

# Assignment 1

## AI Assisted Coding

Name:- M Geethika

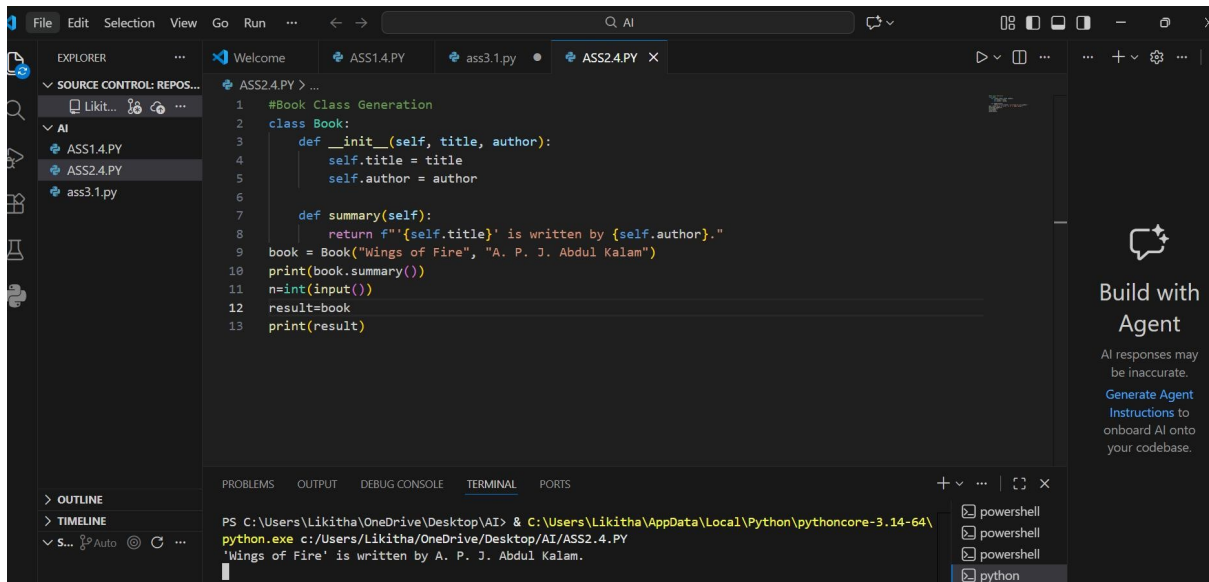
HTNO: 2303A52276

### Task 1:

#### Prompt:

#### #Book Class Generation

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

A screenshot of a code editor interface, likely Visual Studio Code, showing a Python file named ASS2.4.PY. The code defines a class 'Book' with an '\_\_init\_\_' method that takes 'title' and 'author' as arguments and assigns them to 'self.title' and 'self.author'. It also has a 'summary' method that returns a string formatted as '{self.title} is written by {self.author}.'. Below the class definition, an instance of the class is created: 'book = Book("Wings of Fire", "A. P. J. Abdul Kalam")'. The code then prints the summary of the book, takes user input for a number 'n', and prints the book object 'n' times. The terminal at the bottom shows the command 'python.exe c:/Users/Likitha/OneDrive/Desktop/AI/ASS2.4.PY' and the output: 'Wings of Fire' is written by A. P. J. Abdul Kalam. On the right side of the editor, there is a sidebar with the text 'Build with Agent' and a note that 'AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase.'

#### 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.

```

14
15 #Sorting Dictionaries with AI
16
17 def sort_by_age(users):
18     return sorted(users, key=lambda user: user["age"])
19
20 users = [
21     {"name": "Alice", "age": 25},
22     {"name": "Bob", "age": 20},
23     {"name": "Charlie", "age": 30}
24 ]
25
26 print(sort_by_age(users))
27

```

```

PS C:\Users\Likitha\OneDrive\Desktop\AI> & C:\Users\Likitha\AppData\Local\Python\pythoncore-3.14-64\python.exe c:/Users/Likitha/OneDrive/Desktop/AI/ASS2.4.PY
[{'name': 'Bob', 'age': 20}, {'name': 'Alice', 'age': 25}, {'name': 'Charlie', 'age': 30}]
PS C:\Users\Likitha\OneDrive\Desktop\AI>

```

#### 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.

The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project named 'AI' with files 'ASS1.4.PY', 'ASS2.4.PY', and 'ass3.1.py'. The main editor window displays the code for 'ASS2.4.PY', which is a calculator program using functions for addition, subtraction, multiplication, and division. The code includes error handling for division by zero. The Output pane at the bottom shows the execution of the program, displaying the results of the arithmetic operations: Addition: 15, Subtraction: 5, Multiplication: 50, and Division: 2.0. The terminal pane on the right shows the command prompt where the program was executed.

```
27 |
28 | #Calculator Using Functions
29 | def add(a, b):
30 |     return a + b
31 | def subtract(a, b):
32 |     return a - b
33 | def multiply(a, b):
34 |     return a * b
35 | def divide(a, b):
36 |     if b == 0:
37 |         return "Error: Division by zero"
38 |     return a / b
39 | print("Addition:", add(10, 5))
40 | print("Subtraction:", subtract(10, 5))
41 | print("Multiplication:", multiply(10, 5))
42 | print("Division:", divide(10, 5))
43 |
44 |
```

```
PS C:\Users\Likitha\OneDrive\Desktop\AI> & C:\Users\Likitha\AppData\Local\Python\pythoncore-3.14-64\python.exe c:/Users/Likitha/OneDrive/Desktop/AI/ASS2.4.PY
[{'name': 'Bob', 'age': 20}, {'name': 'Alice', 'age': 25}, {'name': 'Charlie', 'age': 30}]
PS C:\Users\Likitha\OneDrive\Desktop\AI> & C:\Users\Likitha\AppData\Local\Python\pythoncore-3.14-64\python.exe c:/Users/Likitha/OneDrive/Desktop/AI/ASS2.4.PY
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.0
PS C:\Users\Likitha\OneDrive\Desktop\AI>
```

#### 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.

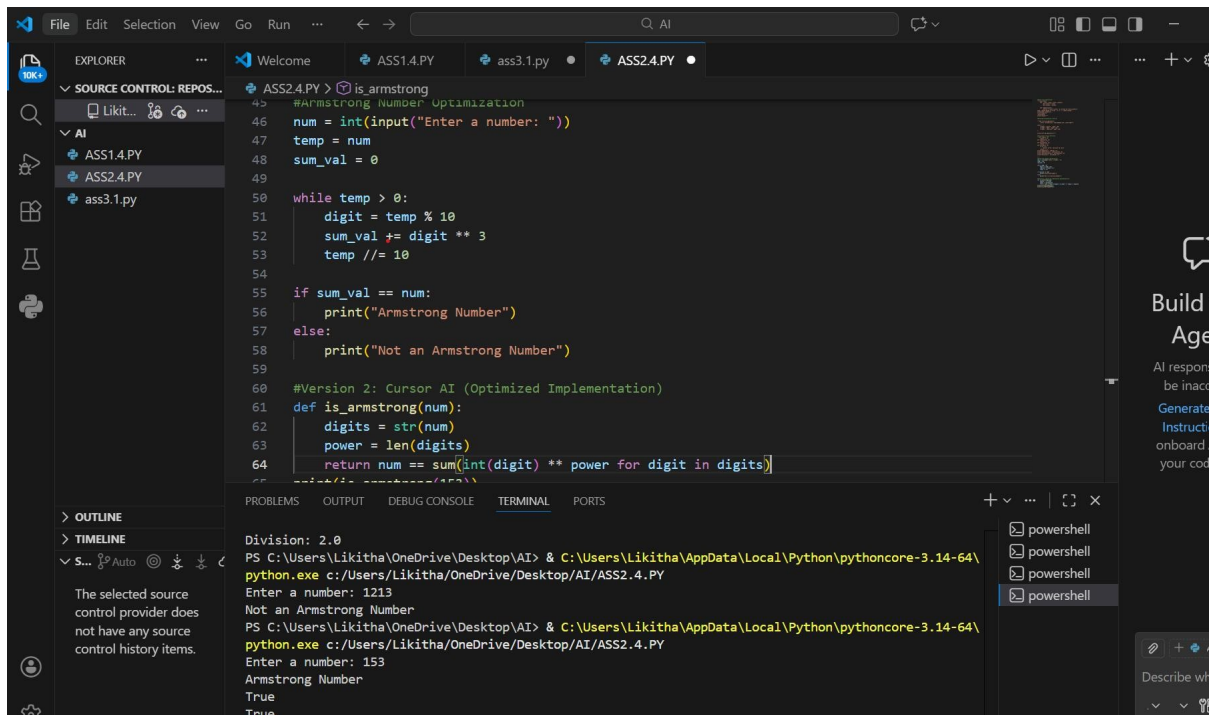
#### Task 4: Armstrong Number Optimization

##### 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.