**AI ASSITED CODING**

**Hallno:2403A51331**

**Name: Pujari Navya**

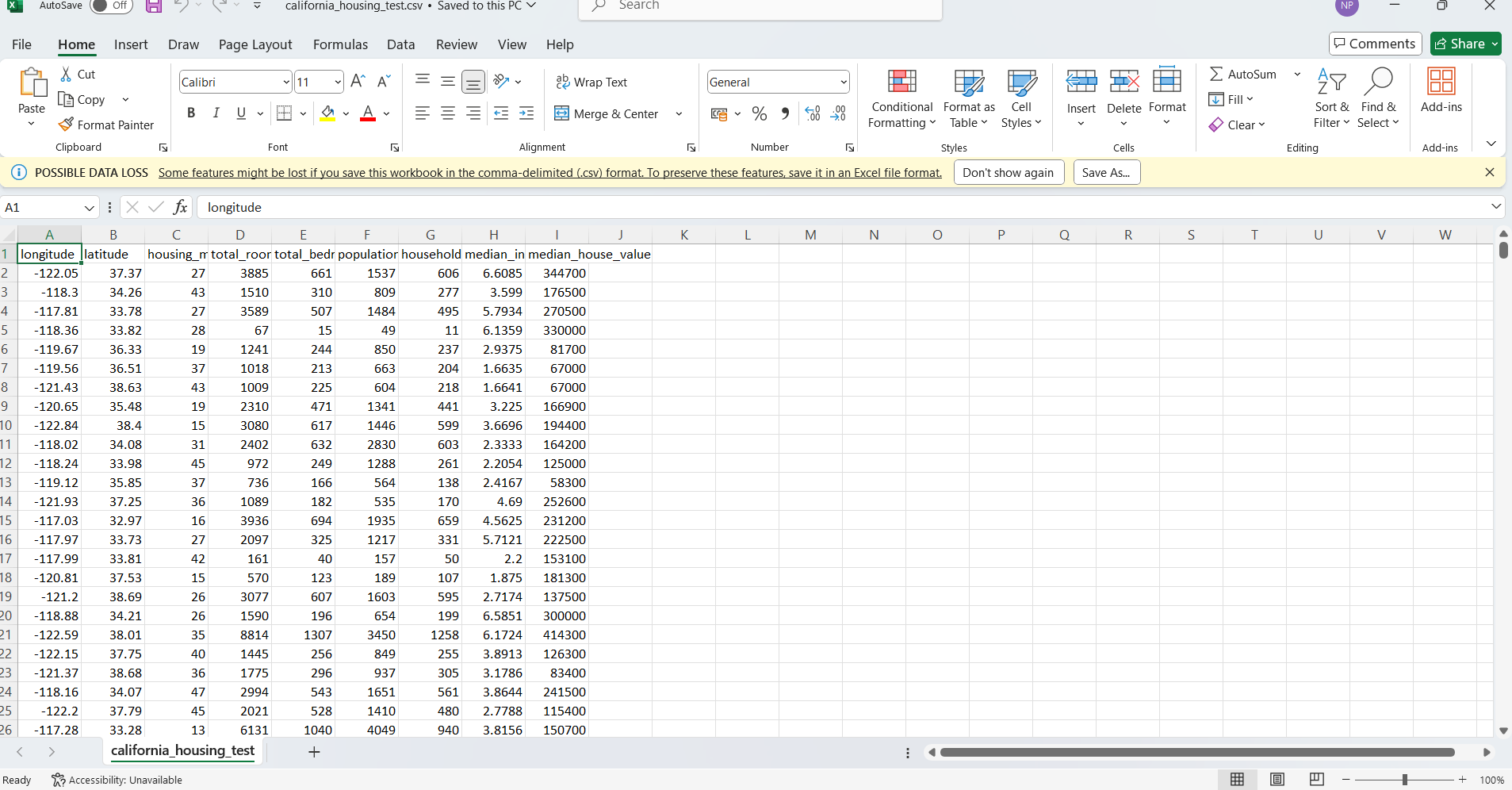
**Batch:24BTCAICSB13**

**Assignment 2.3**

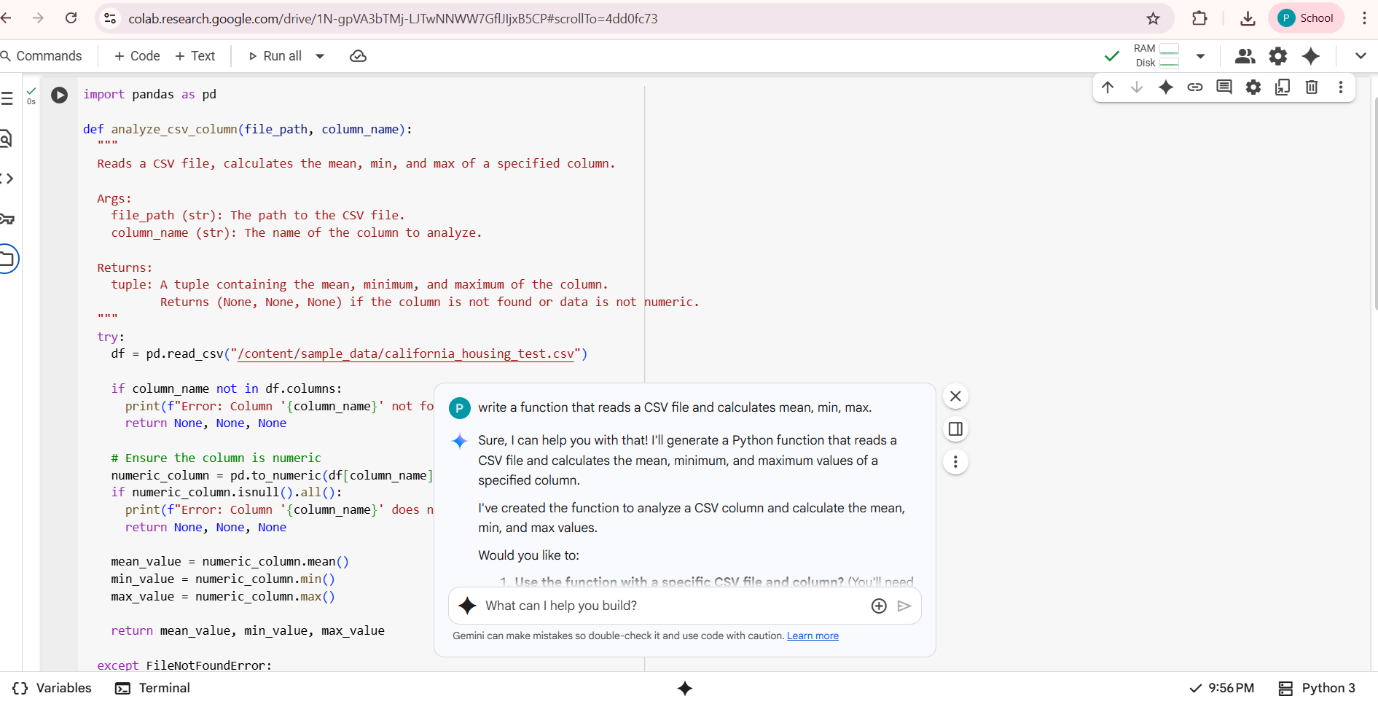
**Task 1:**

* **Prompt:** Use Google Gemini in Colab to write a function that reads a CSV file and calculates mean, min, max.

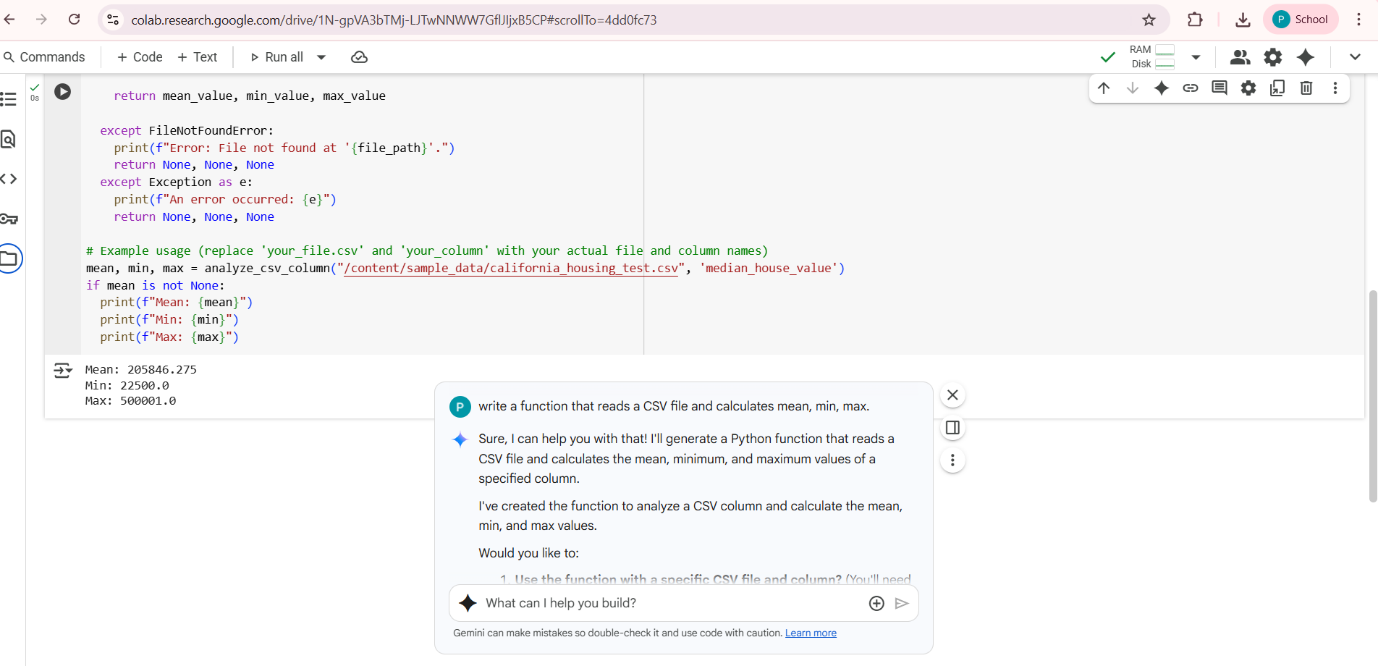
CSV File:



**Calculating mean,min,max:**



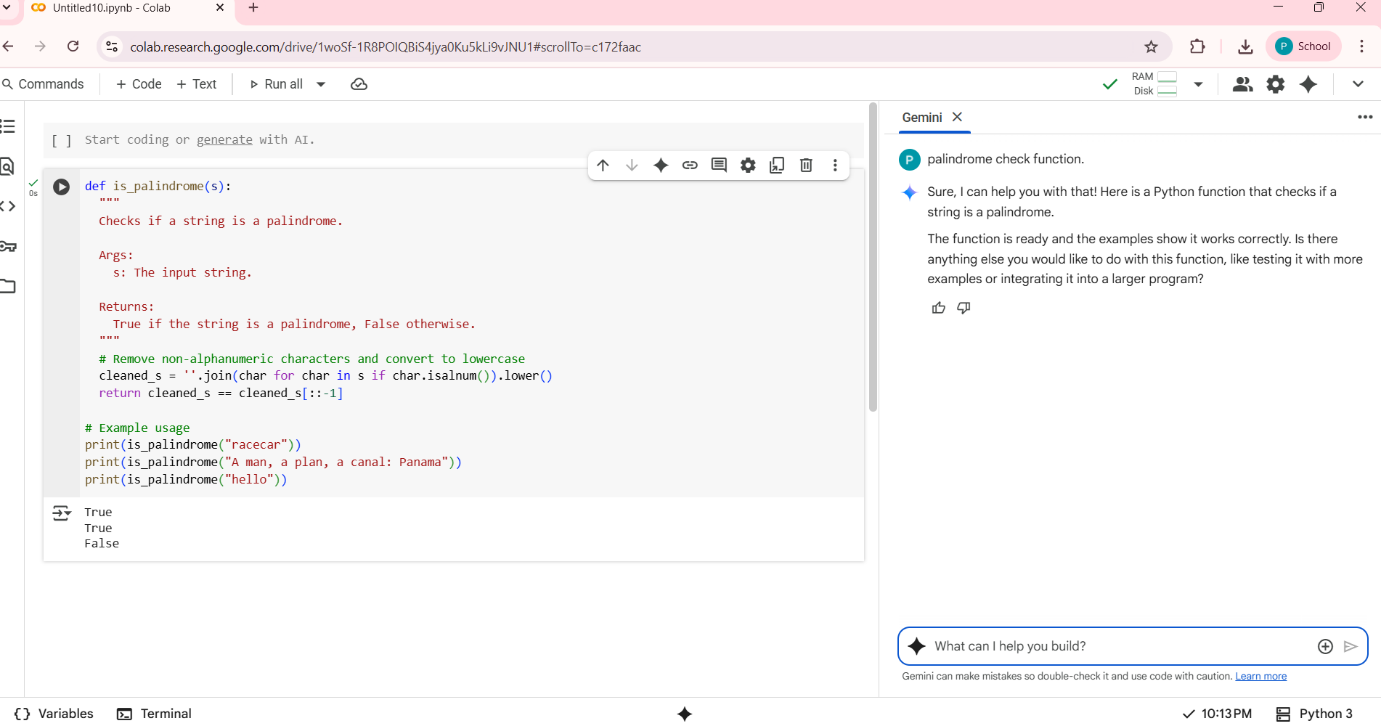
**Output:**



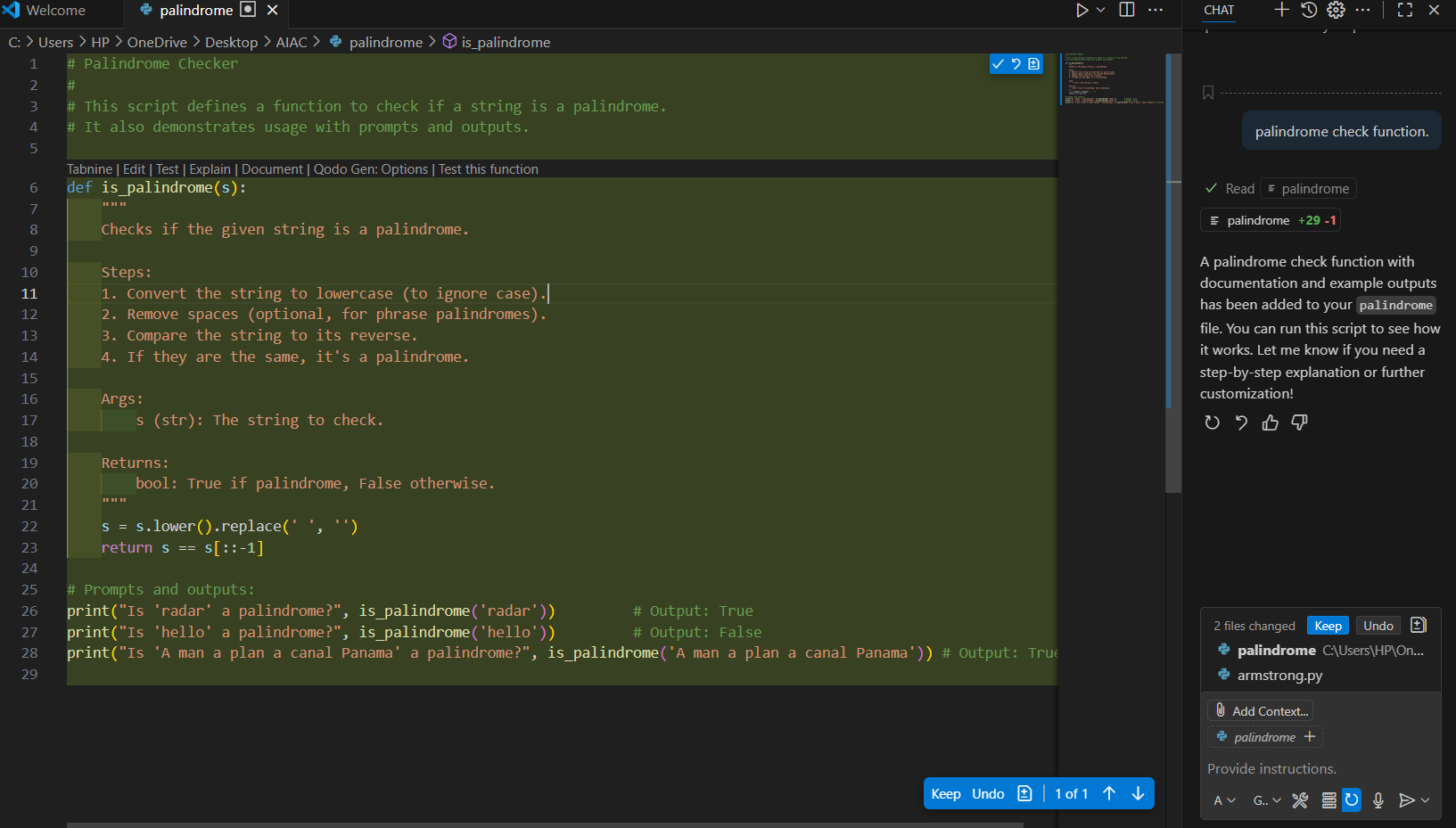
**Task-2:**

* **Prompt:** Compare Gemini and Copilot outputs for a palindrome check function.

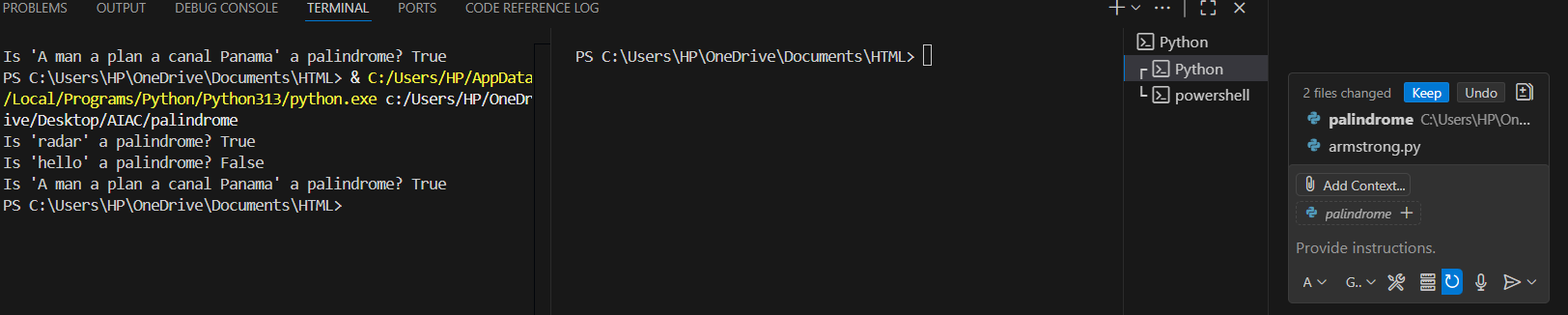
**Gemini:**



**Copilot :**

****

**Output:**

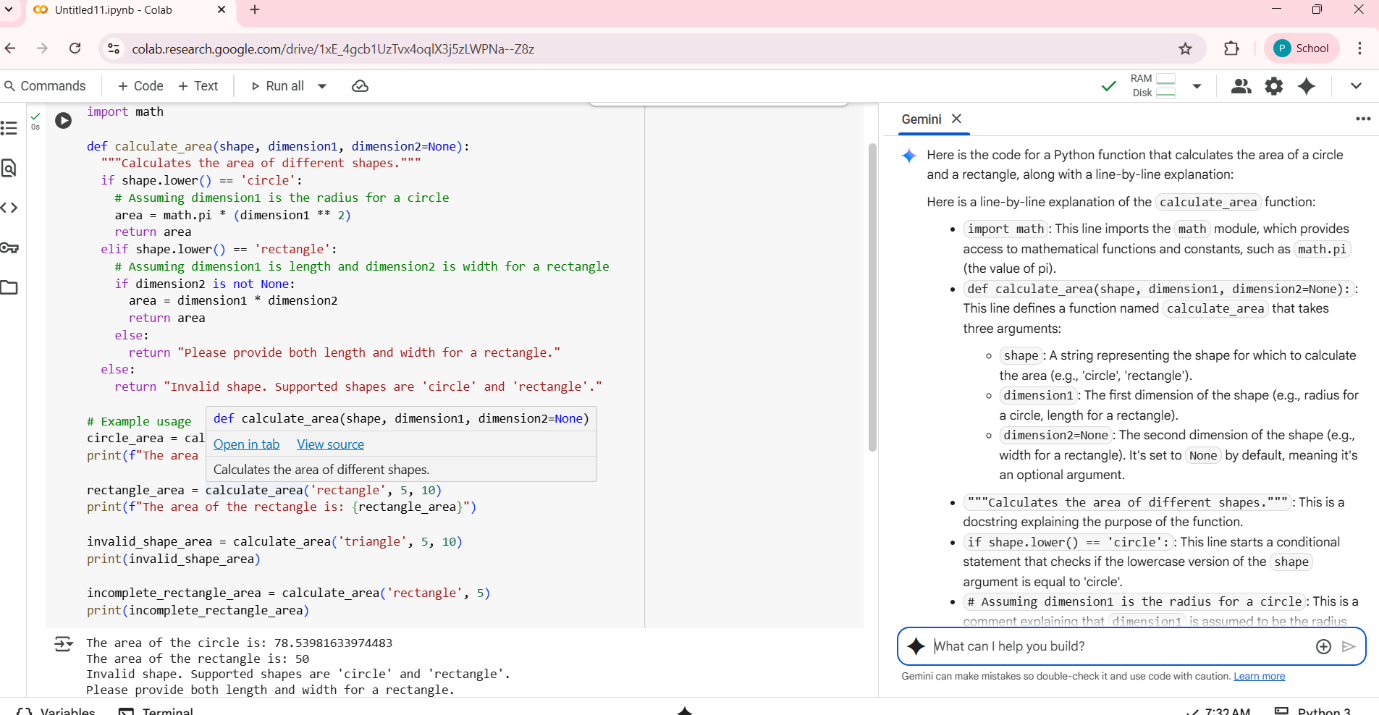
****

**Observation:**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Gemini Output** | **Copilot Output** |
| **Function Name** | **is\_palindrome** | **is\_palindrome** |
| **Spaces Handling** | **Yes** | **Yes** |
| **Readability** | **Beginner-friendly with docstring** | **Concise and practical** |
| **Robustness** | **Handles spaces and case, but not special symbols** | **Handles spaces, case, and special symbols** |
| **examples** | **Racecar,hello** | **Rececar,hello** |

**Task-3:**

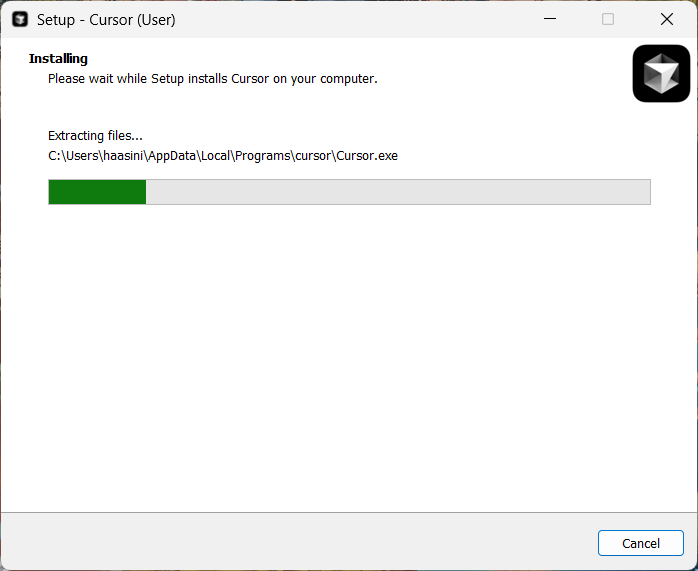
* **Prompt:** Ask Gemini to explain a Python function (to calculate area of various shapes) line by line..

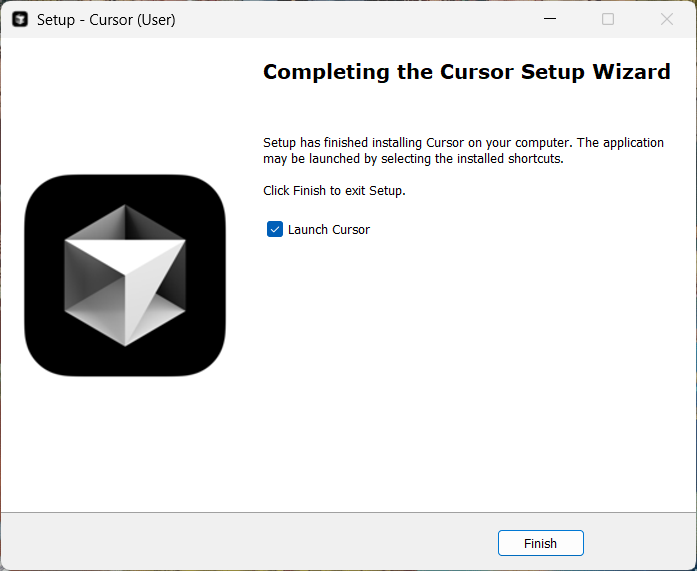


* **EXPLANATION:**
* **import math:** This line imports the math module, which provides access to mathematical functions and constants, such as math.pi (the value of pi).
* **def calculate\_area(shape, dimension1, dimension2=None)::** This line defines a function named calculate\_area that takes three arguments:
  + shape: A string representing the shape for which to calculate the area (e.g., 'circle', 'rectangle').
  + dimension1: The first dimension of the shape (e.g., radius for a circle, length for a rectangle).
  + dimension2=None: The second dimension of the shape (e.g., width for a rectangle). It's set to None by default, meaning it's an optional argument.
* **"""Calculates the area of different shapes.""**": This is a docstring explaining the purpose of the function.
* **if shape.lower() == 'circle'**:: This line starts a conditional statement that checks if the lowercase version of the shape argument is equal to 'circle'.
* **# Assuming dimension1 is the radius for a circle:** This is a comment explaining that dimension1 is assumed to be the radius for a circle.
* **area = math.pi \* (dimension1 \*\* 2):** If the shape is a circle, this line calculates the area using the formula pi \* radius^2 and stores it in the area variable.
* **return area:** This line returns the calculated area of the circle.
* **elif shape.lower() == 'rectangle'::** This line starts another conditional statement that checks if the lowercase version of the shape argument is equal to 'rectangle'.
* **# Assuming dimension1 is length and dimension2 is width for a rectangle:** This is a comment explaining that dimension1 is assumed to be the length and dimension2 is assumed to be the width for a rectangle.
* **if dimension2 is not None::** This line checks if dimension2 is not None, meaning a second dimension was provided for the rectangle.
* **area = dimension1 \* dimension2**: If the shape is a rectangle and dimension2 is provided, this line calculates the area by multiplying dimension1 and dimension2 and stores it in the area variable.
* **return area:** This line returns the calculated area of the rectangle.
* **else::** This line starts a block of code that is executed if dimension2 is None for a rectangle.
* return "Please provide both length and width for a rectangle.": If the shape is a rectangle but dimension2 is not provided, this line returns an error message.
* **else::** This line starts a block of code that is executed if the shape argument is not 'circle' or 'rectangle'.
* **return "Invalid shape. Supported shapes are 'circle' and 'rectangle'.":** If the shape is invalid, this line returns an error message.

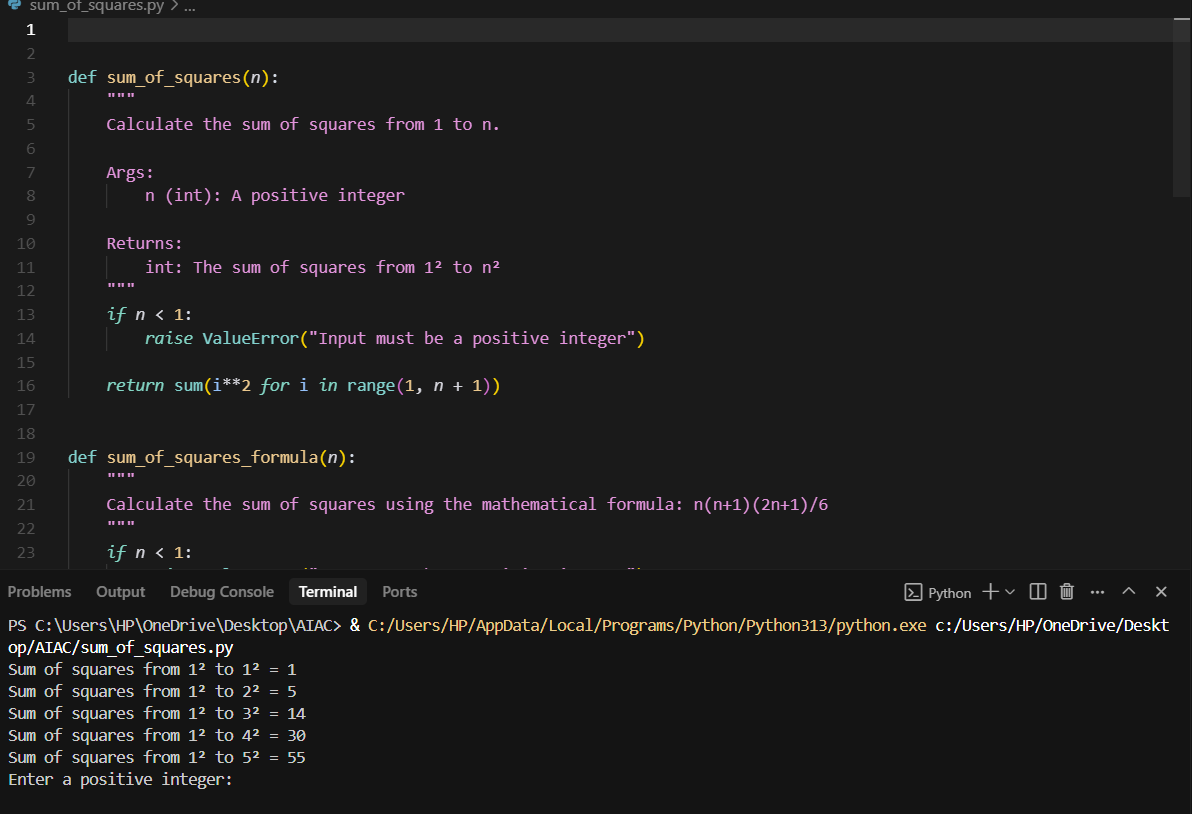
**TASK 4:**

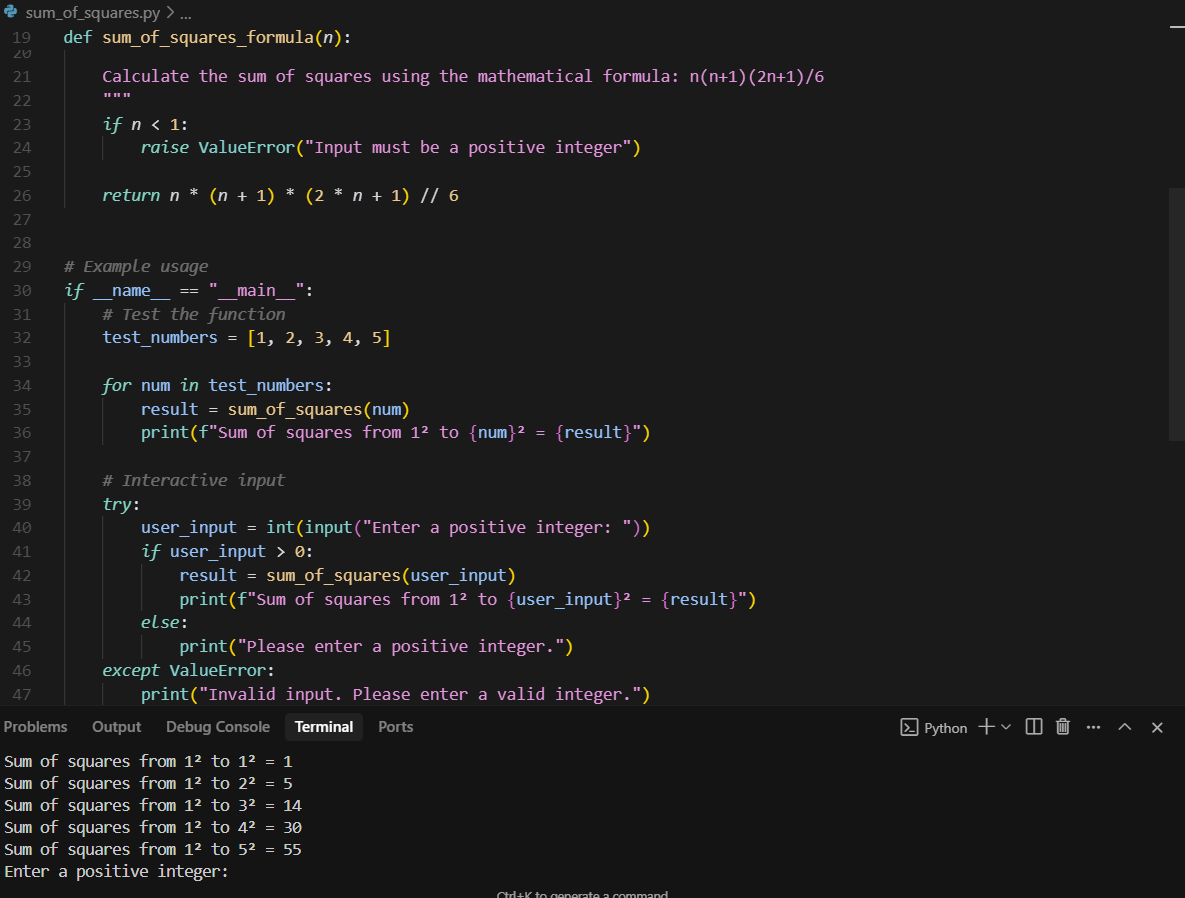
**Installation of cursor ai:**





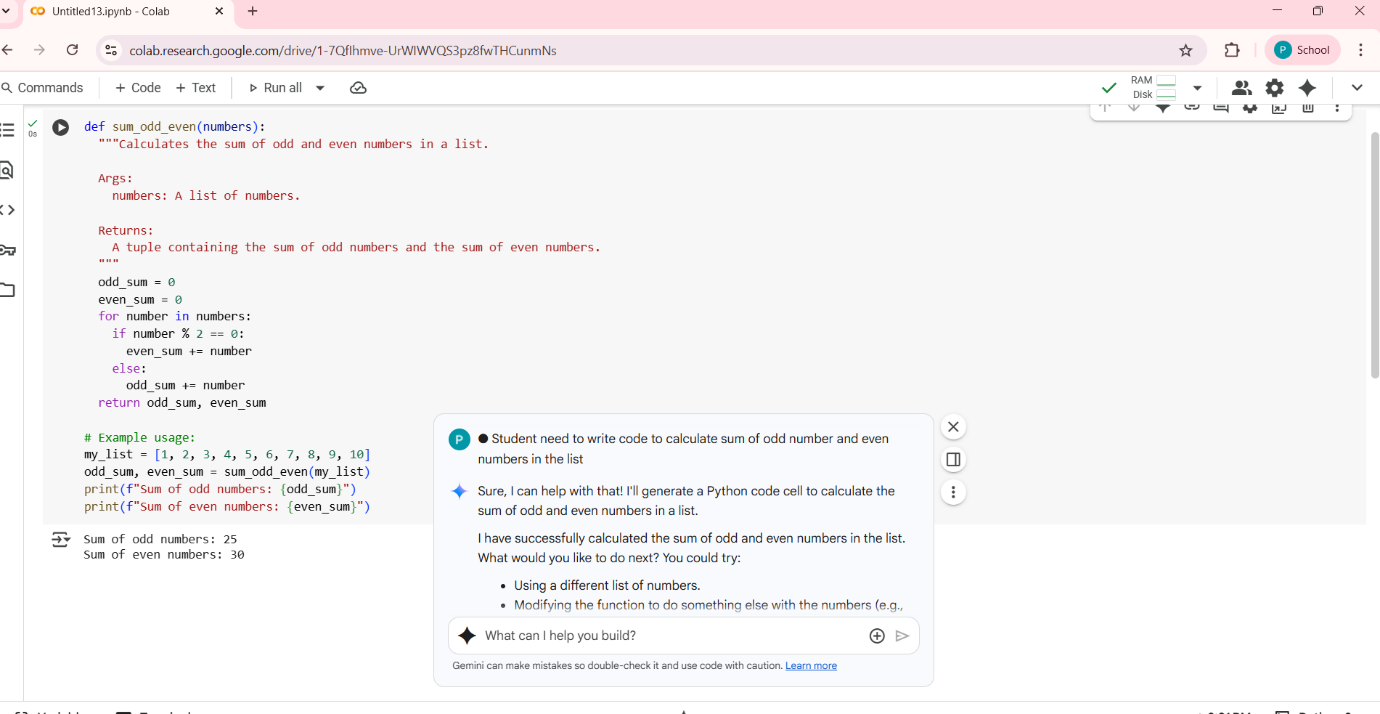
**Prompt:** Write a python function to calculate sum of squares of a number.

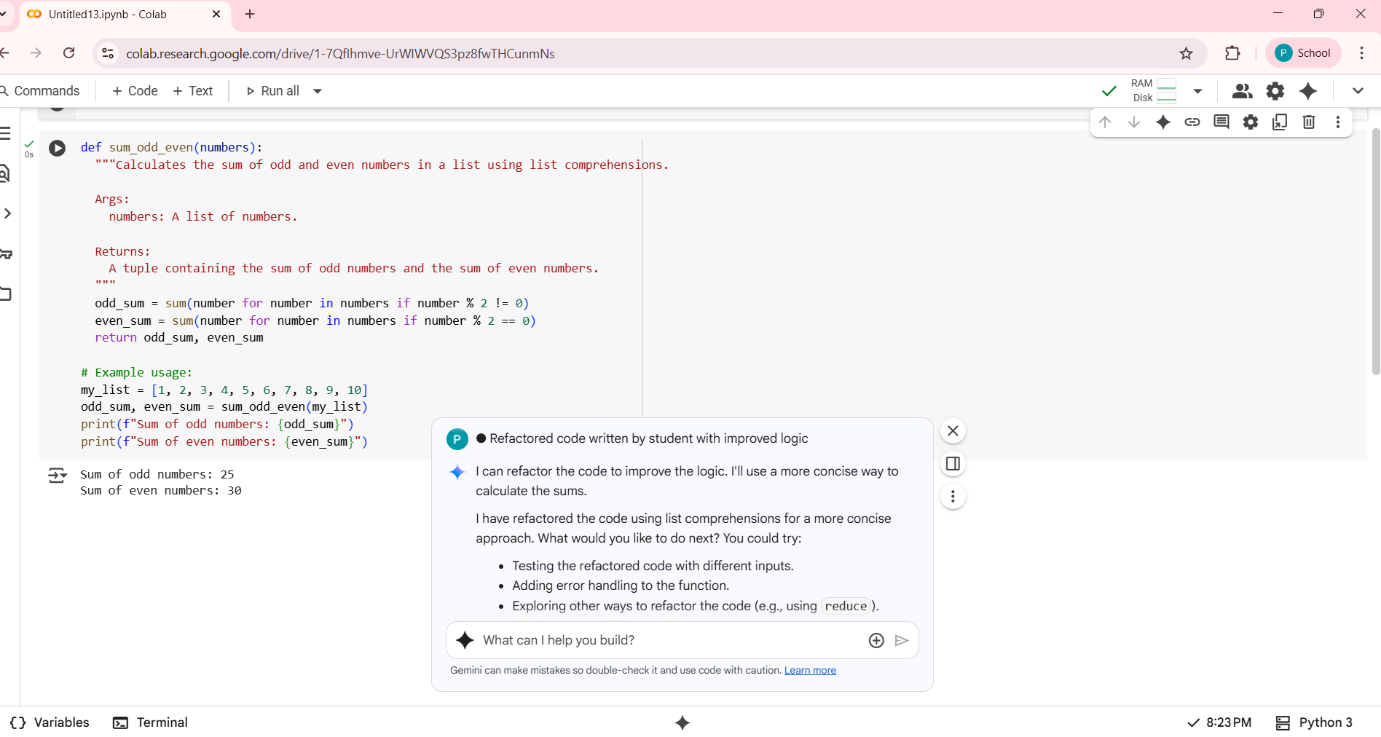




**Task-5:**

* **Prompt:** Student need to write code to calculate sum of odd number and even numbers in the list



* **Refactored code written by student with improved logic**
* 

**Comparison Table**

| **Aspect** | **Original Code** | **Refactored Code** | **Improvement** |
| --- | --- | --- | --- |
| **Approach** | **List comprehensions with sum()** | **Single loop with conditional accumulation** | **More memory-efficient, avoids creating temporary lists** |
| **Readability** | **Compact but slightly abstract for beginners** | **Clear and beginner-friendly logic** | **Easier to understand for new programmers** |
| **Performance** | **Creates two temporary lists before summing** | **Directly adds to sums during iteration** | **Faster and uses less memory** |
| **Flexibility** | **Harder to modify for additional logic** | **Easier to extend (e.g., count odd/even numbers)** | **More adaptable** |
| **Output** | **Correct** | **Correct** | **Same output** |

**Observation:**

* **Original code uses list comprehensions which creates temporary lists.**
* **Refactored code uses a loop for direct accumulation, saving memory.**
* **Both produce the same correct output.**
* **Refactored version is easier to read and modify. It’s more efficient and better for larger datasets or beginners.**