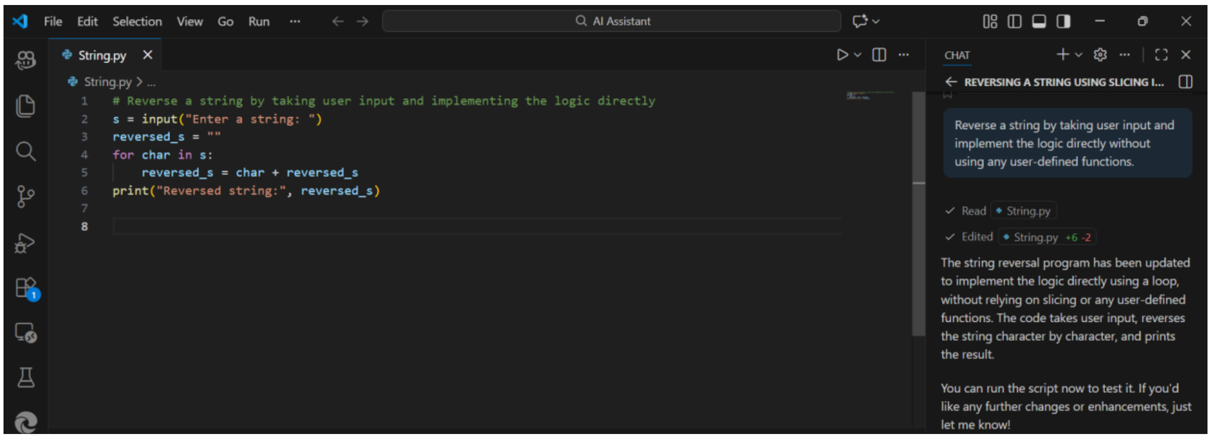
**Assignment – 01**
**2303A51927 (Batch 30)**

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

**PROMPT :**
reverse a given string by giving dynamic input without using user defined function.

**CODE:**



```python
# Reverse a string by taking user input and implementing the logic directly
s = input("Enter a string: ")
reversed_s = ""
for char in s:
    reversed_s = char + reversed_s
print("Reversed string:", reversed_s)

```

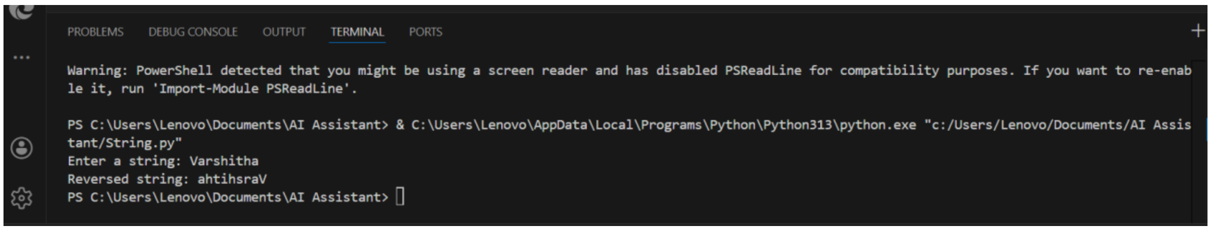
Reverse a string by taking user input and implement the logic directly without using any user-defined functions.

✓ Read String.py
✓ Edited String.py +6 -2

The string reversal program has been updated to implement the logic directly using a loop, without relying on slicing or any user-defined functions. The code takes user input, reverses the string character by character, and prints the result.

You can run the script now to test it. If you'd like any further changes or enhancements, just let me know!

**OUTPUT:**



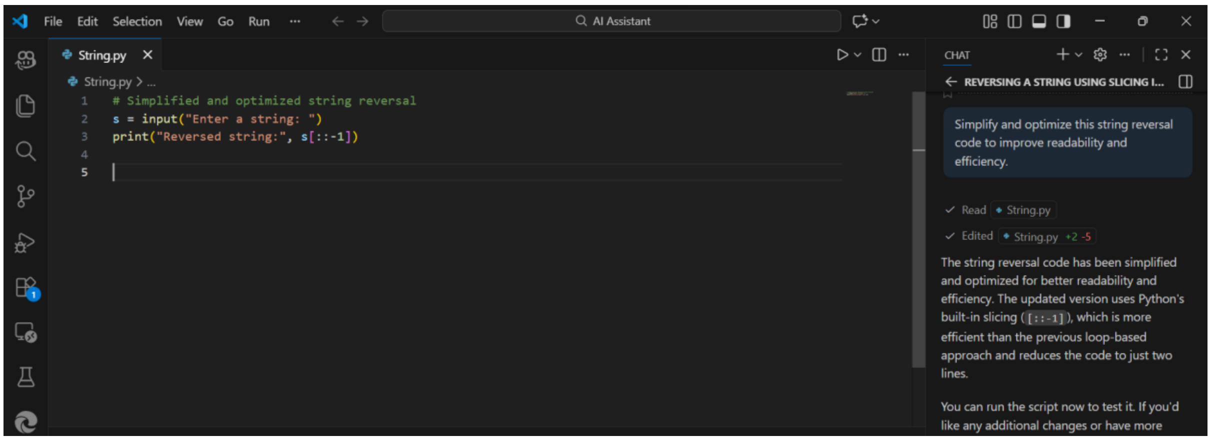
```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enab
le it, run 'Import-Module PSReadLine'.

PS C:\Users\Lenovo\Documents\AI Assistant> & C:\Users\Lenovo\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Lenovo/Documents/AI Assis
tant/String.py"
Enter a string: Varshitha
Reversed string: ahtihsraV
PS C:\Users\Lenovo\Documents\AI Assistant> 
```

**Justification :**
In this task, GitHub Copilot was utilized to assist in generating a Python program that performs string reversal without using any user-defined functions**.** A clear comment-based prompt was provided in the code editor, instructing Copilot to accept user input and reverse the given string. Based on the prompt, Copilot produced a straightforward solution implemented directly in the main program logic. The program takes a string from the user and reverses it by processing the characters sequentially in reverse order. The final reversed string is then printed as the output. This method is simple, readable, and well-suited for beginners learning basic string manipulation. Relevant screenshots were taken to demonstrate Copilot's code suggestions and the successful execution of the program.

## Task 2: Efficiency & Logic Optimization (Readability Improvement)

**PROMPT :**
Simplify this string reversal code.

**CODE :**



```python
# Simplified and optimized string reversal
s = input("Enter a string: ")
print("Reversed string:", s[::-1])
```

REVERSING A STRING USING SLICING I...

Simplify and optimize this string reversal code to improve readability and efficiency.

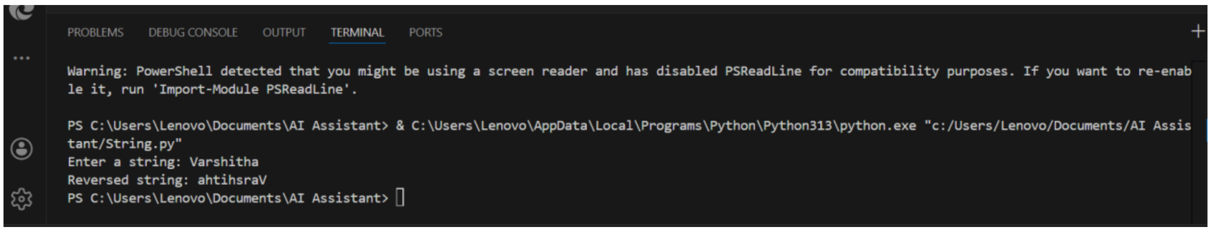
✓ Read String.py

✓ Edited String.py +2 -5

The string reversal code has been simplified and optimized for better readability and efficiency. The updated version uses Python's built-in slicing (`[::-1]`), which is more efficient than the previous loop-based approach and reduces the code to just two lines.

You can run the script now to test it. If you'd like any additional changes or have more

**OUTPUT:**



```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enab
le it, run 'Import-Module PSReadLine'.

PS C:\Users\Lenovo\Documents\AI Assistant> & C:\Users\Lenovo\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Lenovo/Documents/AI Assis
tant/String.py"
Enter a string: Varshitha
Reversed string: ahtihsraV
PS C:\Users\Lenovo\Documents\AI Assistant>
```

**Justification :**

After the initial version of the program was generated, the focus shifted to enhancing the code's clarity and efficiency. GitHub Copilot was guided using prompts such as "simplify this code" and "improve readability." Based on these instructions, Copilot produced a more concise solution that uses Python's slicing technique to reverse the string. This optimized version eliminates extra variables and avoids complex looping constructs. While both the original and revised implementations have the same time complexity of O(n), the improved version is more readable, concise, and better suited for practical applications. This task effectively demonstrates how AI tools can help refine and optimize existing code.

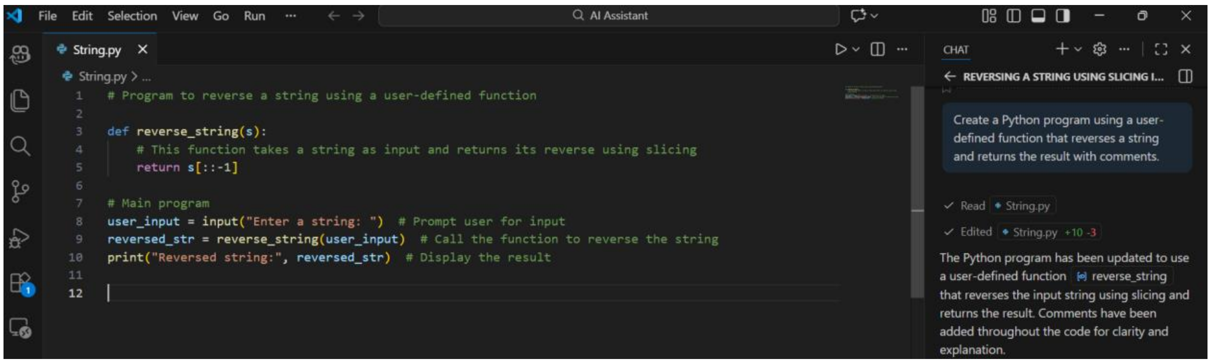
## Task 3:

Modular Design Using AI Assistance (String Reversal Using Functions)

**PROMPT :**

String reversal using functions by including meaningful comments.

**CODE:**



```python
# Program to reverse a string using a user-defined function

def reverse_string(s):
    # This function takes a string as input and returns its reverse using slicing
    return s[::-1]

# Main program
user_input = input("Enter a string: ")  # Prompt user for input
reversed_str = reverse_string(user_input)  # Call the function to reverse the string
print("Reversed string:", reversed_str)  # Display the result
```

REVERSING A STRING USING SLICING I...

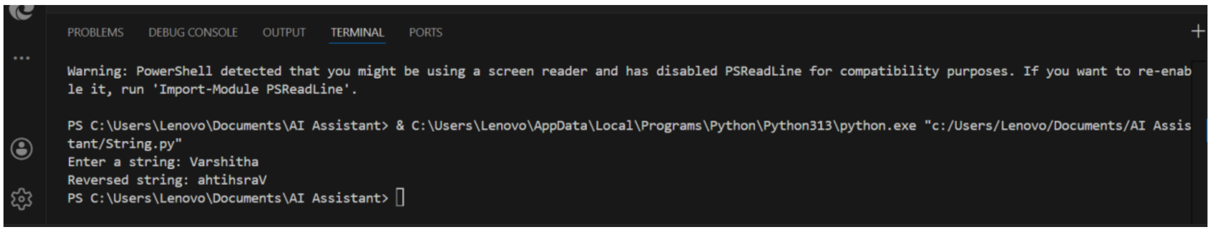
Create a Python program using a user-defined function that reverses a string and returns the result with comments.

✓ Read String.py

✓ Edited String.py +10 -3

The Python program has been updated to use a user-defined function reverse_string that reverses the input string using slicing and returns the result. Comments have been added throughout the code for clarity and explanation.

**OUTPUT:**



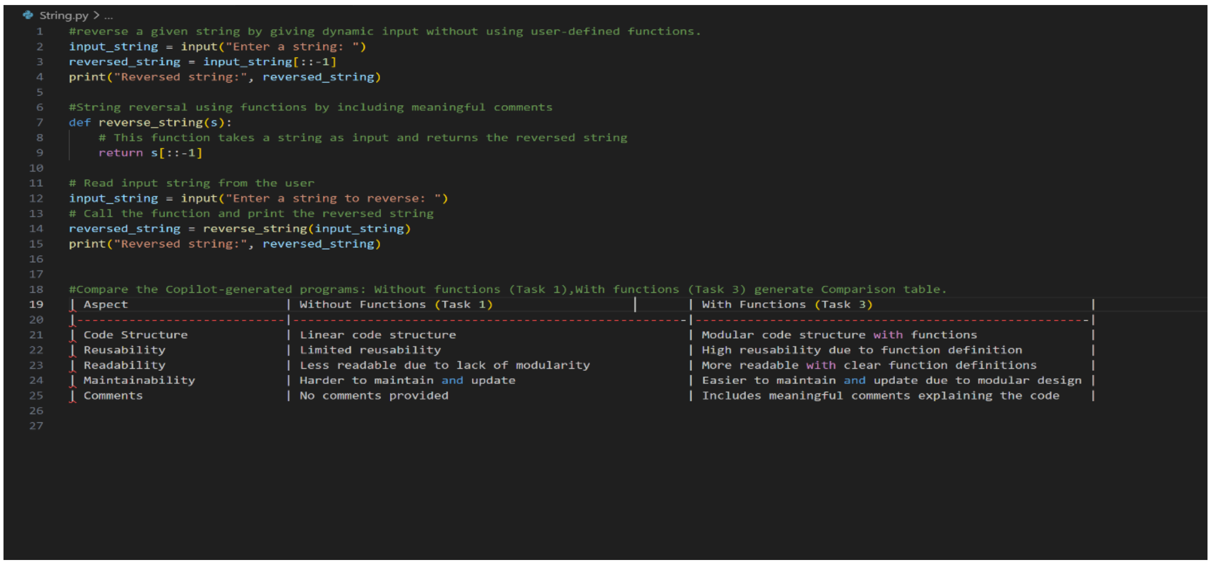
```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enab
le it, run 'Import-Module PSReadLine'.

PS C:\Users\Lenovo\Documents\AI Assistant> & C:\Users\Lenovo\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Lenovo/Documents/AI Assis
tant/String.py"
Enter a string: Varshitha
Reversed string: ahtihsraV
PS C:\Users\Lenovo\Documents\AI Assistant> 
```

**Justification :**
To enhance modularity and code reuse, the string reversal functionality was implemented using a user-defined function**.** GitHub Copilot was guided to generate a function that receives a string as input and returns its reversed version. The AI-produced function made use of Python's slicing method and included appropriate comments to explain each step. This function is then invoked from the main program to display the output. Adopting this modular approach improves maintainability and simplifies testing and debugging. It is particularly effective when the same logic must be applied across different parts of an application. Screenshots were collected to showcase Copilot's involvement in generating the function and its accompanying comments.

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

**PROMPT :**
Compare the Copilot-generated programs: Without functions (Task 1),With functions (Task 3) generate Comparison table.

**CODE and OUTPUT:**



```
#reverse a given string by giving dynamic input without using user-defined functions.
input_string = input("Enter a string: ")
reversed_string = input_string[::-1]
print("Reversed string:", reversed_string)

#String reversal using functions by including meaningful comments
def reverse_string(s):
    # This function takes a string as input and returns the reversed string
    return s[::-1]

# Read input string from the user
input_string = input("Enter a string to reverse: ")
# Call the function and print the reversed string
reversed_string = reverse_string(input_string)
print("Reversed string:", reversed_string)


#Compare the Copilot-generated programs: Without functions (Task 1),With functions (Task 3) generate Comparison table.
| Aspect                     | Without Functions (Task 1)                     | With Functions (Task 3)                          |
|----------------------------|------------------------------------------------|--------------------------------------------------|
| Code Structure             | Linear code structure                          | Modular code structure with functions            |
| Reusability                | Limited reusability                            | High reusability due to function definition      |
| Readability                | Less readable due to lack of modularity        | More readable with clear function definitions    |
| Maintainability            | Harder to maintain and update                  | Easier to maintain and update due to modular design |
| Comments                   | No comments provided                           | Includes meaningful comments explaining the code |
```

**Justification :**

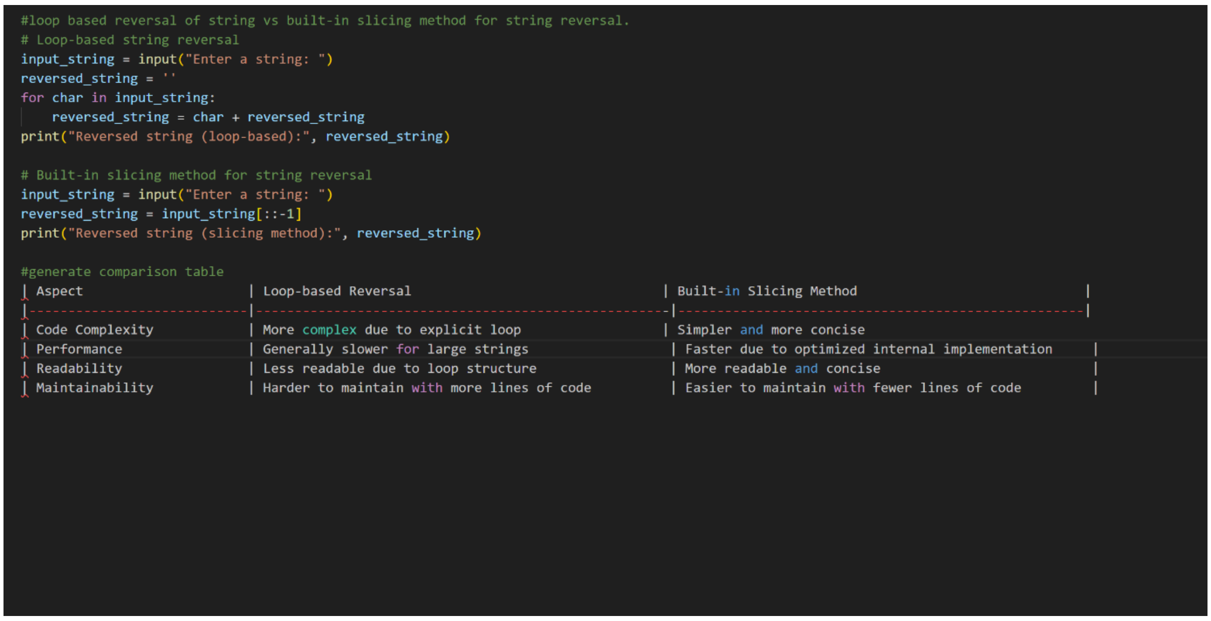
This task focused on evaluating two different programming styles: the procedural approach used in Task 1 and the modular approach applied in Task 3. The procedural style is straightforward and suitable for small, simple programs; however, it becomes harder to maintain and extend as the size of the application increases. In contrast, the modular approach organizes code more effectively by separating logic into functions, which enhances readability and simplifies debugging. It also allows the same logic to be reused across multiple parts of a program. As a result, modular programming is more appropriate for large-scale systems and collaborative development environments. This comparison provided valuable insight into the role of structured code design and informed decision-making in software development.

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

**PROMPT :**

loop based reversal of string vs built-in slicing method for string reversal, generate comparison table

**CODE and OUTPUT:**



```
#loop based reversal of string vs built-in slicing method for string reversal.
# Loop-based string reversal
input_string = input("Enter a string: ")
reversed_string = ''
for char in input_string:
    reversed_string = char + reversed_string
print("Reversed string (loop-based):", reversed_string)

# Built-in slicing method for string reversal
input_string = input("Enter a string: ")
reversed_string = input_string[::-1]
print("Reversed string (slicing method):", reversed_string)

#generate comparison table
| Aspect                      | Loop-based Reversal                                   | Built-in Slicing Method                             |
|-----------------------------|-------------------------------------------------------|-----------------------------------------------------|
| Code Complexity             | More complex due to explicit loop                     | Simpler and more concise
| Performance                 | Generally slower for large strings                     | Faster due to optimized internal implementation     |
| Readability                 | Less readable due to loop structure                    | More readable and concise                           |
| Maintainability             | Harder to maintain with more lines of code             | Easier to maintain with fewer lines of code         |
```

**Justification :**

In the concluding task, GitHub Copilot was leveraged to explore multiple algorithmic methods for reversing a string. One solution employed an iterative loop to reverse the string step by step, while another relied on Python's slicing technique. The loop-based method is useful for gaining a deeper understanding of how string manipulation works at a logical level. In contrast, the slicing-based solution is more concise, easier to read, and better suited for practical applications. Although both techniques have a time complexity of **O(n)**, the slicing approach benefits from Python's internal optimizations, resulting in better performance.