

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.

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER:** Shows files: Ass 2.4.py, Ass 2.4.py (untracked), Ass 14.py, and Ass 2.4.py.
- EDITOR:** Displays the code for a Book class:

```
1 #book Class Generation
2
3 class book:
4     def __init__(self, title, author):
5         self.title = title
6         self.author = author
7
8     def summary(self):
9         return f'{self.title} is written by {self.author}.'
10
11 book = Book("Wings of Fire", "A. P. J. Abdul Kalam")
12 print(book.summary())
13 n=int(input())
14 result=book
15 print(result)
16
17
```
- OUTPUT:** Shows the terminal output:

```
PS C:\Users\Sanja\OneDrive\Desktop\Ai assistant coding> & 'c:\Users\Sanja\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\Sanja\vscode\extensions\ms-python.debugger-2025.18.0-win32-x64\bundled\libs\debugger\launcher' '58718' '--' 'C:\Users\Sanja\OneDrive\Desktop\Ai assistant coding\Ass 2.4.py'
'Wings of Fire' is written by A. P. J. Abdul Kalam.
```
- CHAT:** Shows a message from the AI agent: "Build with Agent".
- SUGGESTED ACTIONS:** Shows a button to "Describe what to build next".

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.

The screenshot shows the VS Code interface. In the Explorer sidebar, there are several files: 'Ass 2.4.py' (the active file), 'Ass 2.4.py > ...', 'AI ASS...', 'Ass 1.4.py', and 'Ass 2.4.PY'. The code editor displays the following Python script:

```
16 #Sorting Dictionaries with AI
17
18 def sort_by_age(users):
19     return sorted(users, key=lambda user: user["age"])
20
21 users = [
22     {"name": "Alice", "age": 25},
23     {"name": "Bob", "age": 20},
24     {"name": "Charlie", "age": 30}
25 ]
26
27 print(sort_by_age(users))
28
```

The terminal below shows the command being run and its output:

```
PS C:\Users\Sanja\OneDrive\Desktop\AI assistant coding> & 'c:\Users\Sanja\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Sanja\vscodeextensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58718' '--' 'c:\Users\Sanja\OneDrive\Desktop\AI assistant coding\Ass 2.4.py'
'Wings of Fire' is written by A. P. J. Abdul Kalam.
PS C:\Users\Sanja\OneDrive\Desktop\AI assistant coding> ^
PS C:\Users\Sanja\OneDrive\Desktop\AI assistant coding>
PS C:\Users\Sanja\OneDrive\Desktop\AI assistant coding> < c: cd 'c:\Users\Sanja\OneDrive\Desktop\AI assistant coding'; & 'c:\Users\Sanja\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\Sanja\vscodeextensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50362' '--' 'c:\Users\Sanja\OneDrive\Desktop\AI assistant coding\Ass 2.4.py'
[{"name": "Bob", "age": 20}, {"name": "Alice", "age": 25}, {"name": "Charlie", "age": 30}]
PS C:\Users\Sanja\OneDrive\Desktop\AI assistant coding>
```

The right-hand sidebar features a 'Build with Agent' panel with a message: 'AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase.' It also includes a 'SUGGESTED ACTIONS' section with options like 'Build Workspace' and 'Show Config'.

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 a Microsoft Visual Studio Code (VS Code) interface. The left sidebar has an 'EXPLORER' view with files 'Ass 2.4.py', 'Ass 2.4.py > ...', 'AI ASS...', 'Ass 1.4.py', and 'Ass 2.4.py'. The main editor area contains the following Python code:

```
#Calculator Using Functions
def add(a, b):
    return a + b
def subtract(a, b):
    return a - b
def multiply(a, b):
    return a * b
def divide(a, b):
    if b == 0:
        return "Error: Division by zero"
    return a / b
print("Addition:", add(10, 5))
print("Subtraction:", subtract(10, 5))
print("Multiplication:", multiply(10, 5))
print("Division:", divide(10, 5))
```

The terminal at the bottom shows the output of running the script:

```
PS C:\Users\Sanja\OneDrive\Desktop\Ai assistant coding> c:; cd "c:\Users\Sanja\OneDrive\Desktop\Ai assistant coding"; & "c:\Users\Sanja\AppData\Local\Programs\Python\Python313\python.exe" "c:\Users\Sanja\.vscode\extensions\ms-python.debugpy-2025.18.0\Sanja\AppData\Local\Programs\Python\Python313\python.exe" "c:\Users\Sanja\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher" "64450" -- 'c:\Users\Sanja\OneDrive\Desktop\Ai assistant coding\Ass 2.4.py'
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.0
PS C:\Users\Sanja\OneDrive\Desktop\Ai assistant coding> []
```

The right side of the interface includes a 'CHAT' panel with an 'AI assistant coding' section, an 'AI Build with Agent' section, and a 'SUGGESTED ACTIONS' panel.

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

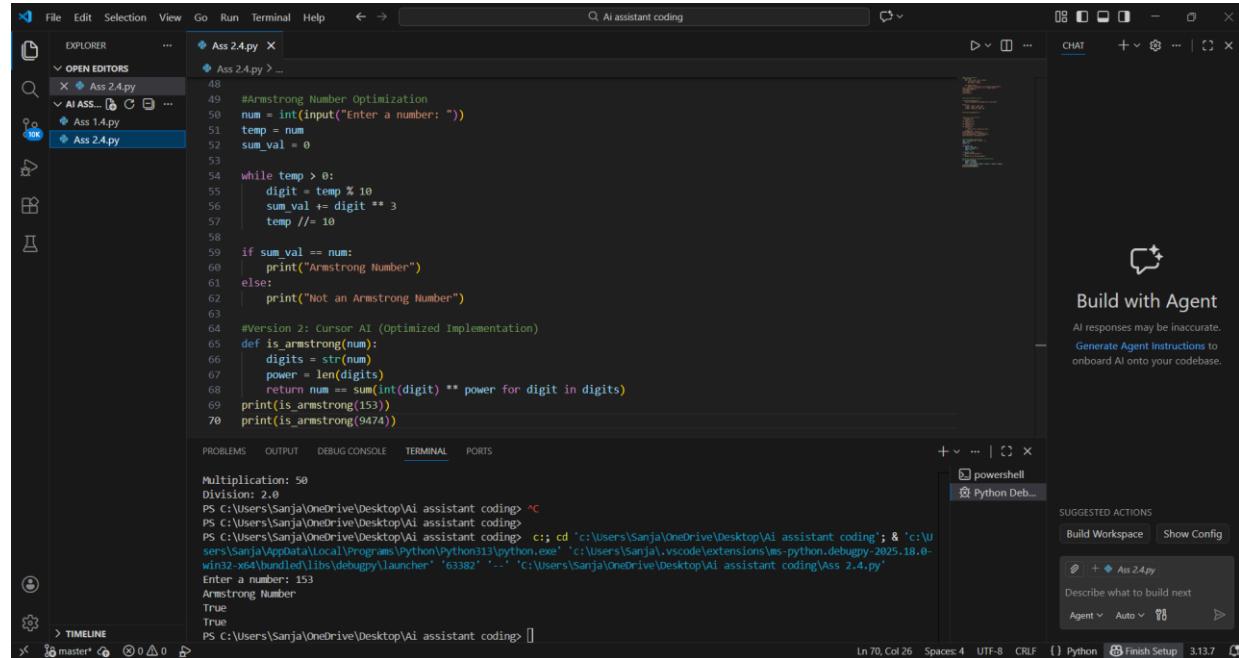
Task 4: Armstrong Number Optimization

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.



The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The left sidebar has an 'EXPLORER' view with files 'Ass 2.4.py', 'Ass 2.4.py > ...', 'AI ASS...', 'Ass 1.4.py', and 'Ass 2.4.py'. The main editor area contains Python code for Armstrong number optimization. The terminal at the bottom shows command-line output for testing the code.

```
File Edit Selection View Go Run Terminal Help ← → Q: Ai assistant coding
OPEN EDITORS Ass 2.4.py ...
AI ASS... Ass 2.4.py > ...
Ass 1.4.py
Ass 2.4.py
EXPLORER Ass 2.4.py
Ass 2.4.py > ...
AI ASS...
Ass 1.4.py
Ass 2.4.py
CHAT + ⚙️ ... | ...
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Multiplication: 50
Division: 2.0
PS C:\Users\Sanja\OneDrive\Desktop\Ai assistant coding> ^C
PS C:\Users\Sanja\OneDrive\Desktop\Ai assistant coding>
PS C:\Users\Sanja\OneDrive\Desktop\Ai assistant coding> c;; cd "c:\Users\Sanja\OneDrive\Desktop\Ai assistant coding"; & 'c:\Users\Sanja\AppData\Local\Programs\Python\3.10\python.exe' 'c:\Users\Sanja\vscode\extensions\ms-python.python.debugv2025.18.0-win32-x64\bundled\lols\debugpy\launcher' '63382' ... 'c:\Users\Sanja\OneDrive\Desktop\Ai assistant coding\Ass 2.4.py'
Enter a number: 153
Armstrong Number
True
True
PS C:\Users\Sanja\OneDrive\Desktop\Ai assistant coding> []
Ln 70, Col 26 Spaces: 4 UTF-8 CRLF { Python Finish Setup 3.13.7
SUGGESTED ACTIONS
Build Workspace Show Config
Agent Auto >
```

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.