

Lab Assignment -1

Name : Gaddam Padmavathi

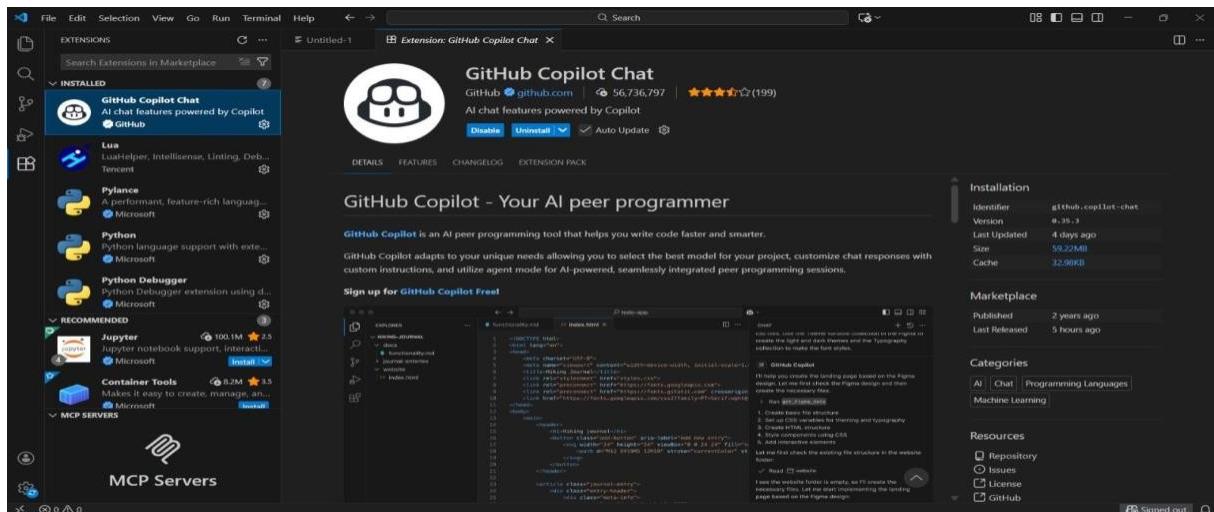
HT.No : 2303A52269

Batch : 36

Environment Setup – *GitHub Copilot and VS Code Integration + Understanding AI-assisted Coding Workflow*

Task 0

Install and configure GitHub Copilot in VS Code. Take screenshots of each step.



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

A screenshot of the VS Code interface showing a Python file named '#Task 1.py'. The code contains a single function that reverses a string input. The terminal at the bottom shows the script running and outputting the reversed string 'padmavathi' when the user inputs 'padmavathi'. The bottom navigation bar includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The status bar indicates the file is in Python mode.

```
C:\ > Users > Padmavathi > #Task 1.py > ...
1  # Task 1
2  # Reverses a given string
3  user_input = input("Enter a string to reverse: ")
4  reversed_string = user_input[::-1]
5  print(f"Reversed string: {reversed_string}")

PS C:\Users\Padmavathi> & C:/Users/Padmavathi/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Padmavathi/#Task 1.py"
Enter a string to reverse: padmavathi
Reversed string: ihtavamadap
PS C:\Users\Padmavathi>
```

Task 2: Efficiency & Logic Optimization (Readability Improvement)

The screenshot shows a code editor interface with two tabs: '#Task 1.py' and '#Task 2.py'. The '#Task 1.py' tab contains the original code, which includes a loop to reverse the string. The '#Task 2.py' tab contains the optimized code, which uses Python's slicing feature to reverse the string directly. Below the code editor is a terminal window showing the execution of both scripts and their output.

```
C: > Users > Padmavathi > #Task 1.py > ...
2  # Reverses a given string
3  user_input = input("Enter a string to reverse: ")
4  reversed_string = user_input[::-1]
5  print(f"Reversed string: {reversed_string}")
6  #Task 2
7  ## Simplify this string reversal code and improve readability
8  def reverse_string(s):
9      return s[::-1]
10 user_input = input("Enter a string to reverse: ")
11 print(f"Reversed string: {reverse_string(user_input)}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
PS C:\Users\Padmavathi> & C:/Users/Padmavathi/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Padmavathi/#Task 1.py"
Enter a string to reverse: padmavathi
Reversed string: ihtavamdap
PS C:\Users\Padmavathi> & C:/Users/Padmavathi/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Padmavathi/#Task 1.py"
Enter a string to reverse: sruniveristy
Reversed string: ytsirevinurs
Enter a string to reverse: █
```

Explanation : In Task 2, the original string reversal code was optimized by removing unnecessary variables and simplifying the logic.

The loop and intermediate variable were eliminated, and Python slicing was used directly.

This improved readability and reduced code length.

Both versions have a time complexity of $O(n)$, but the optimized version is more efficient in practice due to fewer operations.

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

The screenshot shows a code editor interface with a dark theme. At the top, there's a navigation bar with 'Help' and other icons. Below it is a search bar. The main area contains Python code for three tasks. Task 1 defines a function to reverse a string using slicing. Task 2 uses an f-string to print the reversed string. Task 3 is a comment about writing a program using a function to reverse a string. The code editor has tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is active, showing command-line output of the script running. The output shows various strings being reversed, such as 'economics' becoming 'scimonoce' and 'hello world' becoming 'dlrow olleh'. The bottom status bar indicates 'Ln 16, Col 22' and 'Spaces: 4'.

```
C: > Users > Padmavathi > #Task 1.py > ...
6  #Task 2
7  ## Simplify this string reversal code and improve readability
8  def reverse_string(s):
9      return s[::-1]
10 user_input = input("Enter a string to reverse: ")
11 print(f"Reversed string: {reverse_string(user_input)}")
12 #Task 3
13 #Write a Python program using a function to reverse a string.
14 #Add meaningful comments.
15 def reverse_string(s):
16     return s[::-1] #Slicing the string to reverse it
17 #Get user input
18 user_input = input("Enter a string to reverse: ")
19 #call the function and display the reversed string
20 reversed_str = reverse_string(user_input)
21 print(f"Reversed string: {reversed_str}")

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
Enter a string to reverse: & c:/Users/Padmavathi/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Padmavathi/#Task 1.py"
Reversed string: "yp.1 ksaT#/ihtavamdaP/sresu/:c" exe.31.3nohtyp/sppAswodniW/tfosorciM/lacol/ataDppA/ihtavamdaP/sresu/:c &
PS C:\Users\Padmavathi> & c:/Users/Padmavathi/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Padmavathi/#Task 1.py"
Enter a string to reverse: economics
Reversed string: scimonoce
Enter a string to reverse: hello world
Reversed string: dlrow olleh
Enter a string to reverse: repository
Reversed string: yrotisoper
PS C:\Users\Padmavathi> █
```

Explanation

- The function `reverse_string()` encapsulates the string reversal logic
- It uses Python slicing (`[::-1]`) for efficient reversal
- The function returns the reversed string to the caller
- This modular approach allows reuse of the same logic in multiple parts of an application
- Meaningful comments improve code readability and understanding

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

Comparison Table

Criteria	Without Functions (Procedural)	With Functions (Modular)
Code Clarity	Logic is mixed with input/output, making it less clear	Logic is separated into a function, improving clarity
Reusability	Code cannot be reused easily	Function can be reused in multiple parts of the application
Debugging Ease	Harder to debug due to lack of separation	Easier to debug and test individual functions
Maintainability	Changes must be made in multiple places	Changes can be made in one function
Scalability	Not suitable for large programs	Suitable for large-scale applications

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

■ A **loop-based** string reversal approach

The screenshot shows a code editor window in VS Code. The file is named `Task 1.py`. The code defines a function `reverse_string_loop(s)` that takes a string `s` and reverses it character by character using a loop. It also includes a call to this function with user input and prints the result. The terminal below shows the execution of the script and several test runs where the user enters different strings like "engineering", "employee", and "future".

```

# Task 1.py
C: > Users > Padmavathi > #Task 1.py > reverse_string_loop
18     user_input = input("Enter a string to reverse: ")
19     #Call the function and display the reversed string
20     reversed_str = reverse_string(user_input)
21     print(f"Reversed string: {reversed_str}")
22
#Task 5
23     #Write a Python program to reverse a string using a loop.
24     def reverse_string_loop(s):
25         reversed_str = ""
26         for char in s:
27             reversed_str = char + reversed_str # Prepend each character
28         return reversed_str
29
#Get user input
30     user_input= input("Enter a string to reverse: ")
31     #Call the function and display the reversed string
32     reversed_str = reverse_string_loop(user_input)
33     print(f"Reversed string: {reversed_str}")

return reversed_str
^^^^^^^^^
SyntaxError: 'return' outside function
PS C:\Users\Padmavathi> & C:/Users/Padmavathi/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Padmavathi/#Task 1.py"
Enter a string to reverse: engineering
Reversed string: gnireenigne
Enter a string to reverse: employee
Reversed string: eeyolpme
Enter a string to reverse: future
Reversed string: erutuf
Reversed string: erutuf
PS C:\Users\Padmavathi>

```

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

The screenshot shows a code editor window in VS Code. The file is named `Task 1.py`. The code defines a function `reverse_string_slicing(s)` that returns the string `s` reversed using slicing. It also includes a call to this function with user input and prints the result. The terminal below shows the execution of the script and several test runs where the user enters different strings like "welcome", "hello", and "people".

```

# Task 1.py
C: > Users > Padmavathi > #Task 1.py > ...
29     #Get user input
30     user_input= input("Enter a string to reverse: ")
31     #Call the function and display the reversed string
32     reversed_str = reverse_string_loop(user_input)
33     print(f"Reversed string: {reversed_str}")
34
#Write a Python program to reverse a string using slicing.
35     def reverse_string_slicing(s):
36         return s[::-1]
37     user_input = input("Enter a string to reverse: ")
38     print(f"Reversed string: {reverse_string_slicing (user_input)}")

SyntaxError: f-string: single ')' is not allowed
PS C:\Users\Padmavathi> & C:/Users/Padmavathi/AppData/Local/Microsoft/WindowsApps/python3.13.exe "c:/Users/Padmavathi/#Task 1.py"
Enter a string to reverse: welcome
Reversed string: emoclew
Enter a string to reverse: hello
Reversed string: olleh
Enter a string to reverse: people
Reversed string: elpoop
Reversed string: elpoop
Enter a string to reverse: meghana
Reversed string: anahgem
PS C:\Users\Padmavathi>

```

2. Time Complexity

- **Loop-Based Approach:**

Time complexity is $O(n)$, where n is the length of the string.

- **Built-in Approach:**

Time complexity is also **O(n)**, as the string must be traversed internally.

Conclusion

Although both approaches have the same time complexity, the built-in slicing method is more efficient and readable, making it the preferred choice for practical applications.