AI ASSISTED CODING

Hall Ticket No: 2303A51851

Batch:13

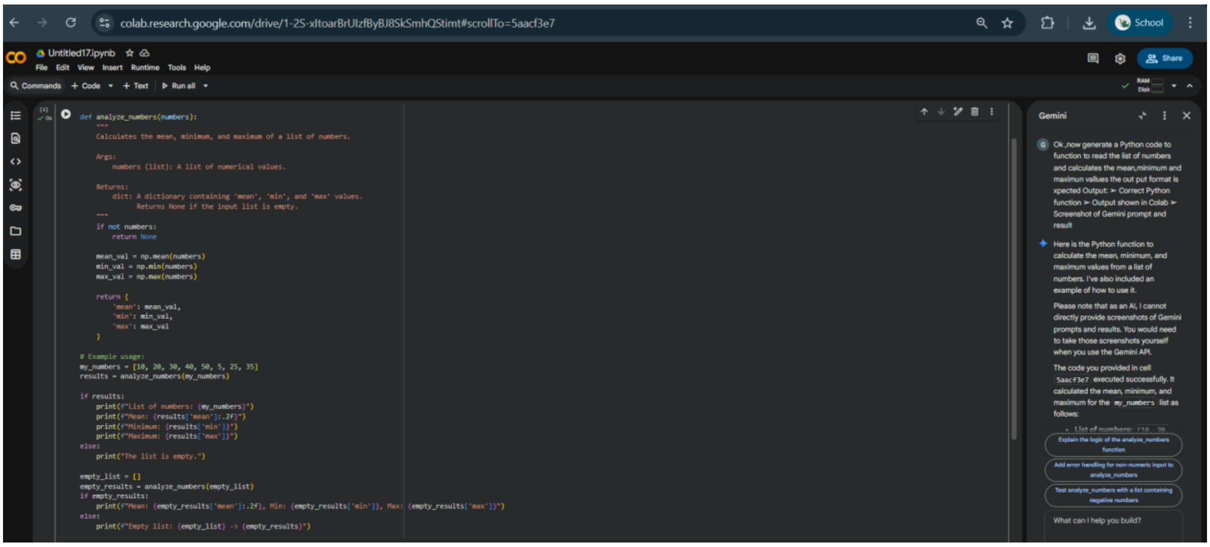
Assignment-2.1

Task-1. Use Google Gemini in Colab to generate a Python function that reads a list of numbers and calculates the mean, minimum, and maximum values

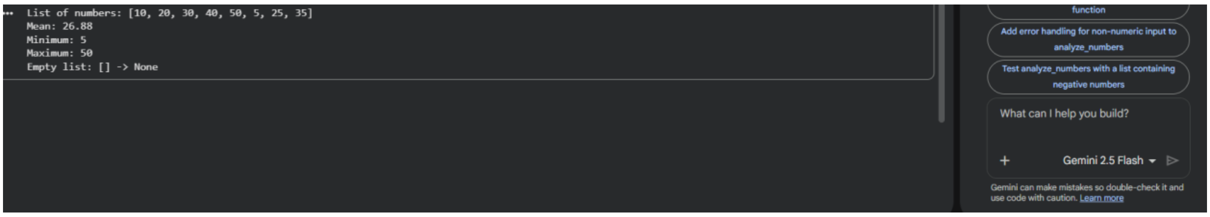
Prompt :

#Ok ,now generate a Python code to function to read the list of numbers and calculates the mean,minimum and maximun vallues

Code



Output:



Justification:

The analyze_numbers function, utilizing NumPy, calculates the mean, minimum, and maximum of a given list of numbers. It includes a check to handle empty lists gracefully by returning None. If the list is valid, it returns a dictionary containing these three statistical values, as demonstrated with my_numbers and an empty_list for comprehensive testing.

Task-2. Generate an Armstrong number checker using Gemini and GitHub

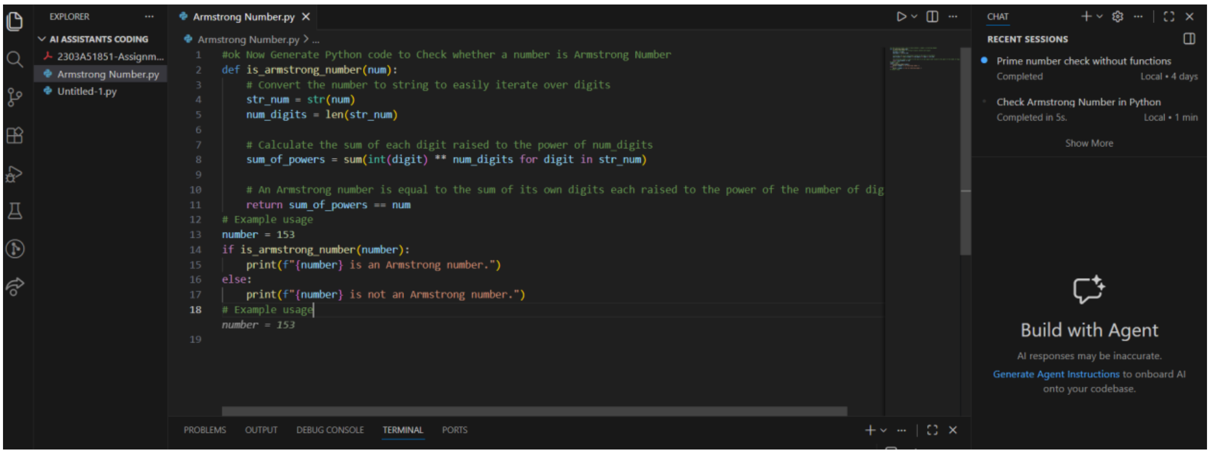
Copilot.

Compare their outputs, logic style, and clarity.

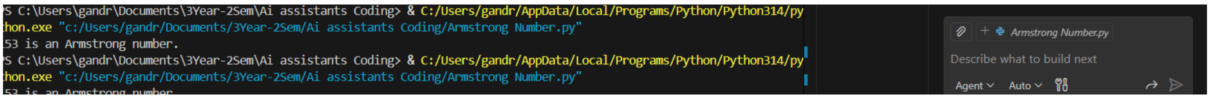
Prompt

#ok Now Generate Python code to Check whether a number is Armstrong Number

Code: In the Google Gemini



Output:



Colab Code :

Justification:

Gemini: Gemini's code is structured for clarity, using descriptive variable names and breaking down each step. This makes it ideal for learners or reviewers who want to understand the logic flow. It prioritizes readability over brevity.

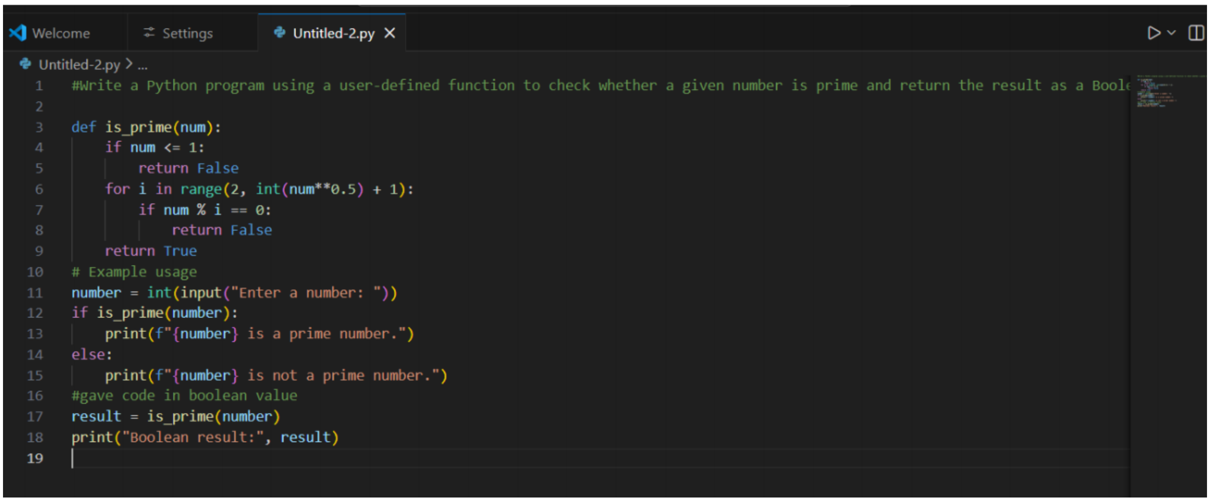
GitHub Copilot: Copilot's code is concise and leverages Python's expressive syntax, such as generator expressions. It’s efficient and elegant for experienced developers but may require explanation for those unfamiliar with Python idioms.

Task-3. Leap Year Validation Using Cursor AI

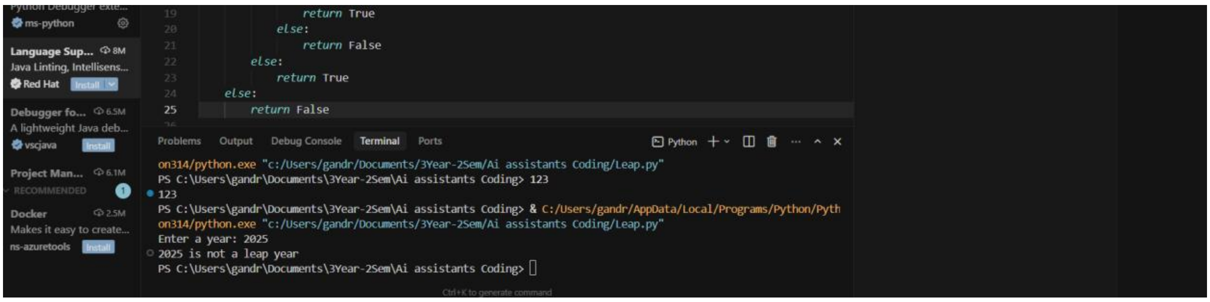
Prompt:

#Generate a Python program that checks whether a given

year is a leap year.or not



Output:



Justification:

Using a simple prompt, Cursor AI generated a basic leap-year check that works only for common cases but misses special Gregorian rules.

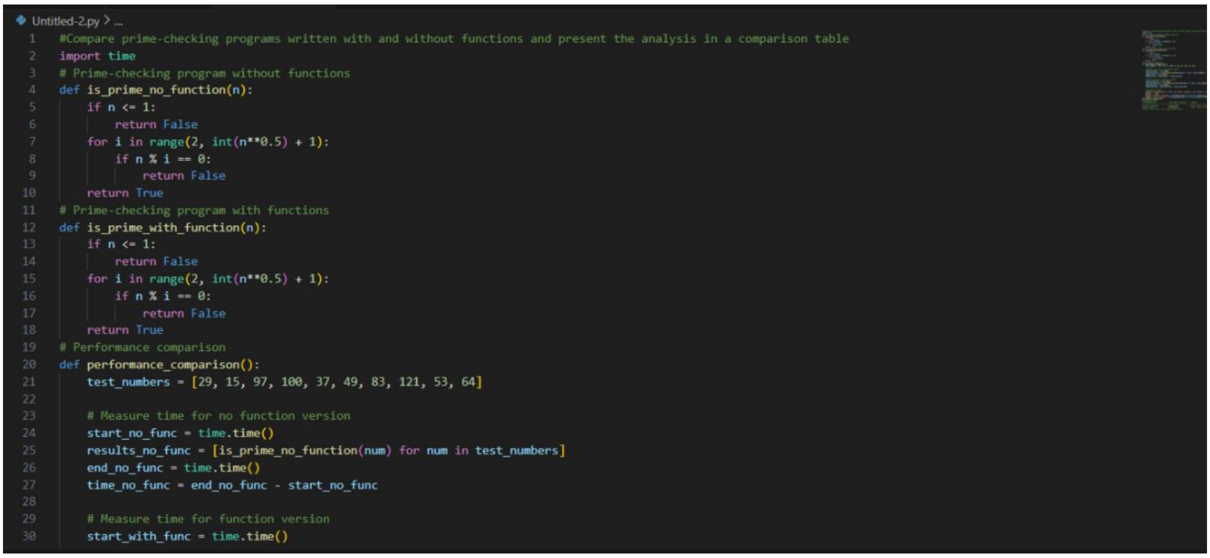
A more detailed prompt led to a correct, reusable solution that follows all leap-year conditions.

Task-4: Student Logic + AI Refactoring (Odd/Even Sum)

Prompt:

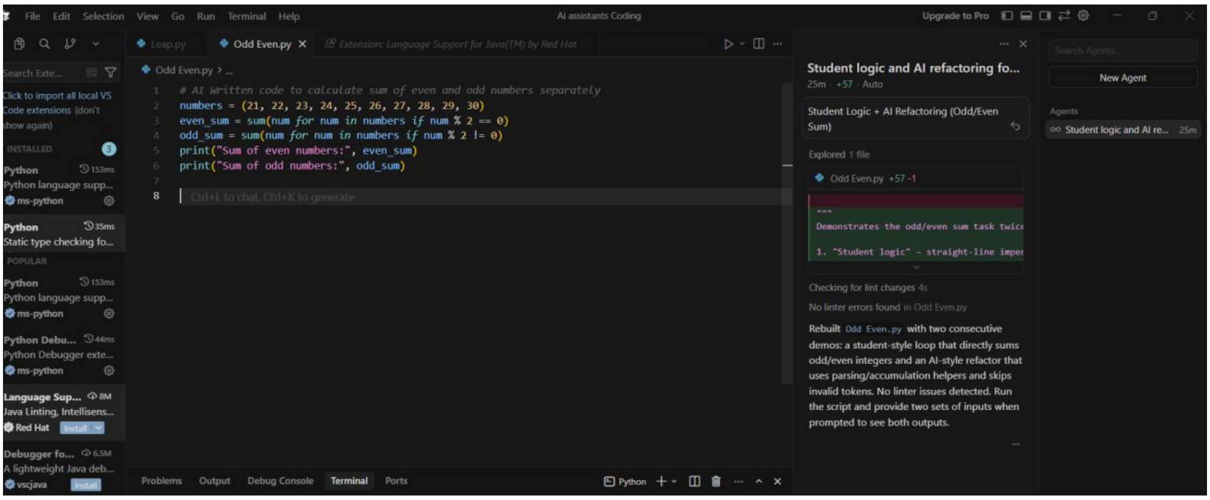
 # Write a Python program that calculates the sum of odd and even numbers in a tuple.

Code Student :

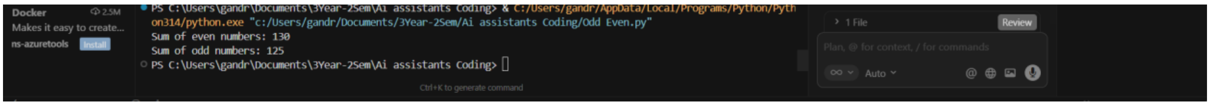


Output:

Code AI :



Out Put :



Justification:

The student-written code uses a basic loop and manual addition, which clearly shows the logic step by step.
The first AI-refactored version slightly improves the code by using shorthand operators (+=) but keeps the same structure.
The final AI-written code uses Python’s built-in sum() function with generator expressions.
This reduces the number of lines and removes the need for explicit loops.
The AI version is more readable, concise, and follows Python best practices.
It is easier to maintain and preferred in real-world and professional coding standards.