triangle').
        **kwargs: Keyword arguments for the dimensions of the shape:
                  - For 'circle': 'radius' (float)
                  - For 'rectangle': 'length' (float), 'width' (float)
                  - For 'triangle': 'base' (float), 'height' (float)

    Returns:
        float: The calculated area of the shape.
        str: An error message if the shape is not recognized or dimensions are missing.
    """
    if shape.lower() == 'circle':
        radius = kwargs.get('radius')
        if radius is not None:
            return math.pi * radius**2
        else:
            return "Error: Radius is required for a circle."
    elif shape.lower() == 'rectangle':
        length = kwargs.get('length')
        width = kwargs.get('width')
        if length is not None and width is not None:
            return length * width
        else:
            return "Error: Length and width are required for a rectangle."
    elif shape.lower() == 'triangle':
```

Gemini

Write a Python function to calculate area of circle, rectangle, and triangle and explain the code

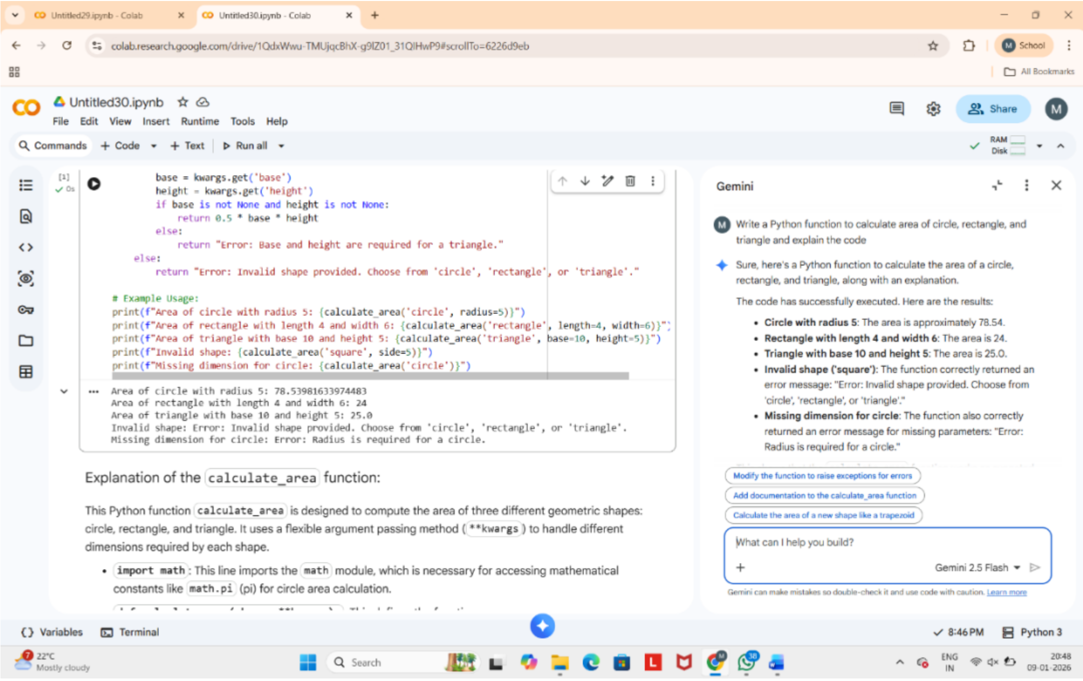
Sure, here's a Python function to calculate the area of a circle, rectangle, and triangle, along with an explanation.

The code has successfully executed. Here are the results:

- **Circle with radius 5**: The area is approximately 78.54.
- **Rectangle with length 4 and width 6**: The area is 24.
- **Triangle with base 10 and height 5**: The area is 25.0.
- **Invalid shape ('square')**: The function correctly returned an error message: "Error: Invalid shape provided. Choose from 'circle', 'rectangle', or 'triangle'."
- **Missing dimension for circle**: The function also correctly returned an error message for missing parameters: "Error: Radius is required for a circle."

Modify the function to raise exceptions for errors
Add documentation to the calculate_area function
Calculate the area of a new shape like a trapezoid
What can I help you build?
Gemini 2.5 Flash

Gemini can make mistakes so double-check it and use code with caution. Learn more



```
        base = kwargs.get('base')
        height = kwargs.get('height')
        if base is not None and height is not None:
            return 0.5 * base * height
        else:
            return "Error: Base and height are required for a triangle."
    else:
        return "Error: Invalid shape provided. Choose from 'circle', 'rectangle', or 'triangle'."

# Example Usage:
print(f"Area of circle with radius 5: {calculate_area('circle', radius=5)}")
print(f"Area of rectangle with length 4 and width 6: {calculate_area('rectangle', length=4, width=6)}"
print(f"Area of triangle with base 10 and height 5: {calculate_area('triangle', base=10, height=5)}")
print(f"Invalid shape: {calculate_area('square', side=5)}")
print(f"Missing dimension for circle: {calculate_area('circle')}")
```

```
Area of circle with radius 5: 78.53981633974483
Area of rectangle with length 4 and width 6: 24
Area of triangle with base 10 and height 5: 25.0
Invalid shape: Error: Invalid shape provided. Choose from 'circle', 'rectangle', or 'triangle'.
Missing dimension for circle: Error: Radius is required for a circle.
```

Explanation of the `calculate_area` function:

This Python function `calculate_area` is designed to compute the area of three different geometric shapes: circle, rectangle, and triangle. It uses a flexible argument passing method (`**kwargs`) to handle different dimensions required by each shape.

- `import math`: This line imports the `math` module, which is necessary for accessing mathematical constants like `math.pi` (pi) for circle area calculation.

Gemini

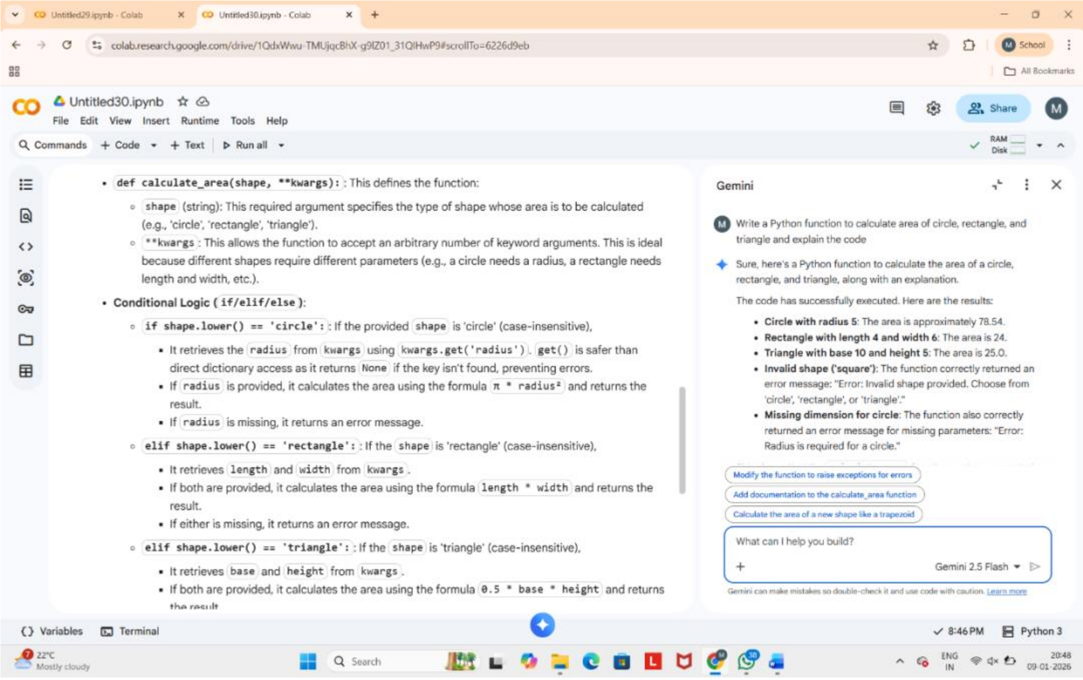
Write a Python function to calculate area of circle, rectangle, and triangle and explain the code

Sure, here's a Python function to calculate the area of a circle, rectangle, and triangle, along with an explanation.

The code has successfully executed. Here are the results:

- **Circle with radius 5**: The area is approximately 78.54.
- **Rectangle with length 4 and width 6**: The area is 24.
- **Triangle with base 10 and height 5**: The area is 25.0.
- **Invalid shape ('square')**: The function correctly returned an error message: "Error: Invalid shape provided. Choose from 'circle', 'rectangle', or 'triangle'."
- **Missing dimension for circle**: The function also correctly returned an error message for missing parameters: "Error: Radius is required for a circle."

Gemini can make mistakes so double-check it and use code with caution. Learn more



- `def calculate_area(shape, **kwargs):` : This defines the function:
  - `shape` (string): This required argument specifies the type of shape whose area is to be calculated (e.g., 'circle', 'rectangle', 'triangle').
  - `**kwargs` : This allows the function to accept an arbitrary number of keyword arguments. This is ideal because different shapes require different parameters (e.g., a circle needs a radius, a rectangle needs length and width, etc.).
- **Conditional Logic (`if/elif/else`):**
  - `if shape.lower() == 'circle':` : If the provided `shape` is 'circle' (case-insensitive),
    - It retrieves the `radius` from `kwargs` using `kwargs.get('radius')`. `get()` is safer than direct dictionary access as it returns `None` if the key isn't found, preventing errors.
    - If `radius` is provided, it calculates the area using the formula `π * radius²` and returns the result.
    - If `radius` is missing, it returns an error message.
  - `elif shape.lower() == 'rectangle':` : If the `shape` is 'rectangle' (case-insensitive),
    - It retrieves `length` and `width` from `kwargs`.
    - If both are provided, it calculates the area using the formula `length * width` and returns the result.
    - If either is missing, it returns an error message.
  - `elif shape.lower() == 'triangle':` : If the `shape` is 'triangle' (case-insensitive),
    - It retrieves `base` and `height` from `kwargs`.
    - If both are provided, it calculates the area using the formula `0.5 * base * height` and returns the result

Write a Python function to calculate area of circle, rectangle, and triangle and explain the code

Sure, here's a Python function to calculate the area of a circle, rectangle, and triangle, along with an explanation.

The code has successfully executed. Here are the results:

- **Circle with radius 5**: The area is approximately 78.54.
- **Rectangle with length 4 and width 6**: The area is 24.
- **Triangle with base 10 and height 5**: The area is 25.0.
- **Invalid shape ('square')**: The function correctly returned an error message: "Error: Invalid shape provided. Choose from 'circle', 'rectangle', or 'triangle'."
- **Missing dimension for circle**: The function also correctly returned an error message for missing parameters: "Error: Radius is required for a circle."

Modify the function to raise exceptions for errors
Add documentation to the calculate_area function
Calculate the area of a new shape like a trapezoid
What can I help you build?
Gemini 2.5 Flash
Gemini can make mistakes so double-check it and use code with caution. Learn more

Explanation :

The function accepts the shape name and required dimensions as input. Conditional statements decide which formula to apply. For a circle, radius is used, while rectangle and triangle require two values. The function returns the calculated area based on the selected shape.

Task 3: Prompt Sensitivity Experiment (Cursor AI)

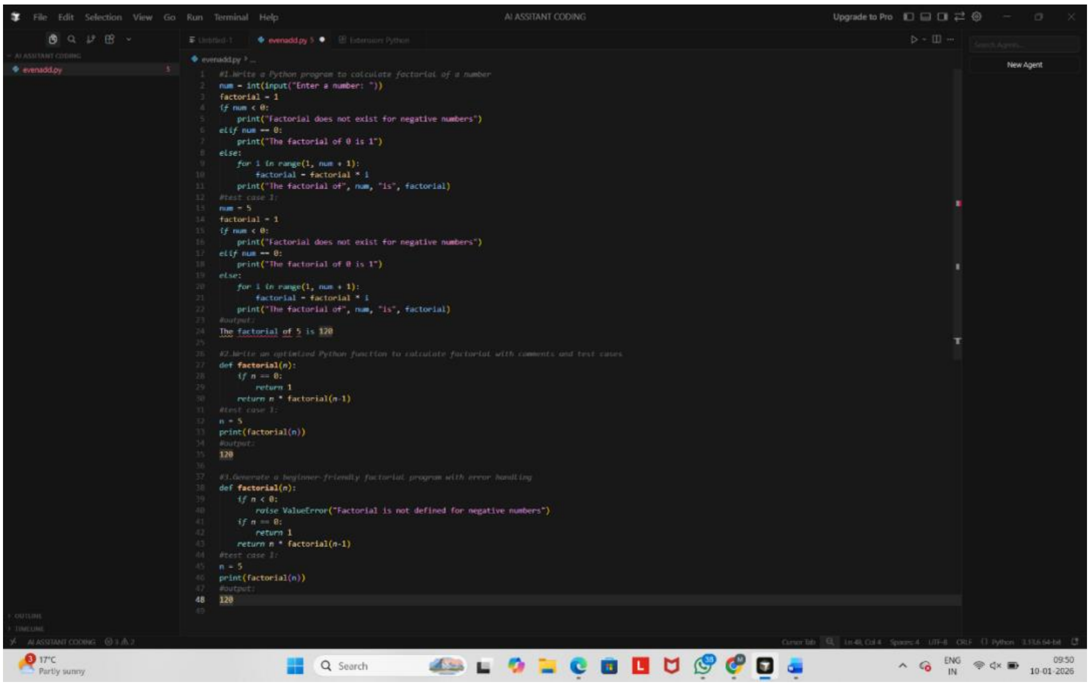
Use Cursor AI with different prompts for the same problem and observe

code changes.

The factorial problem was used to test how different prompts affect AI-generated code.

Prompts Used :

1. Write a Python program to calculate factorial
2. Write an optimized Python function to calculate factorial with comments
3. Generate a beginner-friendly factorial program with error handling



```
#1.Write a Python program to calculate factorial of a number
num = int(input("Enter a number: "))
factorial = 1
if num < 0:
    print("Factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1, num + 1):
        factorial = factorial * i
    print("The factorial of", num, "is", factorial)
#test case 1:
num = 5
factorial = 1
if num < 0:
    print("Factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1, num + 1):
        factorial = factorial * i
    print("The factorial of", num, "is", factorial)
#output:
The factorial of 5 is 120

#2.Write an optimized Python function to calculate factorial with comments and test cases
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n-1)
#test case 1:
n = 5
print(factorial(n))
#output:
120

#3.Generate a beginner-friendly factorial program with error handling
def factorial(n):
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers")
    if n == 0:
        return 1
    return n * factorial(n-1)
#test case 1:
n = 5
print(factorial(n))
#output:
120
```

Observation

The AI produced different code styles depending on the prompt. Optimized prompts improved performance and readability, while beginner-friendly prompts added validation and safety checks.

Task 4: Tool Comparison Reflection

Comparison of Gemini, Copilot, and Cursor AI

Reflection :

Google Gemini is best suited for explanations and learning support. GitHub Copilot provides real-time inline suggestions, improving developer productivity. Cursor AI excels in experimentation, refactoring, and prompt-based exploration. Each tool serves a different purpose, and choosing the right one depends on whether the goal is learning, development speed, or code analysis.