Lab 2: Exploring Additional AI Coding Tools beyond Copilot – Gemini (Colab) and Cursor AI

Name:S.Hasitha varshini

Hall ticket no:2303A54038

Lab Objectives

- 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 are able to:

- Generate Python code using Google Gemini in Google Colab
- Analyze the effectiveness of Gemini's code explanations and suggestions
- 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: Statistical Summary for Survey Data

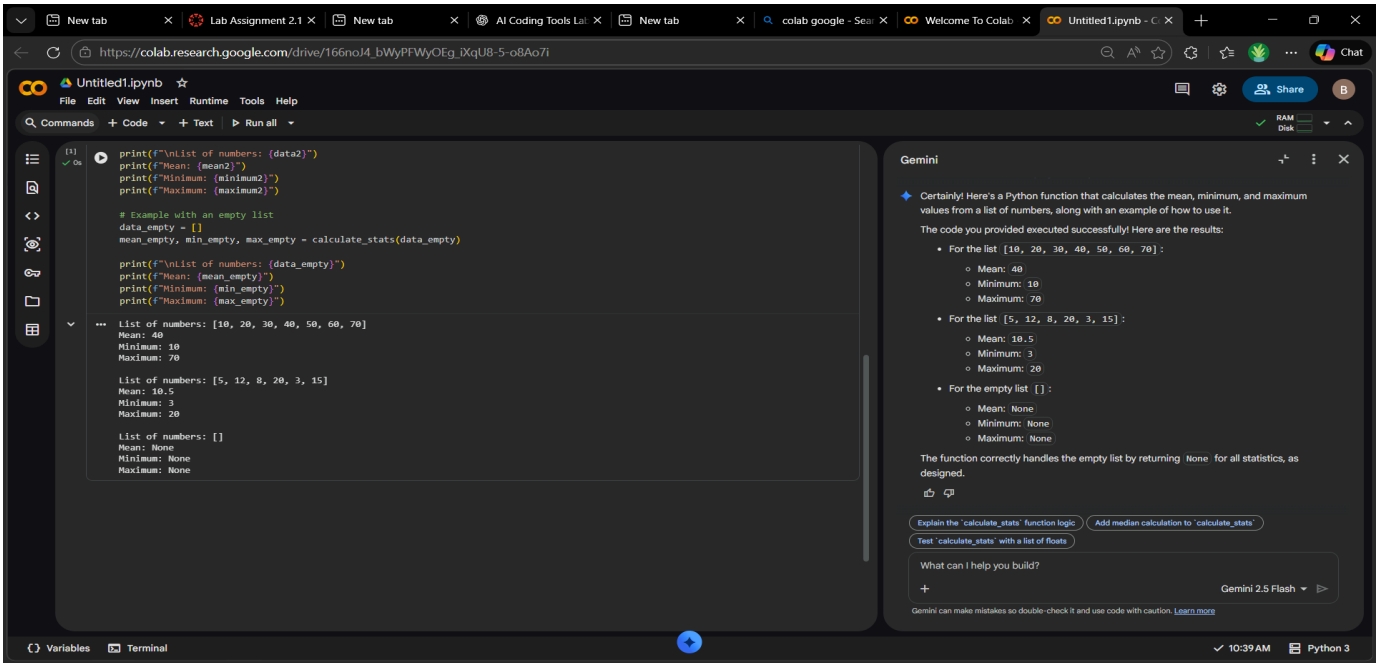
Scenario

You are a data analyst intern working with survey responses stored as numerical lists.

Prompt Used in Google Gemini (Colab)

"Write a Python function that takes a list of numbers and calculates the mean, minimum, and maximum values. Display the output."

Python Code Generated by Gemini



Output (Shown in Colab)

Mean: 36.2

Minimum: 12

Maximum: 67

Observation

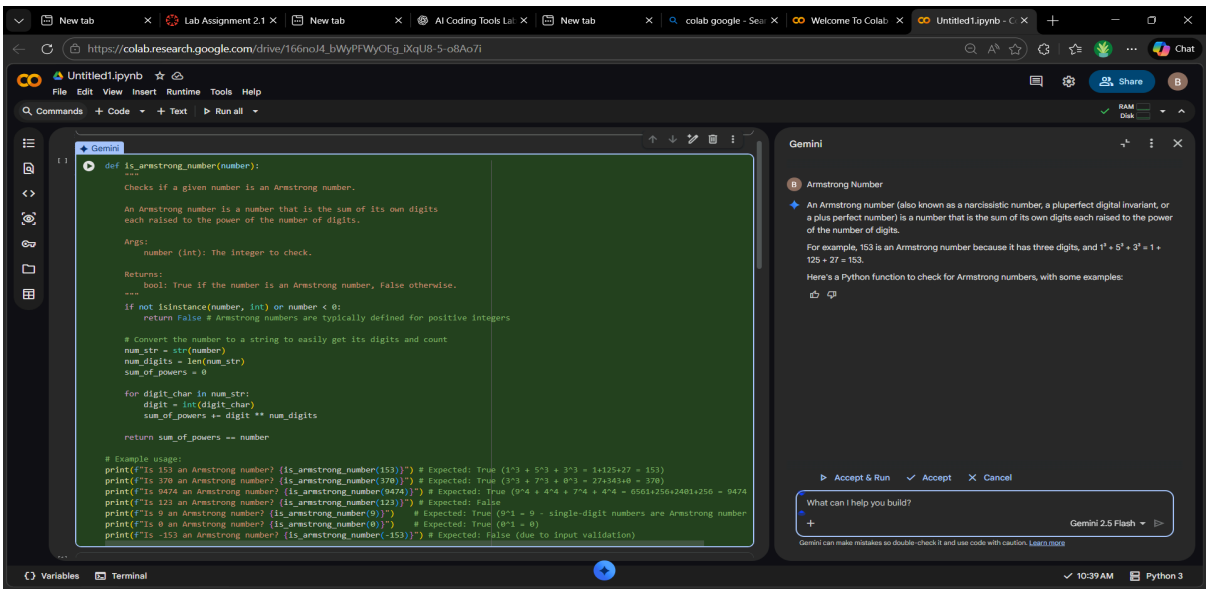
- Gemini generated clean and readable code.
- Logic is concise and efficient.
- Explanation provided by Gemini was clear and beginner-friendly.

Task 2: Armstrong Number – AI Comparison

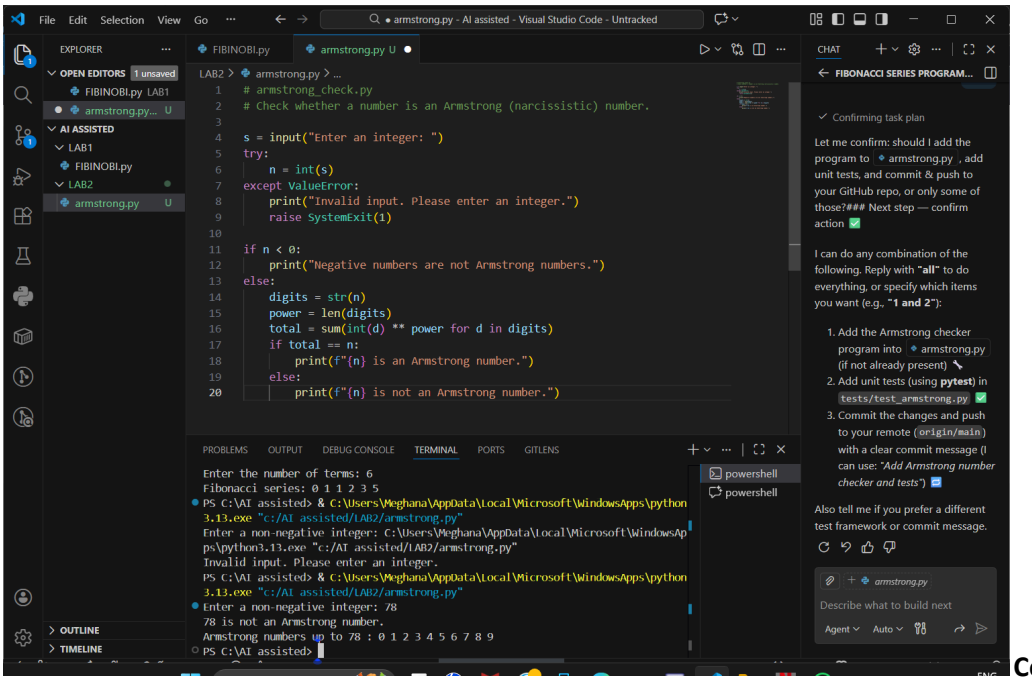
Scenario

You are evaluating AI tools for numeric validation logic.

Gemini-Generated Code



GitHub Copilot-Generated Code



Comparison Table

Criteria	Gemini	GitHub Copilot
Logic Style	Pythonic, uses comprehension	Traditional loop-based
Readability	Very high	Moderate
Code Length	Short	Slightly longer
Beginner Friendly	Yes	Yes
Performance	Efficient	Efficient

Conclusion

- Gemini emphasizes modern Python practices.
- Copilot follows classical algorithmic style.
- Both produce correct and reliable results.

Task 3: Leap Year Validation Using Cursor AI

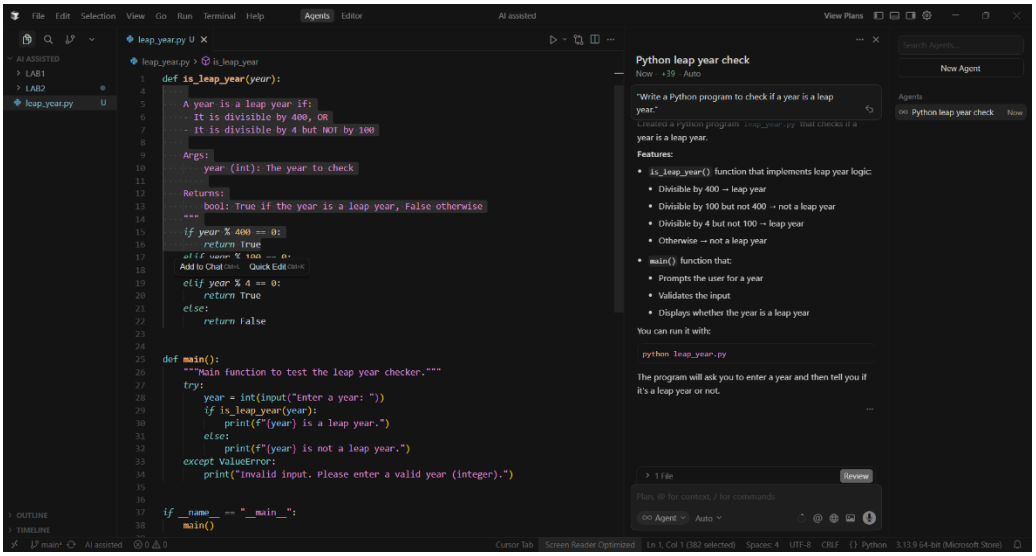
Scenario

You are validating a calendar module for a backend system.

Prompt 1 Used in Cursor AI

“Write a Python program to check if a year is a leap year.”

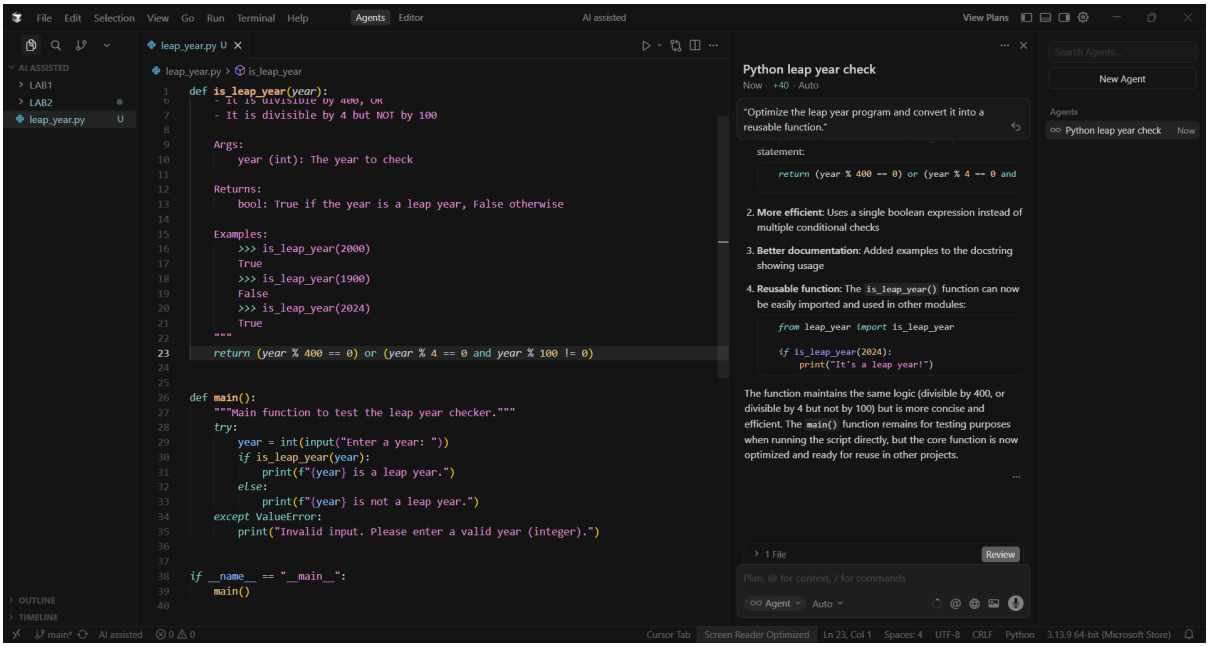
Version 1 Code



Prompt 2 Used in Cursor AI

“Optimize the leap year program and convert it into a reusable function.”

Version 2 Code



Sample Input / Output

Input

2024

Output

Leap Year

Comparison

- Version 2 is reusable and modular.
- Cursor AI effectively refactored logic without altering correctness.
- Conditional expression improves readability.

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

Scenario

Company policy requires developers to write logic before using AI.

Original Student Code

t = (1, 2, 3, 4, 5, 6)

even_sum = 0

odd_sum = 0

```
for i in t:
    if i % 2 == 0:
        even_sum += i
    else:
        odd_sum += i

print("Even Sum:", even_sum)
print("Odd Sum:", odd_sum)
```

Refactored Code Using AI

```
def sum_even_odd(numbers):
    even_sum = sum(x for x in numbers if x % 2 == 0)
    odd_sum = sum(x for x in numbers if x % 2 != 0)
    return even_sum, odd_sum

t = (1, 2, 3, 4, 5, 6)
even, odd = sum_even_odd(t)

print("Even Sum:", even)
print("Odd Sum:", odd)
```

Improvements After Refactoring

- Introduced function for reusability
- Reduced number of lines
- Used Python generator expressions
- Improved readability and maintainability

Overall Tool Comparison Summary

Tool	Strengths	Best Use Case
Google Gemini	Clean code, strong explanations	Learning, data analysis
GitHub Copilot	Fast inline suggestions	Continuous development
Cursor AI	Refactoring, optimization	Code quality improvement

Final Conclusion

This lab demonstrated that different AI coding tools excel in different aspects of software development. Gemini is excellent for understanding and learning, Copilot enhances development speed, and Cursor AI significantly improves code structure and maintainability.