# AI ASSISTED CODING

**NAME:B.SRISHANTH**

**ROLL NO:2403A510G3**

**ASSIGNMENT : 2.3**

**Lab Outcomes (LOs):**

After completing this lab, students will be able to:

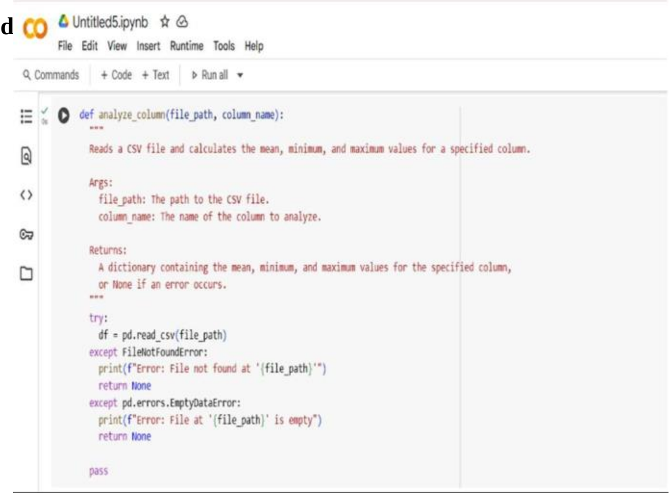
- Generate Python code using Google Gemini in Google Colab.
- Analyze the effectiveness of code explanations and suggestions by Gemini.
- Set up and use Cursor AI for AI-powered coding assistance.
- Evaluate and refactor code using Cursor AI features.
- Compare AI tool behavior and code quality across different platforms.

1

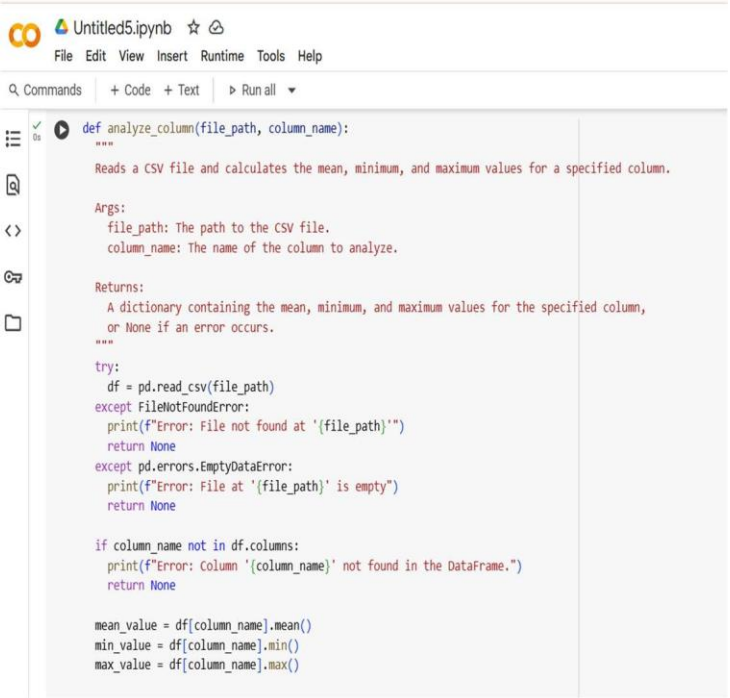
**Task Description#1**

- Use Google Gemini in Colab to write a function that reads a CSV file and calculates mean, min, max

**Expected**

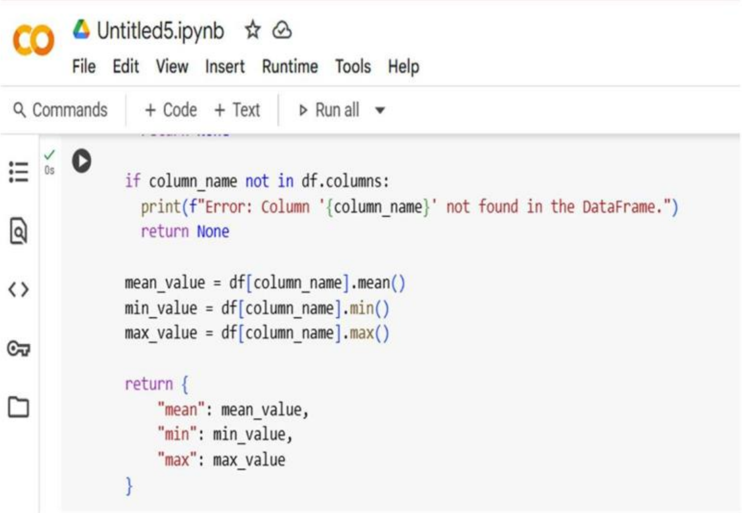


```
def analyze_column(file_path, column_name):
  """
  Reads a CSV file and calculates the mean, minimum, and maximum values for a specified column.

  Args:
    file_path: The path to the CSV file.
    column_name: The name of the column to analyze.

  Returns:
    A dictionary containing the mean, minimum, and maximum values for the specified column,
    or None if an error occurs.
  """
  try:
    df = pd.read_csv(file_path)
  except FileNotFoundError:
    print(f"Error: File not found at '{file_path}'")
    return None
  except pd.errors.EmptyDataError:
    print(f"Error: File at '{file_path}' is empty")
    return None

  pass
```


Untitled5.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

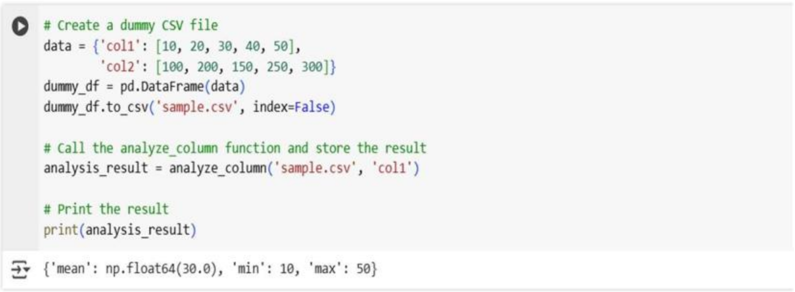
```
def analyze_column(file_path, column_name):
  """
  Reads a CSV file and calculates the mean, minimum, and maximum values for a specified column.

  Args:
    file_path: The path to the CSV file.
    column_name: The name of the column to analyze.

  Returns:
    A dictionary containing the mean, minimum, and maximum values for the specified column,
    or None if an error occurs.
  """
  try:
    df = pd.read_csv(file_path)
  except FileNotFoundError:
    print(f"Error: File not found at '{file_path}'")
    return None
  except pd.errors.EmptyDataError:
    print(f"Error: File at '{file_path}' is empty")
    return None

  if column_name not in df.columns:
    print(f"Error: Column '{column_name}' not found in the DataFrame.")
    return None

  mean_value = df[column_name].mean()
  min_value = df[column_name].min()
  max_value = df[column_name].max()
```

-


Untitled5.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

```
  if column_name not in df.columns:
    print(f"Error: Column '{column_name}' not found in the DataFrame.")
    return None

  mean_value = df[column_name].mean()
  min_value = df[column_name].min()
  max_value = df[column_name].max()

  return {
      "mean": mean_value,
      "min": min_value,
      "max": max_value
  }
```

-



```
# Create a dummy CSV file
data = {'col1': [10, 20, 30, 40, 50],
        'col2': [100, 200, 150, 250, 300]}
dummy_df = pd.DataFrame(data)
dummy_df.to_csv('sample.csv', index=False)

# Call the analyze_column function and store the result
analysis_result = analyze_column('sample.csv', 'col1')

# Print the result
print(analysis_result)
```

```
{'mean': np.float64(30.0), 'min': 10, 'max': 50}
```

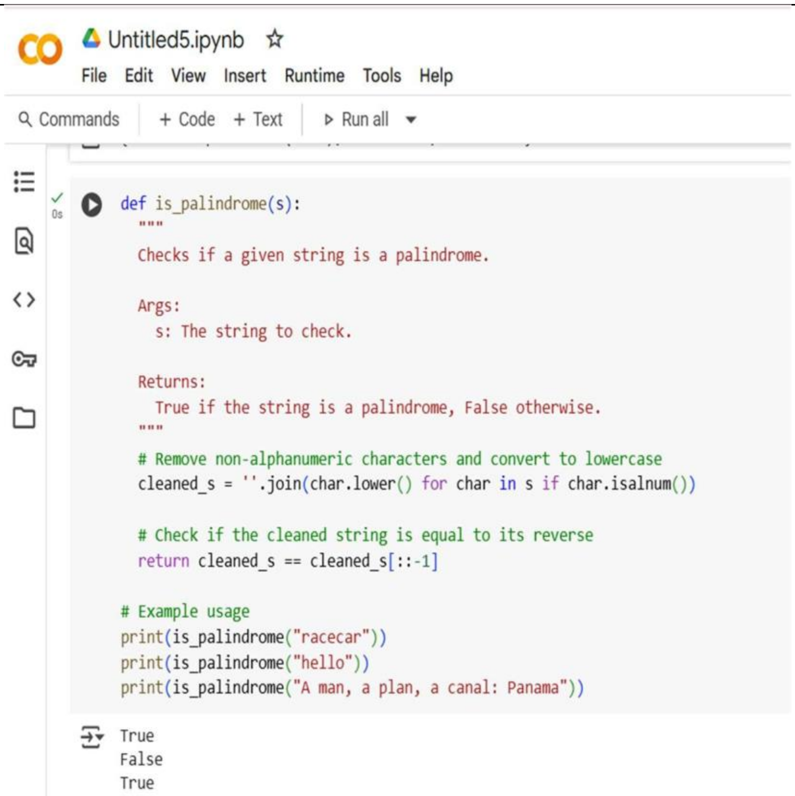
-

**Task Description#2**

- Compare Gemini and Copilot outputs for a palindrome check function.

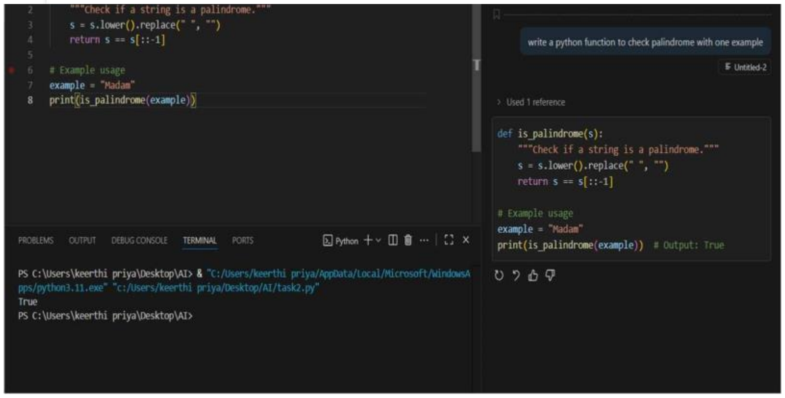
**Expected Output#2**

- Side-by-side comparison and observations
- **GEMINI:**



```python
def is_palindrome(s):
  """
  Checks if a given string is a palindrome.

  Args:
    s: The string to check.

  Returns:
    True if the string is a palindrome, False otherwise.
  """
  # Remove non-alphanumeric characters and convert to lowercase
  cleaned_s = ''.join(char.lower() for char in s if char.isalnum())

  # Check if the cleaned string is equal to its reverse
  return cleaned_s == cleaned_s[::-1]

# Example usage
print(is_palindrome("racecar"))
print(is_palindrome("hello"))
print(is_palindrome("A man, a plan, a canal: Panama"))
```

```
True
False
True
```



```python
    """Check if a string is a palindrome."""
    s = s.lower().replace(" ", "")
    return s == s[::-1]

# Example usage
example = "Madam"
print(is_palindrome(example))
```

```
PS C:\Users\keerthi priya\Desktop\AI> & "C:/Users/keerthi priya/AppData/Local/Microsoft/WindowsA
pps/python3.11.exe" "c:/Users/keerthi priya/Desktop/AI/task2.py"
True
PS C:\Users\keerthi priya\Desktop\AI>
```

write a python function to check palindrome with one example

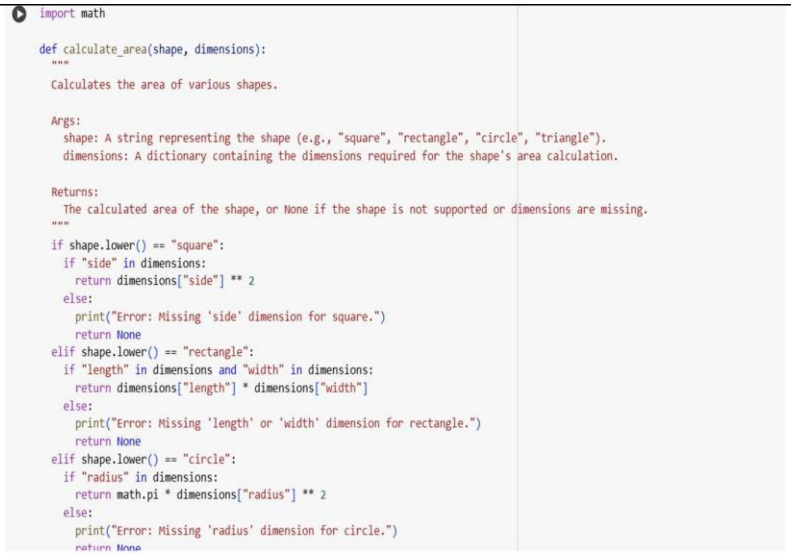
```python
def is_palindrome(s):
    """Check if a string is a palindrome."""
    s = s.lower().replace(" ", "")
    return s == s[::-1]

# Example usage
example = "Madam"
print(is_palindrome(example))  # Output: True
```

**Task Description#3**

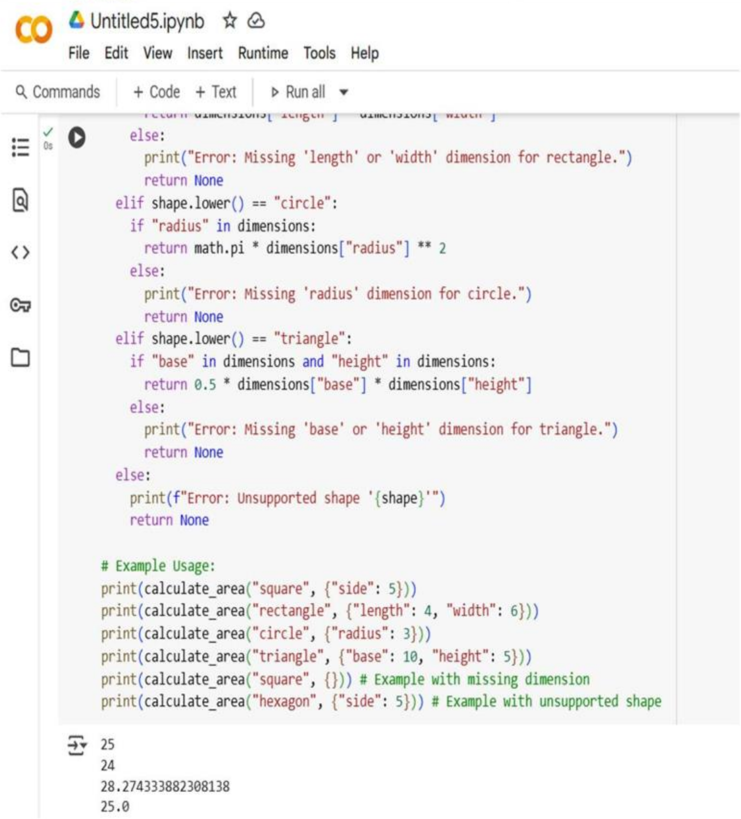
- Ask Gemini to explain a Python function (to calculate area of various shapes) line by line..

**Expected Output#3**

- Detailed explanation with code snippet

- 

```python
import math

def calculate_area(shape, dimensions):
  """
  Calculates the area of various shapes.

  Args:
    shape: A string representing the shape (e.g., "square", "rectangle", "circle", "triangle").
    dimensions: A dictionary containing the dimensions required for the shape's area calculation.

  Returns:
    The calculated area of the shape, or None if the shape is not supported or dimensions are missing.
  """
  if shape.lower() == "square":
    if "side" in dimensions:
      return dimensions["side"] ** 2
    else:
      print("Error: Missing 'side' dimension for square.")
      return None
  elif shape.lower() == "rectangle":
    if "length" in dimensions and "width" in dimensions:
      return dimensions["length"] * dimensions["width"]
    else:
      print("Error: Missing 'length' or 'width' dimension for rectangle.")
      return None
  elif shape.lower() == "circle":
    if "radius" in dimensions:
      return math.pi * dimensions["radius"] ** 2
    else:
      print("Error: Missing 'radius' dimension for circle.")
      return None
```

- 

```python
      return dimensions["length"] * dimensions["width"]
    else:
      print("Error: Missing 'length' or 'width' dimension for rectangle.")
      return None
  elif shape.lower() == "circle":
    if "radius" in dimensions:
      return math.pi * dimensions["radius"] ** 2
    else:
      print("Error: Missing 'radius' dimension for circle.")
      return None
  elif shape.lower() == "triangle":
    if "base" in dimensions and "height" in dimensions:
      return 0.5 * dimensions["base"] * dimensions["height"]
    else:
      print("Error: Missing 'base' or 'height' dimension for triangle.")
      return None
  else:
    print(f"Error: Unsupported shape '{shape}'")
    return None

# Example Usage:
print(calculate_area("square", {"side": 5}))
print(calculate_area("rectangle", {"length": 4, "width": 6}))
print(calculate_area("circle", {"radius": 3}))
print(calculate_area("triangle", {"base": 10, "height": 5}))
print(calculate_area("square", {})) # Example with missing dimension
print(calculate_area("hexagon", {"side": 5})) # Example with unsupported shape
```

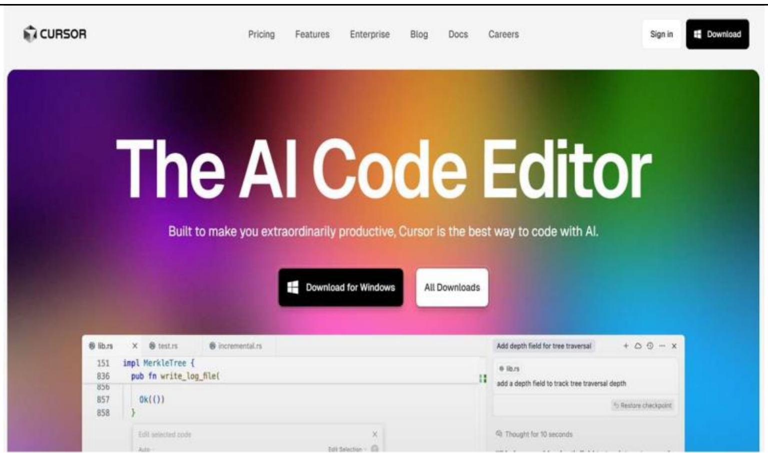
```
25
24
28.274333882308138
25.0
```

**Task Description#4**

- Install and configure Cursor AI. Use it to generate a Python function (e.g., sum of squares).

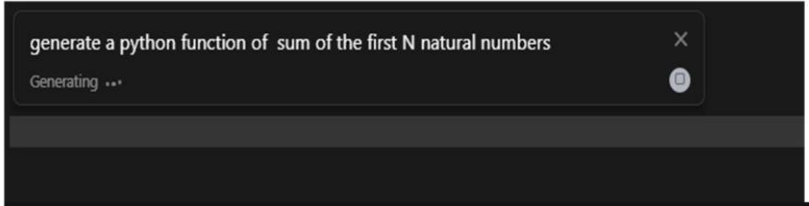
**Expected Output#4**

- Screenshots of working environments with few prompts to generate python code



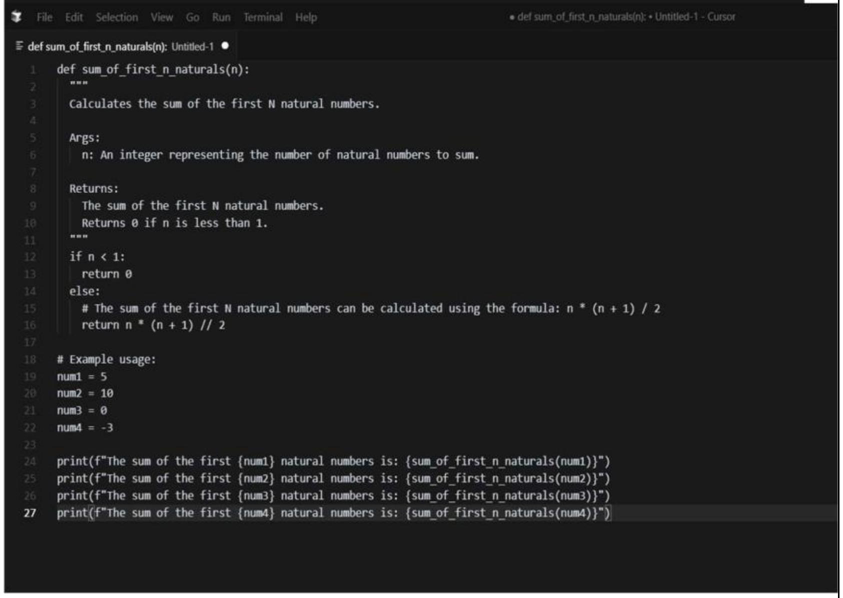
The AI Code Editor

Built to make you extraordinarily productive, Cursor is the best way to code with AI.



generate a python function of  sum of the first N natural numbers

Generating ...


File
Edit
Selection
View
Go
Run
Terminal
Help
• def sum_of_first_n_naturals(n): • Untitled-1 - Cursor
def sum_of_first_n_naturals(n): Untitled-1

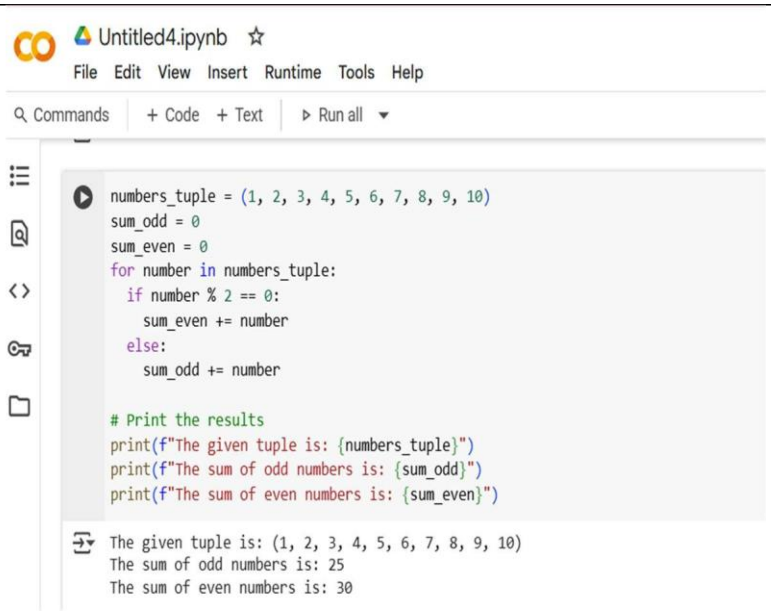
```python
def sum_of_first_n_naturals(n):
  """
  Calculates the sum of the first N natural numbers.

  Args:
    n: An integer representing the number of natural numbers to sum.

  Returns:
    The sum of the first N natural numbers.
    Returns 0 if n is less than 1.
  """
  if n < 1:
    return 0
  else:
    # The sum of the first N natural numbers can be calculated using the formula: n * (n + 1) / 2
    return n * (n + 1) // 2

# Example usage:
num1 = 5
num2 = 10
num3 = 0
num4 = -3

print(f"The sum of the first {num1} natural numbers is: {sum_of_first_n_naturals(num1)}")
print(f"The sum of the first {num2} natural numbers is: {sum_of_first_n_naturals(num2)}")
print(f"The sum of the first {num3} natural numbers is: {sum_of_first_n_naturals(num3)}")
print(f"The sum of the first {num4} natural numbers is: {sum_of_first_n_naturals(num4)}")
```

**Task Description#5**

- Student need to write code to calculate sum of add number and even numbers in the list

**Expected Output#5**

- Refactored code written by student with improved logic



```
numbers_tuple = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
sum_odd = 0
sum_even = 0
for number in numbers_tuple:
  if number % 2 == 0:
    sum_even += number
  else:
    sum_odd += number

# Print the results
print(f"The given tuple is: {numbers_tuple}")
print(f"The sum of odd numbers is: {sum_odd}")
print(f"The sum of even numbers is: {sum_even}")
```

```
The given tuple is: (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
The sum of odd numbers is: 25
The sum of even numbers is: 30
```

**Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots**

**Evaluation Criteria:**

| **Criteria** | **Max Marks** |
|---|---|
| Successful Use of Gemini in Colab (Task#1 & #2) | 1.0 |
| Code Explanation Accuracy (Gemini) (Task#3) | 0.5 |
| Cursor AI Setup and Usage (Task#4) | 0.5 |
| Refactoring and Improvement Analysis (Task#5) | 0.5 |
| **Total** | **2.5 Marks** |