

## Assignment-1.5

Name: A.Sravani

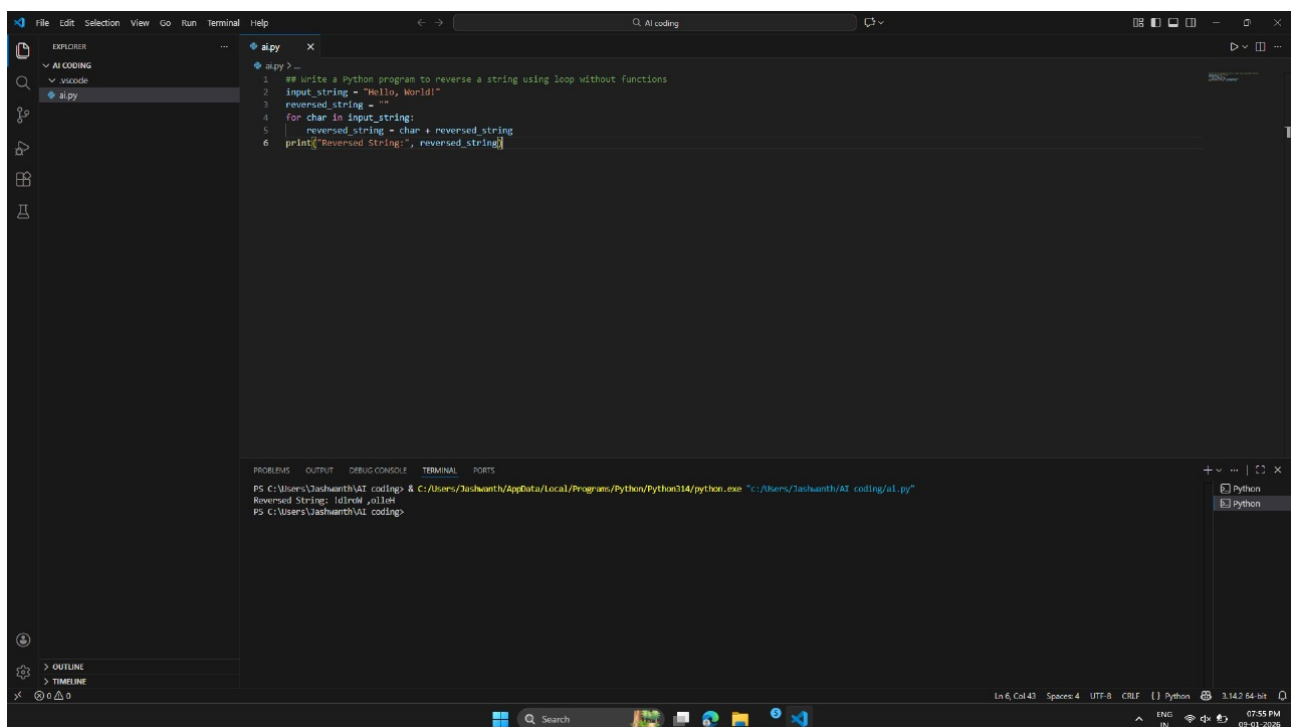
Roll.No:2303A510G7

Batch-30

### Task-1

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

### CODE:



```
ai.py
1 # write a Python program to reverse a string using loop without functions
2 input_string = "Hello, World!"
3 reversed_string = ""
4 for char in input_string:
5     reversed_string = char + reversed_string
6 print("Reversed String:", reversed_string)
```

```
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Programs\Python\Python314\python.exe "c:/Users/Jashwanth/AI coding/ai.py"
Reversed String: !dlroW ,olleH
PS C:\Users\Jashwanth\AI coding>
```

### OBSERVATION:

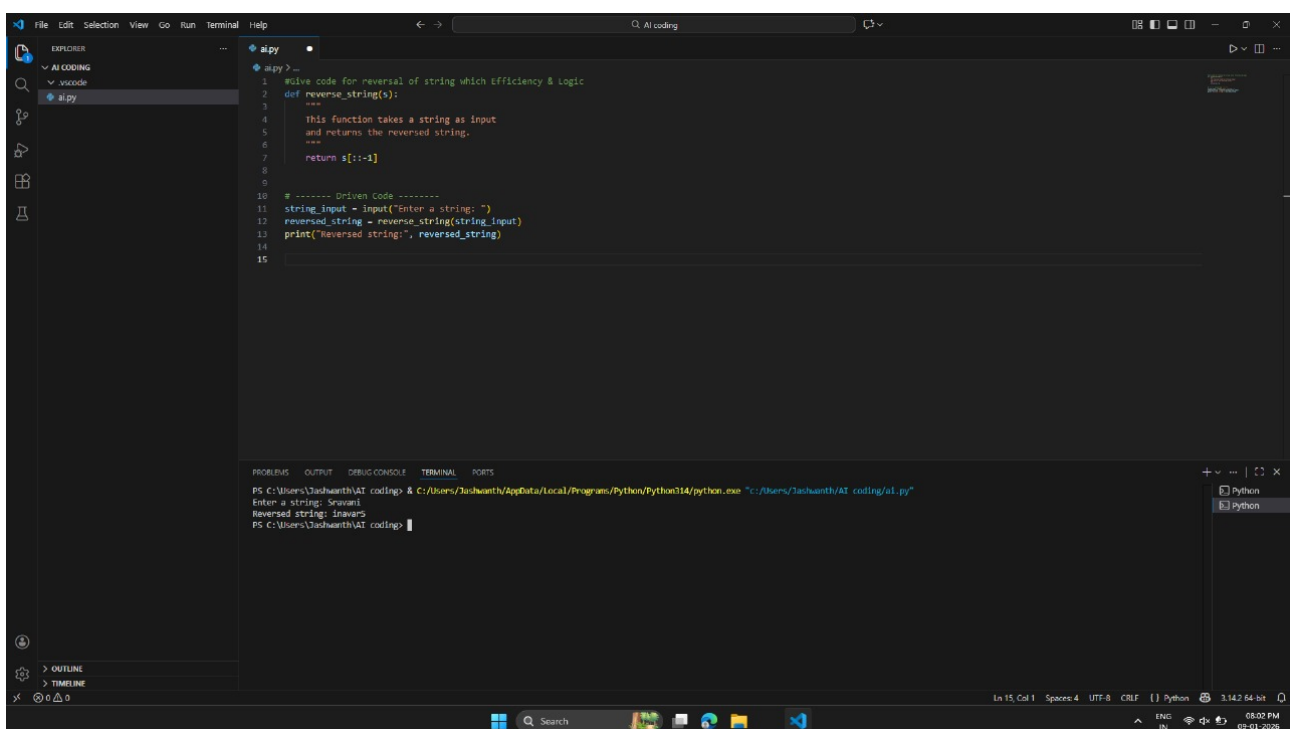
The program successfully reverses the given string using a manual looping approach without built-in reverse functions. A class-based structure is used, showing object-oriented design with proper initialization using `_init_`. The string reversal logic works by iterating from the last index to the

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:



The screenshot shows a Visual Studio Code editor window with a Python file named `ai.py`. The code defines a function `reverse_string(s)` that takes a string `s` as input and returns the reversed string using slicing `s[::-1]`. Below the function, there is a section of driver code that prompts the user to enter a string, calls the `reverse_string` function, and prints the result. The terminal at the bottom shows the execution of the script, where the user enters "Shravan" and the output is "navaerS".

```
1 #Give code for reversal of string which Efficiency & Logic
2 def reverse_string(s):
3     """
4     This function takes a string as input
5     and returns the reversed string.
6     """
7     return s[::-1]
8
9
10 # ----- Driver Code -----
11 string_input = input("Enter a string: ")
12 reversed_string = reverse_string(string_input)
13 print("Reversed string:", reversed_string)
14
15
```

Terminal Output:

```
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Programs\Python\python14\python.exe "c:\Users\Jashwanth\AI coding\ai.py"
Enter a string: Shravan
Reversed string: navaerS
PS C:\Users\Jashwanth\AI coding>
```

## OBSERVATION:

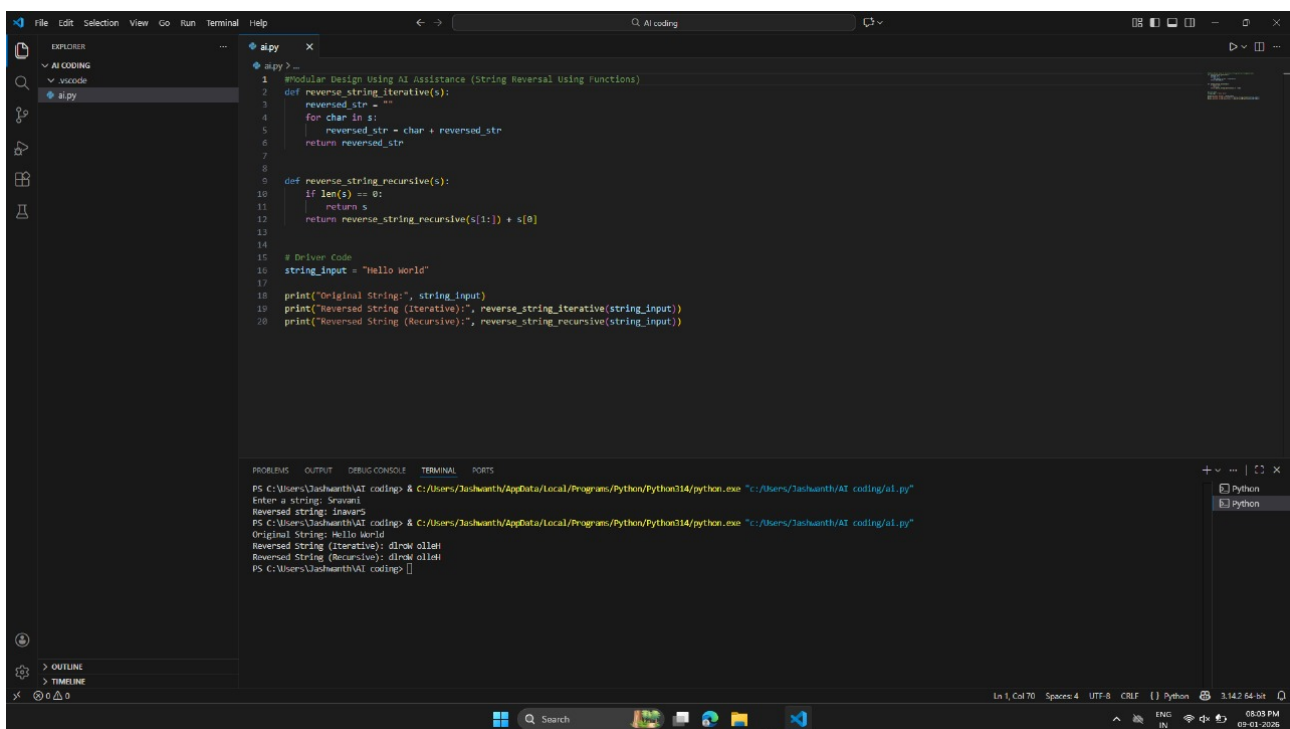
The string reversal is performed using Python slicing, which processes the string from the end to the beginning in a single operation. Since strings are immutable, a new reversed string is created without

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:



```
1 #modular Design Using AI Assistance (String Reversal Using Functions)
2 def reverse_string_iterative(s):
3     reversed_str = ""
4     for char in s:
5         reversed_str = char + reversed_str
6     return reversed_str
7
8
9 def reverse_string_recursive(s):
10    if len(s) == 0:
11        return s
12    return reverse_string_recursive(s[1:]) + s[0]
13
14
15 # Driver Code
16 string_input = "Hello World"
17
18 print("Original String:", string_input)
19 print("Reversed String (Iterative):", reverse_string_iterative(string_input))
20 print("Reversed String (Recursive):", reverse_string_recursive(string_input))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Programs\Python\Python314\python.exe "c:/Users/Jashwanth/AI coding/al.py"
Enter a string: Snovan
Reversed string: navaoS
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Programs\Python\Python314\python.exe "c:/Users/Jashwanth/AI coding/al.py"
Original String: Hello World
Reversed String (Iterative): dlrow olleH
Reversed String (Recursive): dlrow olleH
PS C:\Users\Jashwanth\AI coding>
```

## OBSERVATION:

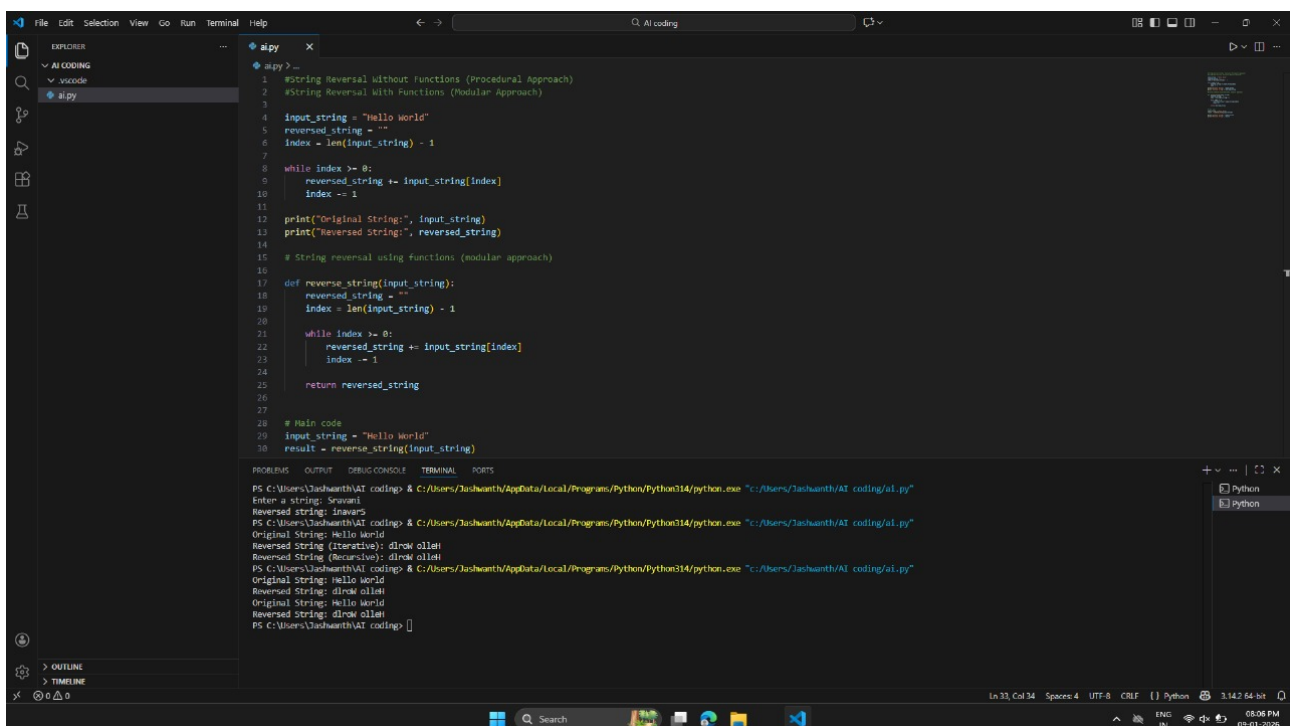
The program uses a separate function to reverse the string, clearly demonstrating modular design. The function takes the string as input and returns the

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:



```
1 #String Reversal without functions (Procedural Approach)
2 #String Reversal with Functions (Modular Approach)
3
4 input_string = "Hello World"
5 reversed_string = ""
6 index = len(input_string) - 1
7
8 while index >= 0:
9     reversed_string += input_string[index]
10    index -= 1
11
12 print("Original String:", input_string)
13 print("Reversed String:", reversed_string)
14
15 # String reversal using functions (modular approach)
16
17 def reverse_string(input_string):
18     reversed_string = ""
19     index = len(input_string) - 1
20
21     while index >= 0:
22         reversed_string += input_string[index]
23         index -= 1
24
25     return reversed_string
26
27
28 # Main code
29 input_string = "Hello World"
30 result = reverse_string(input_string)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Programs\Python\Python114\python.exe "c:/Users/Jashwanth/ai coding/ai.py"
Enter a string: Shruani
Reversed String: inuarv
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Programs\Python\Python114\python.exe "c:/Users/Jashwanth/ai coding/ai.py"
Original String: Hello World
Reversed String (iterative): dlrow olleH
Reversed String (Recursive): dlrow olleH
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Programs\Python\Python114\python.exe "c:/Users/Jashwanth/ai coding/ai.py"
Original String: Hello World
Reversed String: dlrow olleH
Original String: Hello World
Reversed String: dlrow olleH
PS C:\Users\Jashwanth\AI coding> []
```

## 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.

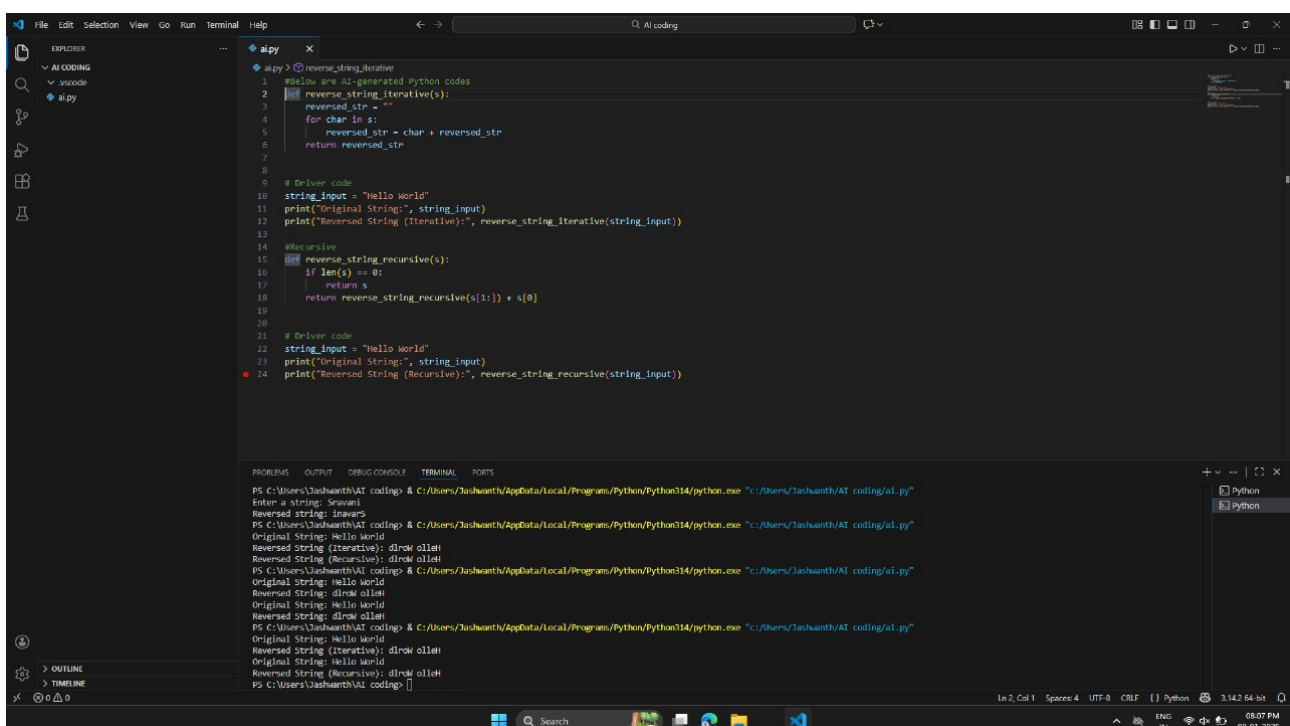
**Debugging Ease:** Procedural code is harder to debug, whereas modular code allows testing and fixing parts independently.

**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:**



```
1 # Iterative
2 def reverse_string_iterative(s):
3     reversed_str = ""
4     for char in s:
5         reversed_str = char + reversed_str
6     return reversed_str
7
8 # Driver code
9 string_input = "Hello World"
10 print("Original String:", string_input)
11 print("Reversed String (Iterative):", reverse_string_iterative(string_input))
12
13 # Recursive
14 def reverse_string_recursive(s):
15     if len(s) == 0:
16         return s
17     return reverse_string_recursive(s[1:]) + s[0]
18
19 # Driver code
20 string_input = "Hello World"
21 print("Original String:", string_input)
22 print("Reversed String (Recursive):", reverse_string_recursive(string_input))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Programs\Python\Python314\python.exe "C:\Users\Jashwanth\AI coding\ai.py"
Enter a string: Swami
Reversed string: iimawS
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Programs\Python\Python314\python.exe "C:\Users\Jashwanth\AI coding\ai.py"
Original String: Hello World
Reversed String (Iterative): dlrow olleH
Reversed String (Recursive): dlrow olleH
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Programs\Python\Python314\python.exe "C:\Users\Jashwanth\AI coding\ai.py"
Original String: Hello World
Reversed String: dlrow olleH
PS C:\Users\Jashwanth\AI coding> & C:\Users\Jashwanth\AppData\Local\Programs\Python\Python314\python.exe "C:\Users\Jashwanth\AI coding\ai.py"
Original String: Hello World
Reversed String (Iterative): dlrow olleH
Original String: Hello World
Reversed String (Recursive): dlrow olleH
PS C:\Users\Jashwanth\AI coding>
```

**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.