

Lab 1: Environment Setup – GitHub Copilot and VS Code Integration + Understanding AI-assisted Coding Workflow

Hall Ticket: 2303A51846

Task 0

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

Expected Output

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

Explanation :

Step 1: Install Visual Studio Code (VS Code)

If VS Code is already installed, you can skip this step.

- 1.Open your browser and go to the official VS Code website
- 2.Download the installer for your operating system (Windows / macOS / Linux)
- 3.Run the installer and complete the installation

Step 2: Open VS Code and Go to Extensions

- 1.Open Visual Studio Code
- 2.Click on the Extensions icon (square icon) on the left sidebar

Step 3: Search and Install GitHub Copilot

- 1.In the Extensions search bar, type GitHub Copilot
- 2.Select GitHub Copilot by GitHub
- 3.Click the Install button

Step 4: Sign In to GitHub Account

- 1.After installation, a popup will ask you to Sign in to GitHub
- 2.Click Sign In
- 3.Your browser will open → log in to your GitHub account
- 4.Authorize GitHub Copilot

Step 5: Verify GitHub Copilot is Enabled

- 1.Return to VS Code after signing in
- 2.Check the status bar (bottom right)
- 3.You should see GitHub Copilot Enabled

Step 6: Test GitHub Copilot (Configuration Check)

- 1.Create a new file (example: test.java or test.py)
- 2.Start typing a comment or code.
- 3.GitHub Copilot will automatically suggest code
- 4.Press Tab to accept the suggestion

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

Comments:

#write a code to reverse a string without using any functions and taking input from user

#taking input from user

Code:

```
3  string = input("Enter a string: ")
4  reversed_string = ""
5  for char in string:
6      reversed_string = char + reversed_string
7  #printing the reversed string
8  print("Reversed string:", reversed_string)
9
```

Output:

The screenshot shows a terminal window with the following content:

```
PROBLEMS    OUTPUT    DASHBOARD    TERMINAL    PORTS
Focus folder in explorer (ctrl + click)  TERMINAL  PORTS
PS C:\Users\madhu\OneDrive\Desktop\AI Assistent coding> python -u "c:\Users\madhu\OneDrive\Desktop\AI Assistent coding\code1.py"
Enter a string: Madhu
Reversed string: uhdaM
PS C:\Users\madhu\OneDrive\Desktop\AI Assistent coding> python -u "c:\Users\madhu\OneDrive\Desktop\AI Assistent coding\code1.py"
Enter a string: karthik
Reversed string: kiiltrak
PS C:\Users\madhu\OneDrive\Desktop\AI Assistent coding> 
```

Observation

- The program successfully reverses a string without using any built-in string functions like `reverse()` or slicing.
 - It takes input from the user, making it interactive and flexible for different strings.
 - The `for` loop iterates through each character of the input string from left to right.
 - In each iteration, the current character is added to the front of `reversed_string`, which gradually builds the reversed string.
 - This approach demonstrates a clear understanding of string manipulation and loop logic.
 - The final output correctly displays the reversed version of the input string.

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”

#prompt or comments:

#optimize the code and make it more efficient

#remove unnecessary comments

Code:

```
string = input("Enter a string: ")
reversed_string = ""
for char in string:
    reversed_string = char + reversed_string
#printing the reversed string
print("Reversed string:", reversed_string)
```

Output:

```
PS C:\Users\madhu\OneDrive\Desktop\AI Assistant coding> python -u "c:\Users\madhu\OneDrive\Desktop\AI Assistant coding\code1.py"
Enter a string: madhu
Reversed string: uhdam
PS C:\Users\madhu\OneDrive\Desktop\AI Assistant coding> python -u "c:\Users\madhu\OneDrive\Desktop\AI Assistant coding\code1.py"
Enter a string: Python
Reversed string: nohtyP
PS C:\Users\madhu\OneDrive\Desktop\AI Assistant coding>
```

In 20 Col 1 Spaces: 4 UTE-8 CR LF {} Python

Observation:

- After reviewing the original code, it is observed that the logic is already simple, clean, and readable.
- The code does not use any unnecessary variables; both string and reversed_string are essential.
- The for loop logic is straightforward and does not involve complex indexing or nested loops.
- Since the task restricts the use of built-in functions, the current approach is one of the most optimal and beginner-friendly solutions available.
- Any further optimization (such as slicing or built-in reverse methods) would violate the task constraint of “without using functions.”

Time Complexity Observation:

- The time complexity of the code is $O(n)$, where n is the length of the input string.
- Each character is processed exactly once, and no redundant operations are performed.
- Since $O(n)$ is the best possible time complexity for reversing a string, no further reduction is possible.

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)

Promt or Comments:

```
#write a code to reverse a string using user define function and use built in functions  
#include meaningful variable names and comments
```

Code:

```
13 def reverse_string(input_string):  
14     # Using slicing to reverse the string  
15     return input_string[::-1]  
16 # Taking input from user  
17 user_input = input("Enter a string: ")  
18 # Calling the function to reverse the string  
19 reversed_result = reverse_string(user_input)  
20 # Printing the reversed string  
21 print("Reversed string:", reversed_result)  
22
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\madhu\OneDrive\Desktop\AI Assistant coding> python -u "c:\Users\madhu\OneDrive\Desktop\AI Assistant coding\code1.py"  
Enter a string: string  
Reversed string: gnirts  
PS C:\Users\madhu\OneDrive\Desktop\AI Assistant coding>
```

Observation:

- The string reversal logic is successfully implemented using a user-defined function, which supports modular design.
- The function returns the reversed string instead of printing it, making the code reusable in multiple parts of an application.
- Meaningful comments clearly explain the purpose of the function and the logic inside it, improving code readability.
- The main program is clean and concise, with the core logic separated from input and output handling.
- The program avoids using built-in string reversal functions, satisfying the given constraints.
- The time complexity remains $O(n)$, as each character in the string is processed exactly once.
- Overall, the modular approach improves maintainability, scalability, and clarity of the code.

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)

Promt or Comments:

comparative Analysis of both codes

Output:

```
# The first code snippet reverses a string without using any built-in functions.  
# by iterating through each character and constructing the reversed string manually. This approach is straightforward but  
# This approach is straightforward but less efficient for longer strings due to the repeated string concatenation.  
# which can lead to higher time complexity.  
# The second code snippet defines a user-defined function that utilizes Python's slicing feature to  
# reverse the string. This method is more efficient and concise, leveraging built-in functionality to  
# achieve the same result with less code and improved performance.# Function to reverse a string using slicing
```

Observation:

- The first code reverses the string using a loop without built-in functions, making it easy to understand.
- Repeated string concatenation in the first approach can reduce efficiency for long strings.
- The second code uses a function and slicing, resulting in shorter and cleaner code.
- Slicing improves performance and readability compared to manual reversal.
- The function-based approach supports better reusability and modularity.
- Overall, the first code is good for learning, while the second is better for practical use.