

## AI Assistant Lab

### Assignment 2.4

Name: ch srinidh rao

2303A52398

#### Task1: Book Class Generation

**Prompt:** *Generate a Python class named Book for a simple library management system.*

```
lab2.4.py > ...
1 class Book:
2     def __init__(self, title, author):
3         self.title = title
4         self.author = author
5
6     def summary(self):
7         return f"Title: {self.title}, Author: {self.author}"
8
```

[Done] exited with code=0 in 0.074 seconds

**Observation:** For this task, a prompt was given to Cursor AI to generate a Python Book class for a simple library management module with attributes title and author and a method. Cursor AI produced a clean and well-structured class using proper object-oriented principles. The constructor correctly initializes the attributes, and the method returns a clear, formatted description of the book. The code follows standard Python naming conventions, making it easy to read and understand. Overall, the generated code is minimal yet complete, meets all the requirements of the task, and demonstrates good code quality in terms of readability, correctness, and reusability.

## Task 2: Sorting Dictionaries with AI

**Prompt :** “Generate Python code to sort a list of dictionaries representing users by the key age in ascending order. Ensure the code is simple and readable.”

```
8
9     ## Task 2
10
11 users = [
12     {"name": "Alice", "age": 25},
13     {"name": "Bob", "age": 20},
14     {"name": "Charlie", "age": 30}
15 ]
16 sorted_users = sorted(users, key=lambda x: x["age"])
17 print(sorted_users)
18 users = [
19     {"name": "Alice", "age": 25},
20     {"name": "Bob", "age": 20},
21     {"name": "Charlie", "age": 30}
22 ]
23 users.sort(key=lambda user: user["age"])
24 print(users)
25
26
```

```
[Done] exited with code=0 in 0.074 seconds
```

Observation: Both Gemini AI and Cursor AI generated correct and efficient Python code for sorting a list of dictionaries by age. Gemini AI's solution is more concise and easier to understand, making it ideal for beginners and quick implementation, as it uses the built-in sorted function with a lambda expression. Cursor AI's solution, while slightly more verbose, improves readability by using a named function and performs in-place sorting using list.sort, which is marginally more memory-efficient. Overall, Gemini AI emphasizes simplicity and clarity, whereas Cursor AI focuses more on structure and performance, making both suitable depending on the use case.

## Task3: Calculator Using Functions

**Prompt :** “Generate a basic calculator program in Python using functions for addition, subtraction, multiplication, and division. Also explain how the calculator works in simple terms.”

```

26     ## task3
27 def add(a, b):
28     return a + b
29
30 def subtract(a, b):
31     return a - b
32
33 def multiply(a, b):
34     return a * b
35
36 def divide(a, b):
37     if b == 0:
38         return "Error: Division by zero"
39     return a / b
40 print("Simple Calculator")
41 print("1. Add")
42 print("2. Subtract")
43 print("3. Multiply")
44 print("4. Divide")
45 choice = int(input("Enter your choice (1-4): "))
46 num1 = float(input("Enter first number: "))
47 num2 = float(input("Enter second number: "))

```

```

44 print("4. Divide")
45 choice = int(input("Enter your choice (1-4): "))
46 num1 = float(input("Enter first number: "))
47 num2 = float(input("Enter second number: "))
48 if choice == 1:
49     print("Result:", add(num1, num2))
50 elif choice == 2:
51     print("Result:", subtract(num1, num2))
52 elif choice == 3:
53     print("Result:", multiply(num1, num2))
54 elif choice == 4:
55     print("Result:", divide(num1, num2))
56 else:
57     print("Invalid choice")
58
59
60

```

*Output:*

```
[Done] exited with code=0 in 0.066 seconds
```

**Observation:** The Gemini AI-generated calculator code is simple, well-structured, and beginner-friendly. It uses separate functions for each operation, improving readability and

maintainability, and correctly handles division by zero. Overall, the code and explanation are clear and suitable for learning and review.

#### Task 4: Armstrong Number Optimization

**Prompt:** *Generate a simple Python program to check whether a number is an Armstrong number."*

```
59     ##Task4
60     num = int(input("Enter a number: "))
61     temp = num
62     order = len(str(num))
63     sum_val = 0
64
65     while temp > 0:
66         digit = temp % 10
67         sum_val += digit ** order
68         temp //= 10
69
70     if sum_val == num:
71         print("Armstrong Number")
72     else:
73         print("Not an Armstrong Number")
74
75
```

Output:

```
[Done] exited with code=0 in 0.08 seconds
```

**Observation:** The Gemini AI solution works correctly but is limited and less reusable. The Cursor AI version improves efficiency by using a function, supports numbers of any length, and has cleaner, more readable code. Overall, Cursor AI provides a more optimized and scalable solution.

