SR UNIVERSITY

AI ASSIST CODING

**LAB-2**: Exploring Additional AI Coding Tools – Gemini (Colab) and Cursor AI

Lab Objectives:

Name : K.SRIKAR

PinNo: 2503A51L03

Batchno:25BTCAICSB19

* To explore and evaluate the functionality of Google Gemini for AI-assisted coding within Google Colab.
* To understand and use Cursor AI for code generation, explanation, and refactoring.
* To compare outputs and usability between Gemini, GitHub Copilot, and Cursor AI.
* To perform code optimization and documentation using AI tools.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

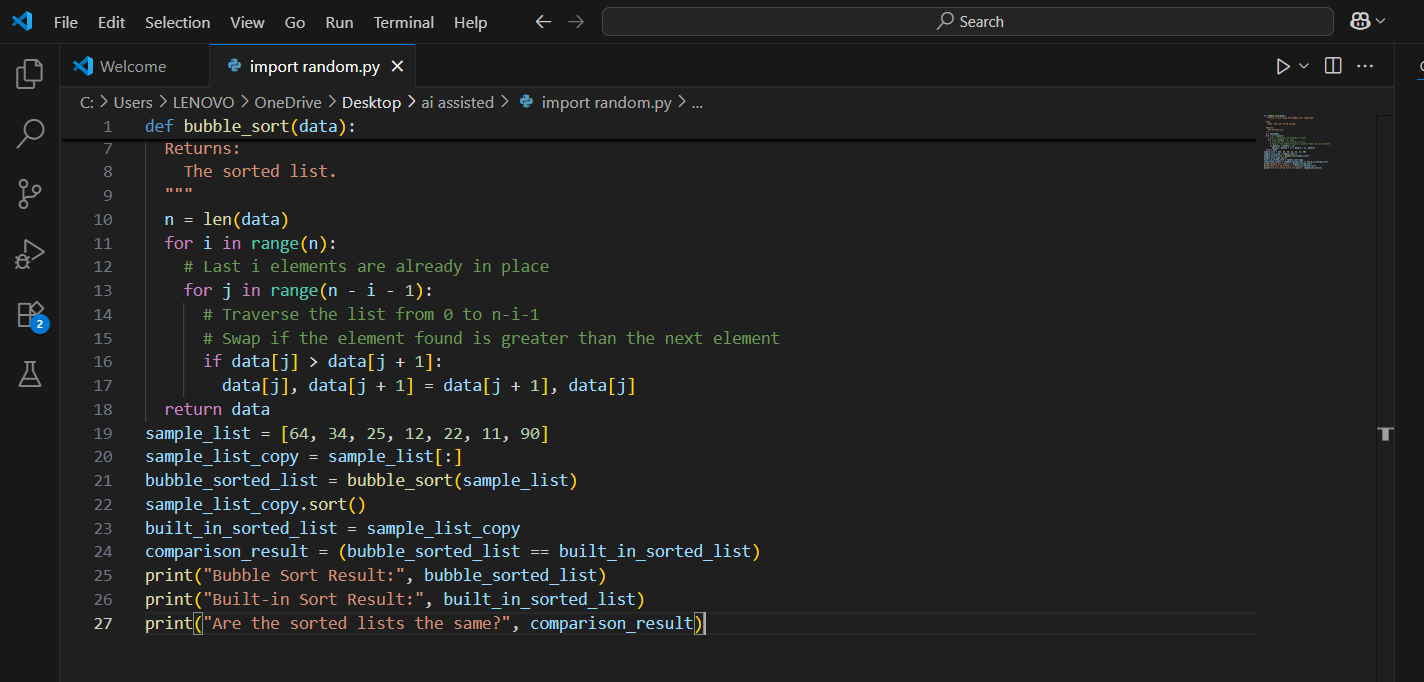
* Generate Python code using Google Gemini in Google Colab.
* Analyze the effectiveness of code explanations and suggestions by Gemini.
* Set up and use Cursor AI for AI-powered coding assistance.
* Evaluate and refactor code using Cursor AI features.
* Compare AI tool behavior and code quality across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

TASK #1:

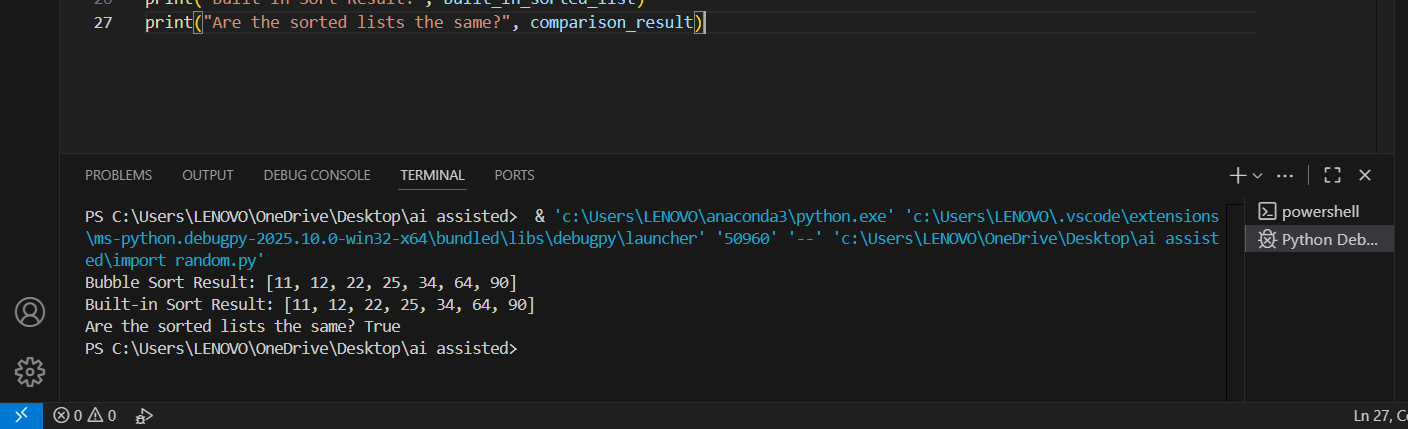
Prompt:

 Open Google Colab and use Google Gemini to generate Python code that performs sorting of a list using both the bubble sort algorithm and Python’s built-in sort() function. Compare the two implementations.

Code Generated:



Output After executing Code:



Your Observations:

1. Correct Implementation of Bubble Sort:

 The function bubble\_sort(arr) correctly implements the bubble sort algorithm using nested loops and value swapping.

1. Data Integrity Preserved:

 The function uses arr.copy() to avoid modifying the original list, which is a good practice.

1. Clear Comparison with Built-in Sort:

 The code includes a custom sort function and compares its result with Python’s built-in sorted() function — great for validating correctness.

1. Readable and Well-Structured:

 The code is neatly organized into:  Custom sort

 Built-in sort

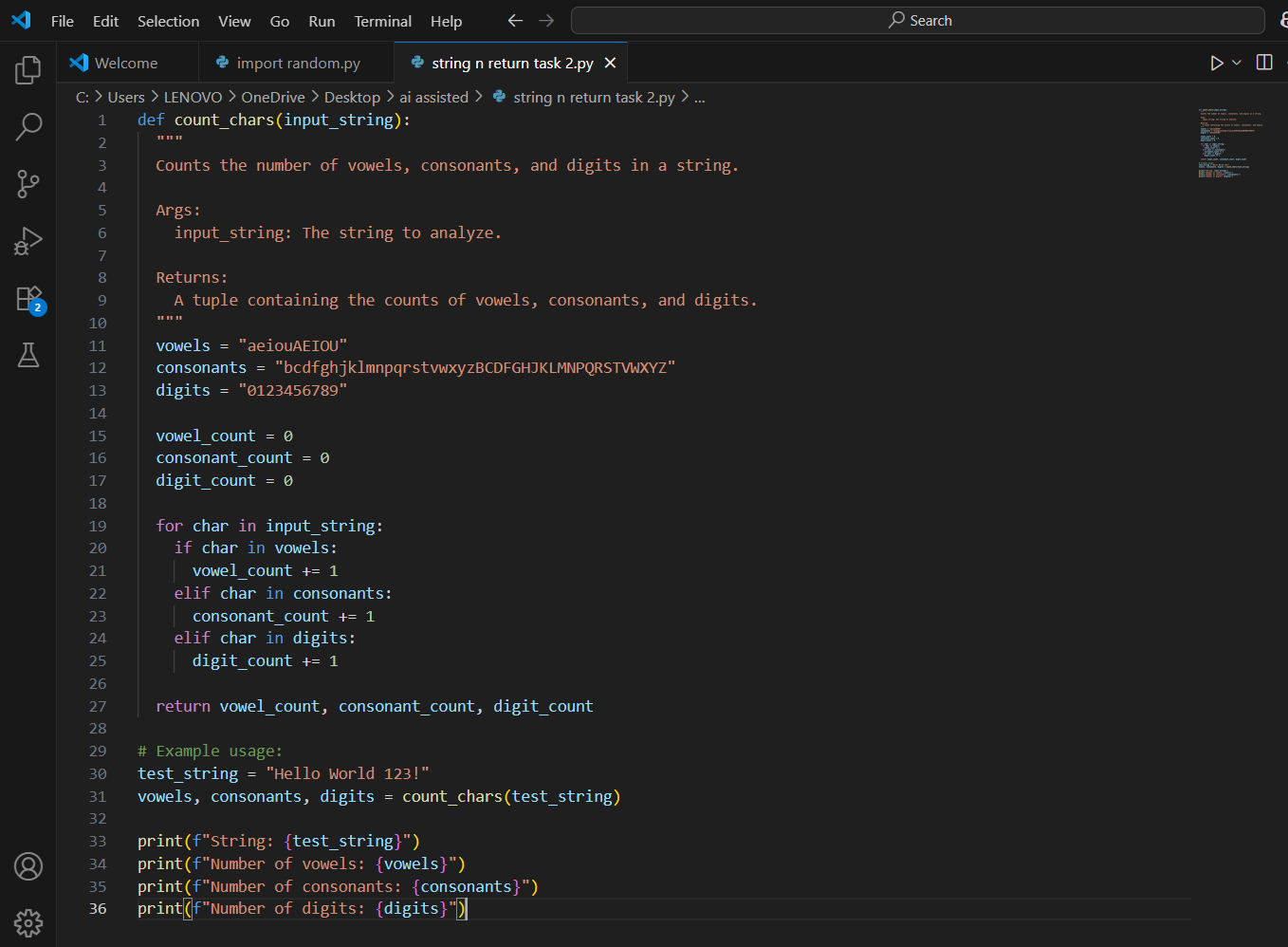
 Example list  Comparison

# TASK #2:

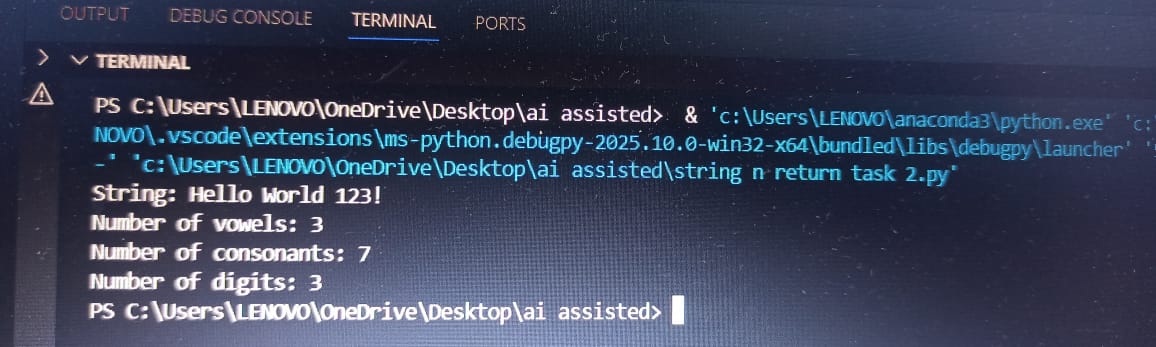
## Prompt:

 In Colab, use Google Gemini to generate a Python function that takes a string and returns: The number of vowels, The number of consonants, The number of digits in the string

Code Generated:



Output After executing Code:



Your Observations:

Function Deflnition

1. def count\_lines(fllename):

 A function count\_lines is defined that takes a filename (e.g., "example.txt") as input.

1. Try Block – Reading the File

try:

with open(filename, 'r') as file: return sum(1 for \_ in file)

 open(fllename, 'r'): Tries to open the file in read mode.



 sum(1 for \_ in flle): Counts each line using a generator expression.

 It iterates through each line and adds 1 per line.

 If the file is found, it returns the line count.

1. Exception Handling – File Not Found
   1. except FileNotFoundError:
   2. If the file doesn’t exist, this block is executed.
2. Created a Sample File

 with open(filename, 'w') as file: file.write("Hello\n")

file.write("This is a test file\n")

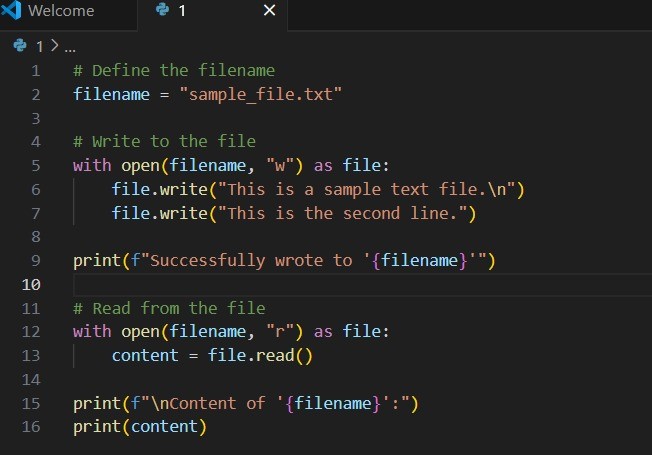
 The file is opened in write mode ('w') which creates a new flle.

# TASK #3:

## Prompt:

* Install and set up Cursor AI. Use it to generate a Python program that performs file handling:
  1. Create a text file
  2. Write sample text
  3. Read and display the content.

Code Generated:



Output After executing Code:



Your Observations:

1**.filename = "sample\_file.txt"**: Sets the name of the file.

1. **with open(filename, 'w') as file:**:

 Opens (or creates) the file in **write mode** ('w').  If the file already exists, it will be **overwritten**.

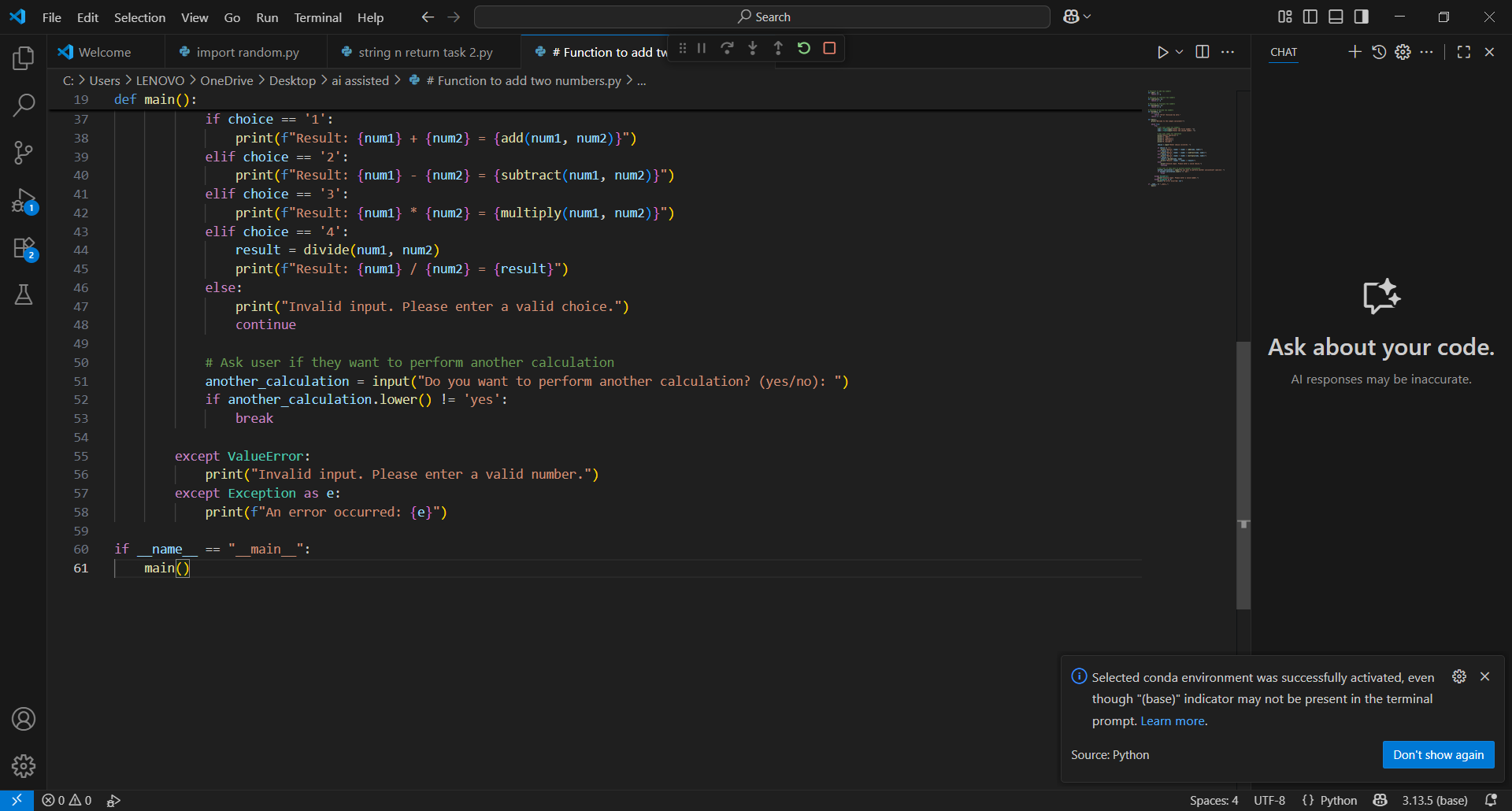
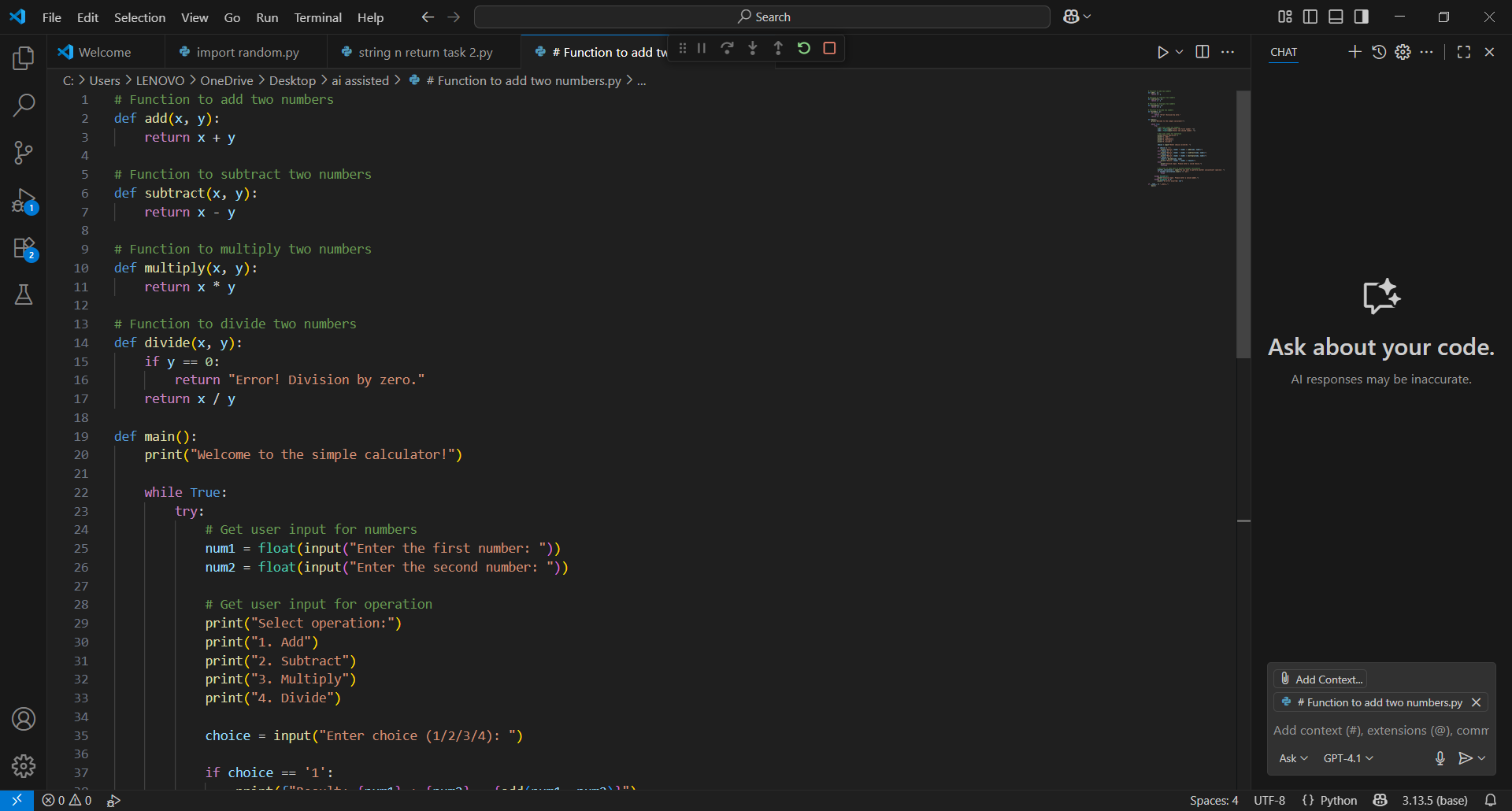
1. **file.write(...)**: Writes 3 lines of text into the file, each ending with a newline (\n).
2. **print(...)**: Confirms that the file was created and written successfully.

# TASK #4:

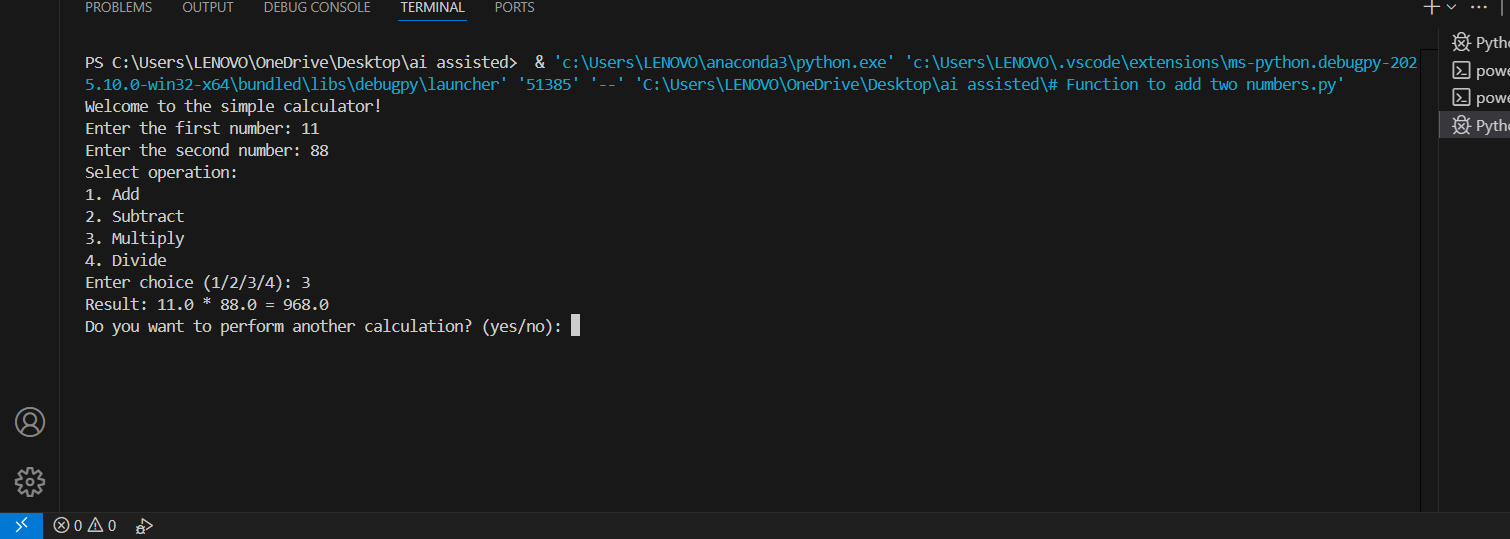
## Prompt:

 Ask Google Gemini to generate a Python program that implements a simple calculator using functions (add, subtract, multiply, divide). Then, ask Gemini to explain how the code works.

Code Generated:



Output After executing Code:



Your Observations:

1. Function Deflnitions

 These functions perform basic arithmetic:

->def add(x, y):

return x + y

-->def subtract(x, y):

return x - y

-->def multiply(x, y):

return x \* y

-->def divide(x, y):

if y == 0:

-->return "Error: Division by zero"

return x / y

 Each function takes two numbers x and y, and returns the result.  The divide() function includes error handling for division by zero.

1. The main() Function

 This is where user interaction happens:

def main():

print("Simple Calculator") print("Select operation:") print("1. Add")

print("2. Subtract") print("3. Multiply")

print("4. Divide")

 The program prints a menu of operations for the user.

choice = input("Enter choice (1/2/3/4): ")

 The user selects an operation (e.g., 3 for multiply).

num1 = float(input("Enter flrst number: "))

num2 = float(input("Enter second number: "))

 The user inputs two numbers, which are converted to floats for accurate calculations.

1. Conditional Execution Based on Choice

if choice == '1':

print("Result:", add(num1, num2))

elif choice == '2':

print("Result:", subtract(num1, num2))

elif choice == '3':

print("Result:", multiply(num1, num2)) elif choice == '4':

print("Result:", divide(num1, num2))

else:

print("Invalid input")

 Depending on the user's choice, the corresponding function is called.  If the choice doesn't match 1–4, it prints "Invalid input.

1. Script Entry Point

if name == "main":

main()

 This ensures that the main() function runs only when the script is executed directly, not when imported.

Terminal Output : Simple Calculator Select operation:

* 1. Add
  2. Subtract
  3. Multiply
  4. Divide

Enter choice (1/2/3/4): 3 enter first number: 11

Enter second number: 88

Result: 968.0

* The user selected 3(multiply).
*  Entered 11 and 88.

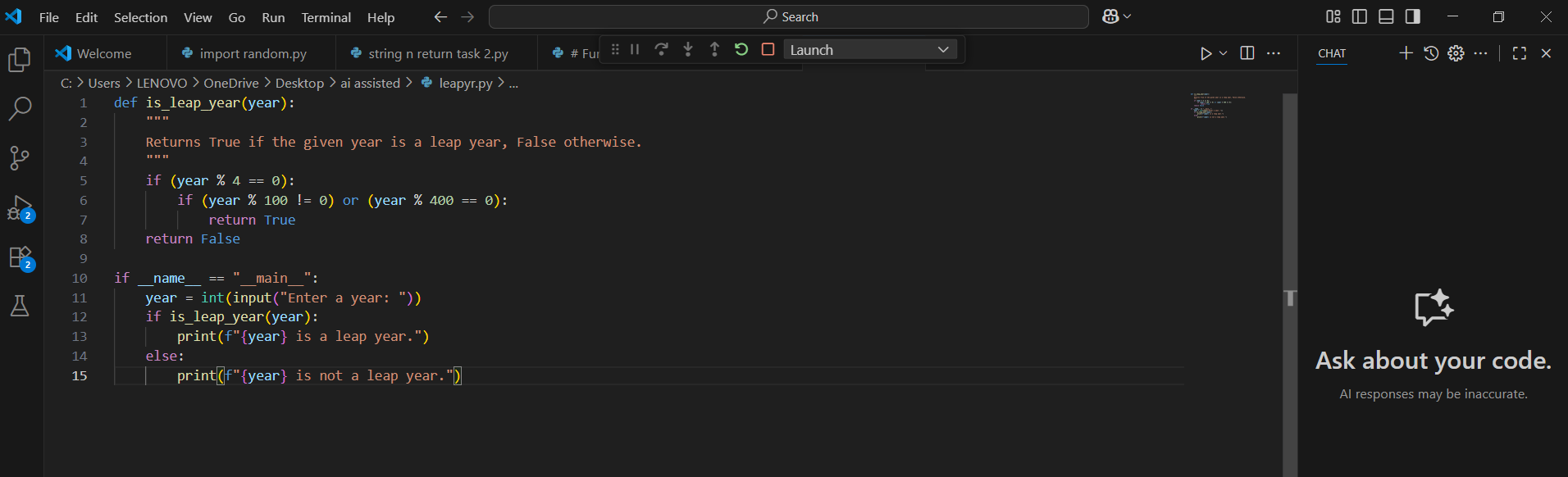
 Got the correct result: 968.0.

## #task5 :

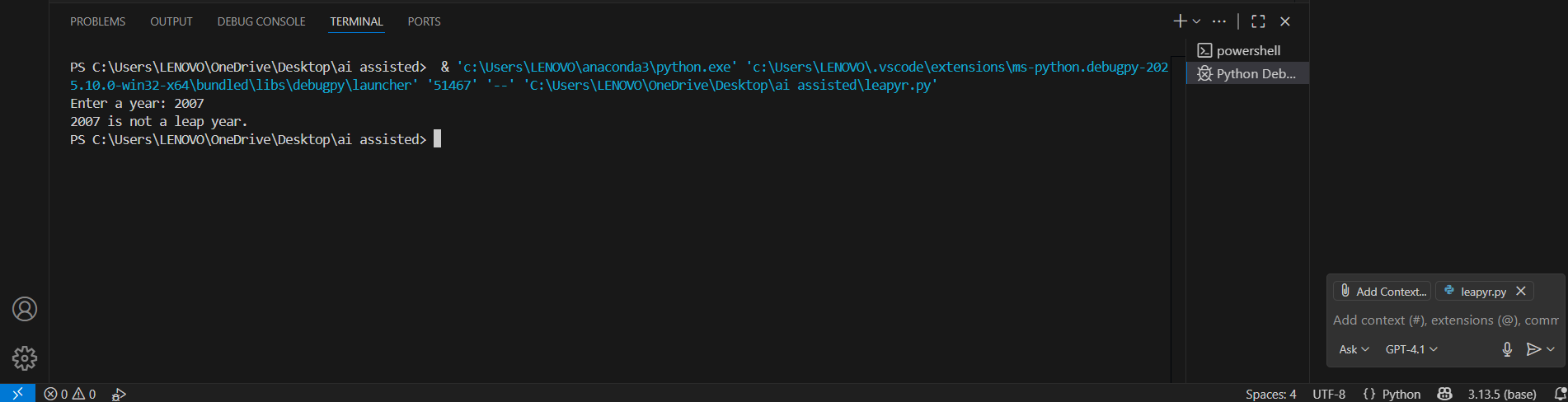
## Prompt:

 Use Cursor AI to create a Python program that checks if a given year is a leap year or not. Try different prompt styles and see how Cursor modifies its code suggestions.

Code Generated:



Output After executing Code:



Your Observations:

1. Function Deflnition

 def is\_leap\_year(year: int) -> bool:

"""

Returns True if the given year is a leap year, False otherwise. """

return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)

 Purpose: Checks leap year condition.  Type hinting:

 year: int → function expects an integer.

 -> bool → function returns a boolean (True or False).  Logic: Implements the leap year condition in one line.

1. Main Execution Block

if name == " main ":

year = int(input("Enter a year: ")) if is\_leap\_year(year):

print(“{year} is a leap year.”)

else:

print(f"{year} is not a leap year.")

 if name == " main ": Ensures this code only runs when the script is executed directly.  input(...): Takes user input and converts it to an integer.

 Conditional check: Calls is\_leap\_year(year) to check and print the appropriate message.

Output from Terminal:

Enter a year: 2007

2007 is not a leap year.

 the output is correct.