AI ASSISTANT CODING
LAB-02

Name : V . Shiva Harsha

Roll No : 2303A5375

Batch No : 33

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 :

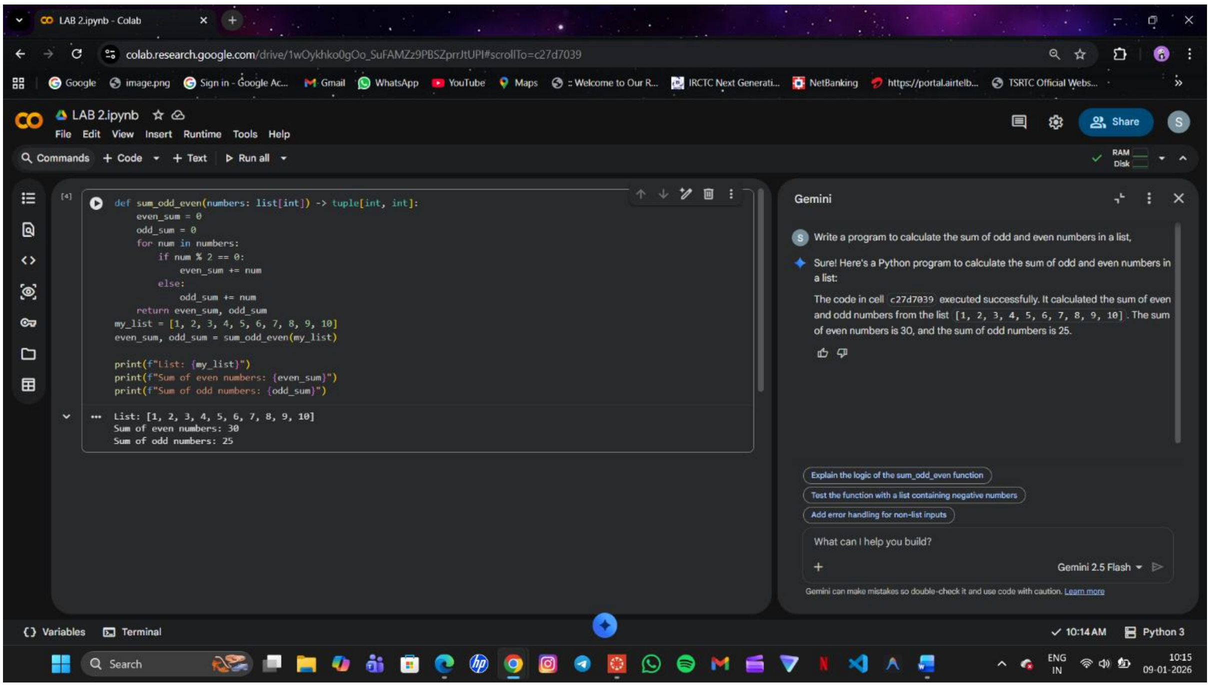
Write a program to calculate the sum of odd and even numbers in a list,
Refactor this Python code to improve readability and efficiency.

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}")
```


LAB 2.ipynb - Colab
colab.research.google.com/drive/1wOykhko0gOo_SuFAMZz9PBSZprrJtUPI#scrollTo=c27d7039
LAB 2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM
Disk
[4]
Gemini
Explain the logic of the sum_odd_even function
Test the function with a list containing negative numbers
Add error handling for non-list inputs
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
Variables
Terminal
10:14 AM
Python 3
10:15
09-01-2026

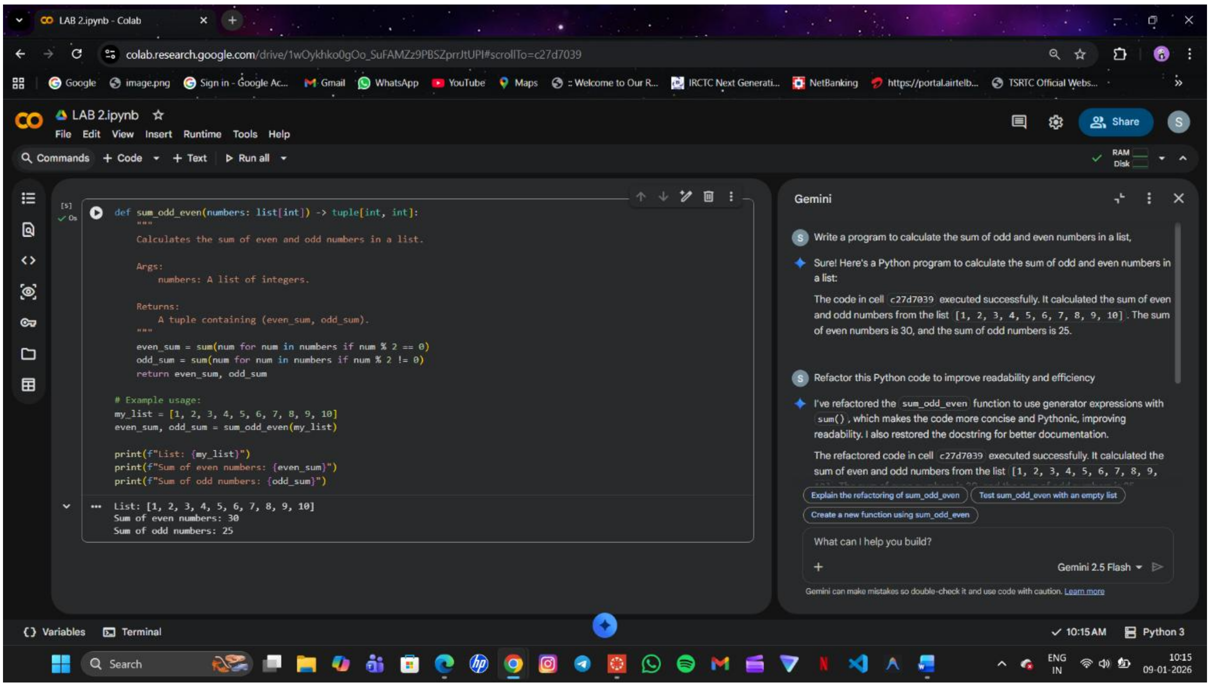
```python
def sum_odd_even(numbers: list[int]) -> tuple[int, int]:
    even_sum = 0
    odd_sum = 0
    for num in numbers:
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num
    return even_sum, odd_sum
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum, odd_sum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

```
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 in cell `c27d7039` executed successfully. It calculated the sum of even and odd numbers from the list `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`. The sum of even numbers is 30, and the sum of odd numbers is 25.


LAB 2.ipynb - Colab
colab.research.google.com/drive/1wOykhko0gOo_SuFAMZz9PBSZprrJtUPI#scrollTo=c27d7039
LAB 2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM
Disk
[5]
0s
Gemini
Explain the refactoring of sum_odd_even
Test sum_odd_even with an empty list
Create a new function using sum_odd_even
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
Variables
Terminal
10:15 AM
Python 3
10:15
09-01-2026

```python
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}")
```

```
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 in cell `c27d7039` executed successfully. It calculated the sum of even and odd numbers from the list `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`. The sum of even numbers is 30, and the sum of odd numbers is 25.

Refactor this Python code to improve readability and efficiency

I've refactored the `sum_odd_even` function to use generator expressions with `sum()`, which makes the code more concise and Pythonic, improving readability. I also restored the docstring for better documentation.

The refactored code in cell `c27d7039` executed successfully. It calculated the sum of even and odd numbers from the list `[1, 2, 3, 4, 5, 6, 7, 8, 9,`

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)

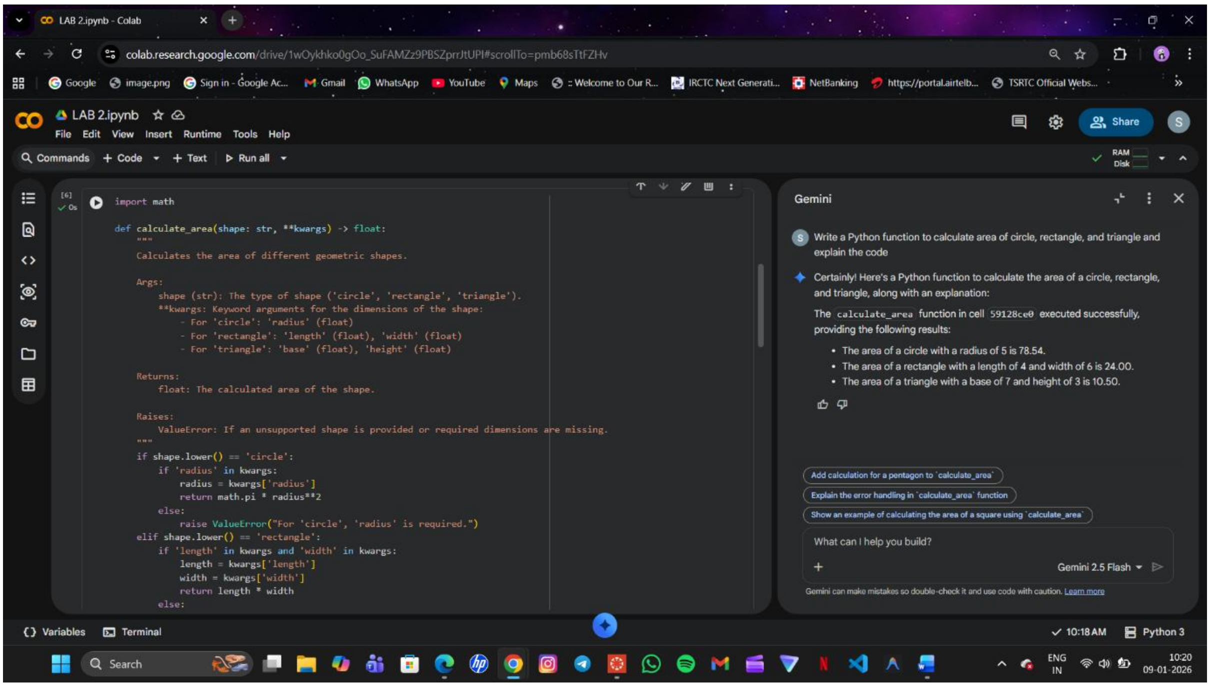
Scenario

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}")
```


LAB 2.ipynb - Colab
colab.research.google.com/drive/1wOykhko0gOo_SuFAMZz9PBSZprrJtUPI#scrollTo=pmb68sTtFZHv
LAB 2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
RAM
Disk
[6]
0s
Variables
Terminal
10:18 AM
Python 3
10:20
09-01-2026

```python
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:
```

Gemini

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

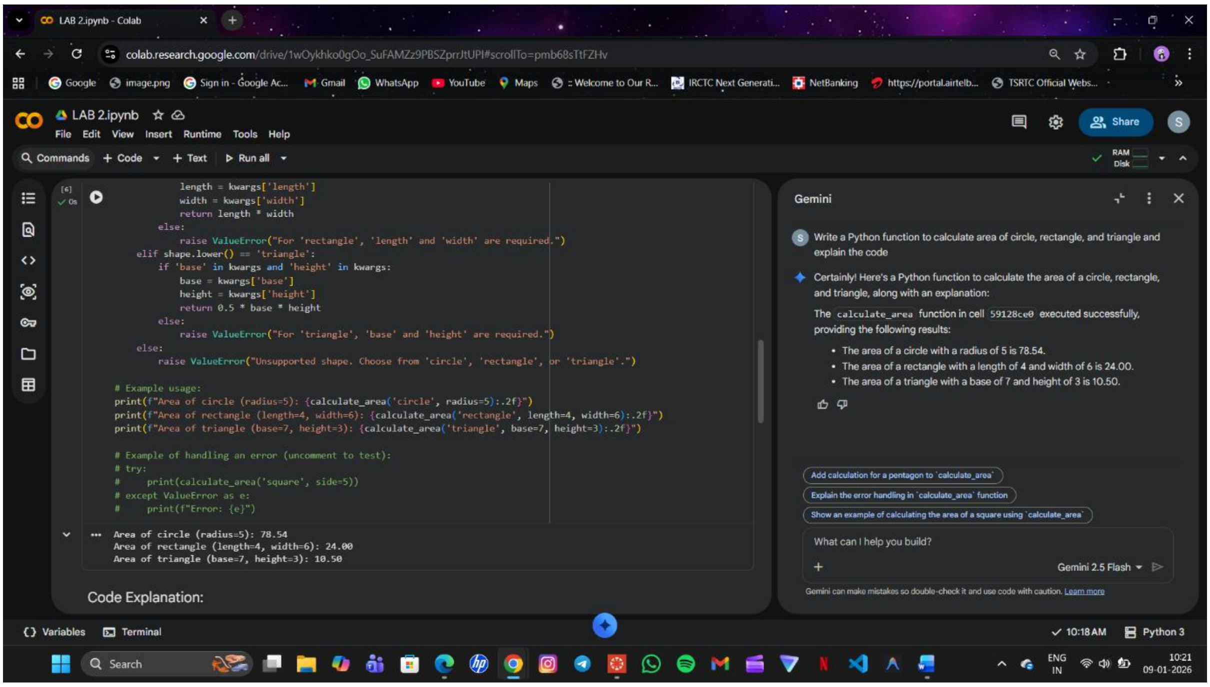
Certainly! Here's a Python function to calculate the area of a circle, rectangle, and triangle, along with an explanation:

The `calculate_area` function in cell `59128ce0` executed successfully, providing the following results:

- The area of a circle with a radius of 5 is 78.54.
- The area of a rectangle with a length of 4 and width of 6 is 24.00.
- The area of a triangle with a base of 7 and height of 3 is 10.50.

Add calculation for a pentagon to `calculate_area`
Explain the error handling in `calculate_area` function
Show an example of calculating the area of a square using `calculate_area`
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


LAB 2.ipynb - Colab
colab.research.google.com/drive/1wOykhko0gOo_SuFAMZz9PBSZprrJtUPI#scrollTo=pmb68sTtFZHv
LAB 2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
[6]
0s
Variables
Terminal
10:18 AM
Python 3
10:21
09-01-2026

```python
            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}")
```

```
Area of circle (radius=5): 78.54
Area of rectangle (length=4, width=6): 24.00
Area of triangle (base=7, height=3): 10.50
```

Code Explanation:

Gemini

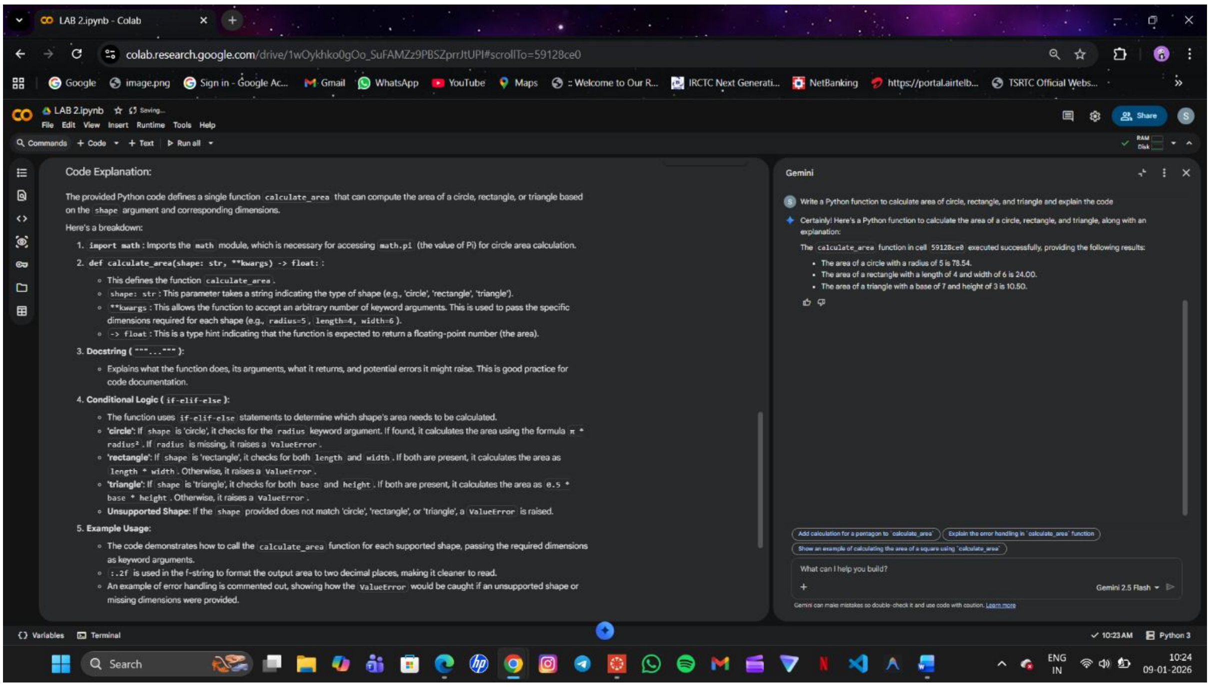
Write a Python function to calculate area of circle, rectangle, and triangle and explain the code

Certainly! Here's a Python function to calculate the area of a circle, rectangle, and triangle, along with an explanation:

The `calculate_area` function in cell `59128ce0` executed successfully, providing the following results:

- The area of a circle with a radius of 5 is 78.54.
- The area of a rectangle with a length of 4 and width of 6 is 24.00.
- The area of a triangle with a base of 7 and height of 3 is 10.50.

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


LAB 2.ipynb - Colab
colab.research.google.com/drive/1wOykhko0gOo_SuFAMZz9PBSZprrJtUPI#scrollTo=59128ce0
LAB 2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Share
Gemini
Variables Terminal
10:23 AM Python 3
10:24 09-01-2026

Code Explanation:

The provided Python code defines a single function `calculate_area` that can compute the area of a circle, rectangle, or triangle based on the `shape` argument and corresponding dimensions.

Here's a breakdown:

1. `import math` : Imports the `math` module, which is necessary for accessing `math.pi` (the value of Pi) for circle area calculation.
2. `def calculate_area(shape: str, **kwargs) -> float:` :
   - This defines the function `calculate_area`.
   - `shape: str` : This parameter takes a string indicating the type of shape (e.g., 'circle', 'rectangle', 'triangle').
   - `**kwargs` : This allows the function to accept an arbitrary number of keyword arguments. This is used to pass the specific dimensions required for each shape (e.g., `radius=5`, `length=4, width=6`).
   - `-> float` : This is a type hint indicating that the function is expected to return a floating-point number (the area).
3. **Docstring (** `"""..."""` **):**
   - Explains what the function does, its arguments, what it returns, and potential errors it might raise. This is good practice for code documentation.
4. **Conditional Logic (** `if-elif-else` **):**
   - The function uses `if-elif-else` statements to determine which shape's area needs to be calculated.
   - **'circle'**: If `shape` is 'circle', it checks for the `radius` keyword argument. If found, it calculates the area using the formula `π * radius²`. If `radius` is missing, it raises a `ValueError`.
   - **'rectangle'**: If `shape` is 'rectangle', it checks for both `length` and `width`. If both are present, it calculates the area as `length * width`. Otherwise, it raises a `ValueError`.
   - **'triangle'**: If `shape` is 'triangle', it checks for both `base` and `height`. If both are present, it calculates the area as `0.5 * base * height`. Otherwise, it raises a `ValueError`.
   - **Unsupported Shape:** If the `shape` provided does not match 'circle', 'rectangle', or 'triangle', a `ValueError` is raised.
5. **Example Usage:**
   - The code demonstrates how to call the `calculate_area` function for each supported shape, passing the required dimensions as keyword arguments.
   - `:.2f` is used in the f-string to format the output area to two decimal places, making it cleaner to read.
   - An example of error handling is commented out, showing how the `ValueError` would be caught if an unsupported shape or missing dimensions were provided.

Gemini

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

Certainly! Here's a Python function to calculate the area of a circle, rectangle, and triangle, along with an explanation:

The `calculate_area` function in cell `59128ce0` executed successfully, providing the following results:

- The area of a circle with a radius of 5 is 78.54.
- The area of a rectangle with a length of 4 and width of 6 is 24.00.
- The area of a triangle with a base of 7 and height of 3 is 10.50.

Add calculation for a pentagon to `calculate_area`

Explain the error handling in `calculate_area` function

Show an example of calculating the area of a square using `calculate_area`

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)

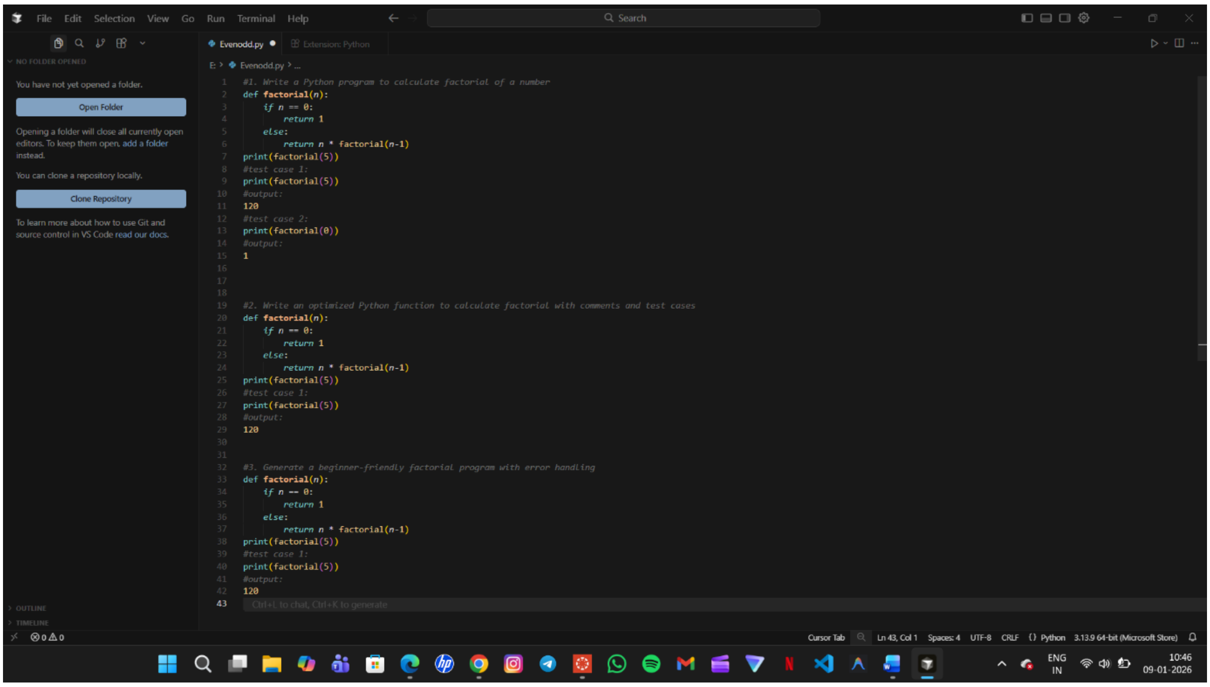
Use Cursor AI with different prompts for the same problem and observe

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



You have not yet opened a folder.

Opening a folder will close all currently open editors. To keep them open, add a folder instead.

You can clone a repository locally.

To learn more about how to use Git and source control in VS Code read our docs.

```python
#1. Write a Python program to calculate factorial of a number
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
print(factorial(5))
#test case 1:
print(factorial(5))
#output:
120
#test case 2:
print(factorial(0))
#output:
1



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


#3. Generate a beginner-friendly factorial program with error handling
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
print(factorial(5))
#test case 1:
print(factorial(5))
#output:
120
Ctrl+L to chat, Ctrl+K to generate
```

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.