Batch:04
23303a51203

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

❖ Scenario

You are developing a basic text-processing utility for a messaging application.

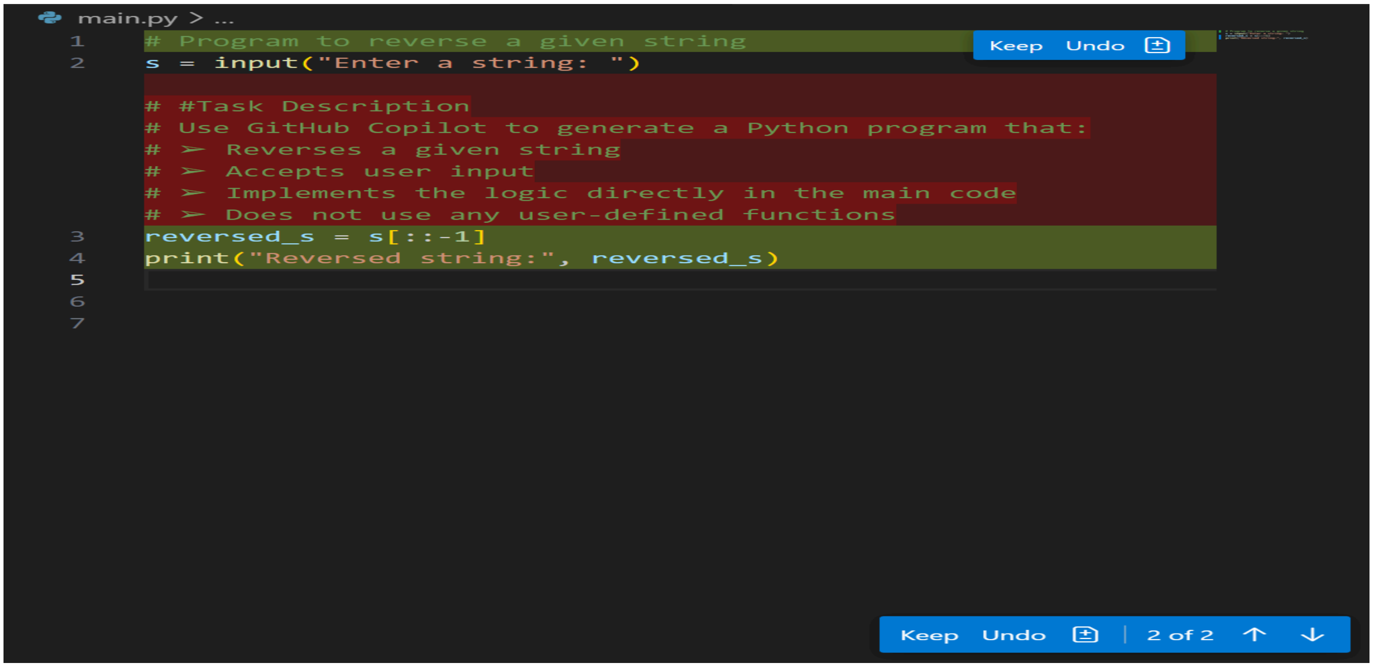
❖ Task Description

Use GitHub Copilot to generate a Python program that:

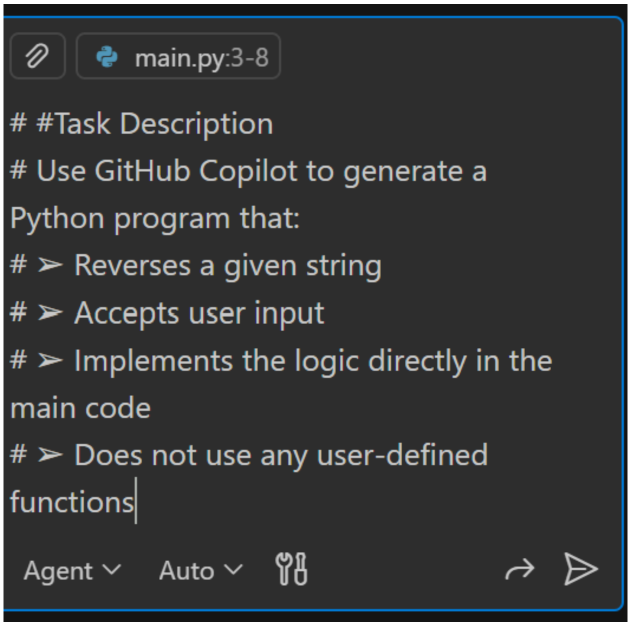
➢ Reverses a given string
➢ Accepts user input
➢ Implements the logic directly in the main code
➢ Does not use any user-defined functions

❖ Expected Output

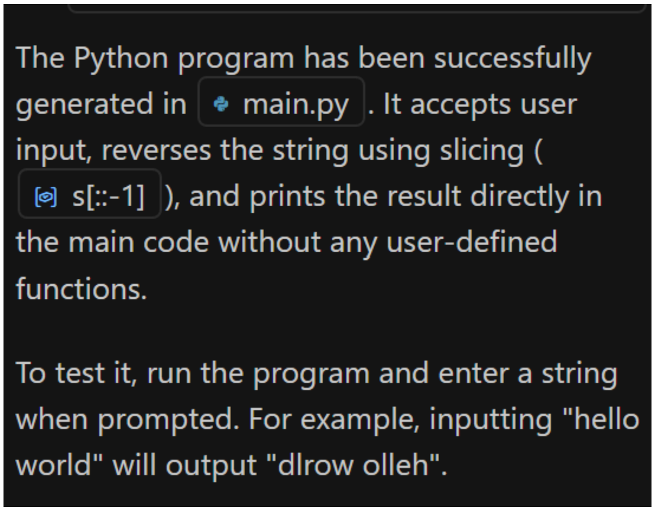
➢ Correct reversed string
➢ Screenshots showing Copilot-generated code suggestions
➢ Sample inputs and outputs



```
# Program to reverse a given string
s = input("Enter a string: ")

# #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
reversed_s = s[::-1]
print("Reversed string:", reversed_s)
```



#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



The Python program has been successfully generated in main.py. It accepts user input, reverses the string using slicing (s[::-1]), and prints the result directly in the main code without any user-defined functions.

To test it, run the program and enter a string when prompted. For example, inputting "hello world" will output "dlrow olleh".

Justification:I used the prompt and asked to generate the code for reverse a string,accepts user input,implements the logic directly in the main code,does nto use any user-defined functions code in python

Task 02:

Task 2: Efficiency & Logic Optimization (Readability Improvement)

❖ Scenario

The code will be reviewed by other developers.

❖ Task Description

Examine the Copilot-generated code from Task 1 and improve it by:

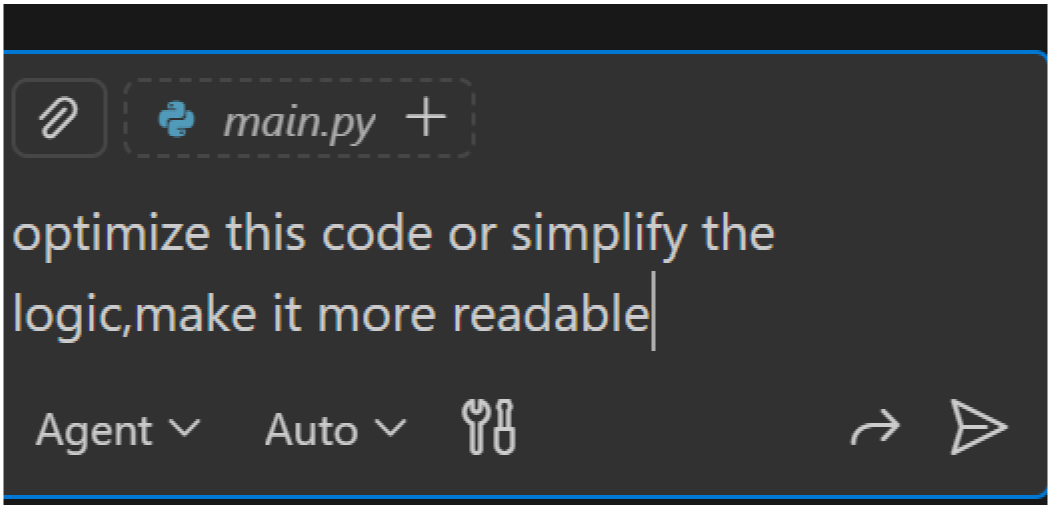
➢ Removing unnecessary variables

➢ Simplifying loop or indexing logic

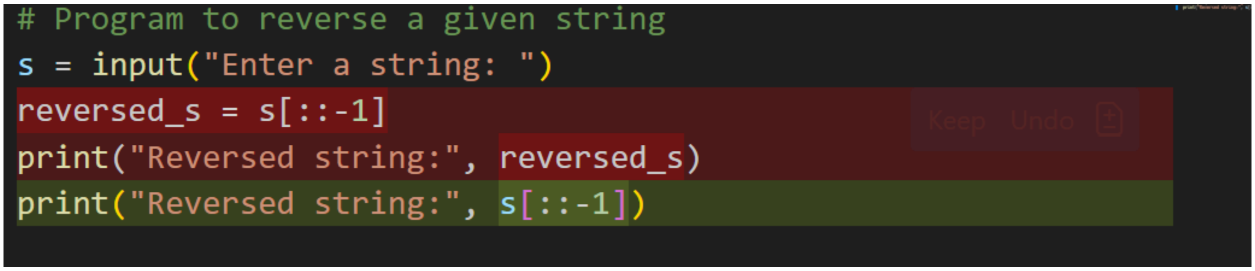
➢ Improving readability

➢ Use Copilot prompts like:

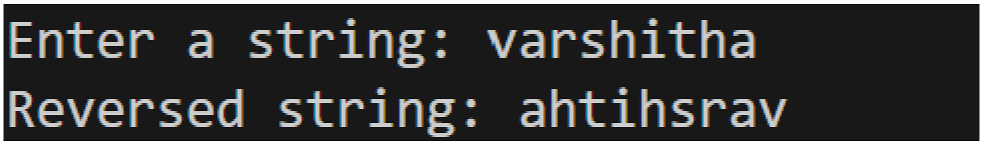
▪ “Simplify this string reversal code”
▪ “Improve readability and efficiency”



optimize this code or simplify the logic,make it more readable



```
# Program to reverse a given string
s = input("Enter a string: ")
reversed_s = s[::-1]
print("Reversed string:", reversed_s)
print("Reversed string:", s[::-1])
```



```
Enter a string: varshitha
Reversed string: ahtihsrav
```

Justification:my prompt is optimize this code or simplify the logic,make it more readable

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

❖ Scenario

The string reversal logic is needed in multiple parts of an application.

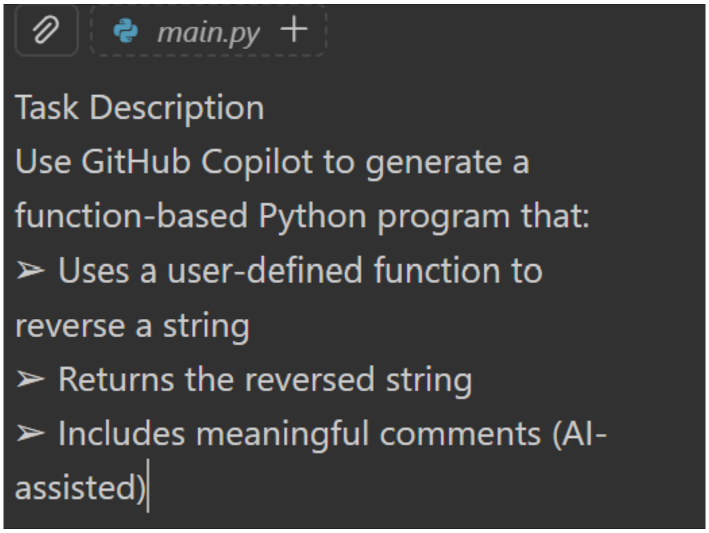
❖ Task Description

Use GitHub Copilot to generate a function-based Python program that:

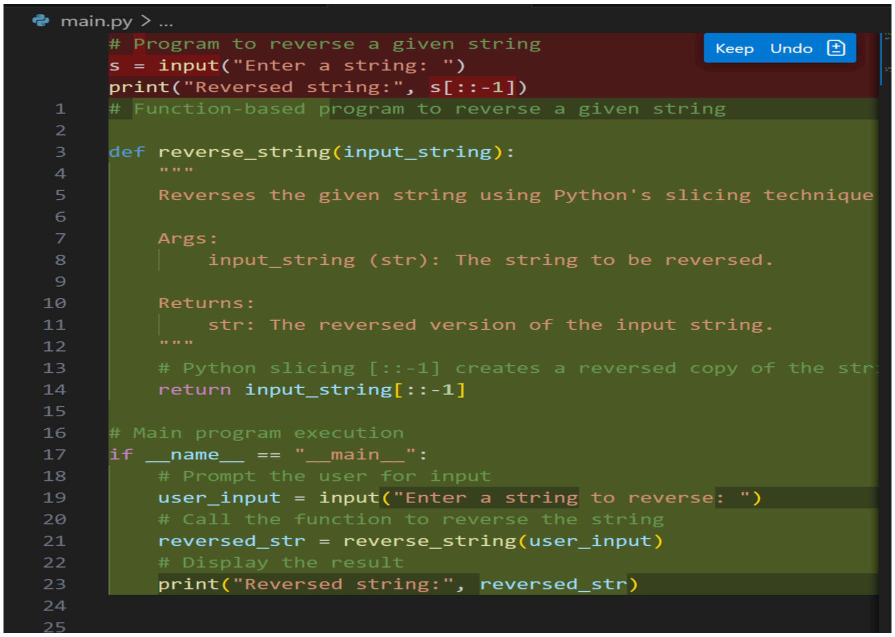
➢ Uses a user-defined function to reverse a string
➢ Returns the reversed string
➢ Includes meaningful comments (AI-assisted)

❖ Expected Output

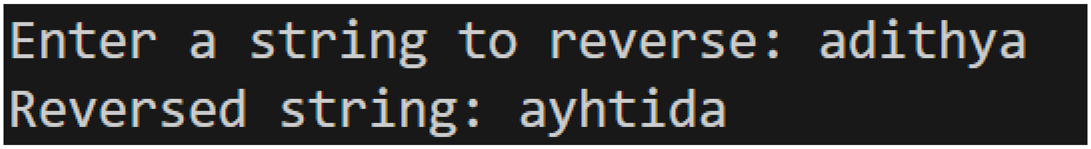
➢ Correct function-based implementation
➢ Screenshots documenting Copilot’s function generation
➢ Sample test cases and outputs



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)



```python
# Program to reverse a given string
s = input("Enter a string: ")
print("Reversed string:", s[::-1])
# Function-based program to reverse a given string

def reverse_string(input_string):
    """
    Reverses the given string using Python's slicing technique

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

    Returns:
        str: The reversed version of the input string.
    """
    # Python slicing [::-1] creates a reversed copy of the str
    return input_string[::-1]

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



```
Enter a string to reverse: adithya
Reversed string: ayhtida
```

Justification:function based python program to define a user defined function to reverse a string ,returns the reversed string,includes meaning comments and make code more readable

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

❖ Scenario

You are asked to justify design choices during a code review.

❖ 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
❖ Expected Output
Comparison table or short analytical report

Comparative Analysis: Procedural vs. Modular Approach

Aspect	Procedural (Without Functions)	Modular (With Functions)
Code Clarity	Simple and straightforward; logic is inline, easy to follow for small scripts.	Enhanced with docstrings, comments, and structure; separates concerns, making intent clearer for readers.
Reusability	Low; reversal logic is embedded and not easily extracted for other uses.	High; the reverse_string function can be reused across the program or imported into other modules.
Debugging Ease	Moderate; issues are traced directly in the main flow, but harder to isolate logic.	High; function can be tested independently (e.g., via unit tests), simplifying bug isolation.
Suitability for Large-Scale Applications	Poor; procedural code becomes unwieldy as complexity grows, leading to maintenance challenges.	Excellent; promotes modularity, scalability, and collaboration in larger codebases with multiple developers.

Prompt :Compare both of the previous code and analyze them based on code clarity,reusability,debugging ease,suitability for large scale applications and a short analytical report.

Justification: For this simple string reversal task, the procedural approach suffices. However, the modular approach is preferable for maintainability and scalability, aligning with best practices in software engineering. In a code review, I'd suggest adopting the function-based version to future-proof the code.

Task 5: AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches to String Reversal)
❖ Scenario
Your mentor wants to evaluate how AI handles alternative logic paths.
❖ Task Description

Prompt GitHub Copilot to generate:
➢ A loop-based string reversal approach
➢ A built-in / slicing-based string reversal approach
❖ Expected Output
➢ Two correct implementations
➢ Comparison discussing:
▪ Execution flow
▪ Time complexity
▪ Performance for large inputs
▪ When each approach is appropriate

Prompt : Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches to String Reversal)
➢ A loop-based string reversal approach
➢ A built-in / slicing-based string reversal approach
❖ Expected Output
➢ Two correct implementations
➢ Comparison discussing:
▪ Execution flow
▪ Time complexity
▪ Performance for large inputs
▪ When each approach is appropriate

1. **Loop-Based Reversal** (loop_reversal.py): Uses a for-loop to prepend each character to a new string, building the reversal iteratively.
2. **Slicing-Based Reversal** (slicing_reversal.py): Uses Python's built-in slicing (`[::-1]`) to reverse the string in a single operation.

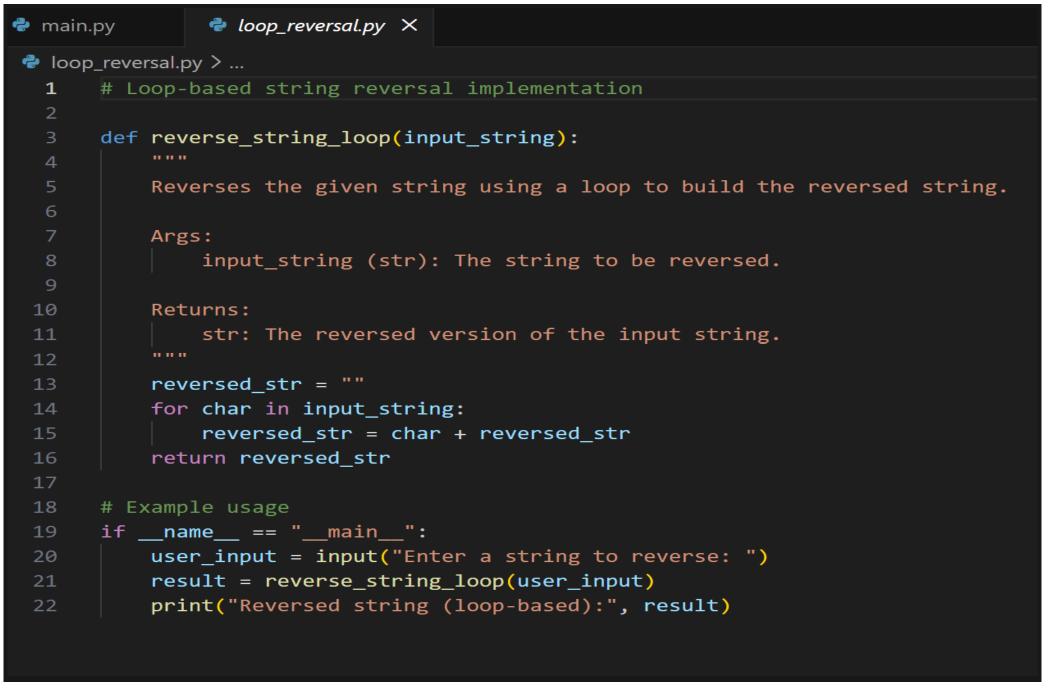
Comparison

Aspect	Loop-Based Approach	Slicing-Based Approach
Execution Flow	Iterates through each character in the string, prepending it to a result string (e.g., for "abc", builds "" → "a" → "ba" → "cba").	Directly creates a reversed copy using Python's internal string slicing mechanism, which handles the reversal at a lower level.

Time Complexity	O(n), where n is the string length, due to linear iteration and string concatenation (which can be inefficient in Python for large n due to immutable strings).	O(n), as slicing involves copying the string, but optimized in CPython's C implementation.
Performance for Large Inputs	Slower for very large strings (e.g., 10^6+ characters) due to repeated string creations and potential memory overhead from immutability.	Faster and more efficient for large inputs, as it's a single, optimized operation without intermediate allocations.
When Appropriate	Use for educational purposes, when slicing is unavailable (e.g., in restricted environments), or for custom logic extensions (e.g., conditional reversal).	Preferred for simplicity, readability, and performance in most Python applications; ideal for quick, standard reversals.

Overall: Slicing is the recommended default for its conciseness and efficiency, while loops are better for learning or specialized cases. Both produce correct results.

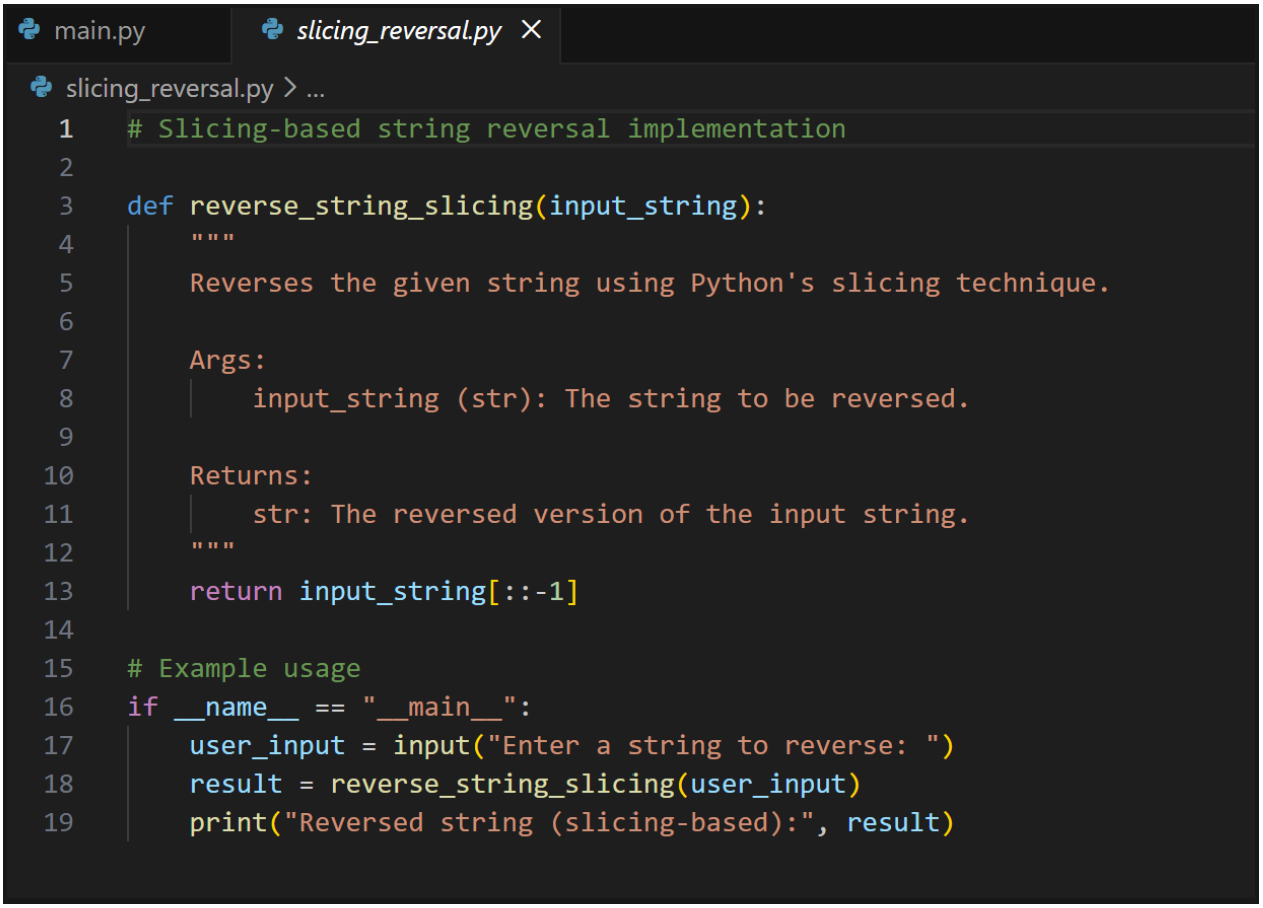
Loop reversal code:



```python
# Loop-based string reversal implementation

def reverse_string_loop(input_string):
    """
    Reverses the given string using a loop to build the reversed string.

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

    Returns:
        str: The reversed version of the input string.
    """
    reversed_str = ""
    for char in input_string:
        reversed_str = char + reversed_str
    return reversed_str

# Example usage
if __name__ == "__main__":
    user_input = input("Enter a string to reverse: ")
    result = reverse_string_loop(user_input)
    print("Reversed string (loop-based):", result)
```

```
Enter a string to reverse: hello Reversed string (slicing-based): olleh
```

Slicing_reversal code



```python
# Slicing-based string reversal implementation

def reverse_string_slicing(input_string):
    """
    Reverses the given string using Python's slicing technique.

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

    Returns:
        str: The reversed version of the input string.
    """
    return input_string[::-1]

# Example usage
if __name__ == "__main__":
    user_input = input("Enter a string to reverse: ")
    result = reverse_string_slicing(user_input)
    print("Reversed string (slicing-based):", result)
```

```
Enter a string to reverse:hello  Reversed string (slicing-based): olleh
```

Justification :

Give me string reversal codes using iterative method and slicing reversal method and compare them based on Execution flow, Time complexity,Performance for large inputs,When each approach is appropriate