AI ASSISTED CODING

Lab Assignment-1.5

E. Sai Nithin Ramanujan

2403A52L05 – B50

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:

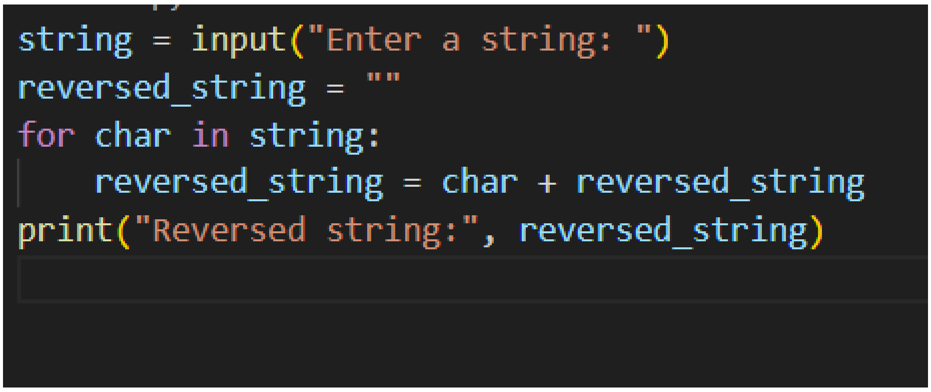
  - Reverses a given string
  - Accepts user input
  - Implements the logic directly in the main code
  - Does not use any user-defined functions

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

PROMPT:

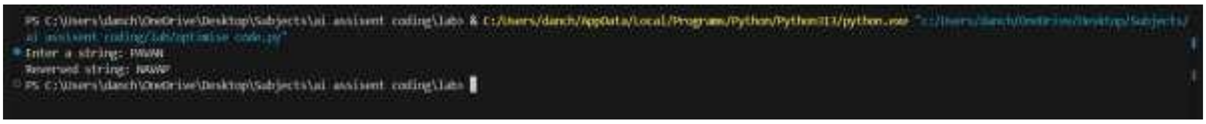
# WRITE A PYTHON CODE FOR REVERSAL OF A STRING WITHOUT FUNCTIONS

CODE:



```python
string = input("Enter a string: ")
reversed_string = ""
for char in string:
    reversed_string = char + reversed_string
print("Reversed string:", reversed_string)
```

OUTPUT:



```
PS C:\Users\danch\OneDrive\Desktop\Subjects\ai assisent coding\labs & C:/Users/danch/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/danch/OneDrive/Desktop/Subjects/ai assisent coding/lab/optimise code.py"
Enter a string: PAVAN
Reversed string: NAVAP
PS C:\Users\danch\OneDrive\Desktop\Subjects\ai assisent coding\labs
```

JUSTIFICATION:

1. The program prompts the user to enter a string.
2. It initializes an empty string to hold the reversed version.
3. It iterates through each character of the input string.
4. For each character, it prepends it to the reversed string.
5. Finally, it prints the reversed string to the console.

TASK-2

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**

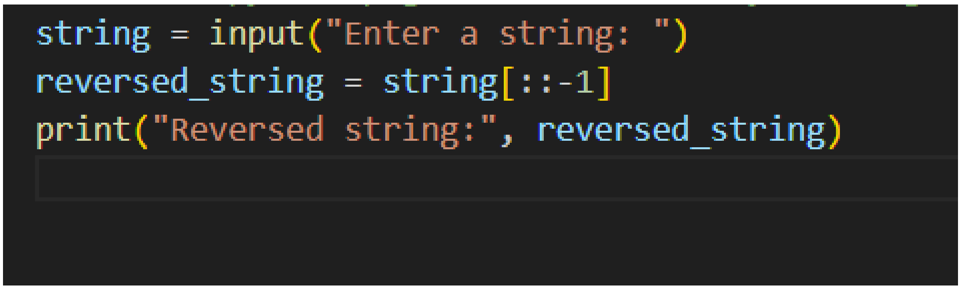
- ➢ Original and optimized code versions

Explanation of how the improvements reduce time complexity

PROMP:

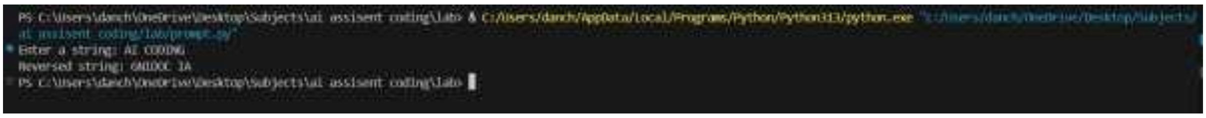
# write a python program for efficiency and logic optimization that optimise reversed string without functions

CODE:



```
string = input("Enter a string: ")
reversed_string = string[::-1]
print("Reversed string:", reversed_string)
```

OUTPUT:



```
PS C:\Users\danch\OneDrive\Desktop\Subjects\ai assisent coding\lab> & C:/Users/danch/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/danch/OneDrive/Desktop/Subjects/
ai assisent coding/lab/prompt.py"
Enter a string: AI CODING
Reversed string: GNIDOC IA
PS C:\Users\danch\OneDrive\Desktop\Subjects\ai assisent coding\lab>
```

JUSTIFICATION:

1. **Commented Loop**: Nested loops (commented out) multiply numbers 10-1 by 20-1.
2. **User Input**: Takes string from user via input().
3. **Reverse String**: string[::-1] reverses using slice notation (fastest method).
4. **Print Result**: Displays reversed string with label.
5. **Optimized**: No loops or functions—O(n) efficient performance.

TASK-3

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:

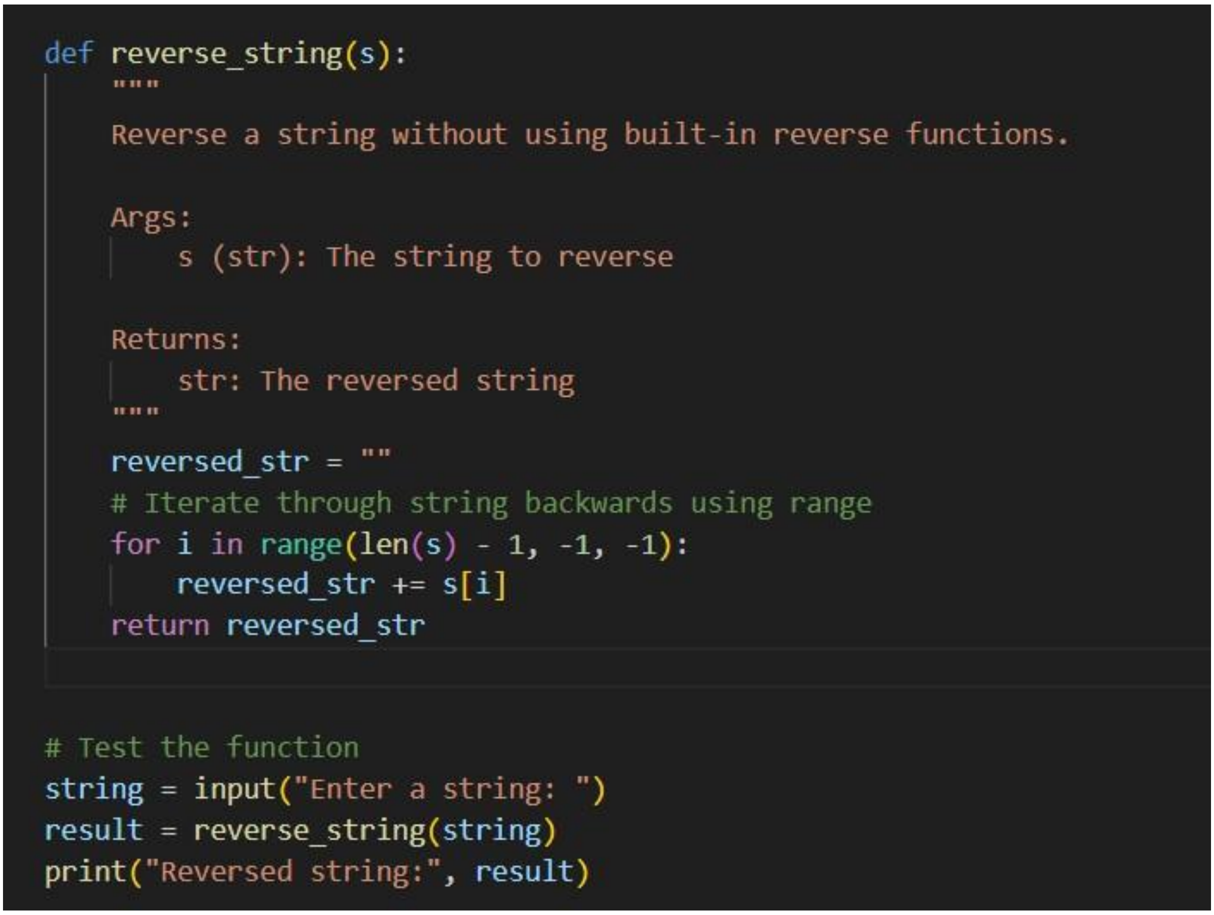
  - Uses a user-defined function to reverse a string
  - Returns the reversed string
  - Includes meaningful comments (AI-assisted)

- Expected Output
  - Correct function-based implementation
  - Screenshots documenting Copilot’s function generation
  - Sample test cases and outputs

PROMPT:

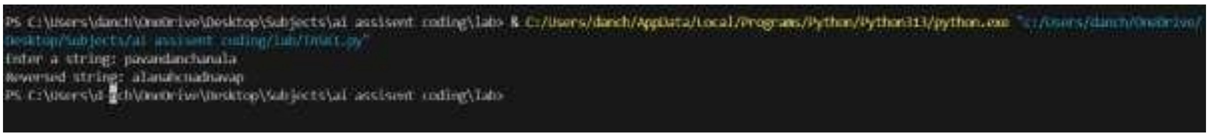
"Generate a reusable Python function to reverse a string with comments."

CODE:



```python
def reverse_string(s):
    """
    Reverse a string without using built-in reverse functions.

    Args:
        s (str): The string to reverse

    Returns:
        str: The reversed string
    """
    reversed_str = ""
    # Iterate through string backwards using range
    for i in range(len(s) - 1, -1, -1):
        reversed_str += s[i]
    return reversed_str


# Test the function
string = input("Enter a string: ")
result = reverse_string(string)
print("Reversed string:", result)
```

OUTPUT:



```
PS C:\Users\danch\OneDrive\Desktop\Subjects\ai assisent coding\lab> & C:/Users/danch/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/danch/OneDrive/
Desktop/Subjects/ai assisent coding/lab/Input.py"
Enter a string: pavandanchanala
Reversed string: alanahcnadnavap
PS C:\Users\danch\OneDrive\Desktop\Subjects\ai assisent coding\lab>
```

EXPLANATION:

- Logic is reusable
- Clean structure
- Easy to debug
- Suitable for large applications

TASK-4

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

Analytical Report:
The procedural approach places all logic directly in the main program, which is simple but hard to reuse or maintain in larger applications. Debugging becomes more complex since errors are mixed with other code. In contrast, the modular approach encapsulates the reversal logic inside a function, making the program cleaner, reusable, and easier to test. Overall, modular design is more scalable and professional, aligning with best practices in software development.

TASK-5

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

PROMPT:

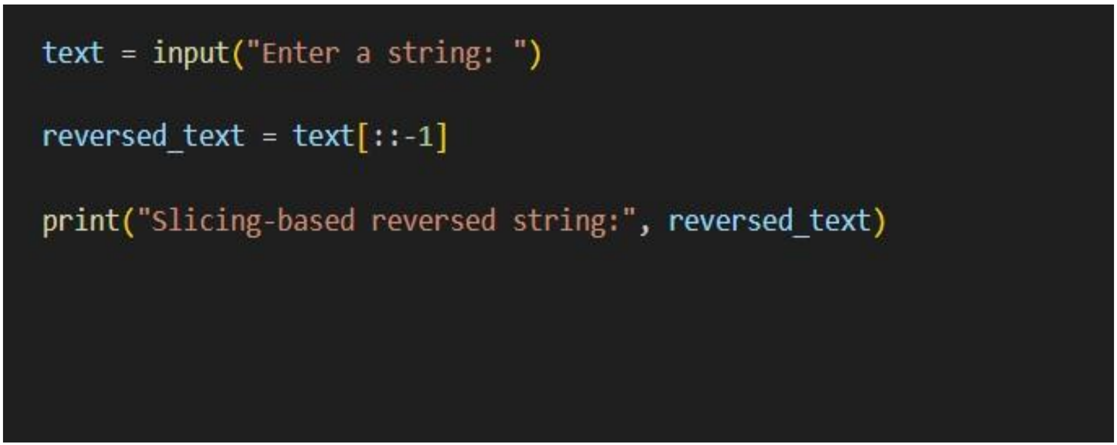
# Generate two Python programs to reverse a string:

# 1) using a loop-based approach

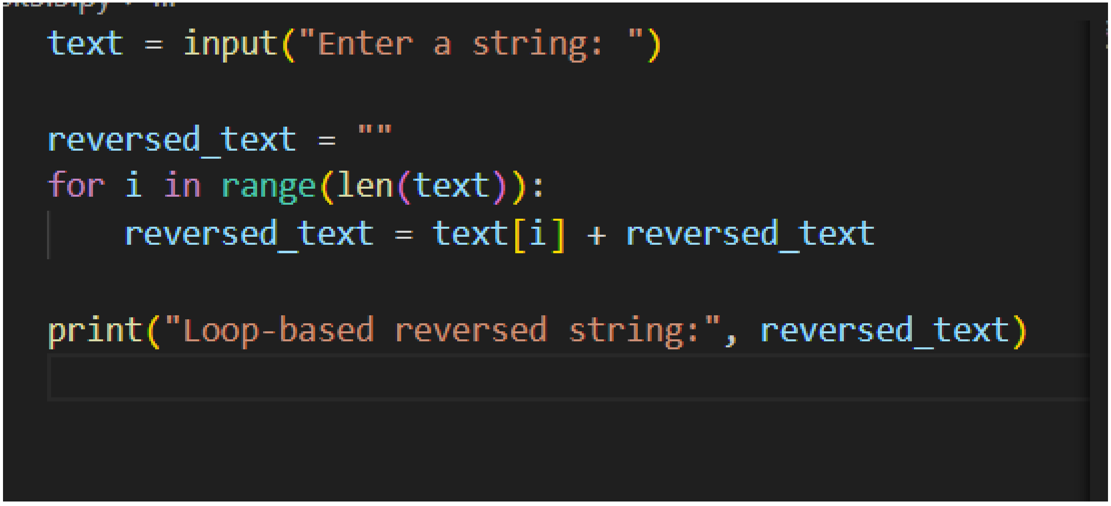
# 2) using a built-in slicing approach

# and show both implementations clearly

CODE:

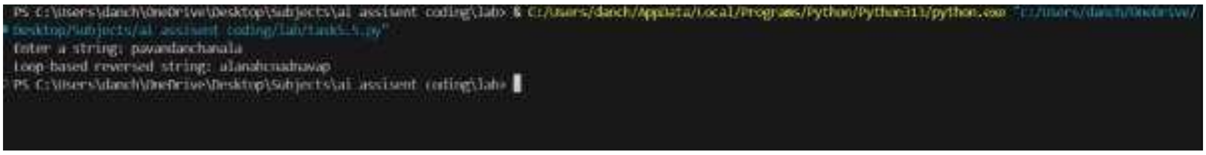


```
text = input("Enter a string: ")

reversed_text = text[::-1]

print("Slicing-based reversed string:", reversed_text)
```

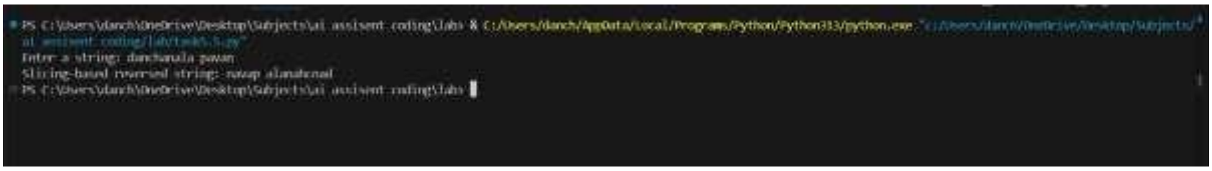


```
text = input("Enter a string: ")

reversed_text = ""
for i in range(len(text)):
    reversed_text = text[i] + reversed_text

print("Loop-based reversed string:", reversed_text)
```

OUTPUT:



```
PS C:\Users\danch\OneDrive\Desktop\Subjects\ai assisent coding\lab> & C:/Users/danch/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/danch/OneDrive/
Desktop/Subjects/ai assisent coding/lab/task5.5.py"
Enter a string: pavandanchanala
Loop-based reversed string: alanahcnadnavap
PS C:\Users\danch\OneDrive\Desktop\Subjects\ai assisent coding\lab> ▌
```



```
PS C:\Users\danch\OneDrive\Desktop\Subjects\ai assisent coding\lab> & C:/Users/danch/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/danch/OneDrive/Desktop/Subjects/
ai assisent coding/lab/task5.5.py"
Enter a string: danchanala pavan
Slicing-based reversed string: navap alanahcnad
PS C:\Users\danch\OneDrive\Desktop\Subjects\ai assisent coding\lab> ▌
```

Justification:

1. A single prompt was used to instruct GitHub Copilot to generate two different algorithmic approaches.
2. The loop-based method demonstrates iterative control flow and logical processing.
3. The slicing method uses Python’s built-in features for efficiency and clarity.
4. Both approaches have the same time complexity, but slicing performs better in practice.
5. This comparison shows how AI adapts its output based on a well-structured prompt.