

AI Assisted Coding

Assignment 1.5

Name: V.Harivamsh

Hall ticket no: 2303A51266

Batch no: 19

Task 1:

Prompt:

Generate string reversal without using functions

Code& Output:

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar displays the file tree with 'Assignment 1.5.py' selected. The main editor window 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>
```

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:

```
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\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>

Ask about your code

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

Python Python Python Debug Cons...

Assignment 1.5.py Explore and understand your code

Ask Claude Haiku 4.5

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:

The screenshot shows the Microsoft Visual Studio Code interface with the 'AI ASSISTANT CODING' extension active. The 'Assignment 1.5.py' file is open in the editor, containing Python code for string reversal. The 'TERMINAL' tab shows the execution of the script, displaying the output of three tasks: Task 1 (original string), Task 2 (optimized reversed string), and Task 3 (Pythonic reversed string). A sidebar on the right titled 'Ask about your code' is visible, along with a message indicating 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
```

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.

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:

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar has a tree view with 'AI ASSISTANT CODING' expanded, showing 'Assignment 1.3.py' and 'Assignment 1.5.py'. The main editor window displays the following Python 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 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
```

The terminal below shows the output of running the script:

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

Output :

The screenshot shows the Microsoft Visual Studio Code interface with the terminal tab selected. The terminal output is as follows:

```
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 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>
```

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:

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 execution results of the code.
- 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")
```

```
Assignment 1.5.py Assignment 1.5.py\

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(n2):<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

All responses may be inaccurate.

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.
- PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, GITLINS, AZURE** tabs at the top.
- EXPLORER** sidebar on the left.
- TERMINAL** tab is active, displaying a PowerShell session (PS) with the command: `PS C:\Users\z303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding & cd 'c:\Users\z303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding' & & "C:\Users\z303a\AppData\Local\Microsoft\WindowsApps\python3.13.exe" "C:\Users\z303a\.vscode\extensions\ms-python.python.debug-2025.19.2025121701-win32\x64\bundled\libs\debugpy\launcher" "64512" ... "C:\Users\z303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding\Assignment 1.5.py"`.
- CHAT** sidebar on the right.
- RECENT SESSIONS** list: Simplifying Fibonacci series variable usage (Completed), Local 1 day.
- TASK 5: ALGORITHMIC APPROACHES TO STRING REVERSAL** section:

 - APPROACH 1: Recursion-based**
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**
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 (JM 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\z303a\OneDrive\Documents\3rd Year\6th sem\AI Assistant coding>**
- Assignment 1.5.py** preview in the bottom right corner.

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.