

AI ASSISTED CODING

2303A52315

BATCH 45

Task 1: AI-Generated Logic Without Modularization (String Reversal Without Functions)

Scenario

You are developing a basic text-processing utility for a messaging application.

Task Description

Use GitHub Copilot to generate a Python program that:

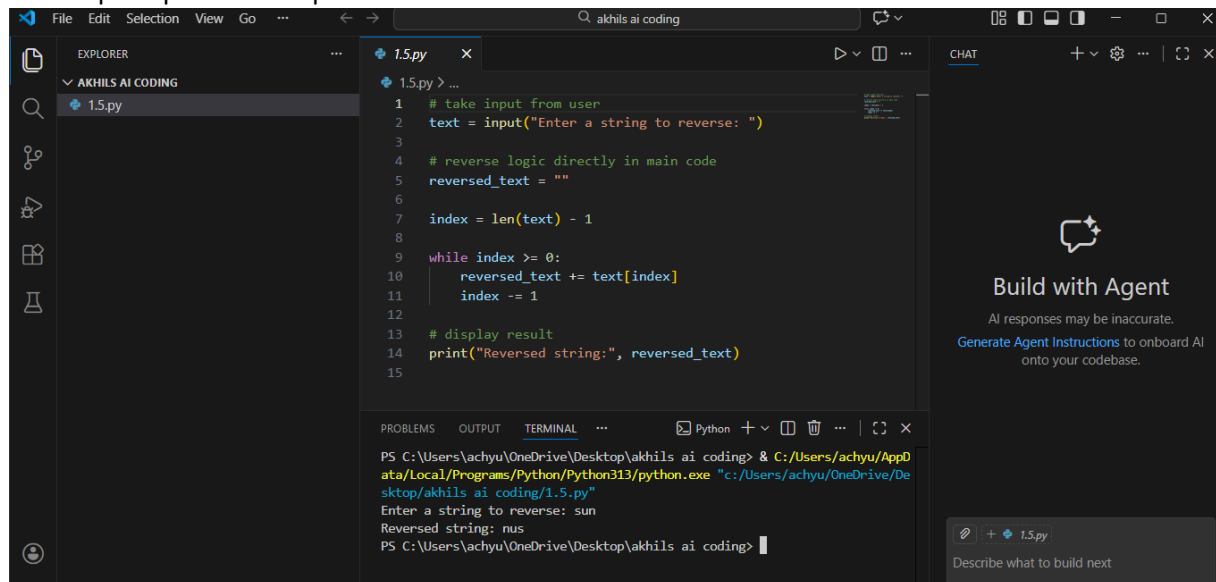
Reverses a given string Accepts user input

Implements the logic directly in the main code

Does not use any user-defined functions

❖ Expected Output

- Correct reversed string
- Screenshots showing Copilot-generated code suggestions
- Sample inputs and outputs



The screenshot displays the Visual Studio Code interface. The Explorer pane on the left shows a file named '1.5.py' under the 'AKHILS AI CODING' folder. The main editor window shows the following Python code:

```
1 # take input from user
2 text = input("Enter a string to reverse: ")
3
4 # reverse logic directly in main code
5 reversed_text = ""
6
7 index = len(text) - 1
8
9 while index >= 0:
10     reversed_text += text[index]
11     index -= 1
12
13 # display result
14 print("Reversed string:", reversed_text)
15
```

The bottom panel shows the TERMINAL output:

```
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/1.5.py"
Enter a string to reverse: sun
Reversed string: nus
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>
```

On the right side of the editor, the 'CHAT' pane is visible with the heading 'Build with Agent' and a prompt: 'Describe what to build next'.

Task 2: Efficiency & Logic Optimization (Readability Improvement)

Scenario

The code will be reviewed by other developers.

Task Description

Examine the Copilot-generated code from Task 1 and improve it by:

Removing unnecessary variables

Simplifying loop or indexing logic

Improving readability

Use Copilot prompts like:

- “Simplify this string reversal code”
- “Improve readability and efficiency”

Hint:

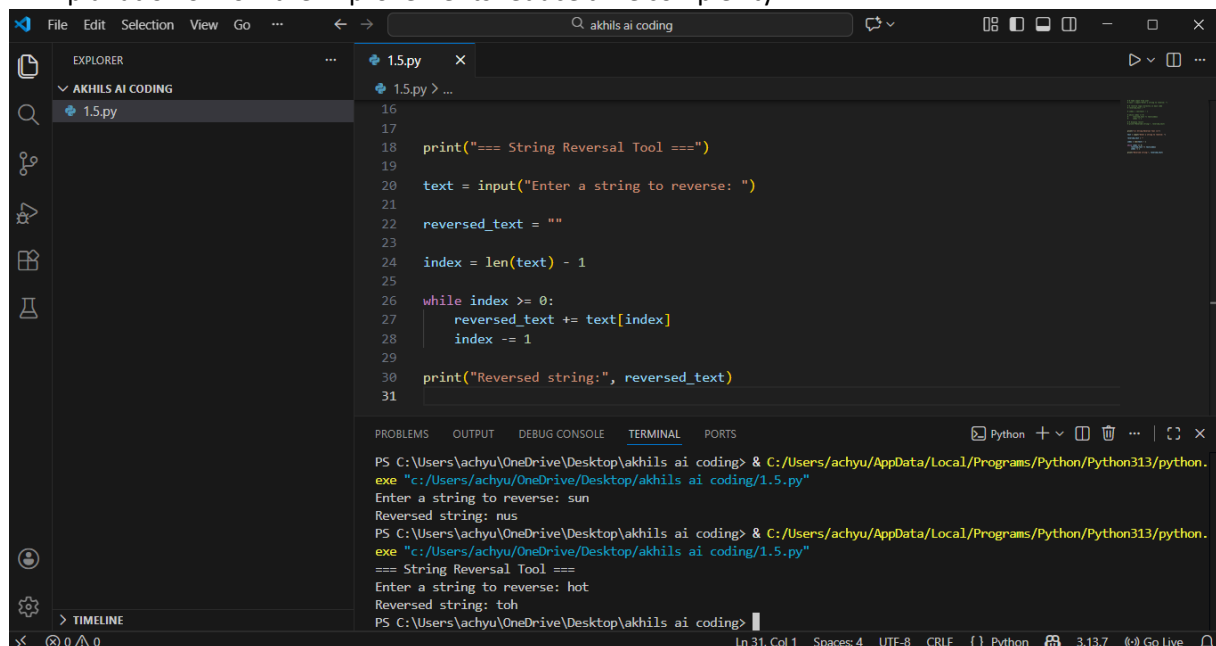
Prompt Copilot with phrases like

“optimize this code”, “simplify logic”, or “make it more readable”

❖ Expected Output

Original and optimized code versions

➤ Explanation of how the improvements reduce time complexity



The screenshot shows a Visual Studio Code editor with a file named `1.5.py` open. The code in the editor is as follows:

```
16
17
18 print("=== String Reversal Tool ===")
19
20 text = input("Enter a string to reverse: ")
21
22 reversed_text = ""
23
24 index = len(text) - 1
25
26 while index >= 0:
27     reversed_text += text[index]
28     index -= 1
29
30 print("Reversed string:", reversed_text)
31
```

Below the editor, the terminal window shows the execution of the script. It prompts the user to enter a string, and the output shows the reversed string for two inputs: 'sun' and 'toh'.

```
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/1.5.py"
Enter a string to reverse: sun
Reversed string: nus
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/1.5.py"
=== String Reversal Tool ===
Enter a string to reverse: toh
Reversed string: hto
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>
```

Task 3: Modular Design Using AI Assistance (String Reversal Using Functions)

Scenario

The string reversal logic is needed in multiple parts of an application.

Task Description

Use GitHub Copilot to generate a function-based Python program that:

Uses a user-defined function to reverse a string

Returns the reversed string

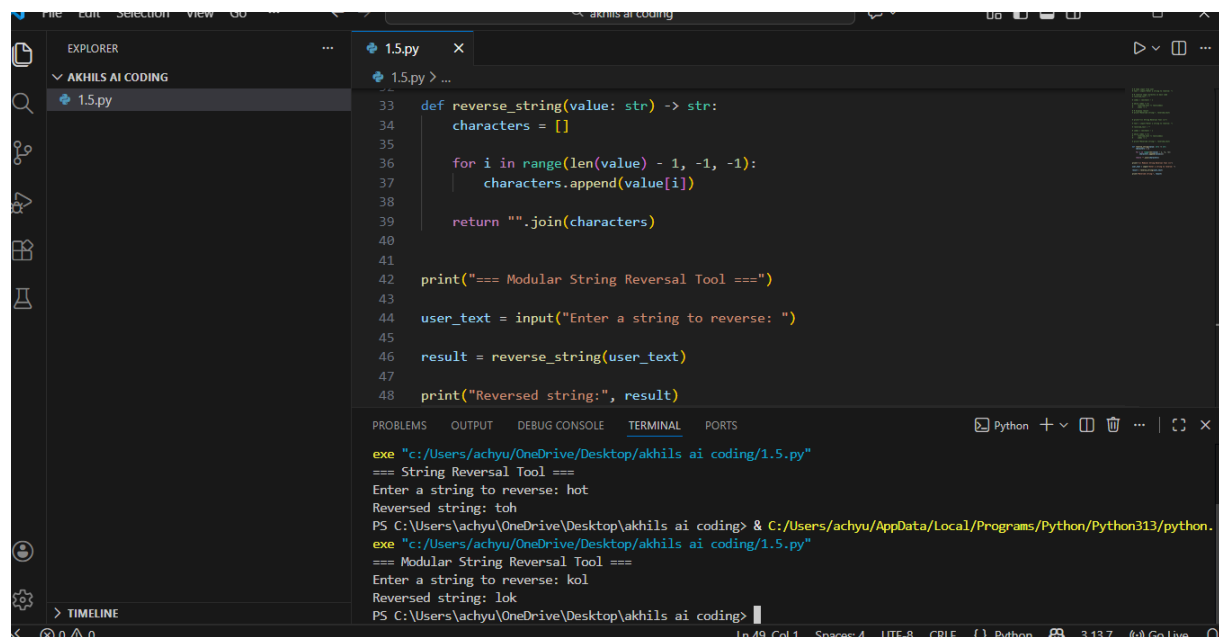
Includes meaningful comments (AI-assisted)

Expected Output

Correct function-based implementation

Screenshots documenting Copilot's function generation

Sample test cases and outputs



The screenshot shows a Visual Studio Code editor with a file named `1.5.py` open. The code defines a function `reverse_string` that takes a string `value` and returns its reverse. The function uses a list `characters` to store characters in reverse order. The main script prompts the user to enter a string, calls the function, and prints the result. The terminal output shows two test cases: reversing 'hot' to 'toh' and 'kol' to 'lok'.

```
33 def reverse_string(value: str) -> str:
34     characters = []
35
36     for i in range(len(value) - 1, -1, -1):
37         characters.append(value[i])
38
39     return "".join(characters)
40
41
42 print("=== Modular String Reversal Tool ===")
43
44 user_text = input("Enter a string to reverse: ")
45
46 result = reverse_string(user_text)
47
48 print("Reversed string:", result)
```

Terminal Output:

```
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/1.5.py"
=== String Reversal Tool ===
Enter a string to reverse: hot
Reversed string: toh
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding> python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/1.5.py"
=== Modular String Reversal Tool ===
Enter a string to reverse: kol
Reversed string: lok
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding>
```

Task 4: Comparative Analysis – Procedural vs Modular Approach (With vs Without Functions)

❖ Scenario

You are asked to justify design choices during a code review.

❖ Task Description

Compare the Copilot-generated programs:

➤ Without functions (Task 1)

➤ With functions (Task 3)

Analyze them based on:

➤ Code clarity

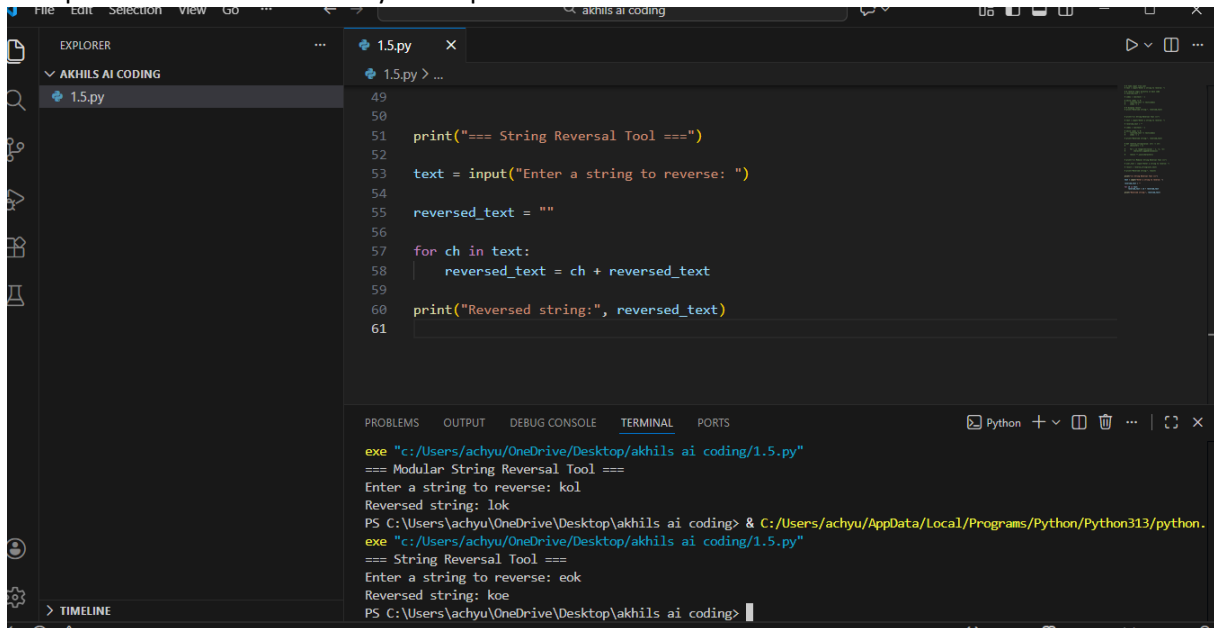
➤ Reusability

➤ Debugging ease

➤ Suitability for large-scale applications

❖ Expected Output

Comparison table or short analytical report



The screenshot shows a VS Code editor with a file named '1.5.py' open. The code in the file is as follows:

```
49
50
51 print("=== String Reversal Tool ===")
52
53 text = input("Enter a string to reverse: ")
54
55 reversed_text = ""
56
57 for ch in text:
58     reversed_text = ch + reversed_text
59
60 print("Reversed string:", reversed_text)
61
```

The terminal at the bottom shows the execution of the script:

```
exe "C:/Users/achyu/OneDrive/Desktop/akhils ai coding/1.5.py"
=== Modular String Reversal Tool ===
Enter a string to reverse: kol
Reversed string: lok
PS C:/Users/achyu/OneDrive/Desktop/akhils ai coding> & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "C:/Users/achyu/OneDrive/Desktop/akhils ai coding/1.5.py"
=== String Reversal Tool ===
Enter a string to reverse: eok
Reversed string: koe
PS C:/Users/achyu/OneDrive/Desktop/akhils ai coding>
```

Task 5: AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches to String Reversal)

❖ Scenario

Your mentor wants to evaluate how AI handles alternative logic paths.

❖ Task Description

Prompt GitHub Copilot to generate:

- A loop-based string reversal approach
- A built-in / slicing-based string reversal approach

❖ Expected Output

➤ Two correct implementations

➤ Comparison discussing:

- Execution flow
- Time complexity
- Performance for large inputs
- When each approach is appropriate

Note: Report should be submitted as a word document for all tasks in a

and if required, screenshots

