ASSIGNMENT 1.5

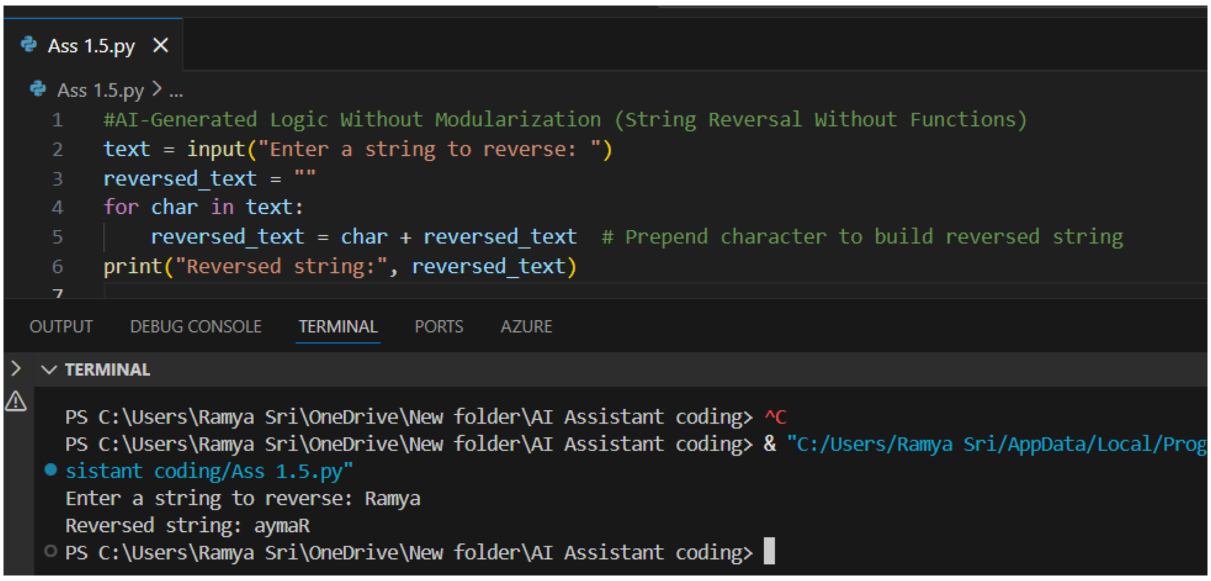
RamyaSri

2303A510i9

Batch :30

Task 1:

Code& Output:



Ass 1.5.py

Ass 1.5.py > ...

```python
#AI-Generated Logic Without Modularization (String Reversal Without Functions)
text = input("Enter a string to reverse: ")
reversed_text = ""
for char in text:
    reversed_text = char + reversed_text  # Prepend character to build reversed string
print("Reversed string:", reversed_text)
```

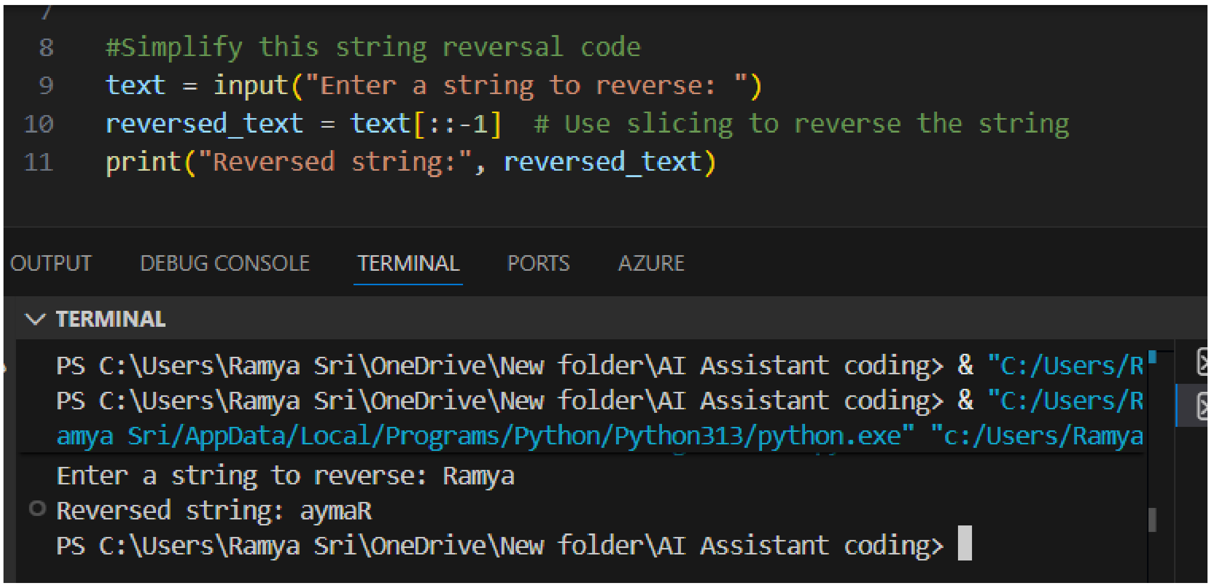
```
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding> ^C
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding> & "C:/Users/Ramya Sri/AppData/Local/Prog
sistant coding/Ass 1.5.py"
Enter a string to reverse: Ramya
Reversed string: aymaR
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>
```

Justification:

The non-modular approach directly places logic in the main code, making it simple and quick to implement. However, it reduces clarity and reusability since the reversal logic cannot be reused elsewhere. It’s suitable for very small, one-time scripts but not for scalable applications.

Task 2:

Code & Output:



```
#Simplify this string reversal code
text = input("Enter a string to reverse: ")
reversed_text = text[::-1]  # Use slicing to reverse the string
print("Reversed string:", reversed_text)
```

OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

TERMINAL

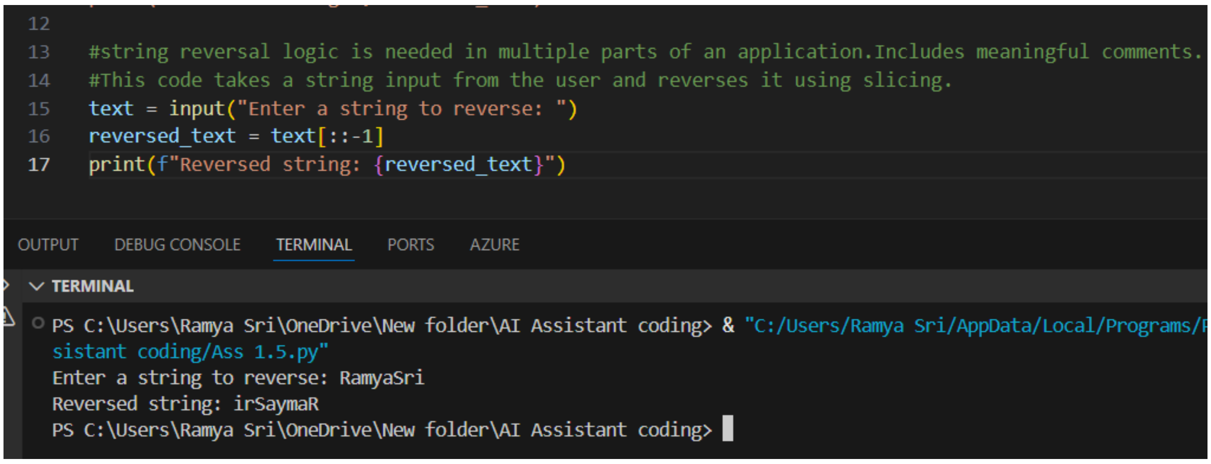
```
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding> & "C:/Users/R
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding> & "C:/Users/R
amya Sri/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/Ramya
Enter a string to reverse: Ramya
Reversed string: aymaR
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>
```

Justification:

Optimizing the Copilot-generated code improves readability and reduces unnecessary operations. By removing extra variables or loops, the program becomes easier to understand and maintain. Although time complexity remains O(n), reduced overhead makes execution slightly faster and cleaner.

Task 3:

Code & Output:



```
#string reversal logic is needed in multiple parts of an application.Includes meaningful comments.
#This code takes a string input from the user and reverses it using slicing.
text = input("Enter a string to reverse: ")
reversed_text = text[::-1]
print(f"Reversed string: {reversed_text}")
```

OUTPUT DEBUG CONSOLE TERMINAL PORTS AZURE

TERMINAL

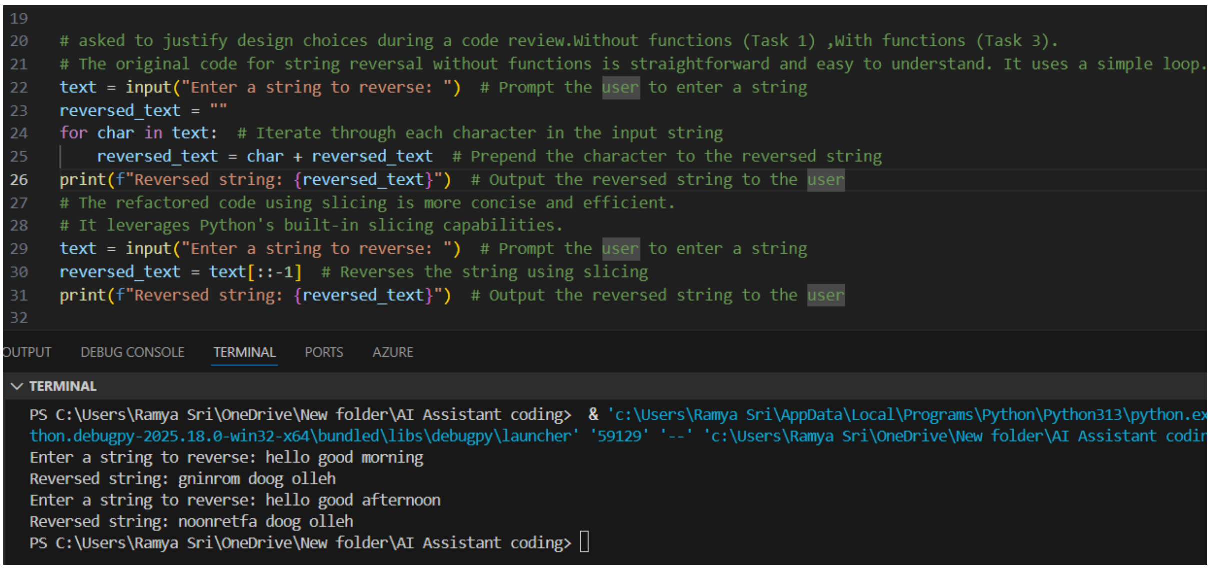
```
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding> & "C:/Users/Ramya Sri/AppData/Local/Programs/P
sistant coding/Ass 1.5.py"
Enter a string to reverse: RamyaSri
Reversed string: irSaymaR
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>
```

Justification:

Using functions introduces modularity, allowing the string reversal logic to be reused across multiple parts of the application. It improves clarity, debugging, and scalability. Functions also make the code more professional and maintainable for larger projects.

Task 4:

Code &Output:



```python

# asked to justify design choices during a code review.Without functions (Task 1) ,With functions (Task 3).
# The original code for string reversal without functions is straightforward and easy to understand. It uses a simple loop.
text = input("Enter a string to reverse: ")  # Prompt the user to enter a string
reversed_text = ""
for char in text:  # Iterate through each character in the input string
    reversed_text = char + reversed_text  # Prepend the character to the reversed string
print(f"Reversed string: {reversed_text}")  # Output the reversed string to the user
# The refactored code using slicing is more concise and efficient.
# It leverages Python's built-in slicing capabilities.
text = input("Enter a string to reverse: ")  # Prompt the user to enter a string
reversed_text = text[::-1]  # Reverses the string using slicing
print(f"Reversed string: {reversed_text}")  # Output the reversed string to the user

```

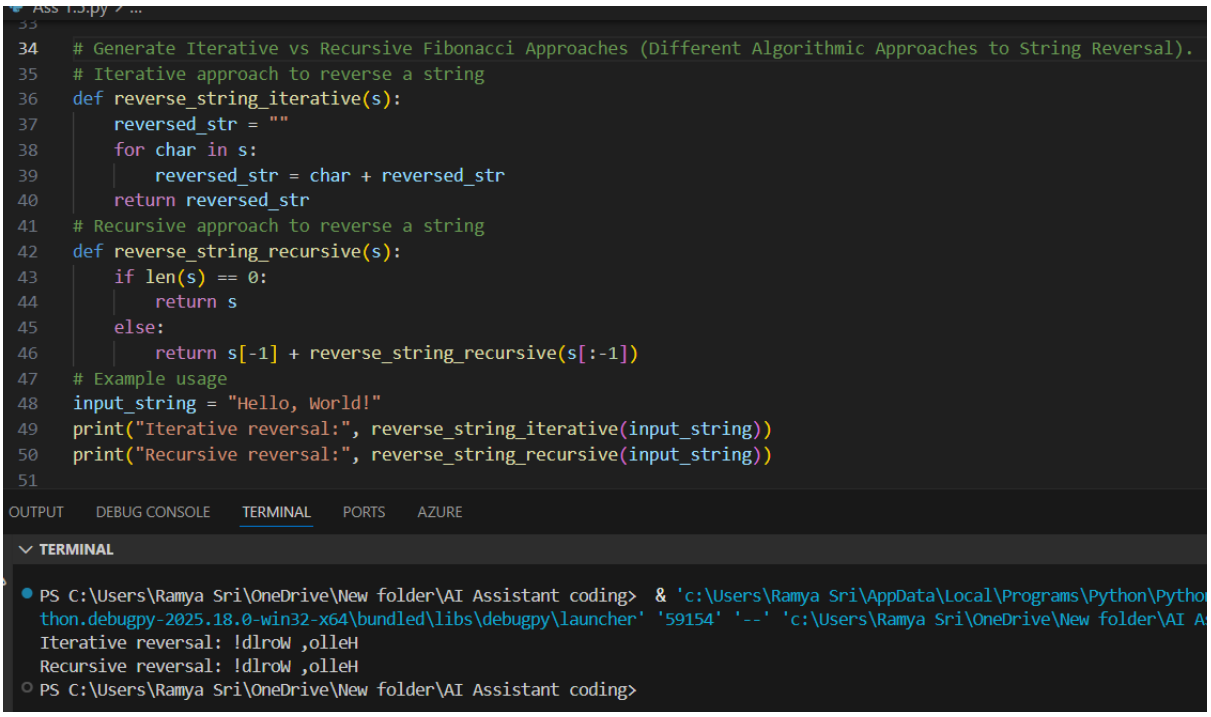
```
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>  & 'c:\Users\Ramya Sri\AppData\Local\Programs\Python\Python313\python.ex
thon.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '59129' '--' 'c:\Users\Ramya Sri\OneDrive\New folder\AI Assistant codir
Enter a string to reverse: hello good morning
Reversed string: gninrom doog olleh
Enter a string to reverse: hello good afternoon
Reversed string: noonretfa doog olleh
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding> 
```

Justification:

Procedural (without functions) is straightforward but limited in clarity and reusability. Modular (with functions) is better for debugging, reuse, and large-scale applications. The comparison highlights that modular design is more suitable for collaborative development and long-term maintenance.

Task 5:

Code &Output:



```python
# Generate Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches to String Reversal).
# Iterative approach to reverse a string
def reverse_string_iterative(s):
    reversed_str = ""
    for char in s:
        reversed_str = char + reversed_str
    return reversed_str
# Recursive approach to reverse a string
def reverse_string_recursive(s):
    if len(s) == 0:
        return s
    else:
        return s[-1] + reverse_string_recursive(s[:-1])
# Example usage
input_string = "Hello, World!"
print("Iterative reversal:", reverse_string_iterative(input_string))
print("Recursive reversal:", reverse_string_recursive(input_string))
```

```
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>  & 'c:\Users\Ramya Sri\AppData\Local\Programs\Python\Pytho
thon.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '59154' '--' 'c:\Users\Ramya Sri\OneDrive\New folder\AI A
Iterative reversal: !dlroW ,olleH
Recursive reversal: !dlroW ,olleH
PS C:\Users\Ramya Sri\OneDrive\New folder\AI Assistant coding>
```

Justification:

The loop-based reversal explicitly shows execution flow, making it educational but slightly verbose. The slicing approach is concise, efficient, and Pythonic, with the same O(n) complexity but less overhead. For large inputs, slicing is faster, while loops are useful for demonstrating algorithmic logic.