

AI ASSISTED CODING-LAB ASSIGNMENT

ASSIGNMENT – 2.4

Name:Ch.venugopal

Batch:13

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 shows the Visual Studio Code interface with the following details:

- Top Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** Welcome, AI Assisted Coding.
- Left Sidebar:** Extensions, Search Extensions in Marketplace, INSTALLED, RECOMMENDED, MCP SERVERS.
- Code Editor:** A Python file named `book.py` with the following code:

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

    def summary(self):
        return f'{self.title} is written by {self.author}'
```
- Bottom Panel:** A terminal window showing command-line interactions with Python code. It includes a scroll bar and a status bar at the bottom.
- Right Panel:** A "Build with Agent" panel with a message: "All responses may be inaccurate. Generate Agent instructions to onboard AI with your codebase." It also shows a progress bar and a status bar.

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 :

A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows extensions like 'Python' and 'AI-Assisted Coding'. The main editor area contains the following Python code:

```
242.py - Write Python code to sort a list of dictionaries by the key age. Explain the code briefly.
1
2  # Write Python code to sort a list of dictionaries by the key age. Explain the code briefly.
3
4 users = [
5     {"name": "Aisha", "age": 25},
6     {"name": "Bobby", "age": 30},
7     {"name": "Mehan", "age": 35}
8 ]
9
10 def sort_by_age(user_list):
11     return sorted(user_list, key=lambda user: user["age"])
12
13 if __name__ == "__main__":
14     sorted_users = sort_by_age(users)
15     print(sorted_users)
```

The bottom right corner has a 'Build with Agent' button. The terminal at the bottom shows command-line history related to the file.

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:

A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows extensions like 'Python' and 'AI-Assisted Coding'. The main editor area contains the following Python code:

```
243.py - Write a Python calculator program using separate functions for add, subtract, multiply, and divide. Then explain how the program works step by step.
1
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 if __name__ == "__main__":
18    a = float(input("Enter first number: "))
19    b = float(input("Enter second number: "))
20
21    print("Addition:", add(a, b))
22    print("Subtraction:", subtract(a, b))
```

The bottom right corner has a 'Build with Agent' button. The terminal at the bottom shows command-line history related to the file.

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:

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar displays extensions like 'All-in-one Python' and 'AI-Assisted Coding'. The main editor area contains Python code for checking if a number is an Armstrong number. The terminal at the bottom shows the code running and outputting 'Not an Armstrong Number' for the input '134'. A floating 'Build with Agent' window is visible on the right, indicating AI assistance.

```
# An Armstrong number is a number that is equal to the sum of its own digits each raised to the power of the number of digits. For example, 153 is an Armstrong number because 1^3 + 5^3 + 3^3 = 153.
def is_armstrong(num):
    digits = str(num)
    power = len(digits)
    total = sum(int(digit) ** power for digit in digits)
    return total == num

if __name__ == "__main__":
    number = int(input("Enter a number: "))
    if is_armstrong(number):
        print("Armstrong Number")
    else:
        print("Not an Armstrong Number")
```