AI Assisted Coding

Assignment 1.5

Name: V.Rithik Reddy

Hall ticket no: 2303A51263

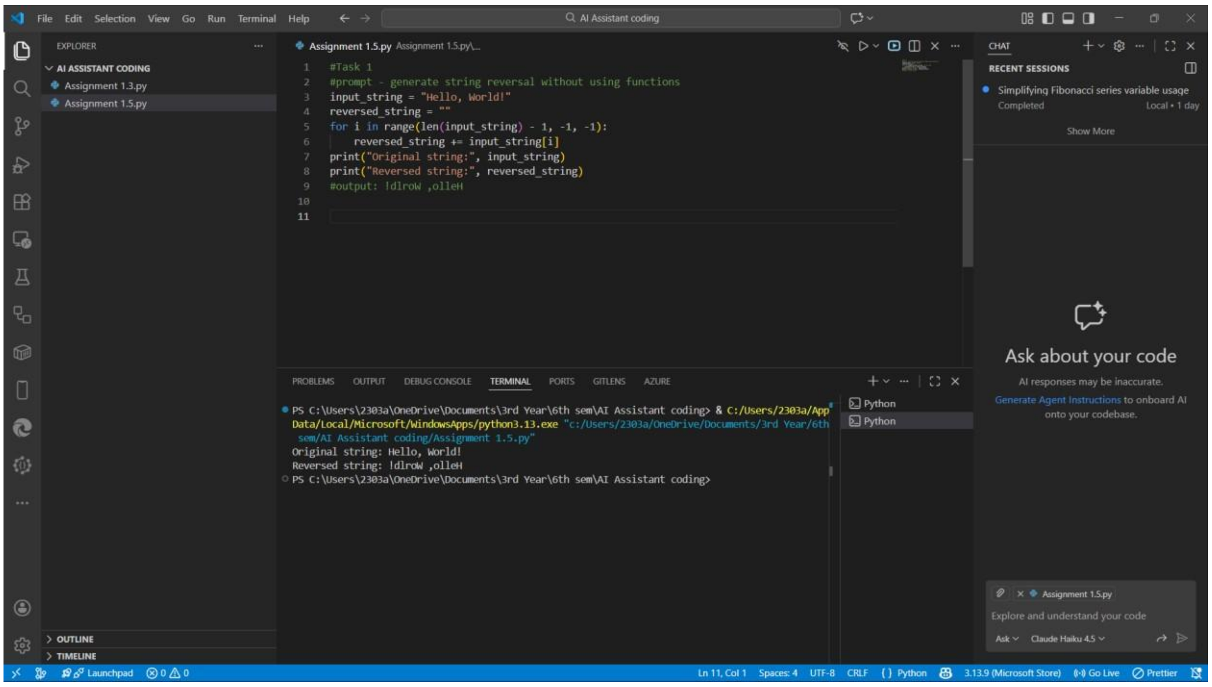
Batch no: 19

Task 1:

Prompt:

Generate string reversal without using functions **Code&**

Output:



```python
#Task 1
#prompt - generate string reversal without using functions
input_string = "Hello, World!"
reversed_string = ""
for i in range(len(input_string) - 1, -1, -1):
    reversed_string += input_string[i]
print("Original string:", input_string)
print("Reversed string:", reversed_string)
#output: !dlroW ,olleH
```

```
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding> & C:/Users/2303a/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/2303a/OneDrive/Documents/3rd Year/6th sem/AI Assistant coding/Assignment 1.5.py"
Original string: Hello, World!
Reversed string: !dlroW ,olleH
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>
```

Explanation:

This task reverses a string without using any predefined functions.

The program reads the string from the end and moves backward to the beginning.

Each character is added to a new variable to form the reversed string.

This method relies only on loops and indexing.

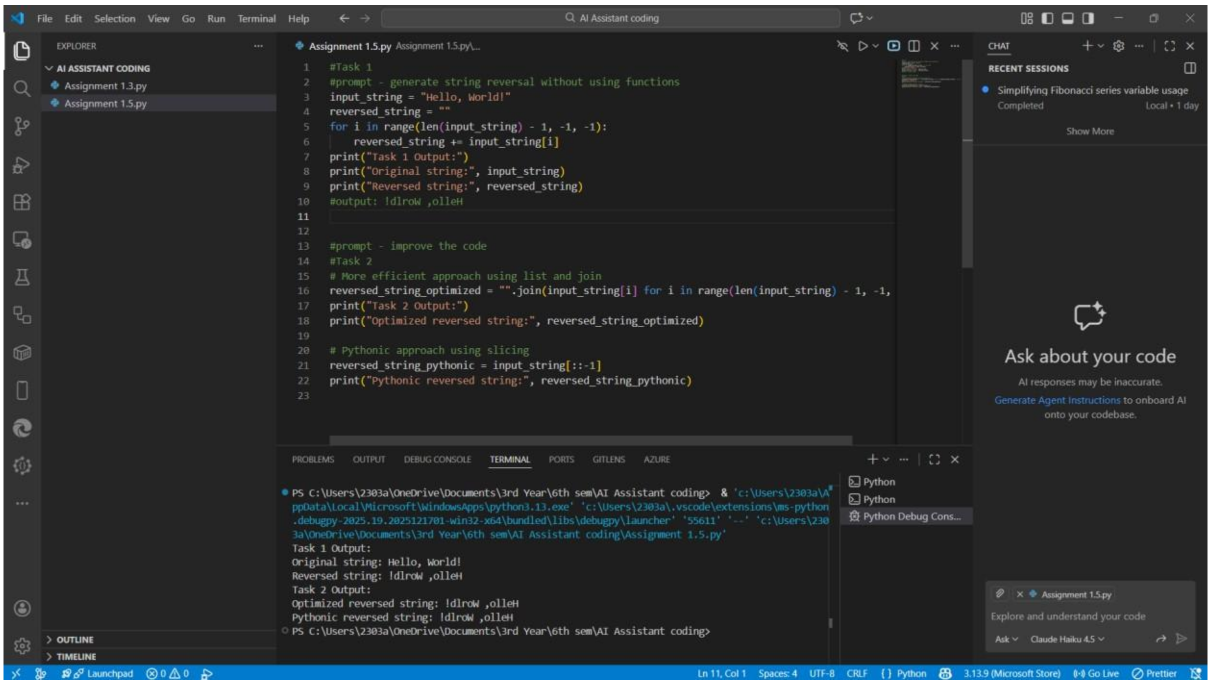
It shows how characters are accessed inside a string.

The logic works for strings of any size.

Task 2: Prompt:

improve the code **Code&**

Output:



```
#Task 1
#prompt - generate string reversal without using functions
input_string = "Hello, World!"
reversed_string = ""
for i in range(len(input_string) - 1, -1, -1):
    reversed_string += input_string[i]
print("Task 1 Output:")
print("Original string:", input_string)
print("Reversed string:", reversed_string)
#output: !dlroW ,olleH


#prompt - improve the code
#Task 2
# More efficient approach using list and join
reversed_string_optimized = "".join(input_string[i] for i in range(len(input_string) - 1, -1,
print("Task 2 Output:")
print("Optimized reversed string:", reversed_string_optimized)

# Pythonic approach using slicing
reversed_string_pythonic = input_string[::-1]
print("Pythonic reversed string:", reversed_string_pythonic)
```

```
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>  & 'c:\Users\2303a\A
ppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\2303a\.vscode\extensions\ms-python
.debugpy-2025.19.2025121701-win32-x64\bundled\libs\debugpy\launcher' '55611' '--' 'c:\Users\230
3a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding\Assignment 1.5.py'
Task 1 Output:
Original string: Hello, World!
Reversed string: !dlroW ,olleH
Task 2 Output:
Optimized reversed string: !dlroW ,olleH
Pythonic reversed string: !dlroW ,olleH
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>
```

File Edit Selection View Go Run Terminal Help
AI Assistant coding
EXPLORER
AI ASSISTANT CODING
Assignment 1.3.py
Assignment 1.5.py
Assignment 1.5.py Assignment 1.5.py\...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS AZURE
Python
Python
Python Debug Cons...
CHAT
RECENT SESSIONS
Simplifying Fibonacci series variable usage
Completed Local • 1 day
Show More
Ask about your code
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
Assignment 1.5.py
Explore and understand your code
Ask Claude Haiku 4.5
OUTLINE
TIMELINE
Launchpad
Ln 11, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.9 (Microsoft Store) Go Live Prettier

Explanation:

This task improves the earlier code by making it simpler and more organized.

Extra steps are removed so the program runs more smoothly.

The loop is written in a more efficient way.

Clear variable names help in understanding the logic better.

Even though the output stays the same, the code quality is higher.

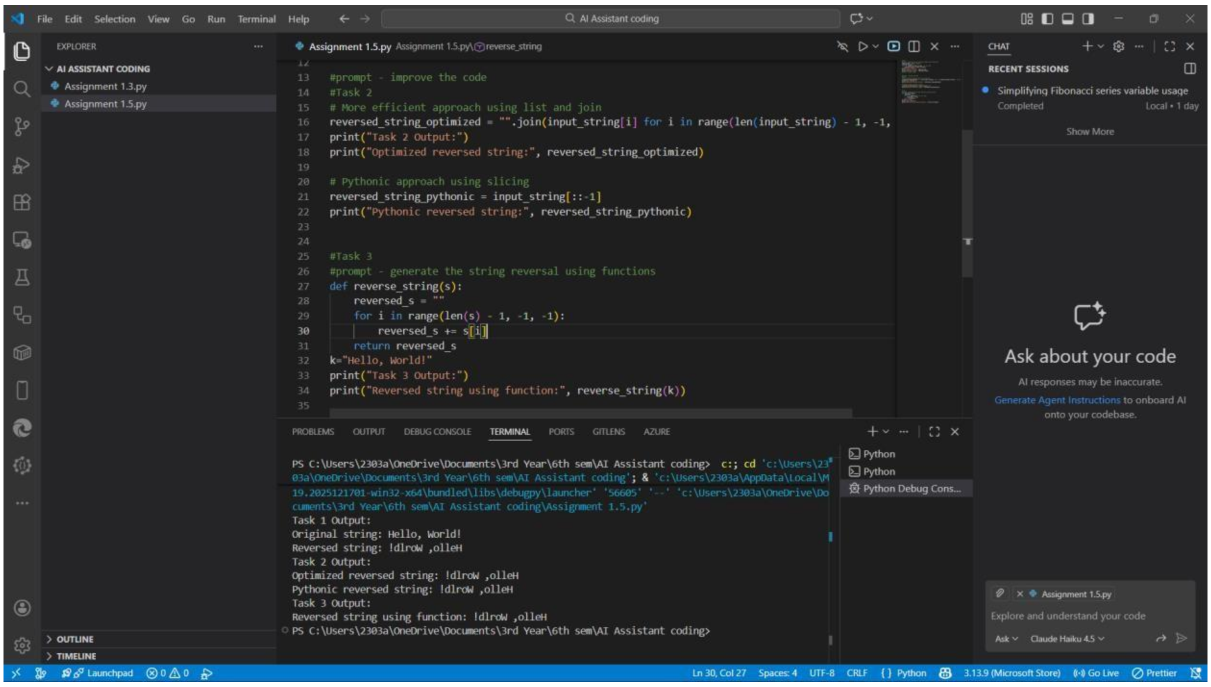
This reflects better programming practice.

Task 3:

Prompt:

Generate the string reversal using functions

Code& Output:



```python
#prompt - improve the code
#Task 2
# More efficient approach using list and join
reversed_string_optimized = "".join(input_string[i] for i in range(len(input_string) - 1, -1,
print("Task 2 Output:")
print("Optimized reversed string:", reversed_string_optimized)

# Pythonic approach using slicing
reversed_string_pythonic = input_string[::-1]
print("Pythonic reversed string:", reversed_string_pythonic)


#Task 3
#prompt - generate the string reversal using functions
def reverse_string(s):
    reversed_s = ""
    for i in range(len(s) - 1, -1, -1):
        reversed_s += s[i]
    return reversed_s
k="Hello, World!"
print("Task 3 Output:")
print("Reversed string using function:", reverse_string(k))
```

```
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>  c:; cd 'c:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding'; & 'c:\Users\2303a\AppData\Local\M
19.2025121701-win32-x64\bundled\libs\debugpy\launcher' '56605' '--' 'c:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding\Assignment 1.5.py'
Task 1 Output:
Original string: Hello, World!
Reversed string: !dlroW ,olleH
Task 2 Output:
Optimized reversed string: !dlroW ,olleH
Pythonic reversed string: !dlroW ,olleH
Task 3 Output:
Reversed string using function: !dlroW ,olleH
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>
```

Explanation:

This task performs string reversal using a function.

The main reversal logic is written inside a separate block of code.

This allows the same function to be reused when needed.

It keeps the main program short and clean.

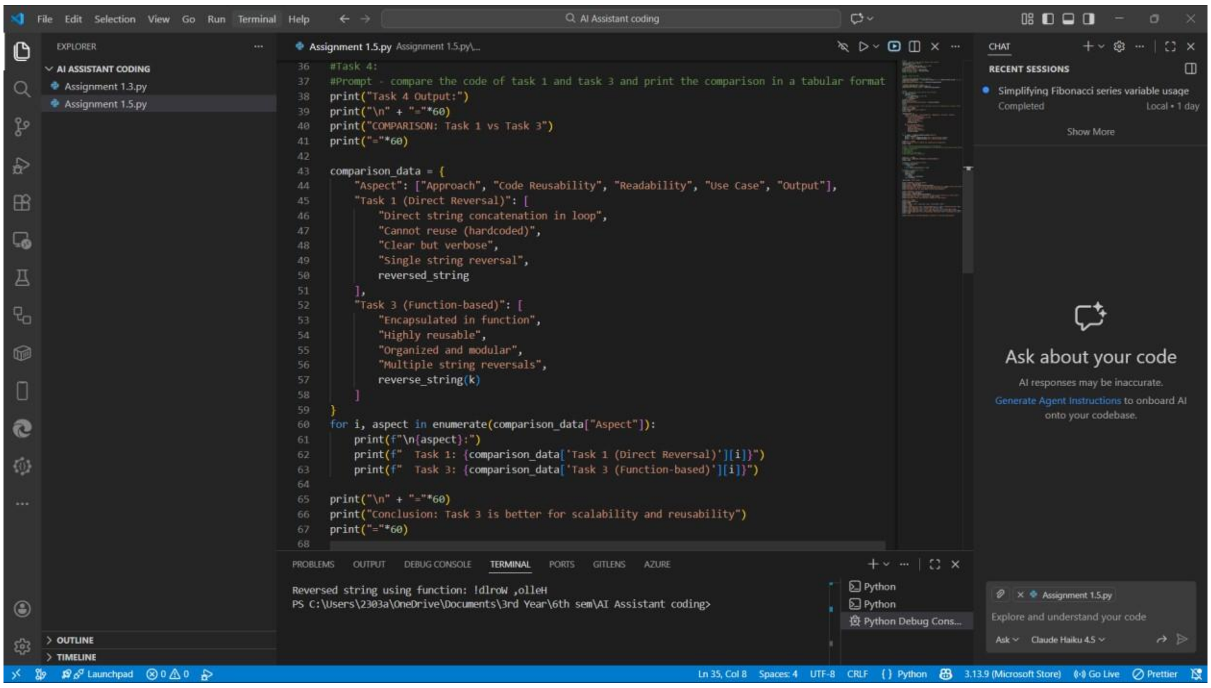
Functions help in managing large programs easily.

This structure is widely used in real software development.

Task 4: Prompt:

compare the code of task 1 and task 3 and print the comparison in a tabular format

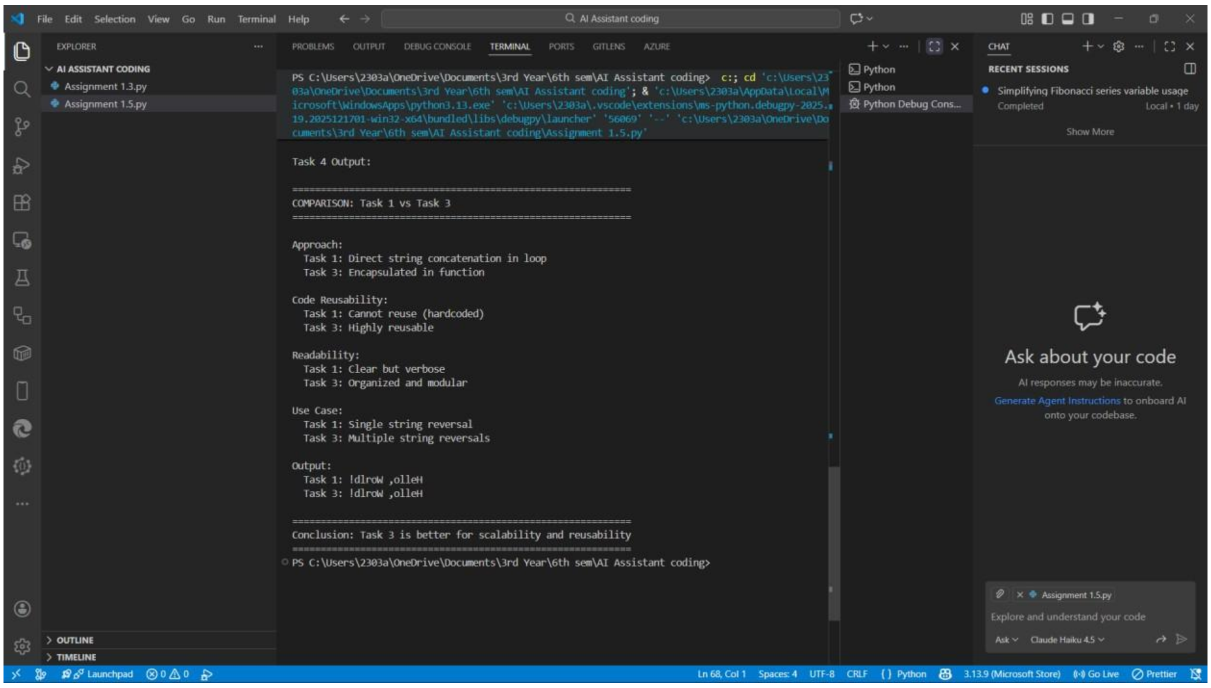
Code:



```python
#Task 4:
#Prompt - compare the code of task 1 and task 3 and print the comparison in a tabular format
print("Task 4 Output:")
print("\n" + "="*60)
print("COMPARISON: Task 1 vs Task 3")
print("="*60)

comparison_data = {
    "Aspect": ["Approach", "Code Reusability", "Readability", "Use Case", "Output"],
    "Task 1 (Direct Reversal)": [
        "Direct string concatenation in loop",
        "Cannot reuse (hardcoded)",
        "Clear but verbose",
        "Single string reversal",
        reversed_string
    ],
    "Task 3 (Function-based)": [
        "Encapsulated in function",
        "Highly reusable",
        "Organized and modular",
        "Multiple string reversals",
        reverse_string(k)
    ]
}
for i, aspect in enumerate(comparison_data["Aspect"]):
    print(f"\n{aspect}:")
    print(f"  Task 1: {comparison_data['Task 1 (Direct Reversal)'][i]}")
    print(f"  Task 3: {comparison_data['Task 3 (Function-based)'][i]}")

print("\n" + "="*60)
print("Conclusion: Task 3 is better for scalability and reusability")
print("="*60)
```

```
Reversed string using function: !dlroW ,olleH
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>
```

Output :



```
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>  c:; cd 'c:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding'; & 'c:\Users\2303a\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\2303a\.vscode\extensions\ms-python.debugpy-2025.19.2025121701-win32-x64\bundled\libs\debugpy\launcher' '56069' '--' 'c:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding\Assignment 1.5.py'

Task 4 Output:

============================================================
COMPARISON: Task 1 vs Task 3
============================================================

Approach:
  Task 1: Direct string concatenation in loop
  Task 3: Encapsulated in function

Code Reusability:
  Task 1: Cannot reuse (hardcoded)
  Task 3: Highly reusable

Readability:
  Task 1: Clear but verbose
  Task 3: Organized and modular

Use Case:
  Task 1: Single string reversal
  Task 3: Multiple string reversals

Output:
  Task 1: !dlroW ,olleH
  Task 3: !dlroW ,olleH

============================================================
Conclusion: Task 3 is better for scalability and reusability
============================================================
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>
```

Explanation:

This task compares the programs from Task 1 and Task 3.

The comparison is displayed in a table for easy understanding.

It shows differences in how the code is written and organized.

One method uses direct logic, while the other uses a function.

This explains why functions are better for structured programs.

The table makes the comparison clear and readable.

Task 5: Prompt:

use Different Algorithmic Approaches to String Reversal and the output should contain as Two correct implementations

Comparison discussing:

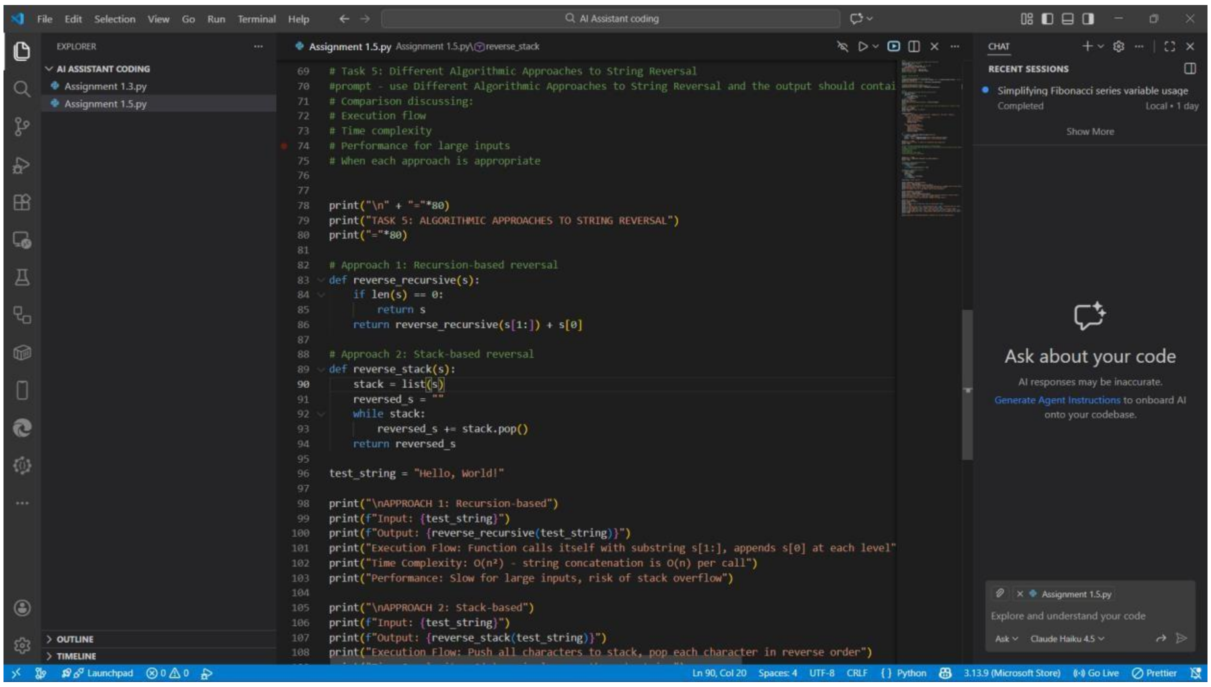
Execution flow

Time complexity

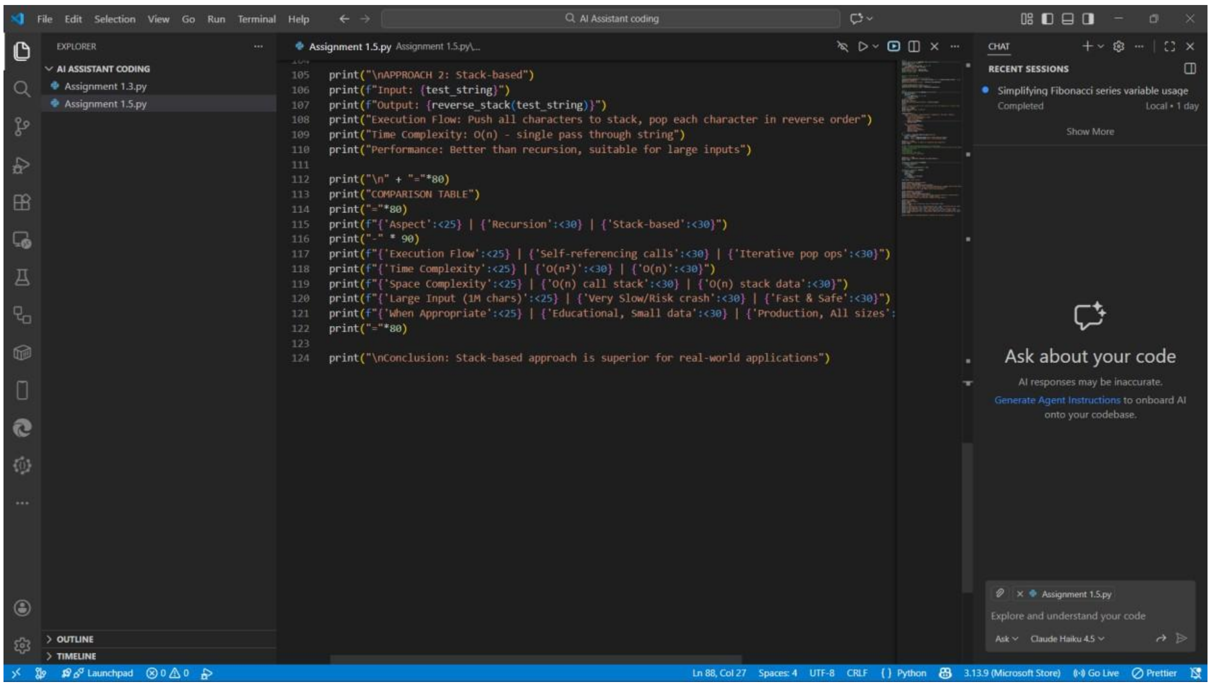
Performance for large inputs

When each approach is appropriate

Code:

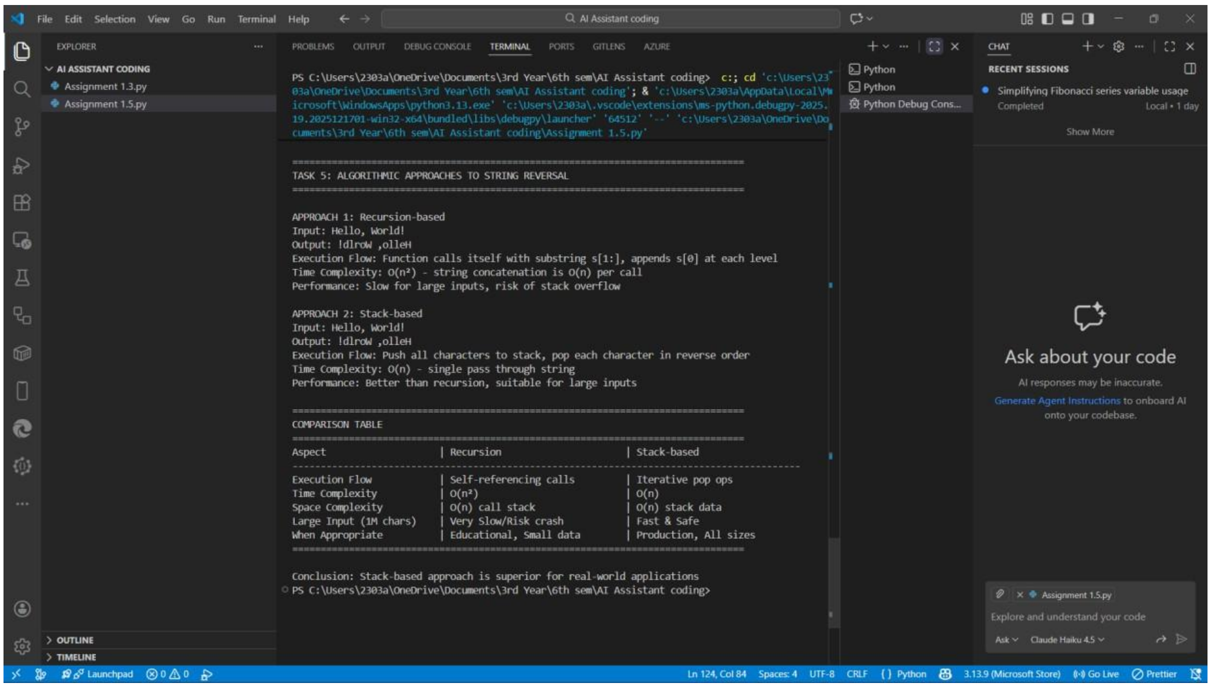


```python
# Task 5: Different Algorithmic Approaches to String Reversal
#prompt - use Different Algorithmic Approaches to String Reversal and the output should contai
# Comparison discussing:
# Execution flow
# Time complexity
# Performance for large inputs
# When each approach is appropriate


print("\n" + "="*80)
print("TASK 5: ALGORITHMIC APPROACHES TO STRING REVERSAL")
print("="*80)

# Approach 1: Recursion-based reversal
def reverse_recursive(s):
    if len(s) == 0:
        return s
    return reverse_recursive(s[1:]) + s[0]

# Approach 2: Stack-based reversal
def reverse_stack(s):
    stack = list(s)
    reversed_s = ""
    while stack:
        reversed_s += stack.pop()
    return reversed_s

test_string = "Hello, World!"

print("\nAPPROACH 1: Recursion-based")
print(f"Input: {test_string}")
print(f"Output: {reverse_recursive(test_string)}")
print("Execution Flow: Function calls itself with substring s[1:], appends s[0] at each level"
print("Time Complexity: O(n²) - string concatenation is O(n) per call")
print("Performance: Slow for large inputs, risk of stack overflow")

print("\nAPPROACH 2: Stack-based")
print(f"Input: {test_string}")
print(f"Output: {reverse_stack(test_string)}")
print("Execution Flow: Push all characters to stack, pop each character in reverse order")
```



```python
print("\nAPPROACH 2: Stack-based")
print(f"Input: {test_string}")
print(f"Output: {reverse_stack(test_string)}")
print("Execution Flow: Push all characters to stack, pop each character in reverse order")
print("Time Complexity: O(n) - single pass through string")
print("Performance: Better than recursion, suitable for large inputs")

print("\n" + "="*80)
print("COMPARISON TABLE")
print("="*80)
print(f"{'Aspect':<25} | {'Recursion':<30} | {'Stack-based':<30}")
print("-" * 90)
print(f"{'Execution Flow':<25} | {'Self-referencing calls':<30} | {'Iterative pop ops':<30}")
print(f"{'Time Complexity':<25} | {'O(n²)':<30} | {'O(n)':<30}")
print(f"{'Space Complexity':<25} | {'O(n) call stack':<30} | {'O(n) stack data':<30}")
print(f"{'Large Input (1M chars)':<25} | {'Very Slow/Risk crash':<30} | {'Fast & Safe':<30}")
print(f"{'When Appropriate':<25} | {'Educational, Small data':<30} | {'Production, All sizes':
print("="*80)

print("\nConclusion: Stack-based approach is superior for real-world applications")
```

Output :



```
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>  c:; cd 'c:\Users\23
03a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding'; & 'c:\Users\2303a\AppData\Local\M
icrosoft\WindowsApps\python3.13.exe' 'c:\Users\2303a\.vscode\extensions\ms-python.debugpy-2025.
19.2025121701-win32-x64\bundled\libs\debugpy\launcher' '64512' '--' 'c:\Users\2303a\OneDrive\Do
cuments\3rd Year\6th sem\AI Assistant coding\Assignment 1.5.py'

================================================================================
TASK 5: ALGORITHMIC APPROACHES TO STRING REVERSAL
================================================================================

APPROACH 1: Recursion-based
Input: Hello, World!
Output: !dlroW ,olleH
Execution Flow: Function calls itself with substring s[1:], appends s[0] at each level
Time Complexity: O(n²) - string concatenation is O(n) per call
Performance: Slow for large inputs, risk of stack overflow

APPROACH 2: Stack-based
Input: Hello, World!
Output: !dlroW ,olleH
Execution Flow: Push all characters to stack, pop each character in reverse order
Time Complexity: O(n) - single pass through string
Performance: Better than recursion, suitable for large inputs

================================================================================
COMPARISON TABLE
================================================================================
Aspect                    | Recursion                      | Stack-based
------------------------------------------------------------------------------------------
Execution Flow            | Self-referencing calls         | Iterative pop ops
Time Complexity           | O(n²)                          | O(n)
Space Complexity          | O(n) call stack                | O(n) stack data
Large Input (1M chars)    | Very Slow/Risk crash           | Fast & Safe
When Appropriate          | Educational, Small data        | Production, All sizes
================================================================================

Conclusion: Stack-based approach is superior for real-world applications
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>
```

Explanation:

This task applies two different techniques to reverse a string.

Both methods produce the same correct result.

The steps of execution vary between the two approaches.

Each method takes time based on the length of the string.

Some approaches are better when working with large inputs.

The comparison helps decide which method is more suitable.