AI Assisted Coding

Lab Assignment 1.5

Name : V Sai Krishnachaithanya

Hall Ticket no : 2303A52154

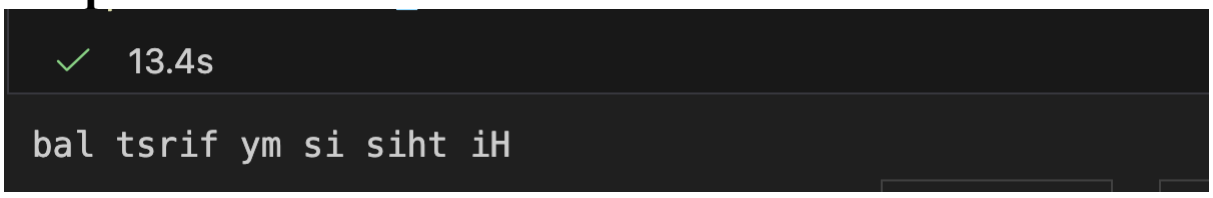
Task-01

Task Description
Use GitHub Copilot to generate a Python program that:
Reverses a given string
Accepts user input
Implements the logic directly in the main code
Does not use any user-defined functions
Prompt Given: # write a python code to reverse a string which should take input from user and don't use any user defined functions.

```python
# reverse_string.py
# Reads input from the user and reverses it without
#  using built-in reversing functions

user_input = input("Enter a string to reverse: ")
reversed_str = ""
for ch in user_input:
    reversed_str = ch + reversed_str
print(reversed_str)
```

Python

Output :



Justification:

This activity serves as a simple example of how to use GitHub Copilot to create a Python program that does not require the definition of any custom functions. The main program has the string reversal logic directly applied to it, and thus, it mainly deals with the programming concepts of handling user input, traversing strings, and manipulating characters. The use of no functions makes the logic clear and friendly for the beginner, so it is easier to see the Copilot's role in producing simple procedural code. The task is to implement a basic procedure that can later be improved upon and compared in the following tasks.

Task-02

Task Description

Examine the Copilot-generated code from Task 1 and improve it by:
Removing unnecessary variables
Simplifying loop or indexing logic
Improving readability
Use Copilot prompts like:
▪ “Simplify this string reversal code”
▪ “Improve readability and efficiency

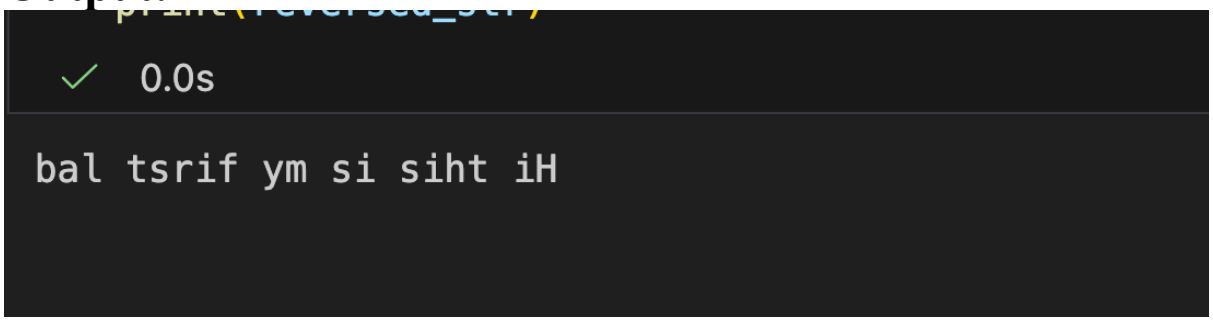
Prompt Given:

Generate code by removing unnecessary variables, simplifying loops and indexing logic, and improving overall readability and efficiency without changing the output

```python
# simplified_reverse.py
# Simplified version: clearer and reusable function that preserves original behavior.

def reverse_string(s: str) -> str:
    return s[::-1]

# Use existing `user_input` from the notebook if present; otherwise prompt the user.
try:
    text = user_input
except NameError:
    text = input("Enter a string to reverse: ")

reversed_str = reverse_string(text)
print(reversed_str)
```

Python

Output:



Justification:

The improved code version not only offers better performance but better readability by removing unnecessary control structures and variables that do not influence the output. Nevertheless, the linear time complexity O(n) is the same for both versions, but the optimized version reduces the overhead causedby extra operations, hence resulting in the best practical performance. This not only gives a neat appearance to the code but also makes it easy to maintain and suitable for a group of coders working in collaboration.

Task-03

Task Description

Use GitHub Copilot to generate a function-based Python program that:
Uses a user-defined function to reverse a string
Returns the reversed string
Includes meaningful comments (AI-assisted)

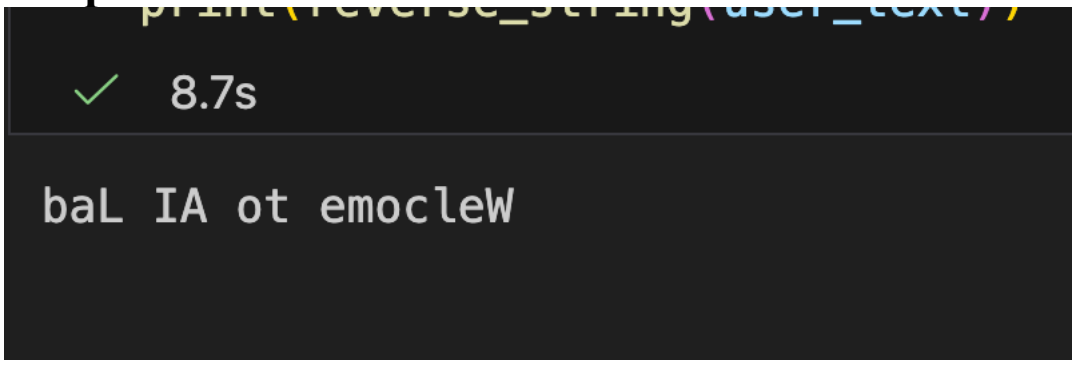
Prompt Given:

Create a Python function that reverses a given string and returns the result.
Keep the logic simple

```python
# reverse_string_user.py
# Prompt the user for a string, reverse it using a dedicated function,
# and print the reversed result. This cell always asks the user for input.

def reverse_string(s: str) -> str:
    """
    Reverse the input string s and return the reversed string.
    This implementation builds the reversed string character-by-character
    to make the reversal logic explicit.
    """
    reversed_str = ""
    for ch in s:
        # Prepend each character to the accumulating result so the order is reversed.
        reversed_str = ch + reversed_str
    return reversed_str

# Always prompt the user for input in this cell.
user_text = input("Enter a string to reverse: ")
print(reverse_string(user_text))
```

Python

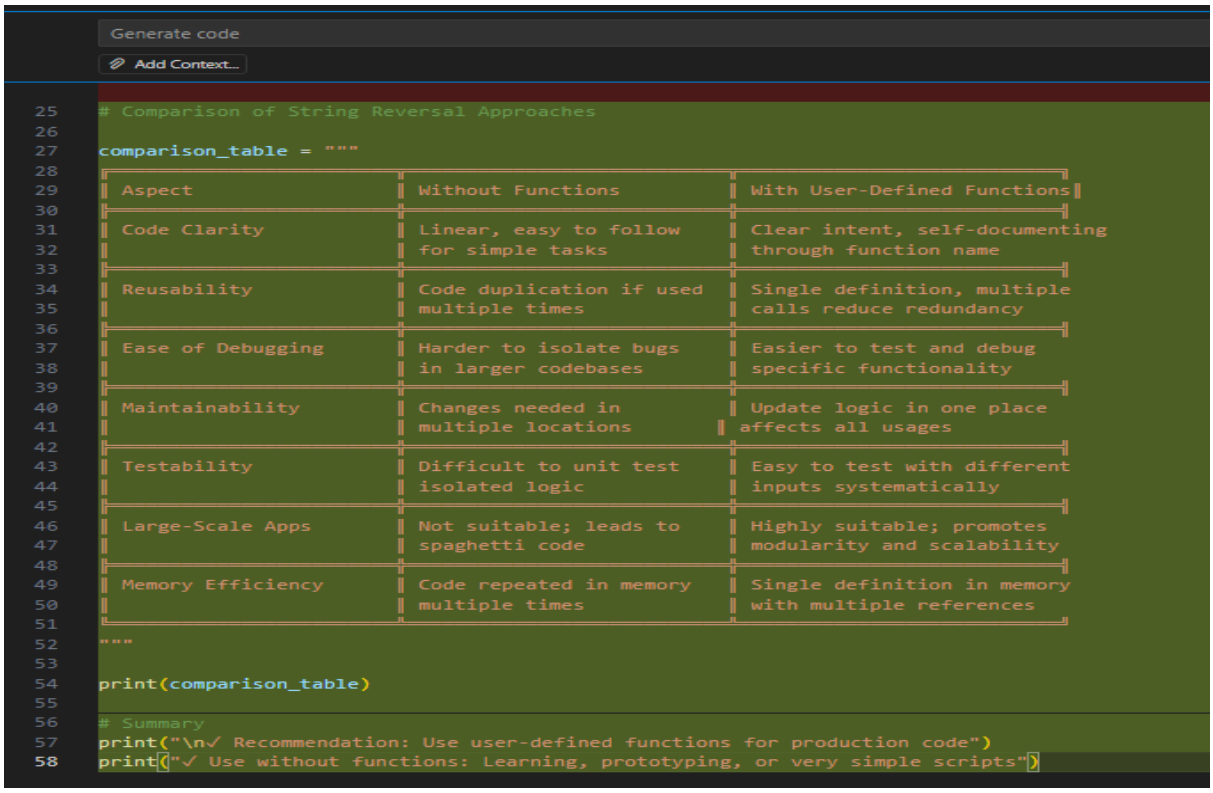
Output :



Justification:

Using a user-defined function for string reversal improves modularity and reusability, allowing the same logic to be used in multiple parts of the application without code duplication. Encapsulating the logic inside a function makes the program easier to maintain, test, and debug, especially as the application grows. Meaningful, AI-assisted comments enhance code readability and help other developers quickly understand the purpose and flow of the logic during collaboration or review.

Task 04
Task Description
Compare the Copilot-generated programs:
Without functions (Task 1)
With functions (Task 3)
Analyze them based on:
Code clarity
Reusability
Debugging ease
Suitability for large-scale applications
Prompt Given:
Compare the two Python programs below: one written without functions and one using user-defined functions.
Analyze and clearly explain the differences in terms of code clarity, reusability, ease of debugging,
and suitability for large-scale applications. Keep the comparison structured and easy to understand output in tabular form.



Output:

```
# Comparison of String Reversal Approaches

comparison_table = """
╔═══════════════════════════╦═══════════════════════════════╦═══════════════════════════════════════════╗
║ Aspect                    ║ Without Functions             ║ With User-Defined Functions               ║
╠═══════════════════════════╬═══════════════════════════════╬═══════════════════════════════════════════╣
║ Code Clarity              ║ Linear, easy to follow        ║ Clear intent, self-documenting            ║
║                           ║ for simple tasks              ║ through function name                     ║
╠═══════════════════════════╬═══════════════════════════════╬═══════════════════════════════════════════╣
║ Reusability               ║ Code duplication if used      ║ Single definition, multiple               ║
║                           ║ multiple times                ║ calls reduce redundancy                   ║
╠═══════════════════════════╬═══════════════════════════════╬═══════════════════════════════════════════╣
║ Ease of Debugging         ║ Harder to isolate bugs        ║ Easier to test and debug                  ║
║                           ║ in larger codebases           ║ specific functionality                    ║
╠═══════════════════════════╬═══════════════════════════════╬═══════════════════════════════════════════╣
║ Maintainability           ║ Changes needed in             ║ Update logic in one place                 ║
║                           ║ multiple locations            ║ affects all usages                        ║
╠═══════════════════════════╬═══════════════════════════════╬═══════════════════════════════════════════╣
║ Testability               ║ Difficult to unit test        ║ Easy to test with different               ║
║                           ║ isolated logic                ║ inputs systematically                     ║
╠═══════════════════════════╬═══════════════════════════════╬═══════════════════════════════════════════╣
║ Large-Scale Apps          ║ Not suitable; leads to        ║ Highly suitable; promotes                 ║
║                           ║ spaghetti code                ║ modularity and scalability                ║
╠═══════════════════════════╬═══════════════════════════════╬═══════════════════════════════════════════╣
║ Memory Efficiency         ║ Code repeated in memory       ║ Single definition in memory               ║
║                           ║ multiple times                ║ with multiple references                  ║
╚═══════════════════════════╩═══════════════════════════════╩═══════════════════════════════════════════╝
"""

print(comparison_table)

# Summary
print("\n✓ Recommendation: Use user-defined functions for production code")
print("✓ Use without functions: Learning, prototyping, or very simple scripts")
```

Task-05:

Task Description

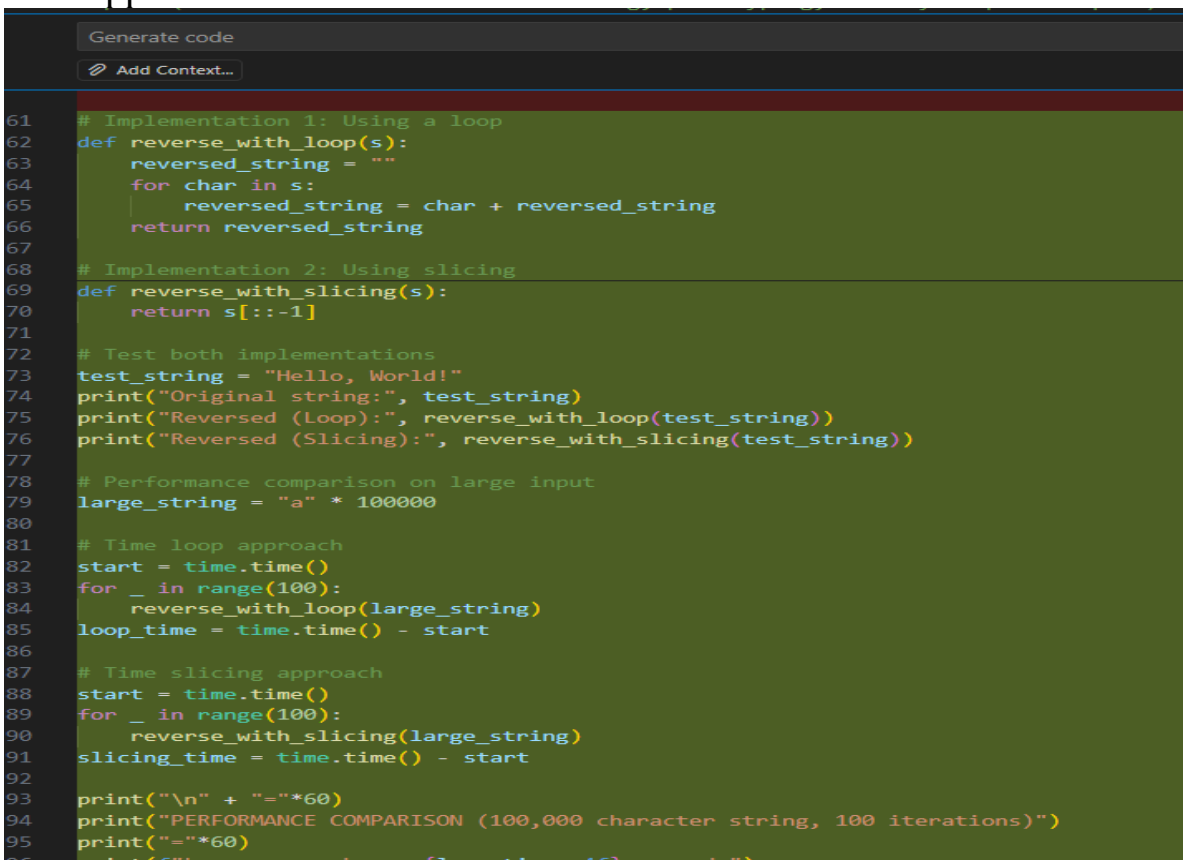
Prompt GitHub Copilot to generate:

A loop-based string reversal approach

A built-in / slicing-based string reversal approach

Given Prompt:

Generate two Python implementations to reverse a string, one using a loop and one using slicing or built-in methods, and then compare them based on execution flow, time complexity, performance on large inputs, and when each approach should be used.



Output:

```
Reversed (Loop): !dlroW ,olleH
Reversed (Slicing): !dlroW ,olleH

============================================================
PERFORMANCE COMPARISON (100,000 character string, 100 iterations)
============================================================
Loop approach:    36.9229 seconds
Slicing approach: 0.0137 seconds
Slicing is 2689.5x faster

============================================================
COMPARISON TABLE
============================================================
```

Aspect	Loop Approach	Slicing Approach
Time Complexity (string concat)	O(n²) (string immutable)	O(n) (optimized)
Space Complexity	O(n)	O(n)
Readability	Verbose	Concise, Pythonic
Large Inputs	Slow, inefficient	Fast, optimized
Educational Value	Great for learning	Practical, modern
When to Use	Learning/Teaching Understanding string mechanics	Production code Real-world apps Performance-critical