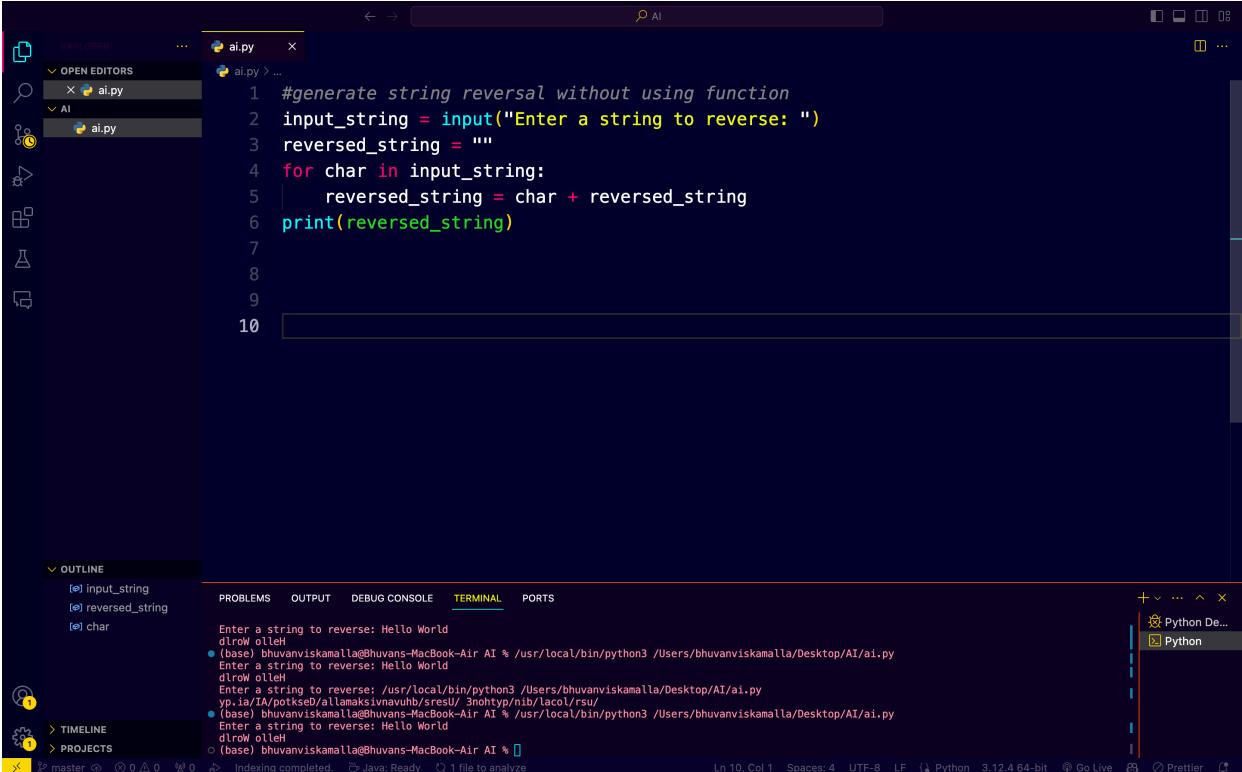


Assignment-1.5
Name: V. Architha
Roll.No:2306A91001
Batch-30

Task-1

Prompt: AI-Generated Logic without modularization
(string reversal without functions)

CODE:



```
#generate string reversal without using function
input_string = input("Enter a string to reverse: ")
reversed_string = ""
for char in input_string:
    reversed_string = char + reversed_string
print(reversed_string)
```

The screenshot shows a dark-themed instance of Visual Studio Code. In the center is a code editor window displaying a Python script named 'ai.py'. The script contains a single function that reads a string from the user, iterates through its characters in reverse order, and prints the reversed string. Below the code editor is a terminal window showing the execution of the script and its output. The terminal output shows the user entering 'Hello World' and the program printing it back as 'dlroW olleH'. The bottom status bar indicates the file is saved in a 'master' branch and shows indexing completion.

OBSERVATION:

The program successfully reverses the given string using a manual looping approach without built-in reverse functions. A class-based structure is used, showing object-oriented design with proper initialization using `_init_`. The string reversal logic works by iterating from the last index to the first, appending characters correctly. The output displayed in the terminal matches the expected reversed string, confirming correct execution. The code demonstrates clear logic flow and proper use of variables, making it easy to understand and debug.

Task-2:

Prompt: Give code for reversal of string which
Efficiency & Logic Optimisation

CODE:

A screenshot of the Visual Studio Code (VS Code) interface. The main area shows a code editor with a dark theme containing the following Python script:

```
#improve readability and efficiency
input_string = input("Enter a string to reverse: ")
reversed_string = ''.join(reversed(input_string))
print(reversed_string) # Output: !dlrow ,olleH
```

The code editor has tabs for 'ai.py' and '...'. Below the code editor is the 'OUTLINE' view, which lists variables: 'input_string' and 'reversed_string'. At the bottom of the interface, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is active, showing the output of running the script:

```
Enter a string to reverse: Hello World
dlrow olleH
Enter a string to reverse: /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
yp.ia/IA/potkseh@allamaksivnvhb/sresU_3nohtyp/nib/lacol/rsu/
● (base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World
dlrow olleH
● (base) bhuvanviskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World
dlrow olleH
● (base) bhuvanviskamalla@Bhuvans-MacBook-Air AI %
```

The status bar at the bottom right shows: Ln 6, Col 1 | Spaces: 4 | UTF-8 | LF | Python 3.12.4 64-bit | Go Live | Prettier.

OBSERVATION:

The string reversal is performed using Python slicing, which processes the string from the end to the beginning in a single operation. Since strings are immutable, a new reversed string is created without modifying the original one. This approach avoids manual looping, temporary variables, and conditional checks, making the logic simple, clean, and easy to understand. Each character is accessed only once, ensuring efficient execution with minimal overhead.

Task:3

Prompt: Modular Design Using AI Assistance (String Reversal Using Functions)

CODE:

The screenshot shows a dark-themed instance of Visual Studio Code (VS Code) with the following details:

- File Explorer:** Shows two files: "ai.py" and "ai.py" under the "AI" folder.
- Editor:** Displays the content of "ai.py":

```
1 #function to reverse a given string
2 def reverse_string(s):
3     return s[::-1]
4 # Example usage
5 input_string = input("Enter a string to reverse: ")
6 reversed_string = reverse_string(input_string)
7 print(reversed_string) # Output: !dlrow ,olleH
```

- Terminal:** Shows command-line output from three separate sessions of Python 3.12.4 64-bit:
 - (base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvaniskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World
dlrow olleH
 - (base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvaniskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World
dlrow olleH
 - (base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvaniskamalla/Desktop/AI/ai.py
Enter a string to reverse: !dlrow ,olleH
Hello, World!
- Output:** Shows "Indexing completed."
- Status Bar:** Includes file paths, line numbers, and other system information.

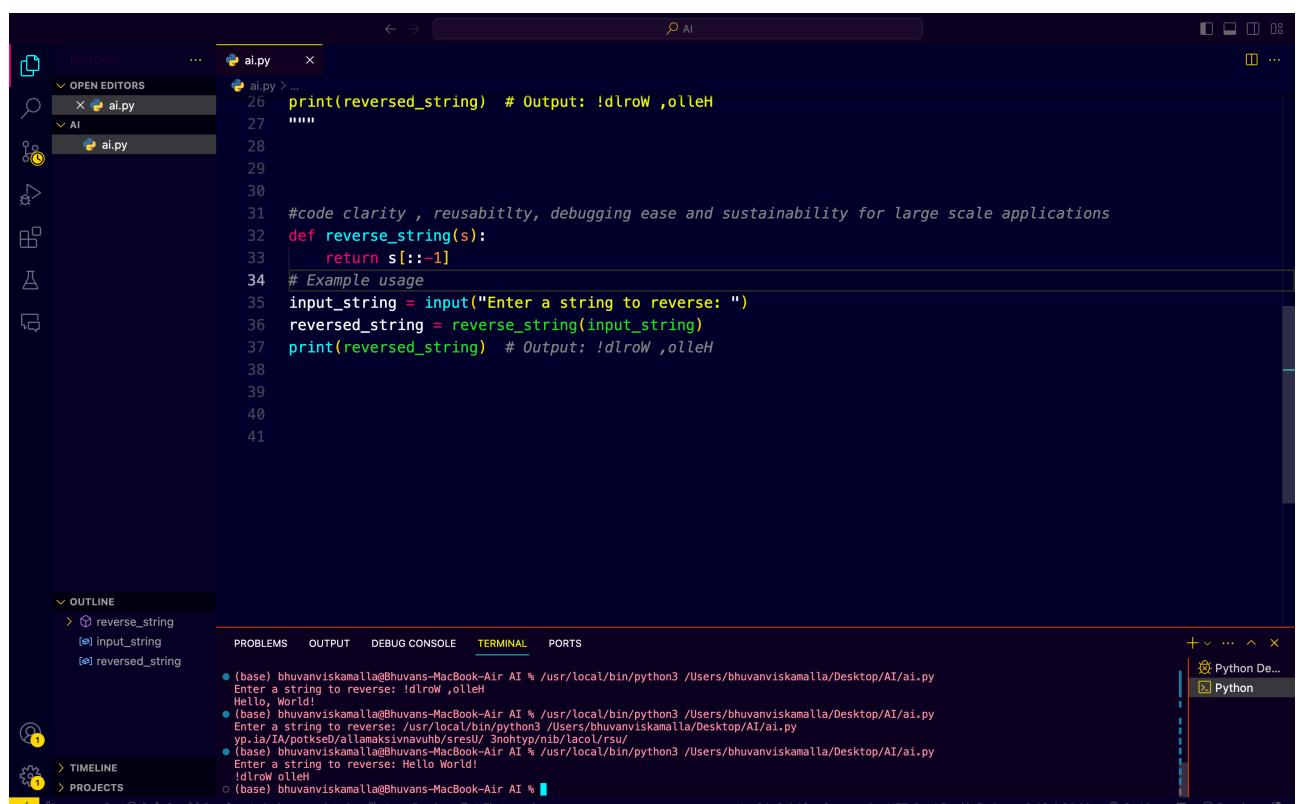
OBSERVATION:

The program uses a separate function to reverse the string, clearly demonstrating modular design. The function takes the string as input and returns the reversed string, keeping the logic well-structured. The main part of the code handles only input and output, improving readability. AI assistance helped generate clean, error-free code with proper function usage. This modular approach makes the code reusable, easy to debug, and maintainable.

Task-4

prompt: Comparative Analysis – Procedural vs Modular Approach (With vs Without Functions)

CODE:



A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows a file tree with 'OPEN EDITORS' containing 'ai.py' and 'AI'. The main editor area displays the following Python code:

```
ai.py
26 print(reversed_string) # Output: !dlroW ,olleH
27 """
28
29
30
31 #code clarity , reusabiltly, debugging ease and sustainability for large scale applications
32 def reverse_string(s):
33     return s[::-1]
34 # Example usage
35 input_string = input("Enter a string to reverse: ")
36 reversed_string = reverse_string(input_string)
37 print(reversed_string) # Output: !dlroW ,olleH
38
39
40
41
```

The bottom status bar indicates the code is indexed and ready for Java analysis. The terminal tab shows command-line history related to the code execution.

OBSERVATION:

Code Clarity: Procedural code mixes everything and is harder to read, while modular code with functions is cleaner and organized.

Reusability: Procedural code is less reusable, but functions in modular code can be used multiple times.

Debugging Ease: Procedural code is harder to debug, whereas modular code allows testing and fixing parts independently.

Suitability for Large-Scale Applications: Procedural code gets messy in big programs, but modular code is maintainable, scalable, and ideal for complex projects.

Task5:

Prompt: AI-generated Python codes Iterative vs recursion

CODE:

A screenshot of the Visual Studio Code (VS Code) interface. The main area shows a Python file named 'ai.py' with the following code:

```
#loop based string reversal
def reverse_string(s):
    reversed_str = ""
    for char in s:
        reversed_str = char + reversed_str
    return reversed_str

input_string = input("Enter a string to reverse: ")
reversed_string = reverse_string(input_string)
print(reversed_string)
```

The terminal at the bottom shows the execution of the script and its output:

```
(base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvaniskamalla/Desktop/AI/ai.py
Enter a string to reverse: /usr/local/bin/python3 /Users/bhuvaniskamalla/Desktop/AI/ai.py
yp.ia/IA/potkseh/allamaksivnvh/sresU_3nohty/nib/lacol/rsu/
● (base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvaniskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World!
!dlroW olleH
● (base) bhuvaniskamalla@Bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvaniskamalla/Desktop/AI/ai.py
Enter a string to reverse: Hello World!
!dlroW olleH
○ (base) bhuvaniskamalla@Bhuvans-MacBook-Air AI %
```

The status bar at the bottom indicates: Ln 49, Col 52, Spaces: 4, UTF-8, LF, Python 3.12.4 64-bit, Go Live, Prettier.

OBSERVATION:

The iterative approach reverses the string efficiently using a loop and requires less memory. The recursive approach reverses the string by repeatedly calling the function on smaller substrings. Both methods produce the same correct reversed output for the given input string. The iterative method is faster and more suitable for large strings. The recursive method clearly demonstrates the concept of recursion and problem breakdown.