

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 shows the Microsoft Visual Studio Code interface. The code editor displays the following Python script:

```
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
```

The terminal tab shows the execution of the script and its output:

```
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/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>
```

The right sidebar shows a "RECENT SESSIONS" list with one entry: "Simplifying Fibonacci series variable usage" (Completed, Local, 1 day ago). A "CHAT" section is visible at the bottom right.

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:

```
Assignment 1.5.py Assignment 1.5.py...
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: olleH ,dlroH
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, -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
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS AZURE

PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding> 8 "c:\Users\2303a\appdata\local\Microsoft\WindowsApps\python3.13.exe" "c:\Users\2303a\vscodeextensions\ms-python.debugger_2025.19.2025121701-win32-x64\bundles\libs\debugpy\launcher" "5561" ... '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: olleH ,dlroH
Task 2 Output:
Optimized reversed string: olleH ,dlroH
Pythonic reversed string: olleH ,dlroH
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>

Ln 11, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.13.9 (Microsoft Store) Go Live Prettier

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:

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows two files: `Assignment 1.3.py` and `Assignment 1.5.py`.
- Code Editor:** Displays the content of `Assignment 1.5.py`. The code includes three approaches for reversing strings: a manual loop, a join operation, and slicing.
- Terminal:** Shows the command-line output for the program, which prints "Hello, World!" and its reverse.
- AI Assistant:** A sidebar on the right provides AI-generated code completion and asks about the code.

```
Assignment 1.5.py Assignment 1.5.py (reverse_string)

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, -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
24
25 #Task 3
26 #prompt - generate the string reversal using functions
27 def reverse_string(s):
28     reversed_s = ""
29     for i in range(len(s) - 1, -1, -1):
30         reversed_s += s[i]
31     return reversed_s
32 k="Hello, World!"
33 print("Task 3 Output:")
34 print("Reversed string using function:", reverse_string(k))
35

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\2025121701.win32-x64\bundled\libs\debug\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:

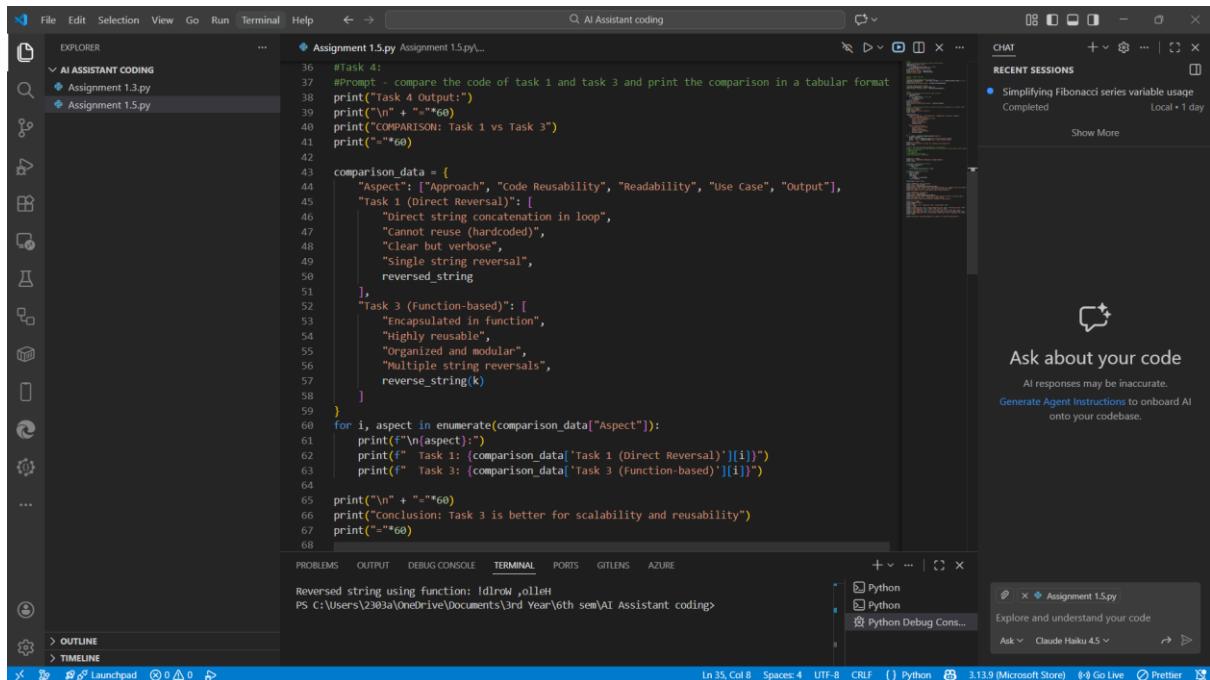
This task performs string reversal using a function, which improves modularity. The reversal logic is placed inside a reusable function. The main program simply calls the function instead of repeating code. This makes the program easier to maintain and modify later. Functions also allow the logic to be reused for multiple inputs. This approach follows proper programming structure.

Task 4:

Prompt:

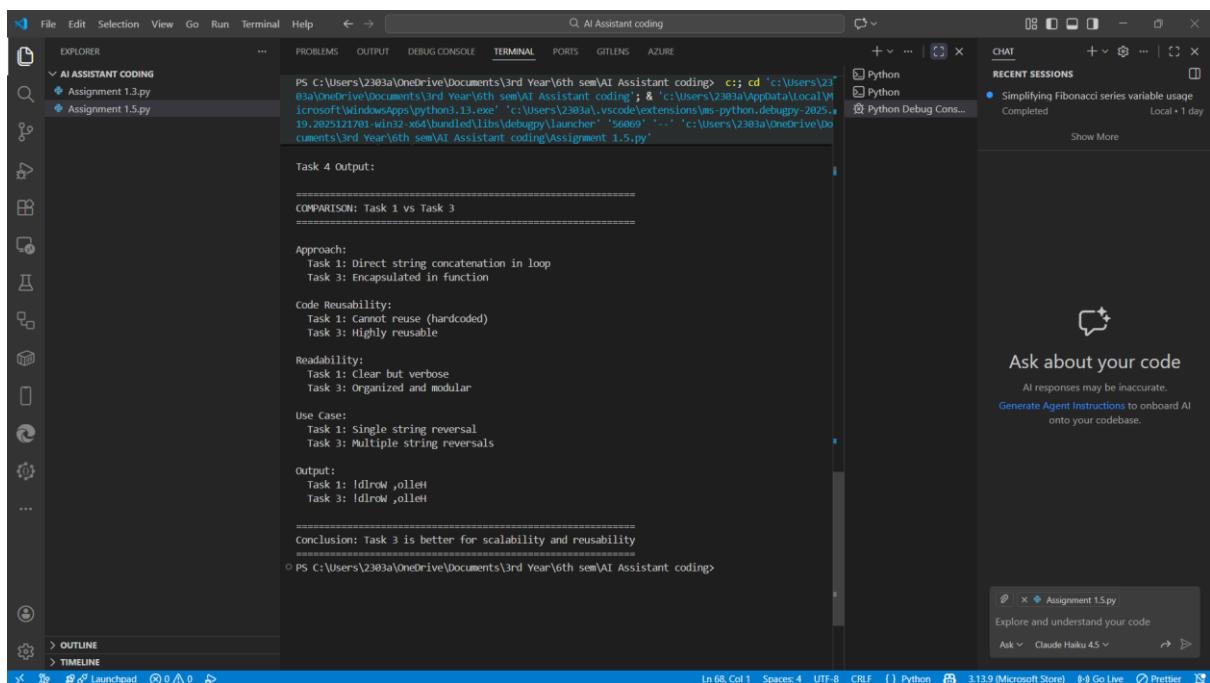
compare the code of task 1 and task 3 and print the comparison in a tabular format

Code :



```
File Edit Selection View Go Run Terminal Help < > Q: AI Assistant coding
EXPLORER AI ASSISTANT CODING Assignment 1.3.py Assignment 1.5.py
RECENT SESSIONS Simplifying Fibonacci series variable usage Completed Local + 1 day Show More
Assignment 1.5.py Assignment 1.5.py...
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 for i, aspect in enumerate(comparison_data["Aspect"]):
61     print(f"\n{aspect}:")
62     print(f" Task 1: {comparison_data['Task 1 (Direct Reversal)'][i]}")
63     print(f" Task 3: {comparison_data['Task 3 (Function-based)'][i]}")
64
65 print("\n" + "="*60)
66 print("Conclusion: Task 3 is better for scalability and reusability")
67 print("="*60)
68
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS AZURE
Reversed string using function: !dirrow ,olleH
PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>
Ln 35, Col 8 Spaces: 4 UTF-8 CRLF { } Python 🏛 3.13.9 (Microsoft Store) ⚡ Go Live ⚡ Prettier 🎨
```

Output :



```
File Edit Selection View Go Run Terminal Help < > Q: AI Assistant coding
EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS AZURE
RECENT SESSIONS Simplifying Fibonacci series variable usage Completed Local + 1 day Show More
Assignment 1.3.py Assignment 1.5.py
Assignment 1.5.py Assignment 1.5.py...
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'; & python -m venv venv; & .\venv\Scripts\activate; & python 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: !dirrow ,olleH
Task 3: !dirrow ,olleH
=====
Conclusion: Task 3 is better for scalability and reusability
=====

PS C:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>
Ln 68, Col 1 Spaces: 4 UTF-8 CRLF { } Python 🏛 3.13.9 (Microsoft Store) ⚡ Go Live ⚡ Prettier 🎨
```

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 screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File Explorer:** Shows two files: "Assignment 1.3.py" and "Assignment 1.5.py".
- Code Editor:** Displays the content of "Assignment 1.5.py".
- Output Panel:** Shows the output of the code execution.
- Terminal:** Shows the command "python Assignment 1.5.py".
- Status Bar:** Shows the file path "C:\Users\user\Assignment 1.5.py", line count (18), column count (90), and other status information.

```
60 # Task 5: Different Algorithmic Approaches to String Reversal
61 #prompt - use Different Algorithmic Approaches to String Reversal and the output should contain as Two
62 # correct implementations
63 # Comparison discussing:
64 # Execution flow
65 # Time complexity
66 # Performance for large inputs
67 # When each approach is appropriate
68
69 print("\n" + "="*80)
70 print("TASK 5: ALGORITHMIC APPROACHES TO STRING REVERSAL")
71 print("="*80)
72
73 # Approach 1: Recursion-based reversal
74 def reverse_recursive(s):
75     if len(s) == 0:
76         return s
77     return reverse_recursive(s[1:]) + s[0]
78
79 # Approach 2: Stack-based reversal
80 def reverse_stack(s):
81     stack = list(s)
82     reversed_s = ""
83     while stack:
84         reversed_s += stack.pop()
85     return reversed_s
86
87 test_string = "Hello, World!"
88
89 print("\nAPPROACH 1: Recursion-based")
90 print(f"Input: {test_string}")
91 print(f"Output: {reverse_recursive(test_string)}")
92 print("Execution Flow: Function calls itself with substring s[1:], appends s[0] at each level")
93 print("Time Complexity: O(n^2) - string concatenation is O(n) per call")
94 print("Performance: Slow for large inputs, risk of stack overflow")
95
96 print("\nAPPROACH 2: Stack-based")
97 print(f"Input: {test_string}")
98 print(f"Output: {reverse_stack(test_string)}")
99 print("Execution Flow: Push all characters to stack, pop each character in reverse order")
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer (Left):** Shows the project structure with files `Assignment 1.3.py` and `Assignment 1.5.py`.
- Code Editor (Center):** Displays the content of `Assignment 1.5.py`. The code prints various performance metrics and complexity analyses for stack-based operations.
- Terminal (Bottom):** Shows the command `ai assistant coding` being run.
- AI Assistant Coding (Right):** A sidebar titled "RECENT SESSIONS" lists a completed session: "Simplifying Fibonacci series variable usage". It also includes a "CHAT" section with a message from "AI Assistant" and a button to "Ask about your code". Below this, there's a note about AI responses being inaccurate and instructions to "Generate Agent Instructions to onboard AI onto your codebase".

Output :

The screenshot shows the Microsoft Visual Studio Code interface with the following details:

- File, Edit, Selection, View, Go, Run, Terminal, Help** menu bar.
- AI Assistant coding** status bar.
- EXPLORER** sidebar on the left.
- PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, GITLENS, AZURE** tabs at the top.
- TERMINAL** tab active, displaying command-line 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\python_3.13.exe 'c:\Users\2303a\vscodeextensions\ms-python.python\debugpy-2025.19.2025121701-win32-x64\bundled\libs\debugpy\launcher' '64512' ... 'c:\Users\2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding\Assignment 1.5.py'
```
- TASK 5: ALGORITHMIC APPROACHES TO STRING REVERSAL** section.
- APPROACH 1: Recursion-based** section:

Input: Hello, World!
Output: olleH
Execution Flow: Function calls itself with substring s[1:], appends s[0] at each level
Time Complexity: $O(n^2)$ - string concatenation is $O(n)$ per call
Performance: Slow for large inputs, risk of stack overflow
- APPROACH 2: Stack-based** section:

Input: Hello, World!
Output: 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** section:

Aspect	Recursion	Stack-based
Execution Flow	Self-referencing calls	Iterative pop ops
Time Complexity	$O(n^2)$	$O(n)$
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>**
- CHAT** sidebar on the right.
- RECENT SESSIONS** sidebar on the right.
- AI Assistant coding** status bar.

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.

Each approach is chosen based on performance needs and code clarity.