Assignment 1

AI Assisted Coding

Name:-B.Sanjana

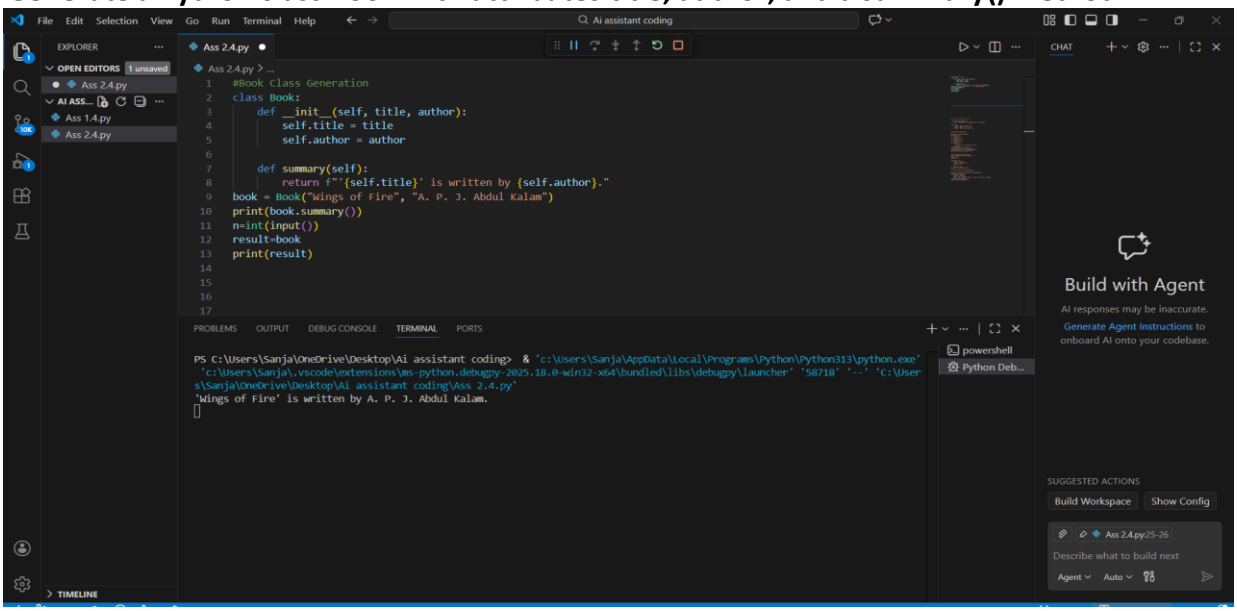
HTNO: 2303A52306

Task 1:

Prompt:

#Book Class Generation

Generate a Python class Book with attributes title, author, and a summary() method.



Observation:

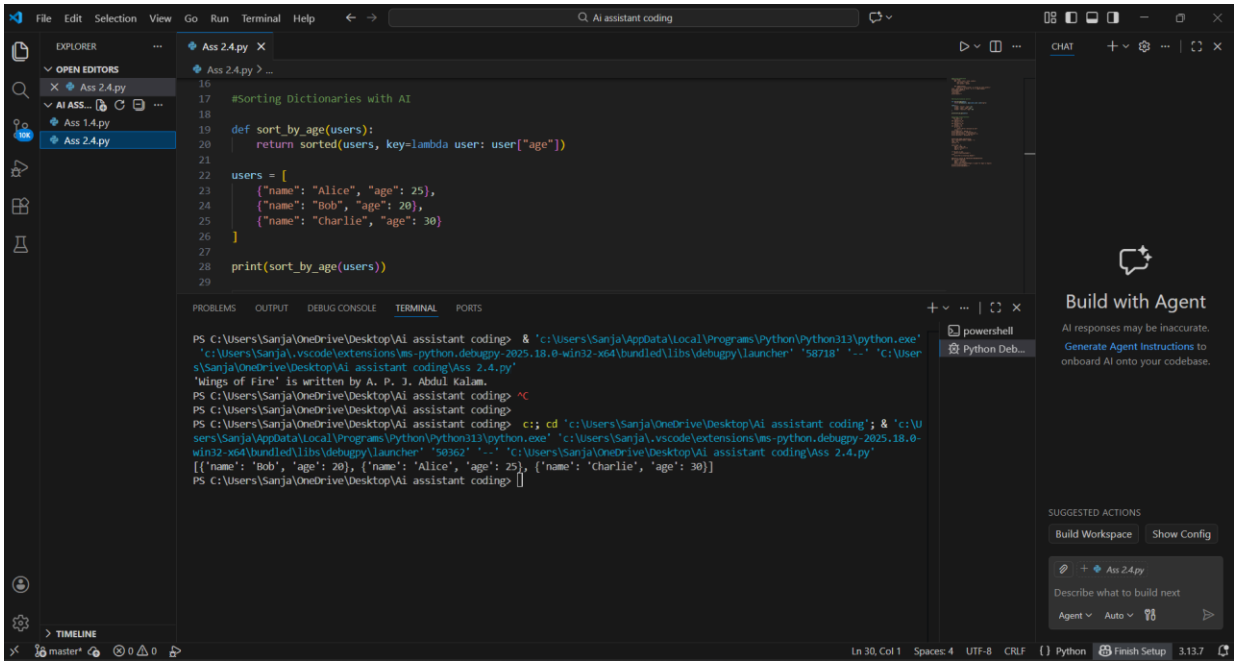
- **The generated Book class follows proper object-oriented programming principles.**
- **The constructor (__init__) is correctly used to initialize the title and author attributes.**
- **The summary() method provides a meaningful and readable description of the book object.**
- **The code is simple, clean, and easy to understand, making it suitable for beginners.**
- **Use of formatted strings (f-strings) improves output clarity and readability.**
- **The class design supports reusability and scalability in a library management system.**
- **The code lacks input validation, which could be improved for real-world applications.**

Task 2:

#Sorting Dictionaries with AI

Prompt:

Generate Python code to sort a list of dictionaries by age.



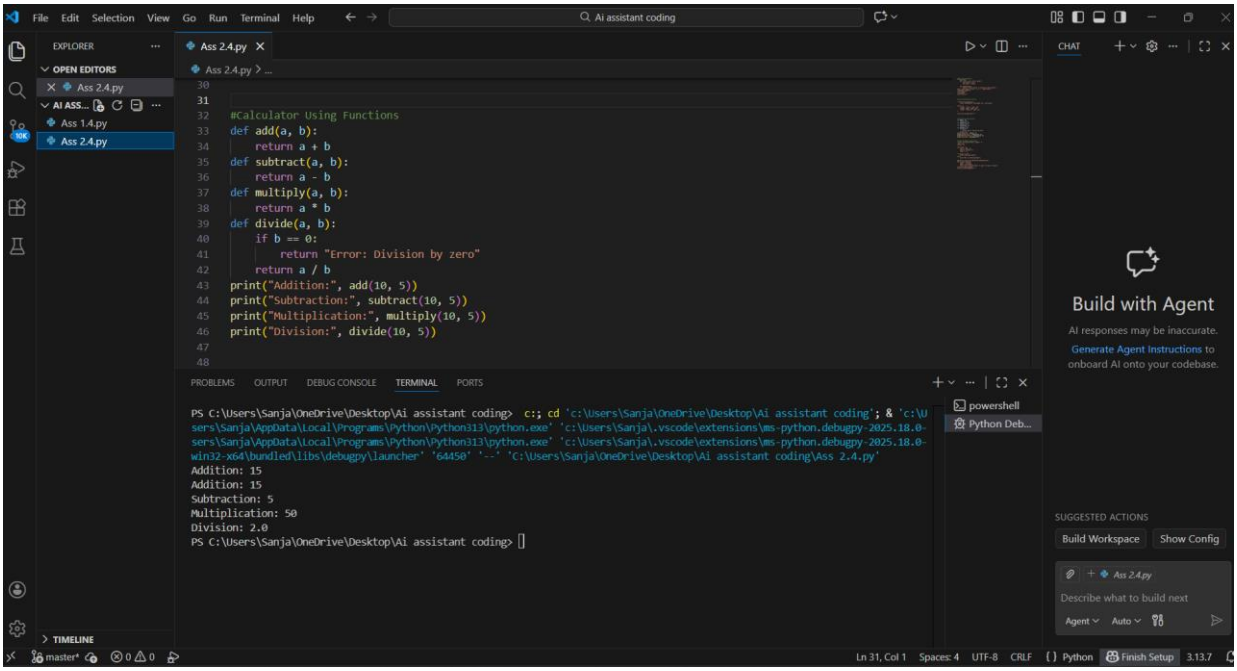
Observation:

- Both Gemini AI and Cursor AI correctly use Python's built-in sorted() function.
- Sorting is performed using a **lambda function** as the key, ensuring concise logic.
- The time complexity for both implementations is **O(n log n)**, which is efficient.
- Gemini AI's solution is shorter and suitable for quick scripting tasks.
- Cursor AI's solution improves **code clarity and reusability** by using a function.
- Cursor AI output is more maintainable for large or scalable applications.
- Both approaches preserve the original data structure while returning sorted results.
- Overall performance is similar, but Cursor AI provides better **readability and structure**.

Task 3: Calculator Using Functions

Prompt:

#Generate a basic calculator using functions and explain how it works.



Observation:

- **The calculator is implemented using separate functions for each arithmetic operation.**
- **Each function performs a single, well-defined task, improving clarity.**
- **The divide() function includes error handling to avoid division by zero.**
- **This modular design makes the program easy to understand, test, and maintain.**
- **Functions can be reused in other programs without modification.**
- **Overall, the calculator follows good programming practices and clean structure**

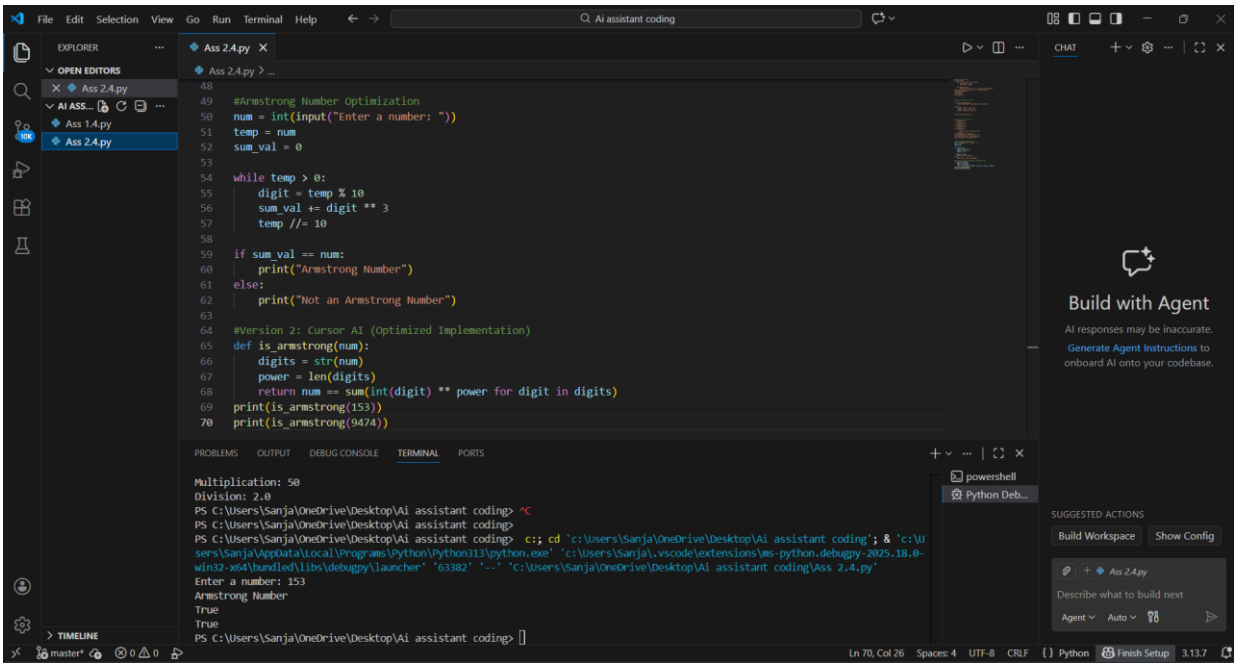
Task 4: Armstrong Number Optimization

Scenario

An existing solution for checking Armstrong numbers is inefficient and limited.

Prompt Used (Gemini AI)

Generate a Python program to check whether a number is an Armstrong number.



Observation:

1. **The optimized version supports Armstrong numbers of any length, not just 3-digit numbers.**
2. **It replaces manual loops with generator expressions, making the code concise.**
3. **Readability is improved through meaningful function naming.**
4. **Temporary variables are reduced, lowering the chance of logical errors.**
5. **The optimized solution is more scalable and reusable.**
6. **Code execution is faster and easier to maintain.**