Lab Assignment-1.5

Mohammed Awez Ali

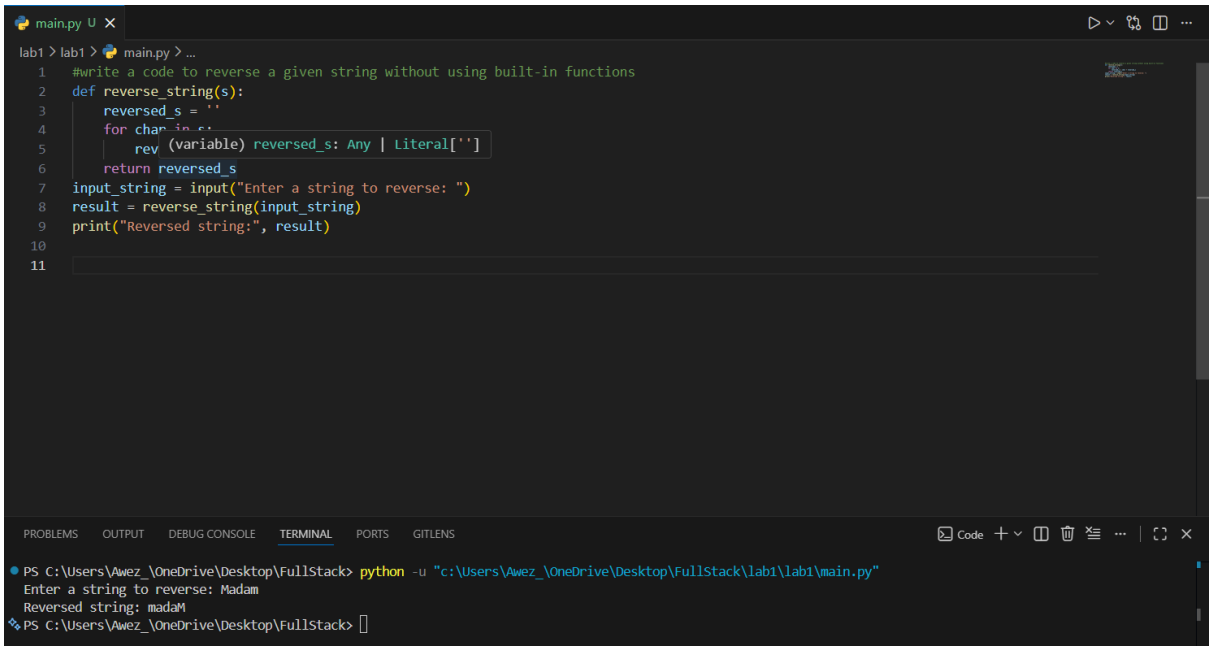
2303A51767

Batch-12

Task 1: AI-Generated Logic Without Modularization (String Reversal Without Functions)

Prompt- #write a code to reverse a given string without using built-in functions

Code:



Output:

```
Enter a string to reverse: Madam
Reversed string: madaM
```

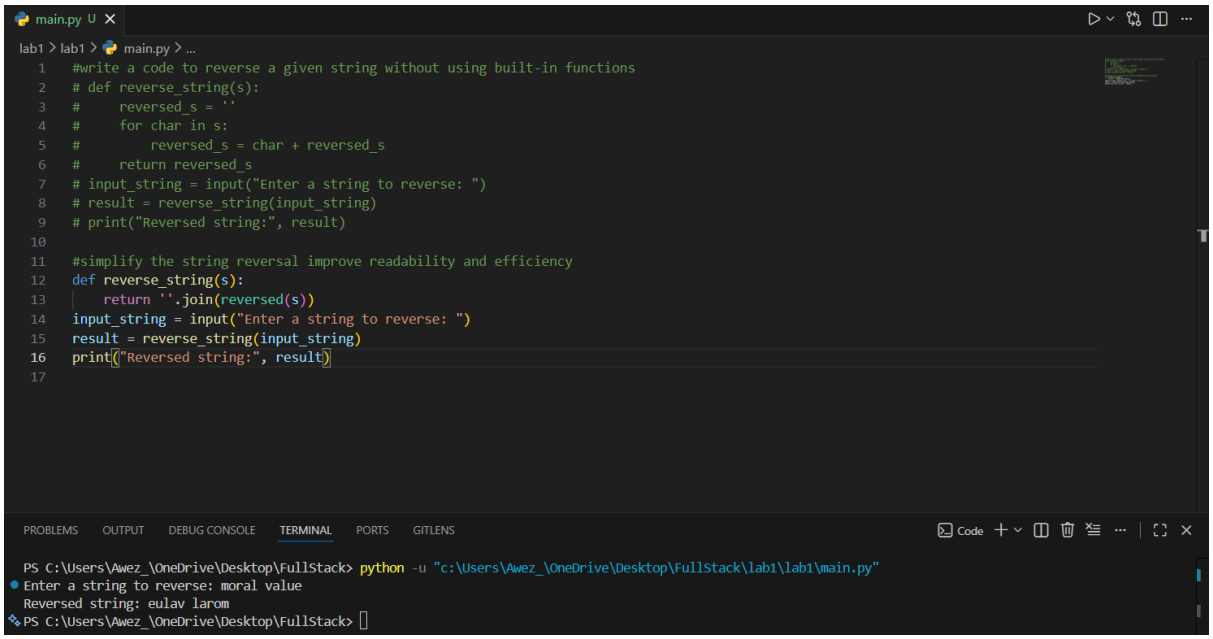
Explanation:

This code defines a function to reverse a string without using built-in functions. It initializes an empty string called reversed and iterates through each character in the input string s. For every character, it prepends the character to reversed, effectively building the reversed string one character at a time. After the loop completes, the function returns the reversed string. The commented-out lines show how to take user input, call the function, and print the reversed result, demonstrating a simple and manual approach to string reversal.

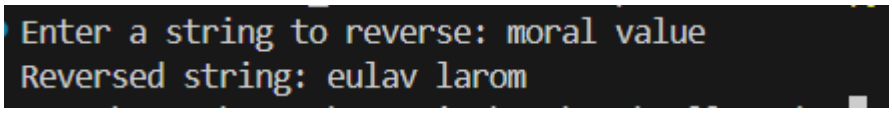
Task 2: Efficiency & Logic Optimization (Readability Improvement)

Prompt- #simplify the string reversal and improve readability and efficiency

Code:



Output:



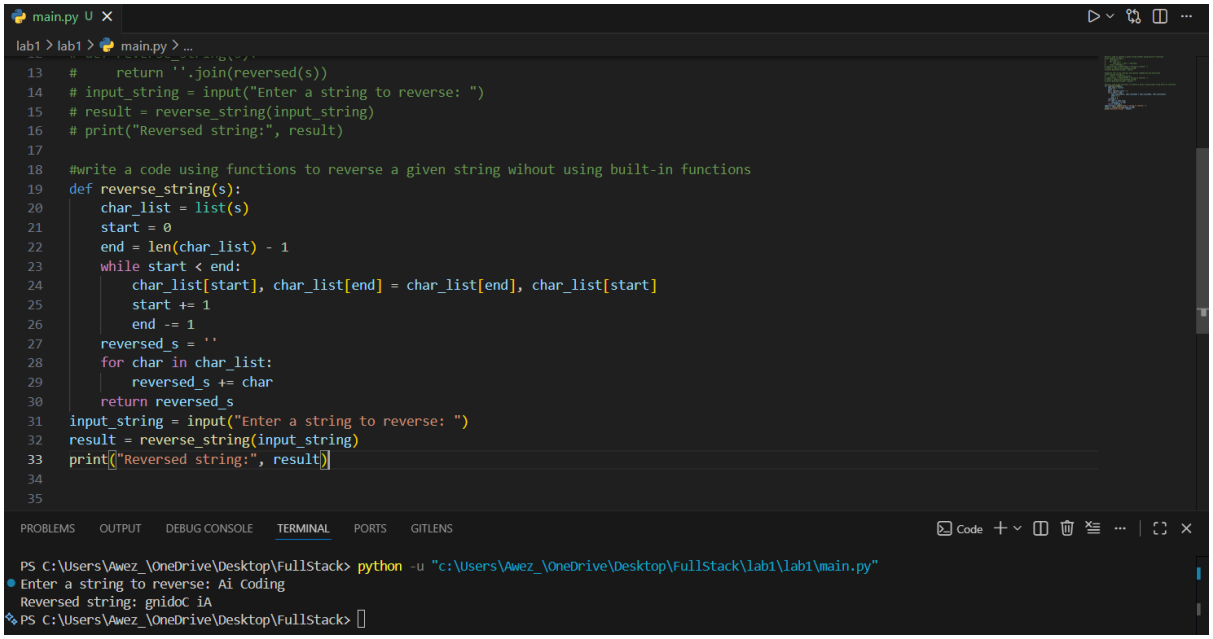
Explanation:

This code snippet demonstrates how to reverse a string using Python's built-in functions for simplicity and efficiency. The function reverse string(s) uses reversed(s) to create an iterator that yields the characters of the string in reverse order. The join () method then concatenates these characters into a new reversed string. The commented lines show how to prompt the user for input, call the function, and print the reversed result. This approach is concise, readable, and leverages Python's optimized internal methods for string manipulation.

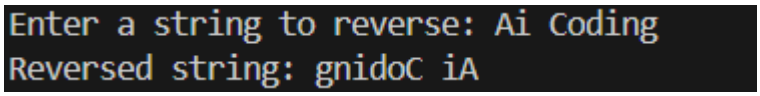
Task 3: Modular Design Using AI Assistance (String Reversal Using Functions)

Prompt- #write a code using functions to reverse a given string without using built-in functions

Code-



Output:



Explanation:

This code defines a function called reverse string that takes a string as input and reverses its characters. It first converts the string into a list of characters to allow swapping. Using a while loop, it swaps characters from the start and end of the list, moving towards the centre, effectively reversing the order. After reversing, it joins the characters back into a string. The program then prompts the user for input, reverses the entered string using the function, and prints the reversed result.

Task 4: Comparative Analysis – Procedural vs Modular Approach (With vs Without Functions)

Prompt- #compare both codes and give comparison table and short Analysis report.

Code-

Aspect	First Version (Concatenation)	Second Version (Swapping)
Approach	Builds reversed string by prepending characters	Swaps characters in a list in-place
Time Complexity	$O(n^2)$ (string concatenation in loop)	O(n) (in-place swaps, then join)
Space Complexity	O(n) (new string created)	O(n) (list of characters, new string for result)
Readability	Simple, concise	Slightly more complex, uses indices
Performance	Slower for long strings	Faster and more efficient for long strings

Analytical Report:

The first version reverses a string by iteratively prepending each character to a new string, which is simple but inefficient for long strings due to repeated string concatenation ($O(n^2)$ time). The second version converts the string to a list, swaps characters from both ends towards the center, and then joins the list back into a string, achieving O(n) time complexity. While the second approach is slightly more complex, it is significantly faster and more suitable for large inputs. Both methods use O(n) space, but the swapping method is generally preferred for performance-critical applications. For small strings, either method is acceptable, but for larger data, the in-place swapping approach is superior.

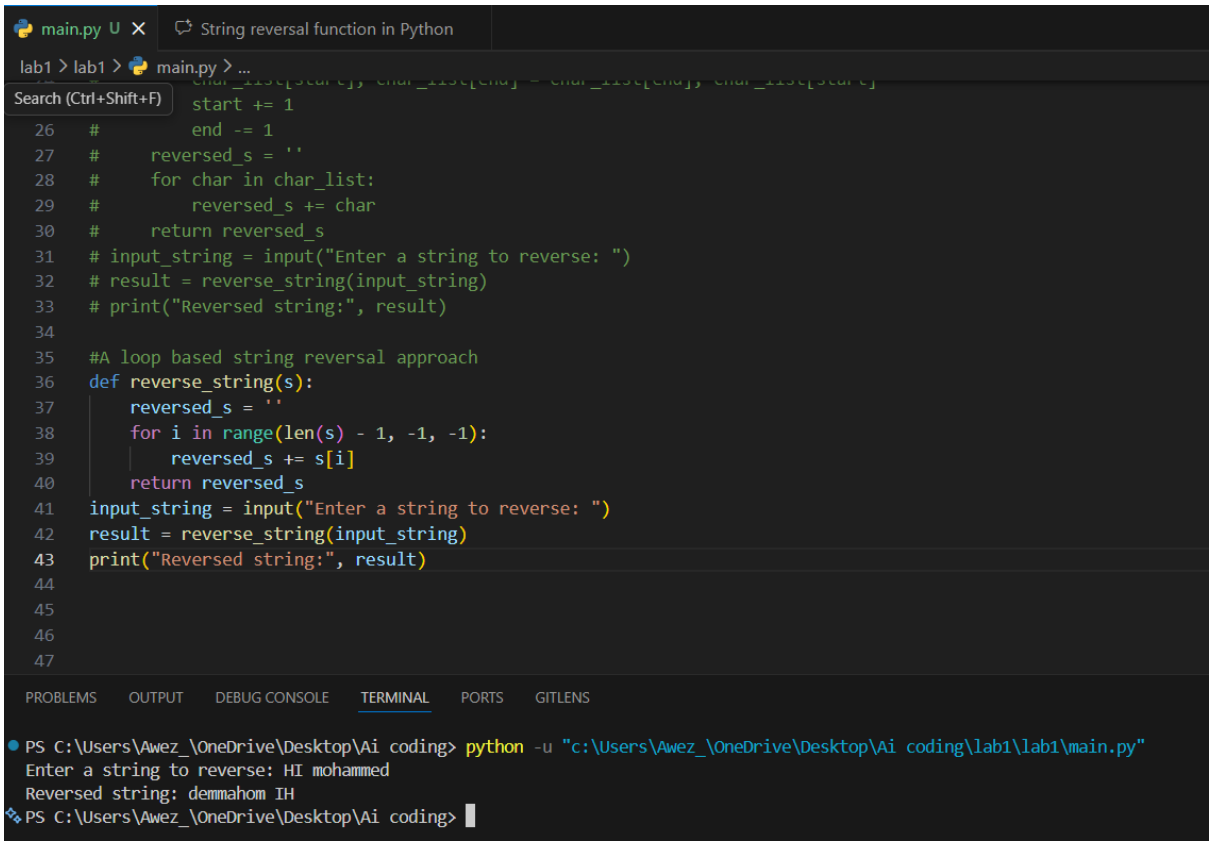
Explanation:

The first program (Task 1) is more reliable and readable for reversing strings. The second program (Task 3) uses functional programming but has a logical flaw with repeated characters, making it less robust. For practical use, Task 1's approach is preferred.

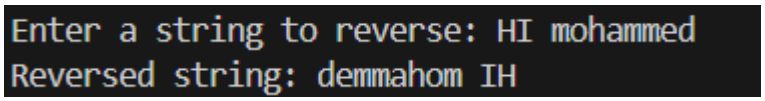
Task 5: AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches to String Reversal)

Prompt- #A loop based string reversal approach.

Code-



Output:



Explanation:

This function reverses a string by iterating over its characters from the end to the beginning using a for loop with a decreasing index. For each character, it appends it to a new string called reversed. After the loop completes, reversed contains the original string in reverse order. The program then prompts the user to enter a string, calls the reverse string function with the input, and prints the reversed result. This approach is straightforward and easy to understand, but it can be less efficient for very long strings due to repeated string concatenation.