

# Assignment-1.5

Name: V. Architha

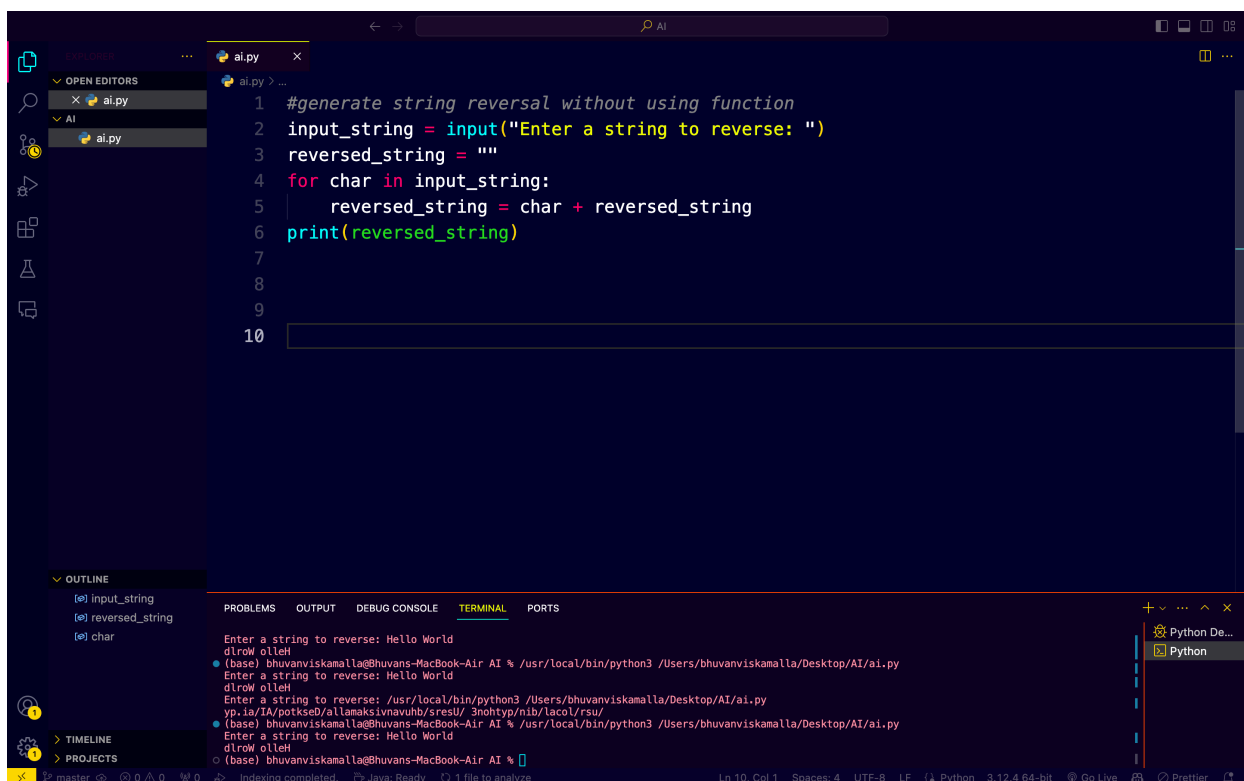
Roll.No:2306A91001

Batch-30

## Task-1

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

## CODE:



```
1 #generate string reversal without using function
2 input_string = input("Enter a string to reverse: ")
3 reversed_string = ""
4 for char in input_string:
5     reversed_string = char + reversed_string
6 print(reversed_string)
7
8
9
10
```

Enter a string to reverse: Hello World  
dlrow olleH  
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.la/IA/potkseB/allamaksiynavuhb/sresU/ 3nohtyp/nib/lacot/rsu/  
Enter a string to reverse: Hello World  
dlrow olleH

## **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 with a Python file named `ai.py` open. The code in the editor is as follows:

```
1 #improve redability and efficiency
2 input_string = input("Enter a string to reverse: ")
3 reversed_string = ''.join(reversed(input_string))
4 print(reversed_string) # Output: !dlroW ,olleH
5
6
7
8
9
```

The left sidebar shows the Explorer view with the file `ai.py` selected. The bottom panel shows the TERMINAL view with the following output:

```
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/potkseD/allamaksivnavuhb/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 indicates the file is at Line 6, Column 1, using UTF-8 encoding, and the Python interpreter is 3.12.4 64-bit.

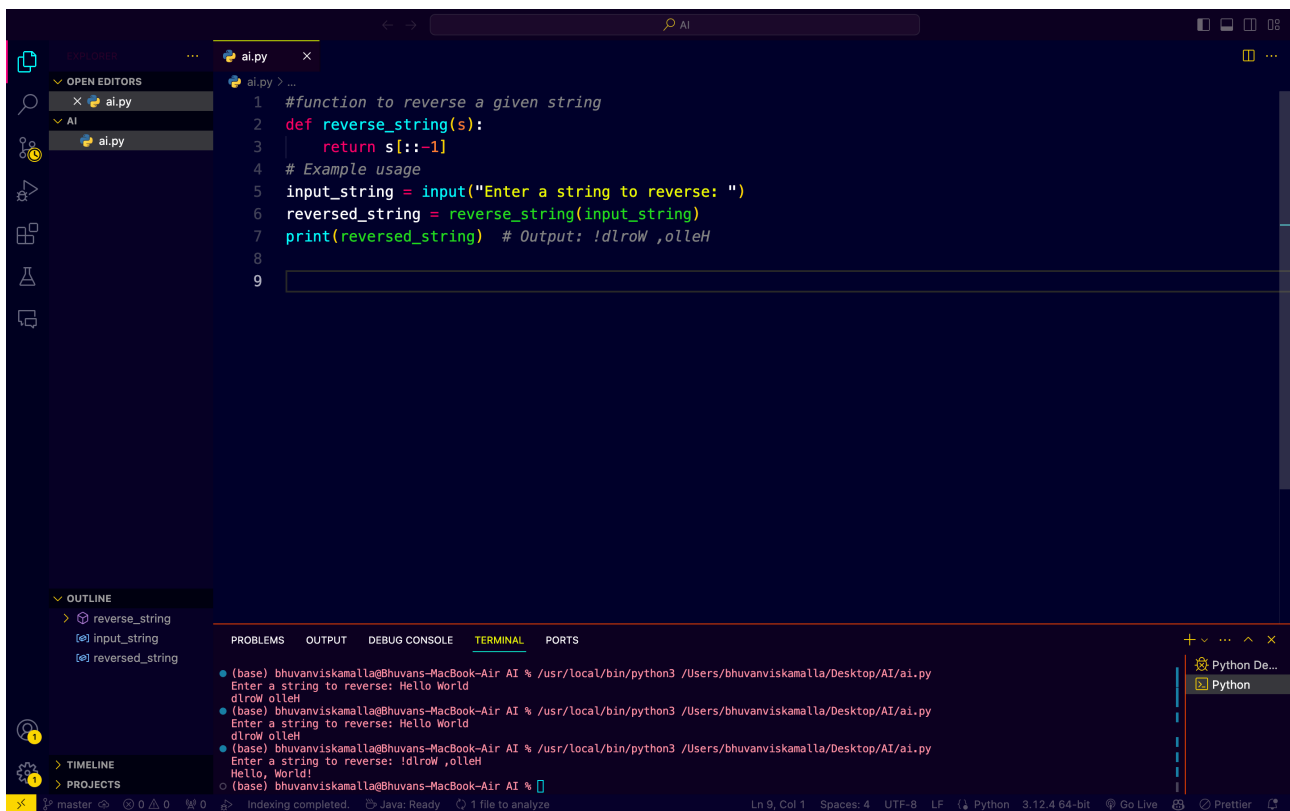
## OBSERVATION:

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 Visual Studio Code editor with a Python file named `ai.py` open. The code defines a function `reverse_string(s)` that returns the reversed string `s[::-1]`. It includes an example usage where the user is prompted to enter a string, and the program prints the reversed string. The terminal at the bottom shows the execution of the script, demonstrating the reversal of "Hello World" to "dlroW ,olleH".

```
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
8
9
```

Terminal Output:

```
(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 % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py
Enter a string to reverse: !dlroW ,olleH
Hello, World!
(base) bhuvanviskamalla@Bhuvans-MacBook-Air AI %
```

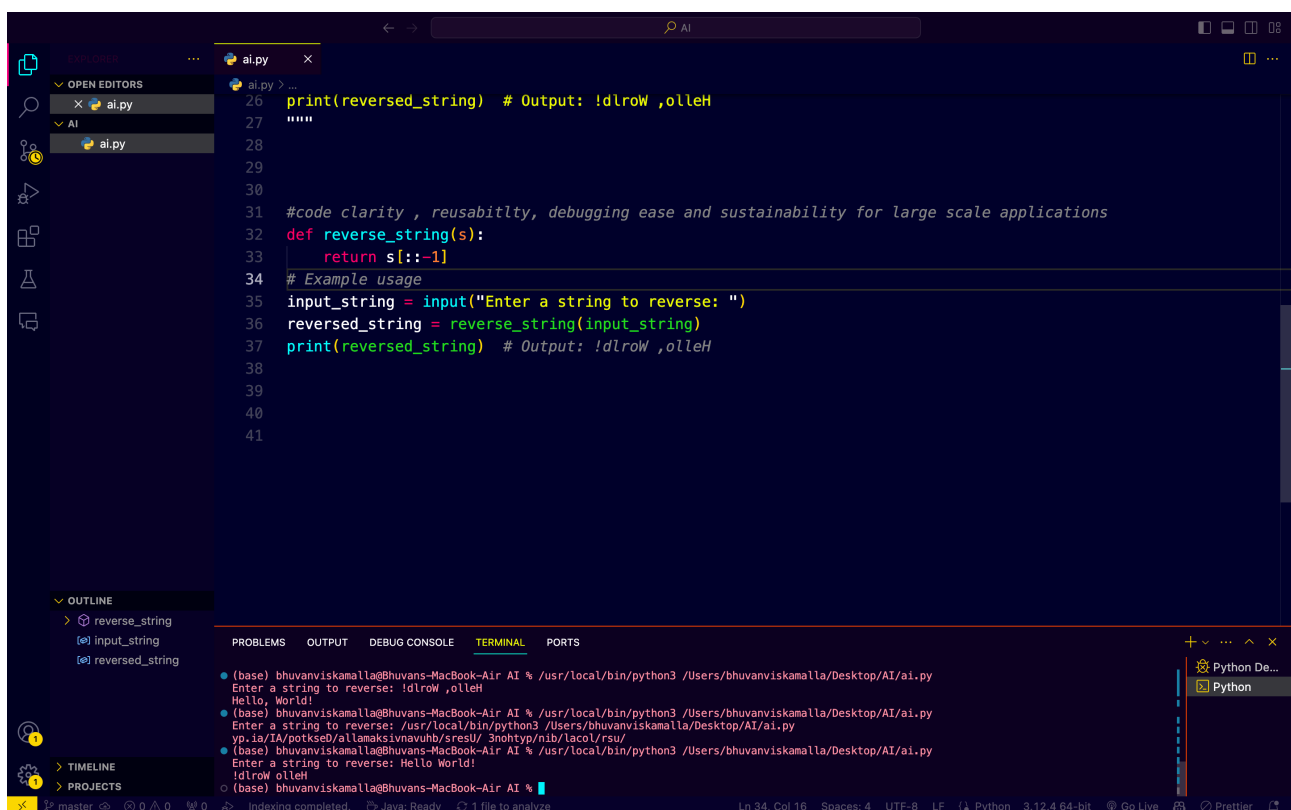
## OBSERVATION:

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:



```
26 print(reversed_string) # Output: !dlroW ,olleH
27
28
29
30
31 #code clarity , reusability, 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 screenshot shows a code editor with a Python file named `ai.py`. The code defines a function `reverse_string(s)` that returns the reversed string `s[::-1]`. It then takes user input, calls the function, and prints the result. The output shown in the terminal is `!dlroW ,olleH`. The editor interface includes a sidebar with 'OPEN EDITORS' and 'OUTLINE' panels, and a bottom status bar showing file encoding and language settings.

## **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:**

```
39
40
41 #loop based string reversal
42 def reverse_string(s):
43     reversed_str = ""
44     for char in s:
45         reversed_str = char + reversed_str
46     return reversed_str
47
48
49 input_string = input("Enter a string to reverse: ")
50 reversed_string = reverse_string(input_string)
51 print(reversed_string)
52
53
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

(base) bhuvanviskamalla@bhuvans-MacBook-Air AI % /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py  
Enter a string to reverse: /usr/local/bin/python3 /Users/bhuvanviskamalla/Desktop/AI/ai.py  
yp.ia/IA/potkseD/allamaksivnavuhb/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 %

## 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.