

AI Assisted Coding

Assignment 1.5

Name: Ch.Koushik

Hall ticket no: 2303A51938

Batch no: 19

Task 1:

Prompt:

Generate string reversal without using functions

Code & Output:

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files: 'Assignment 1.3.py' and 'Assignment 1.5.py'. The main editor area contains the following Python code:

```
#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
```

The terminal at the bottom 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/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>
```

A floating window titled 'Ask about your code' is visible on the right side of the screen.

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: !dlrow ,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, -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> & "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" "55e11" ... '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
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. The left sidebar has a 'RECENT SESSIONS' section with a single entry: 'Simplifying Fibonacci series variable usage' (Completed, Local + 1 day). The main area displays the code for 'Assignment 1.5.py'. The code includes three approaches for reversing a string: a simple loop, a join operation, and slicing. It also defines a function 'reverse_string' for task 3. The terminal at the bottom shows the execution of the script and its output. A floating 'AI Assistant coding' window is visible on the right, with a message 'Ask about your code' and a note that AI responses may be inaccurate.

```
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
```

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 :

The screenshot shows a code editor with Python code for comparing two tasks. The code uses a dictionary to map aspects to their descriptions for both Task 1 (Direct Reversal) and Task 3 (Function-based). It then prints out the descriptions for each aspect and concludes that Task 3 is better for scalability and usability.

```
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 usability")  
67     print("="*60)  
68  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS AZURE  
Reversed string using function: ldrWolleH  
PS C:\Users\2303a\OneDrive\Documents\3rd Year\eth sem\AI Assistant coding>  
+ ... | x  
Python Python Python Debug Cons...  
Assignment 1.5.py  
Explore and understand your code  
Ask Claude Haiku 4.5 Go Live Prettier  
Ln 35, Col 8 Spaces: 4 UFT-8 CRLF { Python 3.13.9 (Microsoft Store) ⌂ Go Live ⌂ Prettier
```

Output :

The screenshot shows the Visual Studio Code (VS Code) interface with the "AI Assistant coding" extension installed. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and several icons for navigating between files and tabs. The left sidebar features the Explorer, Search, and Problems sections, along with a large icon representing the AI Assistant.

The main workspace displays a terminal window with the following command and output:

```
PS C:\Users\j2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding> c;; cd 'c:\Users\j2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding'; & "c:\Users\j2303a\AppData\Local\Microsoft\WindowsApps\python3.13.exe" "c:\Users\j2303a\.vscode\extensions\ms-python.python.debug-2025.19.2025121701-win32\x64\bundle\libs\debug\launcher" "56069" ... "c:\Users\j2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding\Assignment 1.5.py"
```

Below the terminal, there's a section titled "Task 4 Output:" which contains the following text:

```
=====  
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:

```
PS C:\Users\j2303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>
```

Conclusion: Task 3 is better for scalability and reusability

Bottom right corner: Assignment 1.5.py

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 command "3.13.9 (Microsoft Store)".
- Bottom Status Bar:** Shows "Ln 90 Col 20 Spaces:4 UTF-8 CRLF {} Python 3.13.9 (Microsoft Store) Go Live Prettier".

```
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")
```

```

105 print("\nAPPROACH 2: Stack-based")
106 print("Input: {test_string}")
107 print("Output: {reverse_stack(test_string)}")
108 print("Execution Flow: Push all characters to stack, pop each character in reverse order")
109 print("Time Complexity: O(n) - single pass through string")
110 print("Performance: Better than recursion, suitable for large inputs")
111
112 print("\n" + "="*80)
113 print("COMPARISON TABLE")
114 print("-"*80)
115 print("{'Aspect':<25} | {'Recursion':<30} | {'Stack-based':<30}")
116 print("-" * 90)
117 print("{'Execution Flow':<25} | {'Self-referencing calls':<30} | {'Iterative pop ops':<30}")
118 print("{'Time complexity':<25} | {'O(n^2)':<30} | {'O(n)':<30}")
119 print("{'Space complexity':<25} | {'O(n) call stack':<30} | {'O(n) stack data':<30}")
120 print("{'Large Input (1M chars)':<25} | {'Very Slow/Risk crash':<30} | {'Fast & Safe':<30}")
121 print("{'When Appropriate':<25} | {'Educational, Small data':<30} | {'Production, All sizes':<30}")
122 print("-" * 80)
123
124 print("\nConclusion: Stack-based approach is superior for real-world applications")

```

Ask about your code
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

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' & & python -m venv venv & & venv\Scripts\python.exe -m vscode_extensions.vs-python.debugger --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  

-----  

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(n <sup>2</sup> )      | 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

```

Ask about your code
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

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.