

AI Assisted Coding

Assignment 1.5

Name: P. Vineeth Kumar

Hall ticket no: 2303A51256

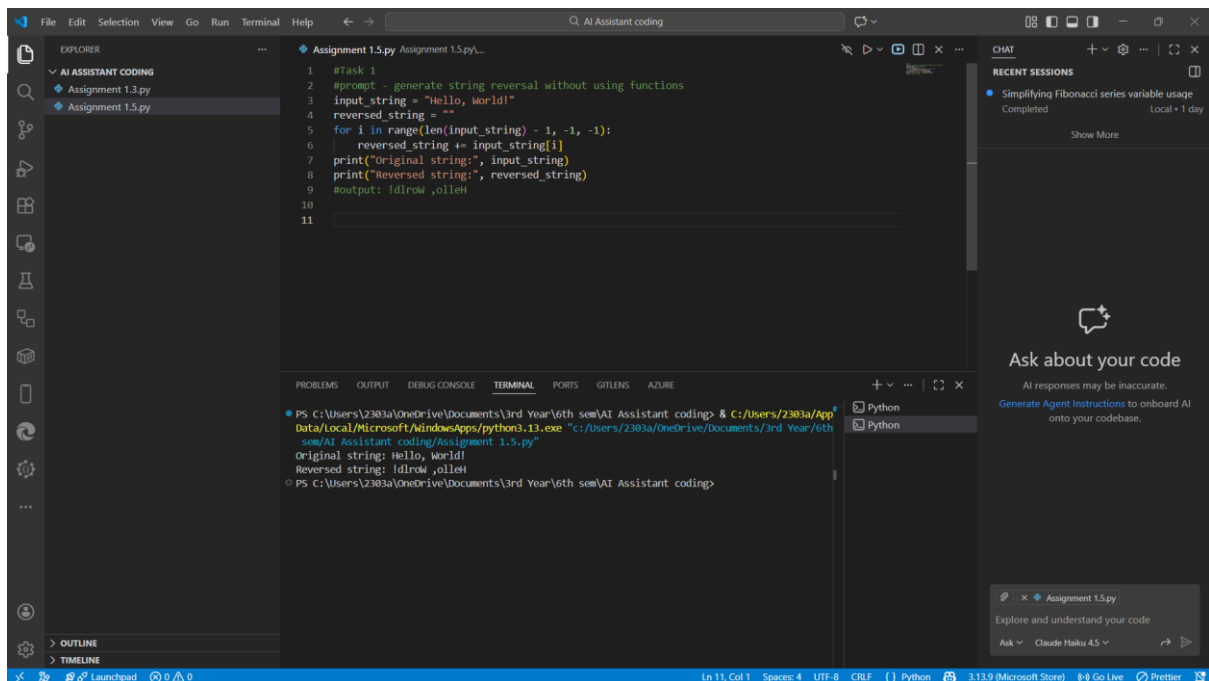
Batch no: 19

Task 1:

Prompt:

Generate string reversal without using functions

Code & Output:



The screenshot displays the Visual Studio Code interface. The Explorer pane on the left shows a project named 'AI ASSISTANT CODING' with files 'Assignment 1.3.py' and 'Assignment 1.5.py'. The main editor window is open to 'Assignment 1.5.py', which contains the following Python code:

```
1 #Task 1
2 #prompt - generate string reversal without using functions
3 input_string = "Hello, World!"
4 reversed_string = ""
5 for i in range(len(input_string) - 1, -1, -1):
6     reversed_string += input_string[i]
7 print("Original string:", input_string)
8 print("Reversed string:", reversed_string)
9 #output: !dlrow ,olleH
10
11
```

The TERMINAL pane at the bottom shows the execution of the script using the command `python "C:/Users/2303a/OneDrive/Documents/3rd Year/6th sem/AI Assistant coding/Assignment 1.5.py"`. The output is:

```
Original string: Hello, World!
Reversed string: !dlrow ,olleH
```

The right sidebar contains the CHAT pane with 'RECENT SESSIONS' and a section titled 'Ask about your code' with a note that 'AI responses may be inaccurate' and a link to 'Generate Agent Instructions to onboard AI onto your codebase'.

Explanation:

This task reverses a string without using any built-in functions, so the logic depends entirely on manual looping.

Each character of the string is accessed one by one from the last index to the first index.

The characters are appended into a new variable in reverse order.

This proves understanding of string indexing and loops instead of shortcuts.

The algorithm is simple but works for any string length.

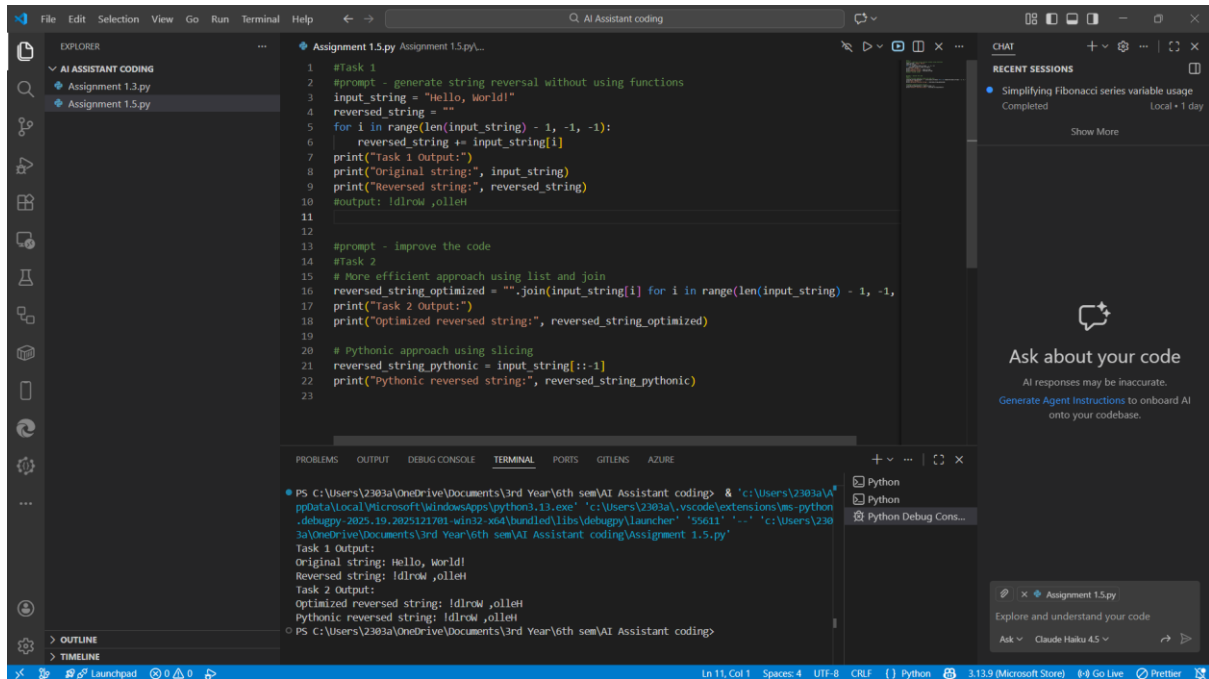
This approach is good for learning how strings behave internally.

Task 2:

Prompt:

improve the code

Code & Output:



```
1 #Task 1
2 #prompt - generate string reversal without using functions
3 input_string = "Hello, World!"
4 reversed_string = ""
5 for i in range(len(input_string) - 1, -1, -1):
6     reversed_string += input_string[i]
7 print("Task 1 Output:")
8 print("Original string:", input_string)
9 print("Reversed string:", reversed_string)
10 #output: ldlrow ,olleH
11
12
13 #prompt - improve the code
14 #Task 2
15 # More efficient approach using list and join
16 reversed_string_optimized = "".join(input_string[i] for i in range(len(input_string) - 1, -1,
17 print("Task 2 Output:")
18 print("Optimized reversed string:", reversed_string_optimized)
19
20 # Pythonic approach using slicing
21 reversed_string_pythonic = input_string[::-1]
22 print("Pythonic reversed string:", reversed_string_pythonic)
23
```

Task 1 Output:
Original string: Hello, World!
Reversed string: ldlrow ,olleH

Task 2 Output:
Optimized reversed string: ldlrow ,olleH
Pythonic reversed string: ldlrow ,olleH

Explanation:

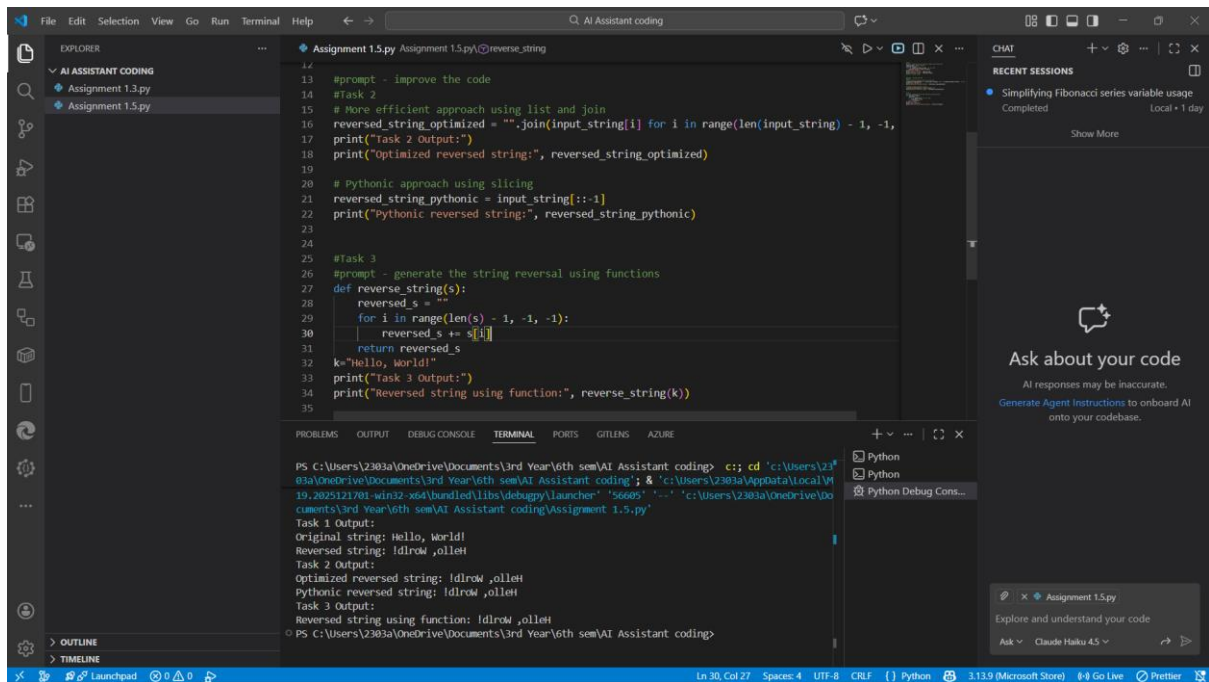
This task improves the first program by making the code cleaner, more readable, and more efficient. Unnecessary variables or steps are removed to reduce confusion. The loop logic is optimized to avoid redundant operations. Better variable names make the code easier to understand. The output remains the same, but the code quality is higher. This shows how the same logic can be written in a better professional way.

Task 3:

Prompt:

Generate the string reversal using functions

Code & Output:



Explanation:

This task performs string reversal using a function, which improves modularity.

The reversal logic is placed inside a reusable function.

The main program simply calls the function instead of repeating code.

This makes the program easier to maintain and modify later.

Functions also allow the logic to be reused for multiple inputs.

This approach follows proper programming structure.

Task 4:

Prompt:

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

Code :

```
36 #Task 4:
37 #Prompt - compare the code of task 1 and task 3 and print the comparison in a tabular format
38 print("Task 4 Output:")
39 print("\n" + "="*60)
40 print("COMPARISON: Task 1 vs Task 3")
41 print("="*60)
42
43 comparison_data = {
44     "Aspect": ["Approach", "Code Reusability", "Readability", "Use Case", "Output"],
45     "Task 1 (Direct Reversal)": [
46         "Direct string concatenation in loop",
47         "Cannot reuse (hardcoded)",
48         "Clear but verbose",
49         "Single string reversal",
50         reversed_string
51     ],
52     "Task 3 (Function-based)": [
53         "Encapsulated in function",
54         "Highly reusable",
55         "Organized and modular",
56         "Multiple string reversals",
57         reverse_string(k)
58     ]
59 }
60
61 for i, aspect in enumerate(comparison_data["Aspect"]):
62     print(f"\n{aspect}:")
63     print(f"Task 1: {comparison_data['Task 1 (Direct Reversal)'][i]}")
64     print(f"Task 3: {comparison_data['Task 3 (Function-based)'][i]}")
65
66 print("\n" + "="*60)
67 print("Conclusion: Task 3 is better for scalability and reusability")
68 print("="*60)
```

Output :

```
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding> 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\bundle\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: ldlrow ,olleH
Task 3: ldlrow ,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 manual reversal (Task 1) and the function-based reversal (Task 3). The comparison is printed in a table format to clearly show differences. It highlights differences in structure, reusability, and readability. Task 1 is direct but not reusable, while Task 3 is modular. This helps in understanding why functions are preferred in real applications. The table makes technical comparison easy to understand.

Task 5:

Prompt:

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

Comparison discussing:

Execution flow

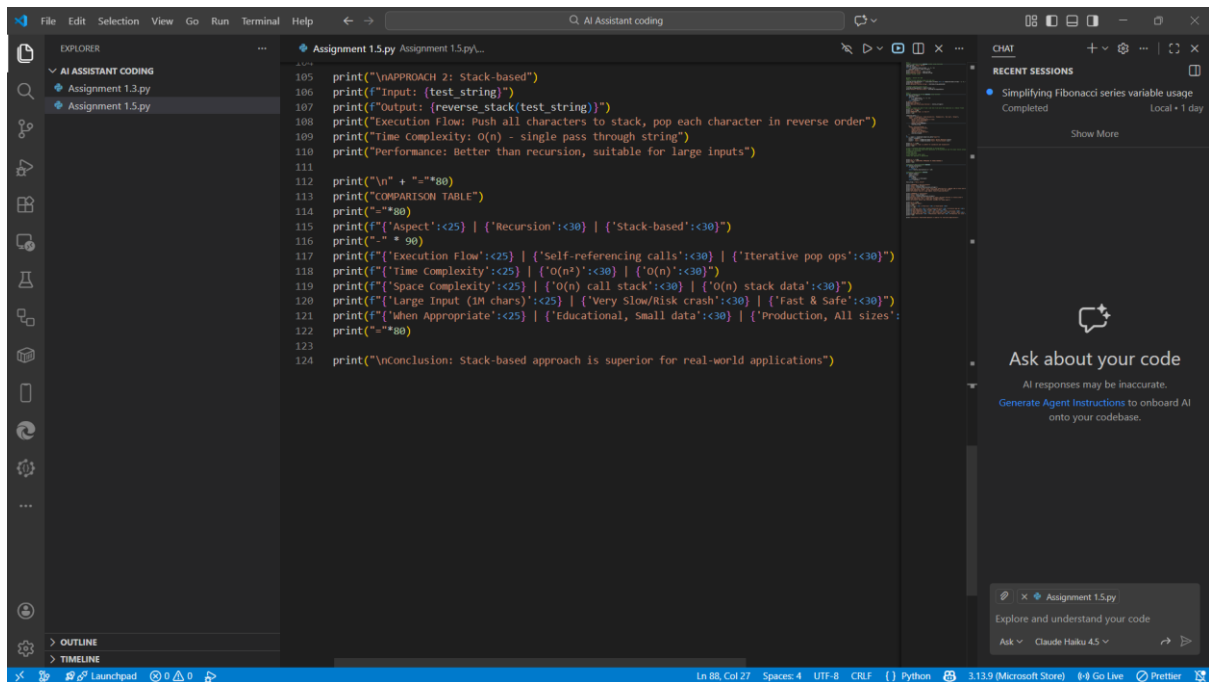
Time complexity

Performance for large inputs

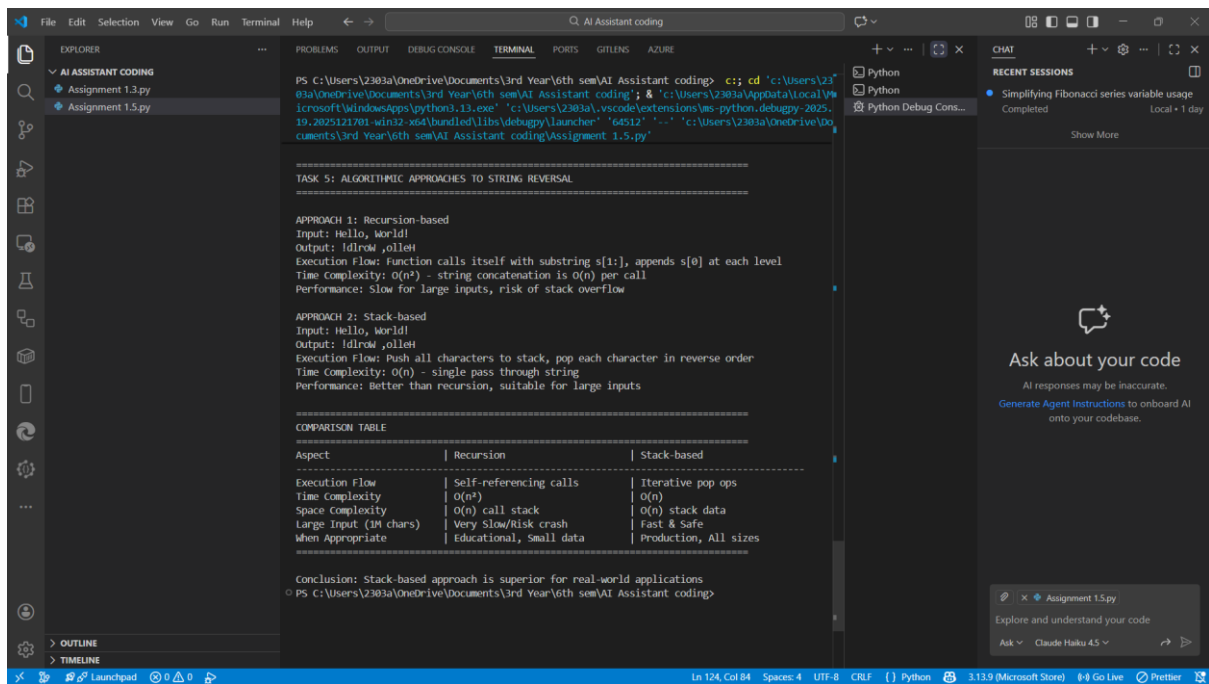
When each approach is appropriate

Code :

The image shows a screenshot of a Visual Studio Code editor window. The main editor area displays a Python file named 'Assignment 1.5.py'. The code is a script for reversing a string using two different methods: recursion and a stack. The script includes comments explaining the task, the prompt, and the execution flow. It defines two functions: 'reverse_recursive(s)' and 'reverse_stack(s)'. The 'reverse_recursive' function uses a recursive approach, and the 'reverse_stack' function uses a stack to reverse the string. A test case is provided at the bottom of the script, which prints the output of both functions for the input string 'Hello, World!'. The right sidebar shows the 'CHAT' panel with a recent session titled 'Simplifying Fibonacci series variable usage'. The session is completed and shows a 'Show More' link. The bottom status bar indicates the current file is 'Assignment 1.5.py' and the editor is at line 90, column 20.



Output :



Explanation:

This task uses two different algorithms to reverse a string. One approach uses a loop-based method, and the other uses a function-based or slicing method. Execution flow shows how each method processes characters differently. Both have O(n) time complexity, but their memory usage differs. For large strings, optimized methods perform better and are cleaner. Each approach is chosen based on performance needs and code clarity.