AI-Assisted String Reversal Using GitHub Copilot

Name: P.Sathwik Goud

Batch : 12

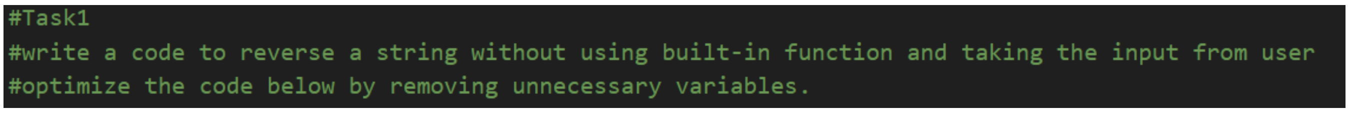
Rollno: 2303A51780

Task 1: AI-Generated Logic Without Modularization

Scenario

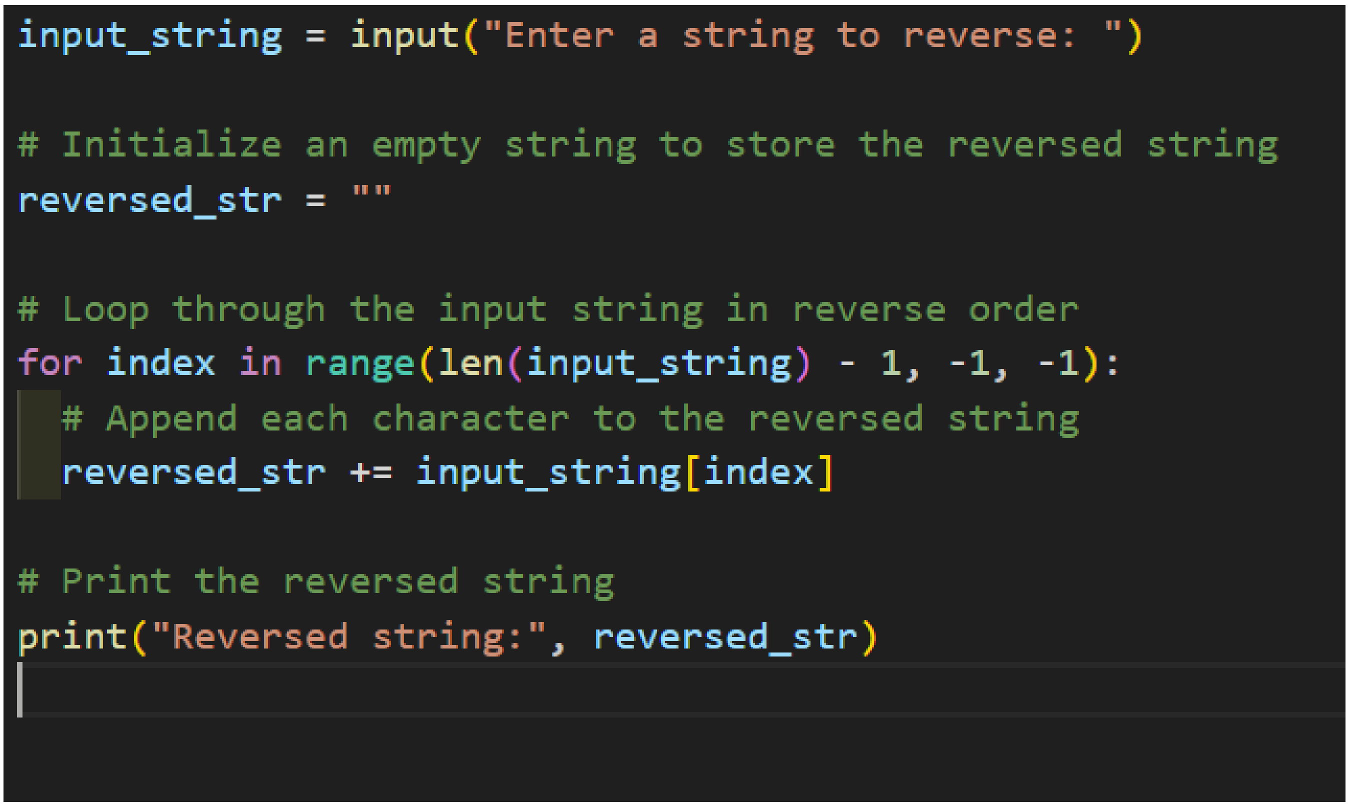
A basic text-processing utility is developed for a messaging application.

Copilot Prompt Used



```
#Task1
#write a code to reverse a string without using built-in function and taking the input from user
#optimize the code below by removing unnecessary variables.
```

Code :



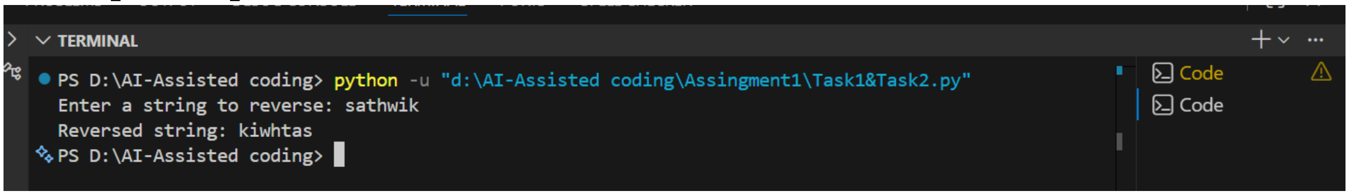
```python
input_string = input("Enter a string to reverse: ")

# Initialize an empty string to store the reversed string
reversed_str = ""

# Loop through the input string in reverse order
for index in range(len(input_string) - 1, -1, -1):
    # Append each character to the reversed string
    reversed_str += input_string[index]

# Print the reversed string
print("Reversed string:", reversed_str)
```

Sample Output



```
PS D:\AI-Assisted coding> python -u "d:\AI-Assisted coding\Assingment1\Task1&Task2.py"
Enter a string to reverse: sathwik
Reversed string: kiwhtas
PS D:\AI-Assisted coding> 
```

Explanation

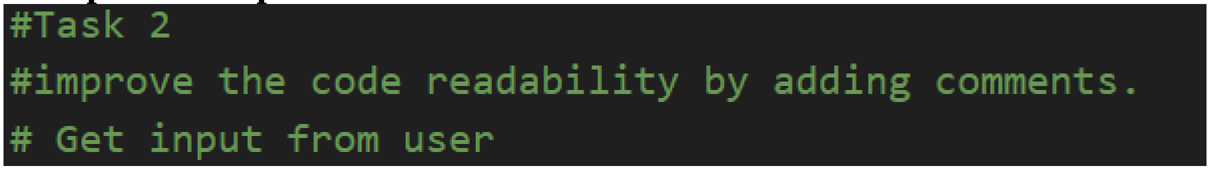
The program reads user input and reverses the string using a loop without defining any functions. The logic is implemented directly in the main code.

Task 2: Efficiency & Logic Optimization

Scenario

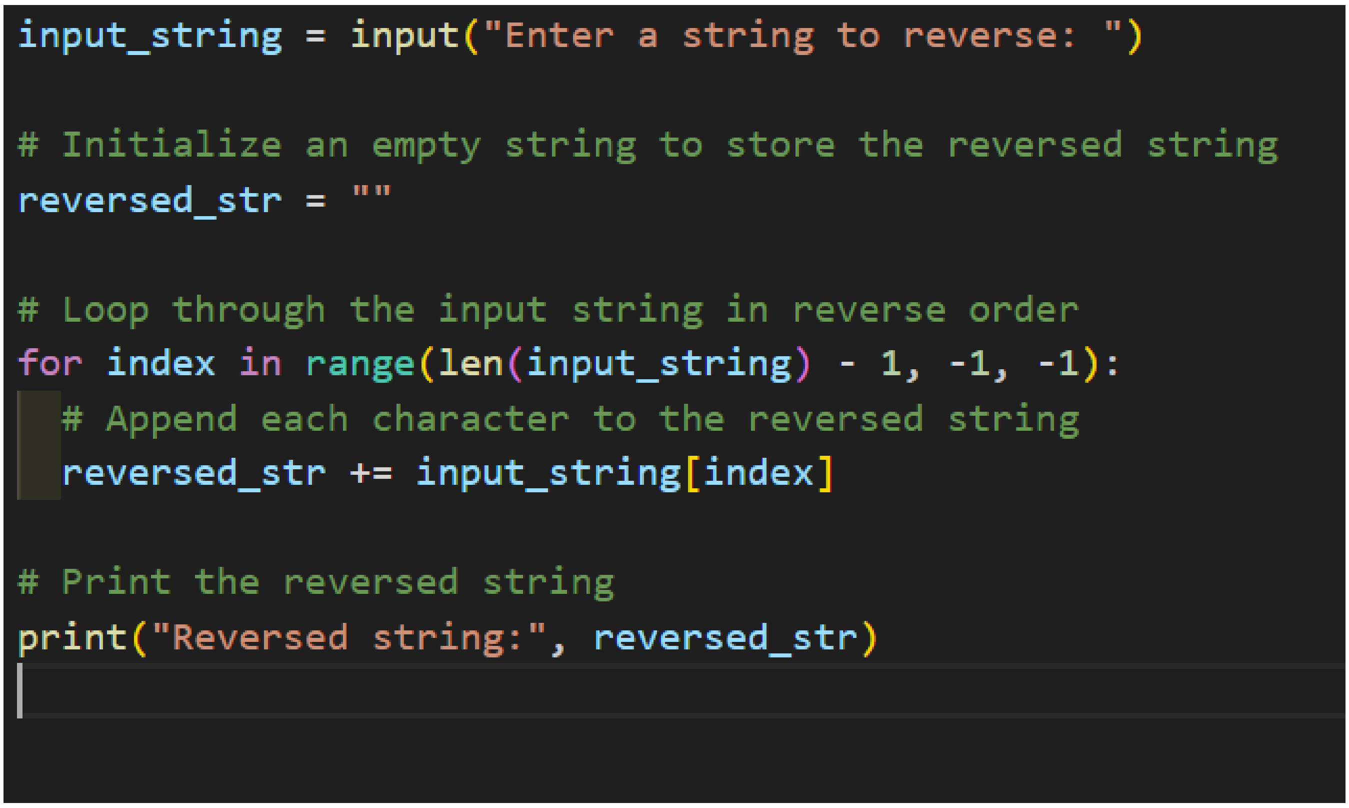
The code is reviewed for readability and efficiency.

Copilot Prompt Used



```
#Task 2
#improve the code readability by adding comments.
# Get input from user
```

Code :



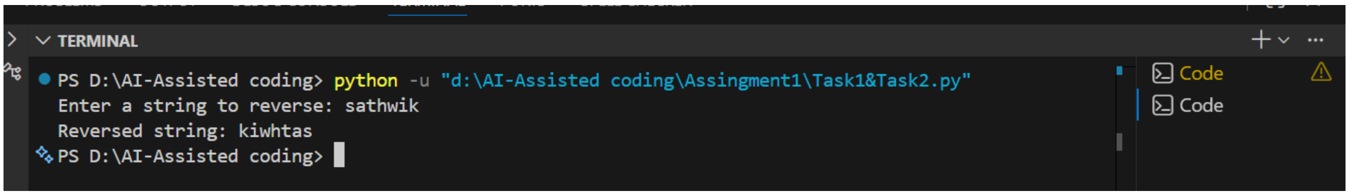
```python
input_string = input("Enter a string to reverse: ")

# Initialize an empty string to store the reversed string
reversed_str = ""

# Loop through the input string in reverse order
for index in range(len(input_string) - 1, -1, -1):
    # Append each character to the reversed string
    reversed_str += input_string[index]

# Print the reversed string
print("Reversed string:", reversed_str)
```

Sample Output



```
PS D:\AI-Assisted coding> python -u "d:\AI-Assisted coding\Assingment1\Task1&Task2.py"
Enter a string to reverse: sathwik
Reversed string: kiwhtas
PS D:\AI-Assisted coding>
```

Explanation

The slicing method simplifies the logic and removes unnecessary variables and loops.

The slicing method is faster in practice due to internal optimizations in Python.

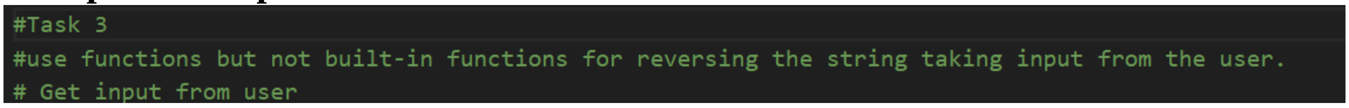
Readability is significantly improved.

Task 3: Modular Design Using Functions

Scenario

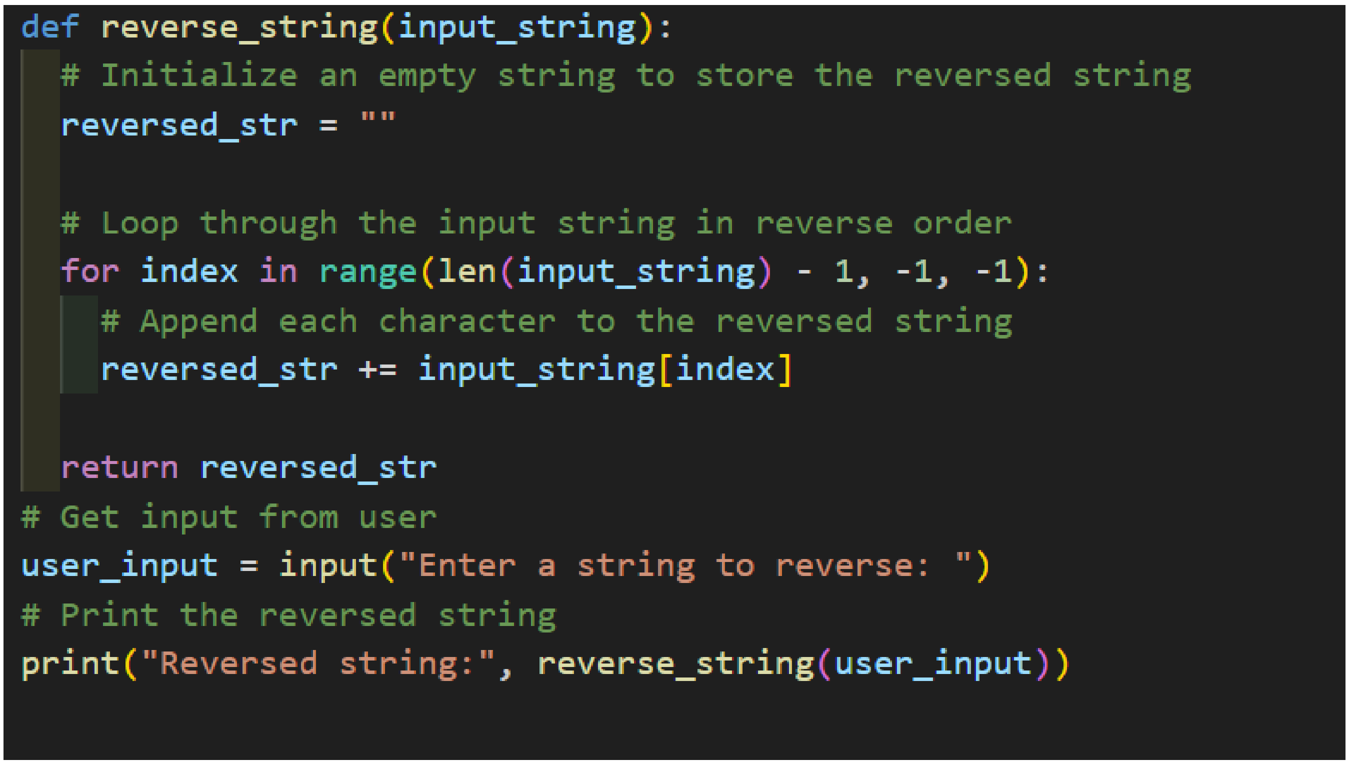
The string reversal logic is reused in multiple parts of an application

Copilot Prompt Used



```
#Task 3
#use functions but not built-in functions for reversing the string taking input from the user.
# Get input from user
```

Code :

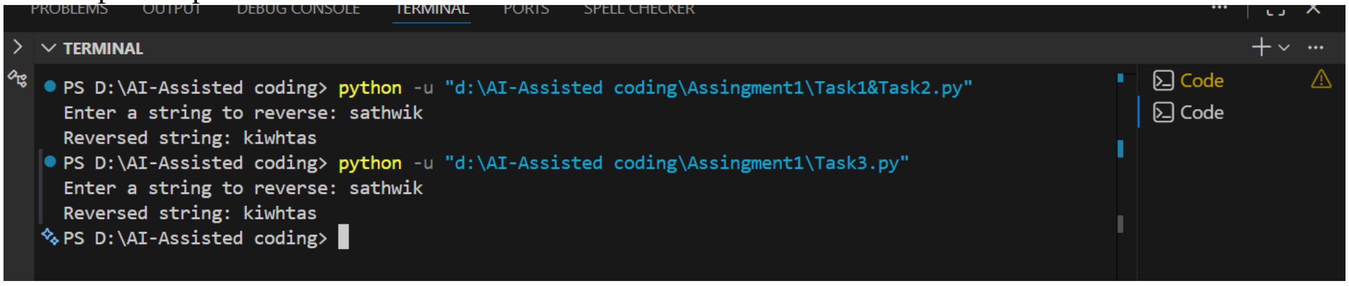


```python
def reverse_string(input_string):
    # Initialize an empty string to store the reversed string
    reversed_str = ""

    # Loop through the input string in reverse order
    for index in range(len(input_string) - 1, -1, -1):
        # Append each character to the reversed string
        reversed_str += input_string[index]

    return reversed_str
# Get input from user
user_input = input("Enter a string to reverse: ")
# Print the reversed string
print("Reversed string:", reverse_string(user_input))
```

Sample Output



```
PS D:\AI-Assisted coding> python -u "d:\AI-Assisted coding\Assingment1\Task1&Task2.py"
Enter a string to reverse: sathwik
Reversed string: kiwhtas
PS D:\AI-Assisted coding> python -u "d:\AI-Assisted coding\Assingment1\Task3.py"
Enter a string to reverse: sathwik
Reversed string: kiwhtas
PS D:\AI-Assisted coding> 
```

Explanation

Using a function improves reusability, readability, and maintainability. The logic can now be reused across different modules.

Task 4: Comparative Analysis – With vs Without Functions

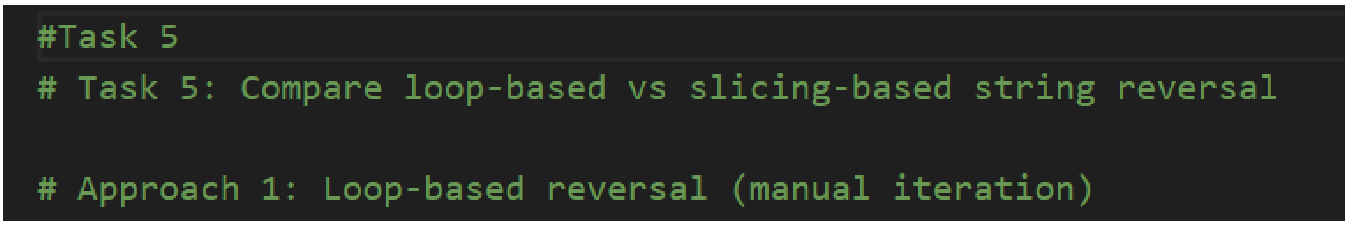
Aspect	Without Functions	With Functions
Code Clarity	Less organized	More structured
Reusability	Cannot be reused	Easily reusable
Debugging	Harder	Easier
Scalability	Not suitable	Suitable
Best Use	Small scripts	Large applications

Conclusion

Function-based design is preferred for large-scale applications due to better organization and reusability.

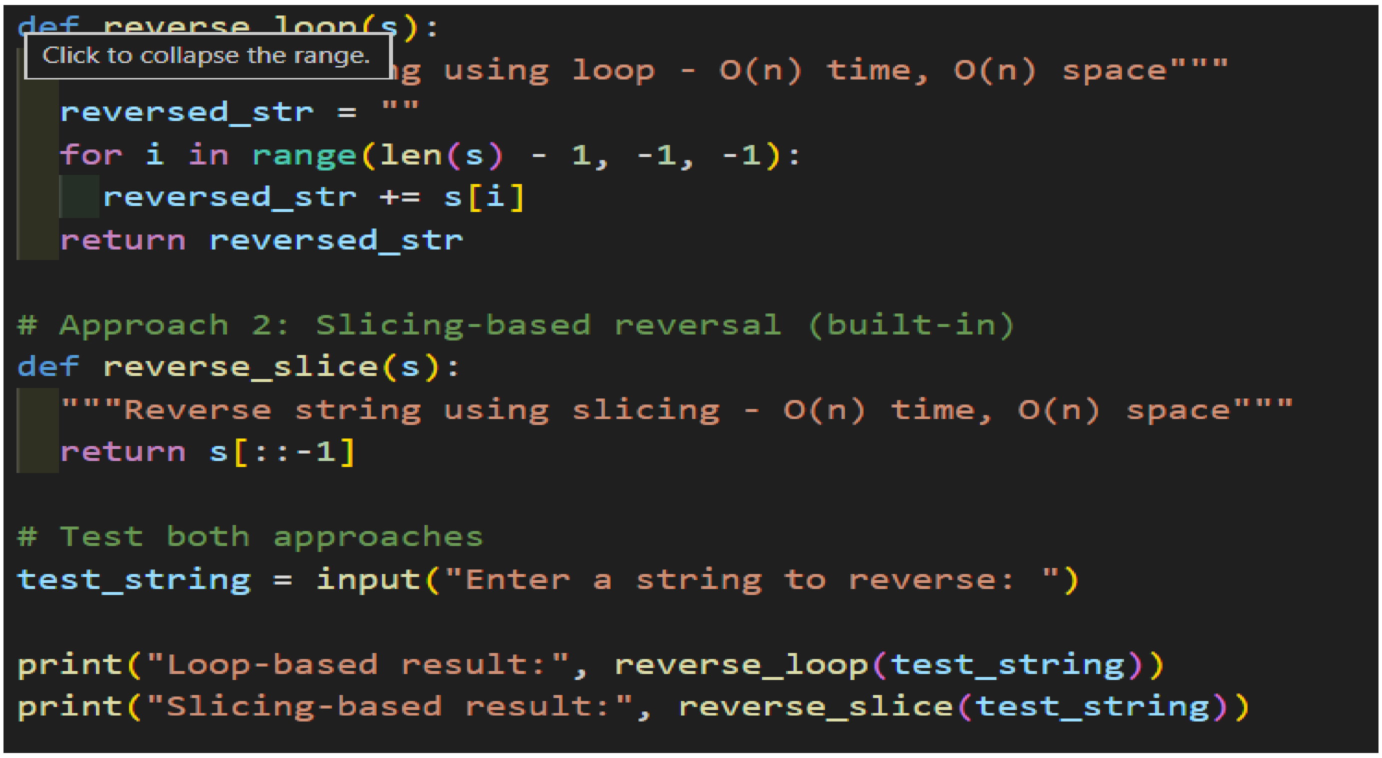
Task 5: Loop-Based vs Built-In String Reversal

Copilot Prompt Use



```
#Task 5
# Task 5: Compare loop-based vs slicing-based string reversal

# Approach 1: Loop-based reversal (manual iteration)
```



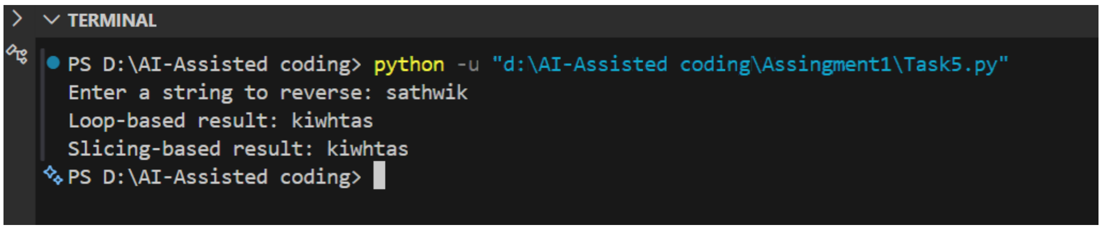
```python
def reverse_loop(s):
    g using loop - O(n) time, O(n) space"""
    reversed_str = ""
    for i in range(len(s) - 1, -1, -1):
        reversed_str += s[i]
    return reversed_str

# Approach 2: Slicing-based reversal (built-in)
def reverse_slice(s):
    """Reverse string using slicing - O(n) time, O(n) space"""
    return s[::-1]

# Test both approaches
test_string = input("Enter a string to reverse: ")

print("Loop-based result:", reverse_loop(test_string))
print("Slicing-based result:", reverse_slice(test_string))
```

Output :



```
PS D:\AI-Assisted coding> python -u "d:\AI-Assisted coding\Assingment1\Task5.py"
Enter a string to reverse: sathwik
Loop-based result: kiwhtas
Slicing-based result: kiwhtas
PS D:\AI-Assisted coding> 
```

Comparison Discussion

The loop-based approach reverses the string by checking each character one by one. It helps beginners understand how string reversal works.

The slicing-based approach uses Python’s built-in feature to reverse the string. It is shorter and easier to read.

Both methods take the same amount of time, which is **O(n)**, where *n* is the length of the string.

However, slicing works faster for large strings because it is optimized inside Python.

The loop-based method is good for learning, while the slicing method is better for real applications.

Conclusion

GitHub Copilot helped in writing, improving, and comparing different string reversal programs. Using functions and built-in features makes the code easier to read, faster, and better for large programs.

Declaration

All the code and explanations were created using GitHub Copilot and were checked manually to ensure correctness.