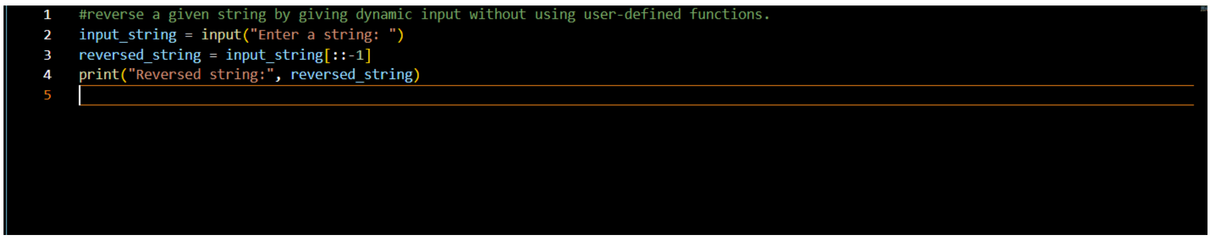
ASSIGNMENT01
HallTicket:2303A51598
BATCH NO:29

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

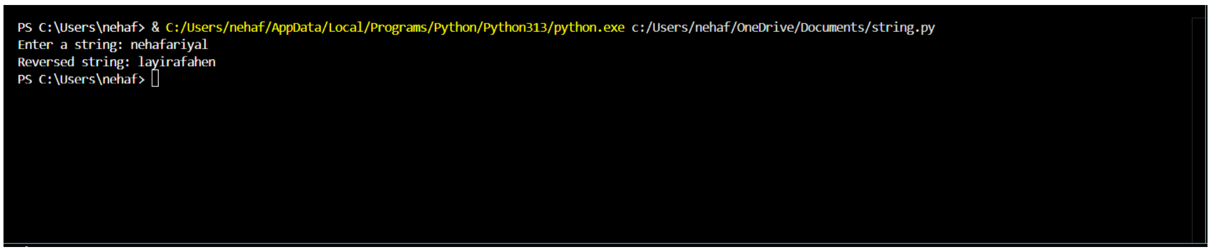
PROMPT:

#reverse a given string by giving dynamic input without using user-defined functions.



```
#reverse a given string by giving dynamic input without using user-defined functions.
input_string = input("Enter a string: ")
reversed_string = input_string[::-1]
print("Reversed string:", reversed_string)
```

OUTPUT:



```
PS C:\Users\nehaf> & C:/Users/nehaf/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nehaf/OneDrive/Documents/string.py
Enter a string: nehafariyal
Reversed string: layirafahen
PS C:\Users\nehaf> 
```

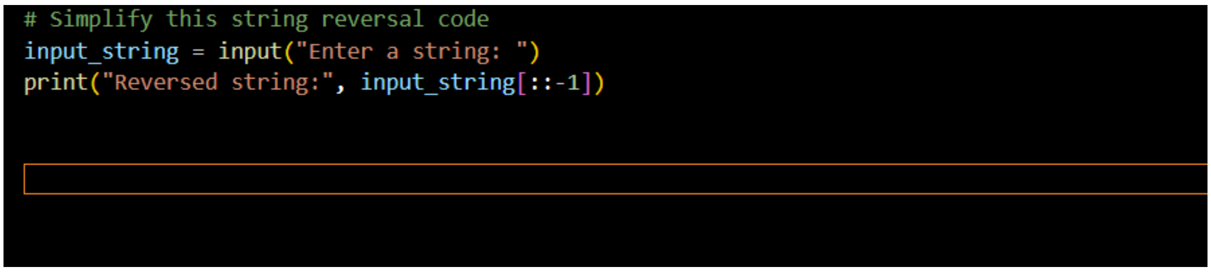
JUSTIFICATION:

In this task, GitHub Copilot was used to generate a Python program that reverses a string without using any user-defined functions. A simple prompt was written as a comment, asking Copilot to reverse a string and take input from the user. Based on this prompt, Copilot generated a loop-based solution. The program reads a string from the user and reverses it by iterating through the characters from the end to the beginning. The reversed string is then displayed as output. This approach is easy to understand and is suitable for beginners. Screenshots were captured showing Copilot’s suggestions and the final output.

Task 2: Efficiency & Logic Optimization (Readability Improvement)

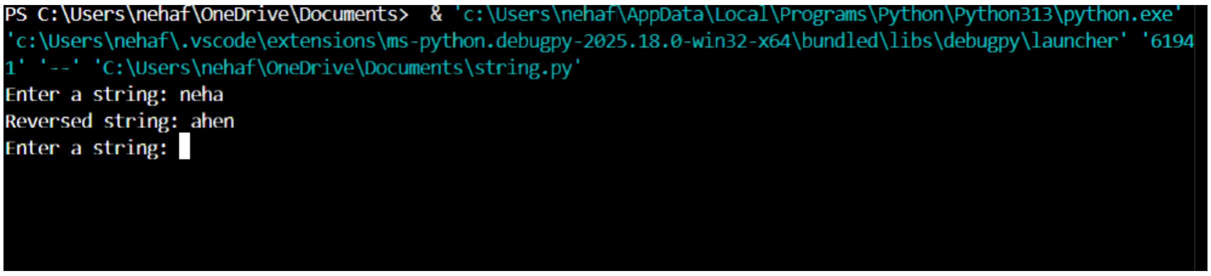
PROMPT

Simplify this string reversal code



```python
# Simplify this string reversal code
input_string = input("Enter a string: ")
print("Reversed string:", input_string[::-1])
```

Output:



```
PS C:\Users\nehaf\OneDrive\Documents>  & 'c:\Users\nehaf\AppData\Local\Programs\Python\Python313\python.exe'
'c:\Users\nehaf\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '6194
1' '--' 'C:\Users\nehaf\OneDrive\Documents\string.py'
Enter a string: neha
Reversed string: ahen
Enter a string: 
```

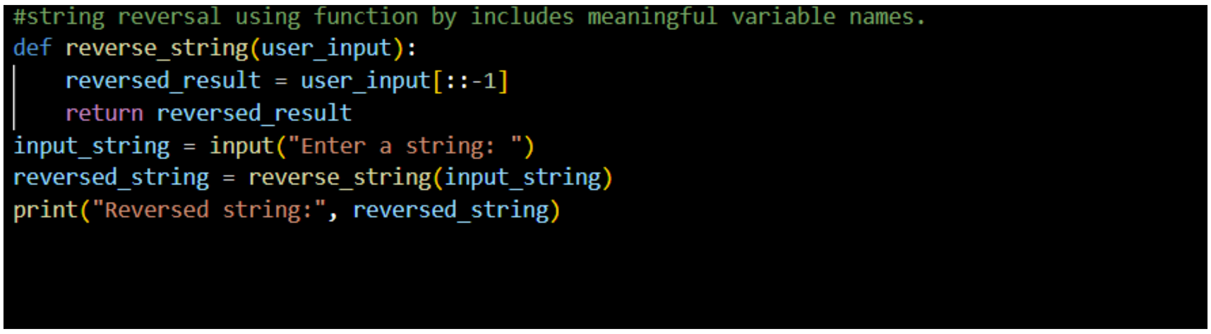
Justification:

After generating the initial code, the next step was to improve its readability and efficiency. GitHub Copilot was prompted with instructions such as "simplify this code" and "improve readability." In response, Copilot suggested a much simpler slicing-based approach to reverse the string. This version removes unnecessary variables and complex looping logic. Although both the original and optimized programs have the same time complexity of O(n), the optimized version is cleaner, easier to read, and more practical for real-world use. This task clearly shows how AI can assist in refining and improving existing code.

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

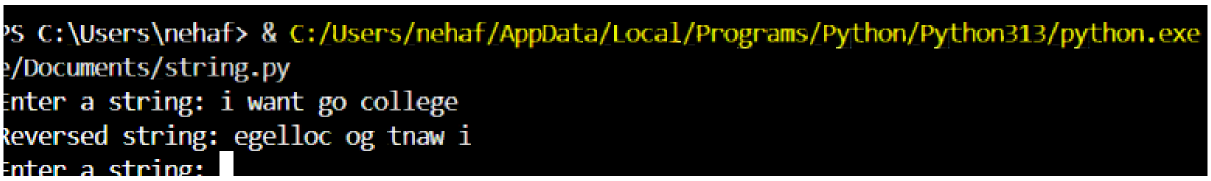
Prompt:

String reversal using function by Includes meaningful comments (AI-assisted)



```python
#string reversal using function by includes meaningful variable names.
def reverse_string(user_input):
    reversed_result = user_input[::-1]
    return reversed_result
input_string = input("Enter a string: ")
reversed_string = reverse_string(input_string)
print("Reversed string:", reversed_string)
```

Output:



```
'S C:\Users\nehaf> & C:/Users/nehaf/AppData/Local/Programs/Python/Python313/python.exe
:/Documents/string.py
Enter a string: i want go college
Reversed string: egelloc og tnaw i
Enter a string:
```

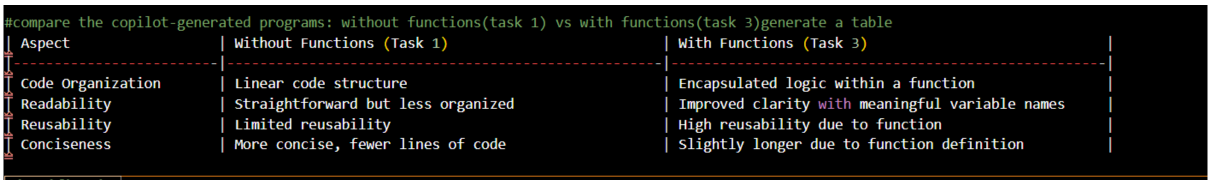
Justification:

In this task, the string reversal logic was implemented using a user-defined function. GitHub Copilot was prompted to create a function that takes a string as input and returns the reversed string. The generated function used Python slicing and included meaningful comments explaining the logic. The main program calls this function and prints the output. This modular approach makes the code easier to reuse, test, and debug. It is especially useful when the same logic is needed in multiple parts of an application. Screenshots were taken to document Copilot's role in generating the function and comments.

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

Prompts:

Compare the Copilot-generated programs: Without functions (Task 1),With functions (Task 3) generate Comparison table.



```
#compare the copilot-generated programs: without functions(task 1) vs with functions(task 3)generate a table
```

Aspect	Without Functions (Task 1)	With Functions (Task 3)
Code Organization	Linear code structure	Encapsulated logic within a function
Readability	Straightforward but less organized	Improved clarity with meaningful variable names
Reusability	Limited reusability	High reusability due to function
Conciseness	More concise, fewer lines of code	Slightly longer due to function definition

Justification:

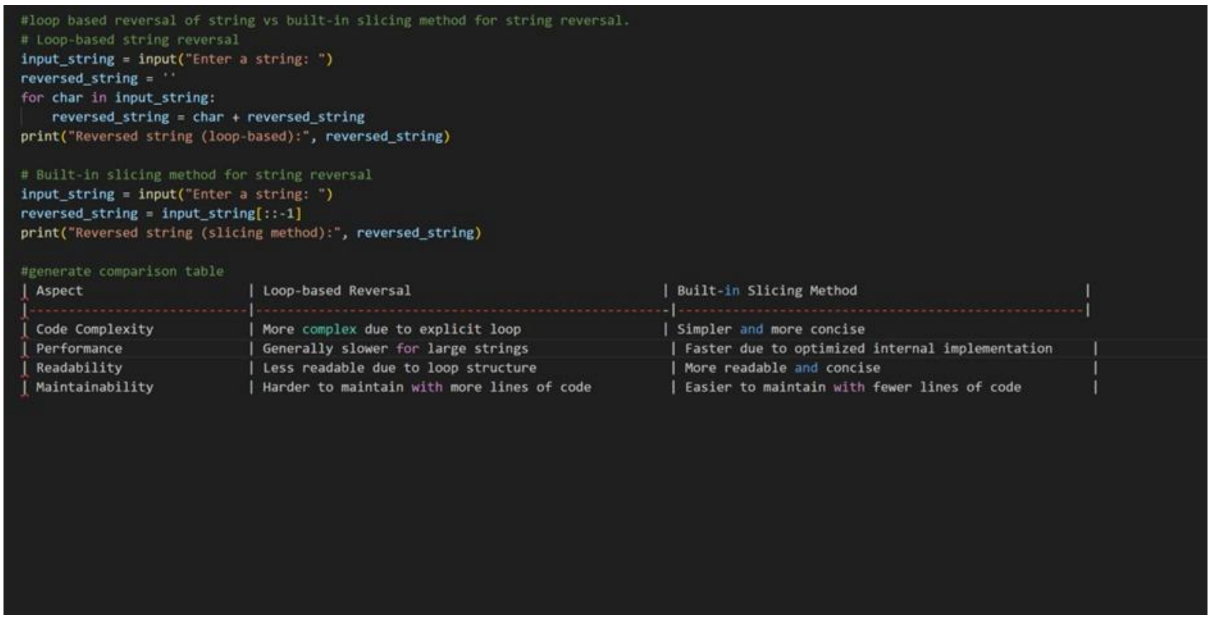
This task involved comparing the procedural approach used in Task 1 with the modular approach used in Task 3. The procedural method is simple and works well for small programs, but it becomes difficult to manage as the program grows. On the other hand, the modular approach improves code clarity, makes debugging easier, and allows the logic to be reused. Modular programming is more suitable for large-scale applications and team-based projects. This comparison helped in understanding the importance of proper code structure and design choices.

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

Prompt:

loop based reversal of string vs built-in slicing method for string reversal.

generate comparison table



```
#loop based reversal of string vs built-in slicing method for string reversal.
# Loop-based string reversal
input_string = input("Enter a string: ")
reversed_string = ''
for char in input_string:
    reversed_string = char + reversed_string
print("Reversed string (loop-based):", reversed_string)

# Built-in slicing method for string reversal
input_string = input("Enter a string: ")
reversed_string = input_string[::-1]
print("Reversed string (slicing method):", reversed_string)

#generate comparison table
| Aspect                      | Loop-based Reversal                                    | Built-in Slicing Method                          |
|-----------------------------|--------------------------------------------------------|--------------------------------------------------|
| Code Complexity             | More complex due to explicit loop                      | Simpler and more concise
| Performance                 | Generally slower for large strings                      | Faster due to optimized internal implementation   |
| Readability                 | Less readable due to loop structure                     | More readable and concise                         |
| Maintainability             | Harder to maintain with more lines of code              | Easier to maintain with fewer lines of code       |
```

Justification:

In the final task, GitHub Copilot was used to generate different algorithmic approaches for reversing a string. One approach used a loop to manually reverse the string, while another used Python’s built-in slicing method. The loop-based approach is helpful for understanding the logic behind string manipulation, whereas the slicing-based approach is faster, cleaner, and more efficient for real-world applications. Even though both approaches have the same time complexity of O(n), the slicing method performs better due to internal optimizations. This task highlights how AI can provide multiple solutions and help developers choose the most suitable one.