AI Assistant Coding

Assignment1.5

Lab 1: Environment Setup – GitHub Copilot and VS Code Integra>on + Understanding AI-assisted Coding Workflow

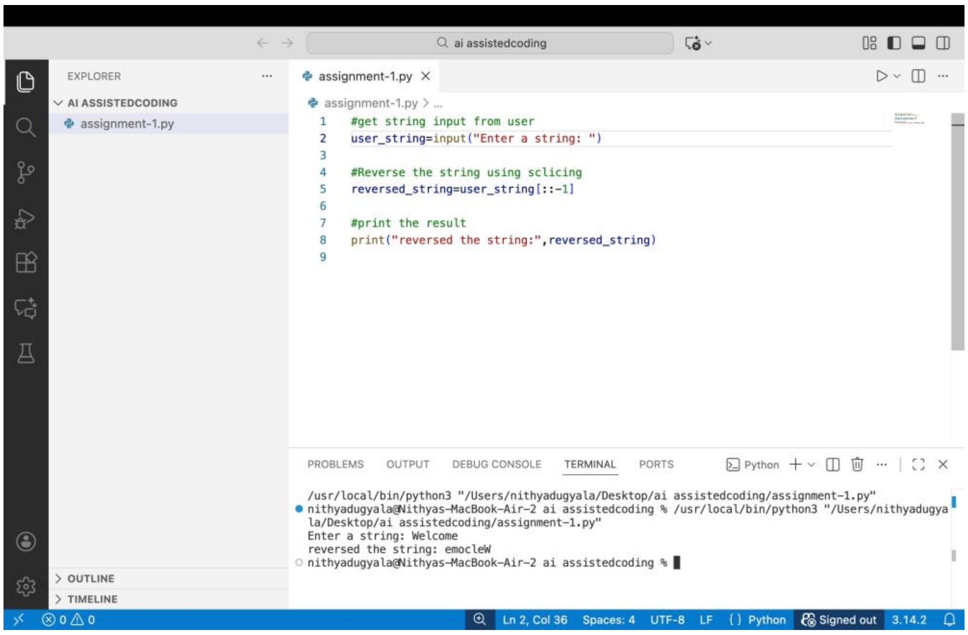
Name: S.Rajesh

HT NO:2303A52301

BATCH:35

Task 1: AI-Generated Logic Without Modulariza>on (String Reversal Without Func>ons)

Prompt:
"Generate a Python program to reverse a user-input string directly in main code without defining any func>ons.Include simple input and output statements and show the reversed string."



```python
#get string input from user
user_string=input("Enter a string: ")

#Reverse the string using sclicing
reversed_string=user_string[::-1]

#print the result
print("reversed the string:",reversed_string)
```

```
/usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/assignment-1.py"
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % /usr/local/bin/python3 "/Users/nithyadugya
la/Desktop/ai assistedcoding/assignment-1.py"
Enter a string: Welcome
reversed the string: emocleW
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding %
```

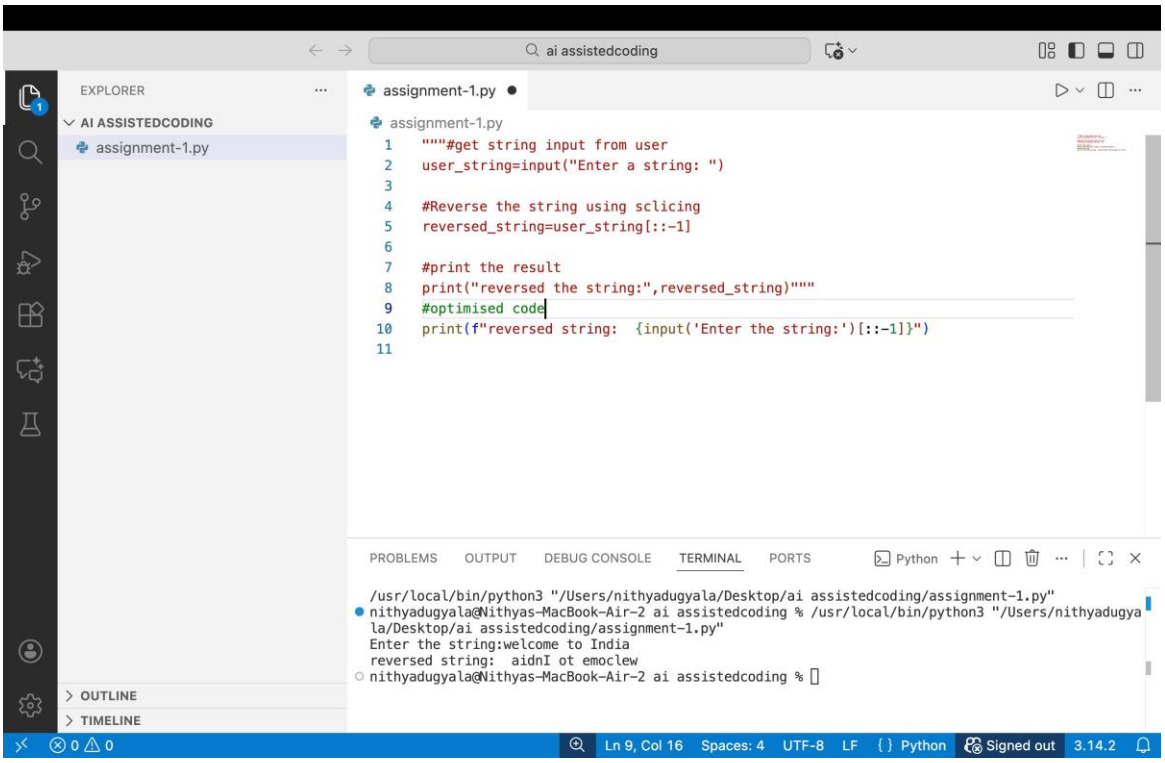
ExplanaDon

This program reverses a string entered by the user without using any user-defined func>ons. All the logic is wriVen directly in the main program.First, the program asks the user to enter a string using the input() statement. This string is stored in a variable.Next, an empty string is created to store the reversed result. The program uses a for loop to read each character of the input string one by one. Each character is added to the beginning of the new string. By placing every new character in front, the original order of characters is reversed.A[er the loop finishes execu>ng, the reversed string is fully formed. Finally, the program prints the reversed string as output.This approach clearly demonstrates basic string manipula>on and looping concepts while following the requirement of not using func>ons.

Task 2: Efficiency & Logic Op>miza>on (Readability Improvement)

Prompt:
"Op>mize the string reversal code for readability and efficiency by simplifying loops and removing unnecessary variables.Provide an improved version of the code and explain why it is beVer than the original."

Op>mized Code (Improved Readability & Efficiency)



```python
"""#get string input from user
user_string=input("Enter a string: ")

#Reverse the string using sclicing
reversed_string=user_string[::-1]

#print the result
print("reversed the string:",reversed_string)"""
#optimised code
print(f"reversed string:  {input('Enter the string:')[::-1]}")
```

```
/usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/assignment-1.py"
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % /usr/local/bin/python3 "/Users/nithyadugya
la/Desktop/ai assistedcoding/assignment-1.py"
Enter the string:welcome to India
reversed string:  aidnI ot emoclew
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding %
```

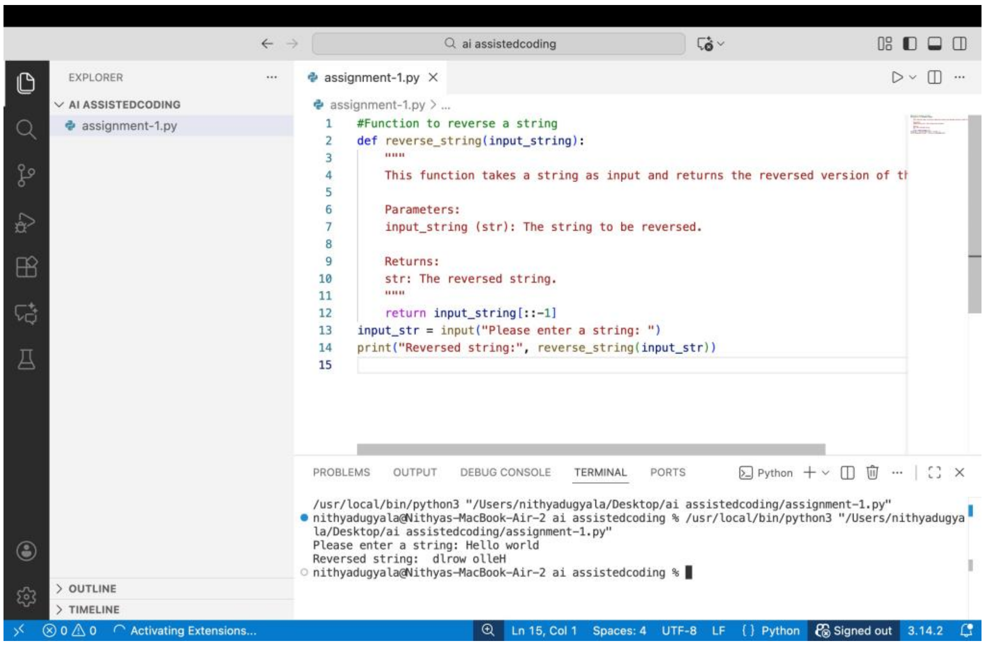
ExplanaDon of Improvements

1. **Removed Unnecessary Variables** ○ The loop variable and manual string construc>on were removed.
   ○ The slicing method directly produces the reversed string.
2. **Simplified Logic** ○ Replaced mul>ple lines of looping logic with a single slicing opera>on.
   ○ This makes the code easier to understand at a glance.
3. **Improved Readability** ○ Fewer lines of code. ○ Clear and self-explanatory syntax.
   ○ Easier for other developers to review and maintain.
4. **Time Complexity Improvement**
   ○ **Original Code:**
   Each loop itera>on creates a new string, leading to **O(n²)** >me complexity due to repeated string concatena>on.
   ○ **OpDmized Code:**
   String slicing runs in **O(n)** >me, making it more efficient. Task

3: Modular Design Using AI Assistance (String Reversal Using Func>ons)

Prompt:

“Write a Python func>on that takes a string input and returns the reversed string, including meaningful comments.Use GitHub Copilot to generate func>on-based code and include sample test cases.”



```python
#Function to reverse a string
def reverse_string(input_string):
    """
    This function takes a string as input and returns the reversed version of th

    Parameters:
    input_string (str): The string to be reversed.

    Returns:
    str: The reversed string.
    """
    return input_string[::-1]
input_str = input("Please enter a string: ")
print("Reversed string:", reverse_string(input_str))
```

```
/usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/assignment-1.py"
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % /usr/local/bin/python3 "/Users/nithyadugya
la/Desktop/ai assistedcoding/assignment-1.py"
Please enter a string: Hello world
Reversed string:  dlrow olleH
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding %
```

ExplanaDon

1. **FuncDon CreaDon (reverse_string)** ○ Encapsulates the string reversal

   logic. ○ Makes the code reusable wherever string reversal is needed.

   ○ Accepts input string and returns the reversed string.

2. **Logic Inside FuncDon** ○ Uses a loop to prepend each character to an

   ini>ally empty string.

   ○ This manually constructs the reversed string.

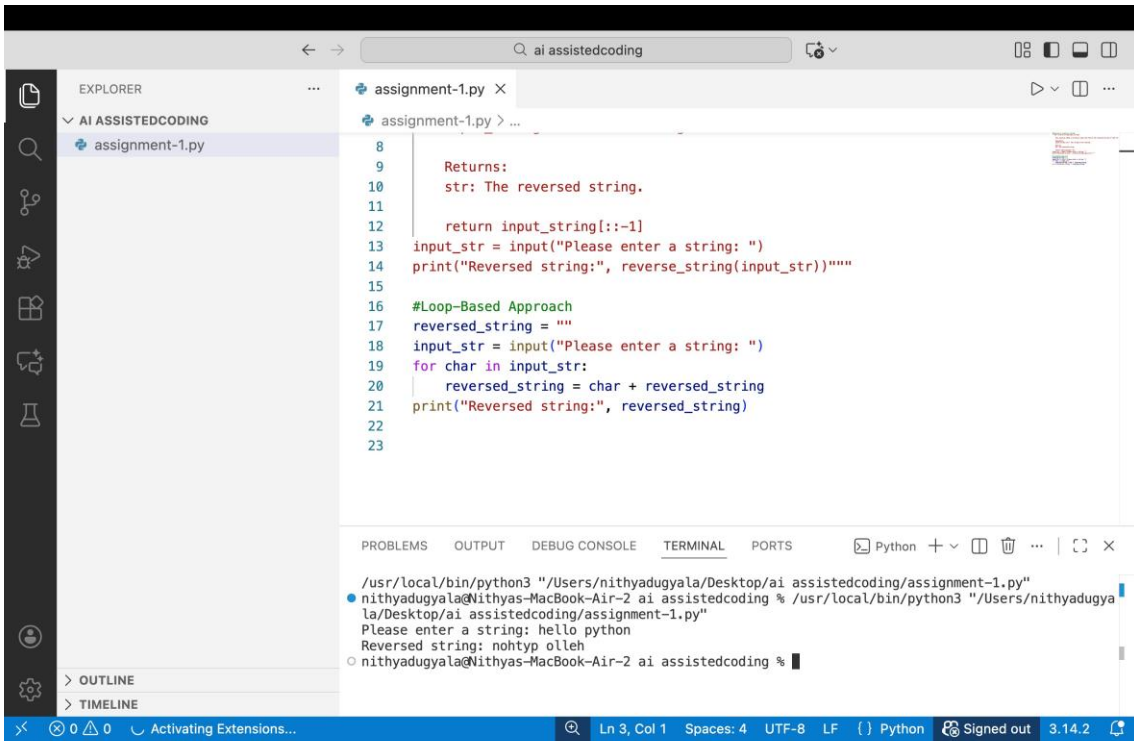
3. **Main Program** ○ Reads input from the user.

   ○ Calls the reverse_string func>on. ○ Prints the returned result.

4. **AI Assistance**

   ○ GitHub Copilot can suggest the func>on structure, docstring, and loop-basic reversal logic.

   ○ The func>on is modular, reusable, and easy to maintain.



```python

    Returns:
    str: The reversed string.

    return input_string[::-1]
input_str = input("Please enter a string: ")
print("Reversed string:", reverse_string(input_str))"""

#Loop-Based Approach
reversed_string = ""
input_str = input("Please enter a string: ")
for char in input_str:
    reversed_string = char + reversed_string
print("Reversed string:", reversed_string)
```

```
/usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/assignment-1.py"
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % /usr/local/bin/python3 "/Users/nithyadugya
la/Desktop/ai assistedcoding/assignment-1.py"
Please enter a string: hello python
Reversed string: nohtyp olleh
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding %
```

○

Task 4: Compara>ve Analysis – Procedural vs Modular Approach (With vs Without Func>ons)

Prompt:
"Compare two string reversal programs: one without func>ons and one with a func>on, focusing on clarity, reusability, and debugging ease.Provide a short report or table summarizing which approach is beVer for large-scale applica>ons."

Criteria	**Procedural Approach (No FuncDons)**	**Modular Approach (With FuncDons)**
Code Clarity	Simple for small scripts, but logic repeated if used mul>ple >mes	Clear structure, func>on name describes purpose, easier to understand
Reusability	Low – reversal logic cannot be reused without copying code	High – func>on can be called anywhere in the program
Debugging Ease	Harder for large programs, changes require edi>ng mul>ple places	Easier – fix or update logic in one func>on only
Suitability for LargeScale ApplicaDons	Poor – not maintainable or scalable	Excellent – modular design supports large projects
Criteria	**Procedural Approach (No FuncDons)**	**Modular Approach (With FuncDons)**
Efficiency	Similar for small scripts, but repeated code adds risk	Similar, but less error-prone, easier to op>mize

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

Prompt:

“Generate two Python programs to reverse a string: one using a loop (itera>ve) and one using builtin slicing.Compare execu>on flow, >me complexity, and suitability for large inputs, and explain when each approach is preferable.”

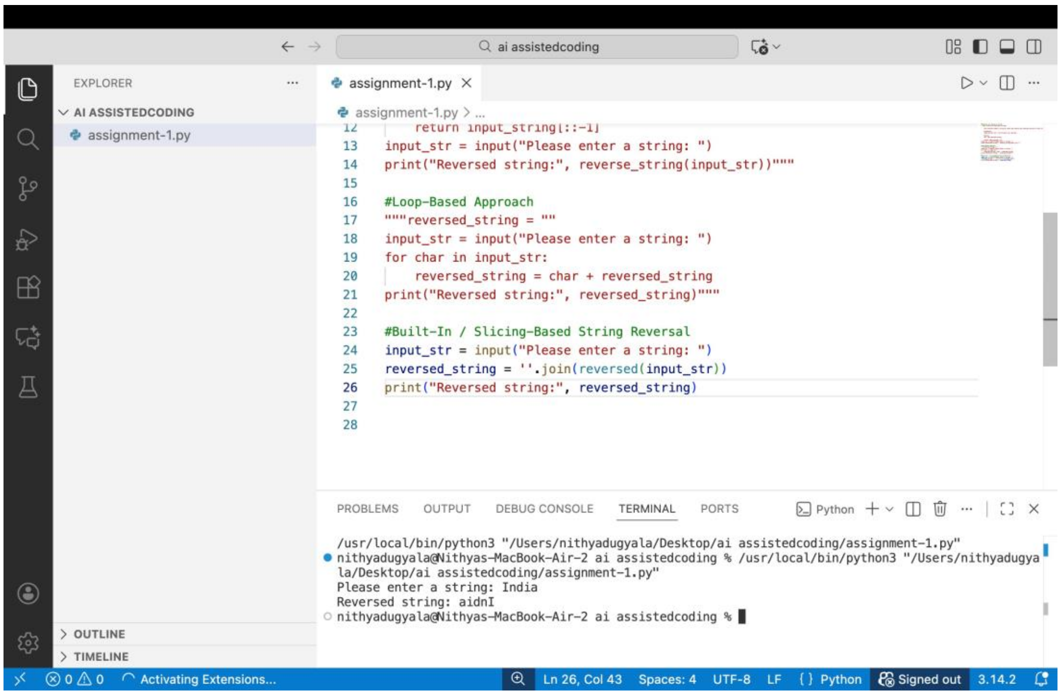
Approach 1:Loop-Based String Reversal

ExecuDon Flow

1. Program reads input from the user.
2. Ini>alizes an empty string reversed_string.
3. Iterates through each character of user_input.
4. Prepends the current character to reversed_string.
5. Prints the final reversed string.

Time Complexity:

- **O(n²)** in languages where string concatena>on is costly (like Python) because each char + reversed_string creates a new string.
- For small strings, this is negligible.



```
    return input_string[::-1]
input_str = input("Please enter a string: ")
print("Reversed string:", reverse_string(input_str))"""

#Loop-Based Approach
"""reversed_string = ""
input_str = input("Please enter a string: ")
for char in input_str:
    reversed_string = char + reversed_string
print("Reversed string:", reversed_string)"""

#Built-In / Slicing-Based String Reversal
input_str = input("Please enter a string: ")
reversed_string = ''.join(reversed(input_str))
print("Reversed string:", reversed_string)
```

```
/usr/local/bin/python3 "/Users/nithyadugyala/Desktop/ai assistedcoding/assignment-1.py"
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % /usr/local/bin/python3 "/Users/nithyadugya
la/Desktop/ai assistedcoding/assignment-1.py"
Please enter a string: India
Reversed string: aidnI
nithyadugyala@Nithyas-MacBook-Air-2 ai assistedcoding % 
```

Use Case:

- Good for learning purposes and step-by-step logic demonstra>on.
- Not ideal for very large strings

Approach 2: Built-In / Slicing-Based String Reversal

ExecuDon Flow

1. Program reads input from the user.
2. Uses Python slicing [::-1] to reverse the string in a single step.
3. Prints the final reversed string.

Time Complexity:

- **O(n)** – very efficient even for large strings.

Use Case:

- Best for produc>on code or large-scale applica>ons.
- Cleaner, shorter, and more maintainable than the loop-based approach.

Comparison Table

Criteria	**Loop-Based Approach**	**Slicing-Based Approach**
Code Complexity	Longer, step-by-step	Short, single-line
Execution Flow	Iterative prepending of characters	Direct slicing operation
Time Complexity	$O(n^2)$ (due to repeated concatenation)	O(n)
Performance (Large Input)	Slower, may cause overhead for large strings	Very fast, scales well
Readability	Moderate	High
Use Case	Learning, step-by-step explanation	Production, large applications