

# AI ASSISTED CODING

## ASSIGNMENT – 2.4

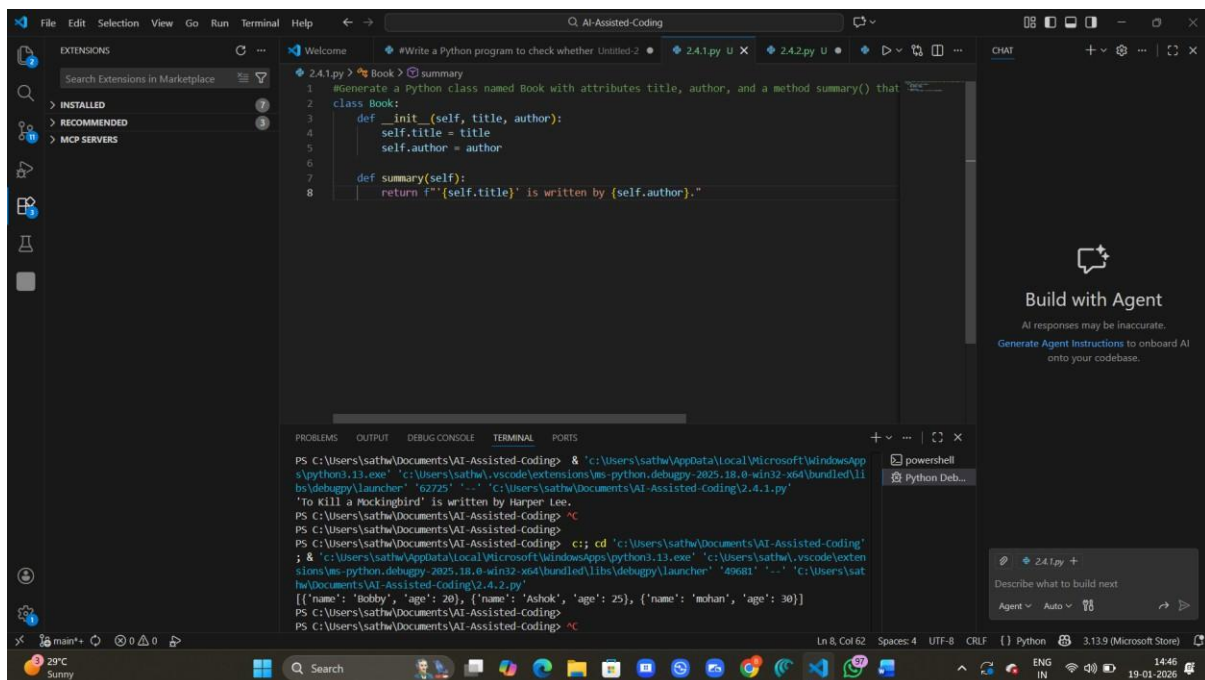
H Ti.No:2303A510B4

Batch:14

Task 1: Use Cursor AI to generate a Python class Book with attributes title, author, and a summary () method.

Prompt : “Generate a Python class named Book with attributes title, author, and a method summary() that returns a formatted string with the title and author.”

Code and output :



The screenshot displays the Cursor AI IDE interface. The main editor window shows a Python file named '2.4.1.py' containing the following code:

```
1 #Generate a Python class named Book with attributes title, author, and a method summary() that
2 class Book:
3     def __init__(self, title, author):
4         self.title = title
5         self.author = author
6
7     def summary(self):
8         return f'{self.title} is written by {self.author}.'
```

The terminal window at the bottom shows the execution of the code:

```
PS C:\Users\sathw\Documents\AI-Assisted-Coding> & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\Documents\AI-Assisted-Coding\2.4.1.py'
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
PS C:\Users\sathw\Documents\AI-Assisted-Coding> cd 'c:\Users\sathw\Documents\AI-Assisted-Coding'
PS C:\Users\sathw\Documents\AI-Assisted-Coding> & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\Documents\AI-Assisted-Coding\2.4.1.py'
[[{'name': 'Bobby', 'age': 20}, {'name': 'Ashok', 'age': 25}, {'name': 'mohan', 'age': 30}]]
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
```

The right sidebar shows the 'CHAT' panel with the prompt: "Build with Agent" and the response: "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase."

Justication:

Object-Oriented Programming helps model real-world entities like books using classes and objects

Encapsulation allows data (title, author) and behavior (summary) to be bundled together logically.

This approach improves code organization, scalability, and ease of maintenance in software systems.

Task 2: Use Gemini and Cursor AI to generate code that sorts a list of dictionaries

by a key.

Prompt: Write Python code to sort a list of dictionaries by the key age. Explain the code

briefly.

Code and output :

```
1 #Write Python code to sort a list of dictionaries by the key age. Explain the code briefly."
2
3
4 users = [
5     {"name": "Ashok", "age": 25},
6     {"name": "Bobby", "age": 20},
7     {"name": "mohan", "age": 30}
8 ]
9
10 def sort_by_age(user_list):
11     return sorted(user_list, key=lambda user: user["age"])
12
13
14 if __name__ == "__main__":
15     sorted_users = sort_by_age(users)
16     print(sorted_users)
```

```
PS C:\Users\sathw\Documents\AI-Assisted-Coding> cd 'c:\Users\sathw\Documents\AI-Assisted-Coding'
PS C:\Users\sathw\Documents\AI-Assisted-Coding> & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\vscode\extensions\ms-python.debugpy-2025.18.8-win32-x64\bundle\libs\debugpy\launcher' '49681' '-...' 'c:\Users\sathw\Documents\AI-Assisted-Coding\2.4.2.py'
[{'name': 'Bobby', 'age': 20}, {'name': 'Ashok', 'age': 25}, {'name': 'mohan', 'age': 30}]
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
PS C:\Users\sathw\Documents\AI-Assisted-Coding> cd 'c:\Users\sathw\Documents\AI-Assisted-Coding'
PS C:\Users\sathw\Documents\AI-Assisted-Coding> & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\vscode\extensions\ms-python.debugpy-2025.18.8-win32-x64\bundle\libs\debugpy\launcher' '62628' '-...' 'c:\Users\sathw\Documents\AI-Assisted-Coding\2.4.2.py'
```

Justification:

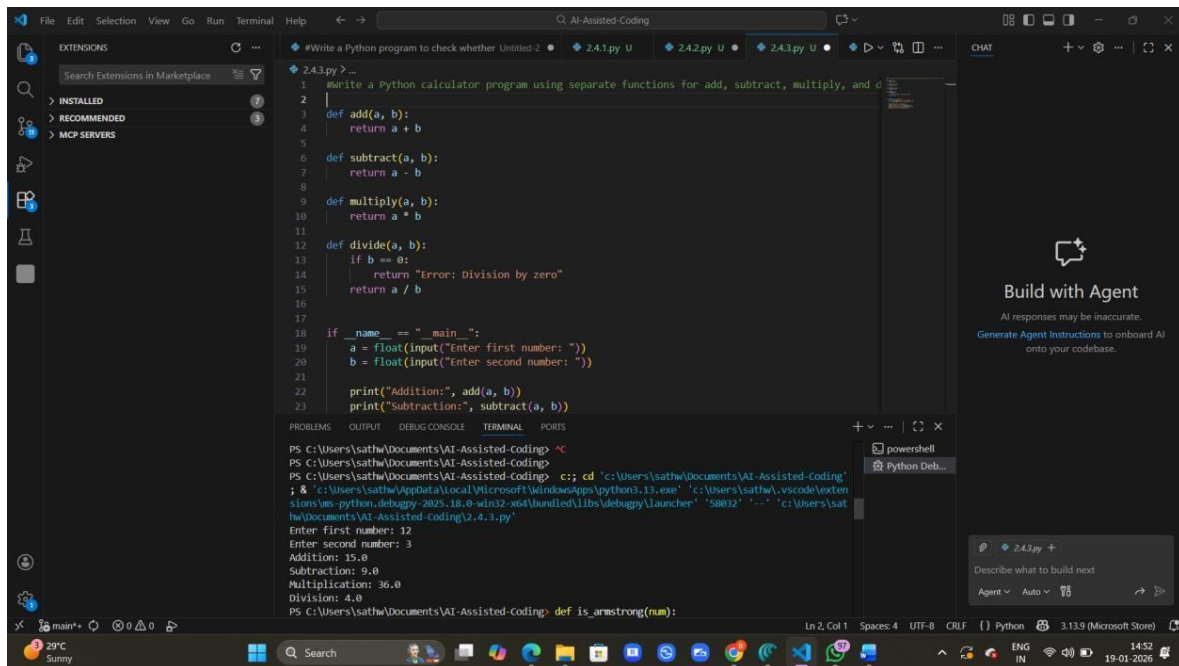
Gemini’s code is more concise and easy to understand for quick tasks.

Cursor’s approach is slightly more structured and reusable due to function encapsulation.

Task 3: Ask Gemini to generate a calculator using functions and explain how it works.

Prompt: Write a Python calculator program using separate functions for add, subtract, multiply, and divide. Then explain how the program works step by step.

Code and Output:



```
#Write a Python program to check whether
1  Write a Python calculator program using separate functions for add, subtract, multiply, and divide.
2
3  def add(a, b):
4      return a + b
5
6  def subtract(a, b):
7      return a - b
8
9  def multiply(a, b):
10     return a * b
11
12 def divide(a, b):
13     if b == 0:
14         return "Error: Division by zero"
15     return a / b
16
17
18 if __name__ == "__main__":
19     a = float(input("Enter first number: "))
20     b = float(input("Enter second number: "))
21
22     print("Addition:", add(a, b))
23     print("Subtraction:", subtract(a, b))
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
PS C:\Users\sathw\Documents\AI-Assisted-Coding> cd 'c:\Users\sathw\Documents\AI-Assisted-Coding'
PS C:\Users\sathw\Documents\AI-Assisted-Coding> c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe 'c:\Users\sathw\Documents\AI-Assisted-Coding\2.4.3.py'
Enter first number: 12
Enter second number: 3
Addition: 15.0
Subtraction: 9.0
Multiplication: 36.0
Division: 4.0
PS C:\Users\sathw\Documents\AI-Assisted-Coding> def is_armstrong(num):
```

Justification:

Each arithmetic operation is written as a separate function, improving modularity.

Functions take two inputs, perform the operation, and return the result.

Division includes a safety check to prevent division by zero errors.

Task 4: Generate an Armstrong number program using Gemini, then improve it using Cursor AI.

Prompt: Write a Python program to check whether a given number is an Armstrong number. Use basic Python constructs and explain briefly.

Code and Input:

```
1 #Write a Python program to check whether a given number is an Armstrong number. Use basic Python constructs and explain briefly.
2 def is_armstrong(num):
3     digits = str(num)
4     power = len(digits)
5     total = sum(int(digit) ** power for digit in digits)
6     return total == num
7
8
9 if __name__ == "__main__":
10     number = int(input("Enter a number: "))
11
12     if is_armstrong(number):
13         print("Armstrong Number")
14     else:
15         print("Not an Armstrong Number")
16
```

```
Enter a number: 157
Not an Armstrong Number
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
PS C:\Users\sathw\Documents\AI-Assisted-Coding> cd 'c:\Users\sathw\Documents\AI-Assisted-Coding'
; & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bin\debugpy\launcher' '50589' '-...' 'c:\Users\sathw\Documents\AI-Assisted-Coding\2.4.4.py'
Enter a number: 153
Armstrong Number
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
PS C:\Users\sathw\Documents\AI-Assisted-Coding>
PS C:\Users\sathw\Documents\AI-Assisted-Coding> cd 'c:\Users\sathw\Documents\AI-Assisted-Coding'
```

Justification:

The optimized version avoids string conversion, reducing unnecessary overhead.

It uses arithmetic operations, which are more efficient and closer to low-level computation.

The logic is clearer for understanding numeric processing and improves overall performance.