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

Hall Ticket: 2303A51710

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:

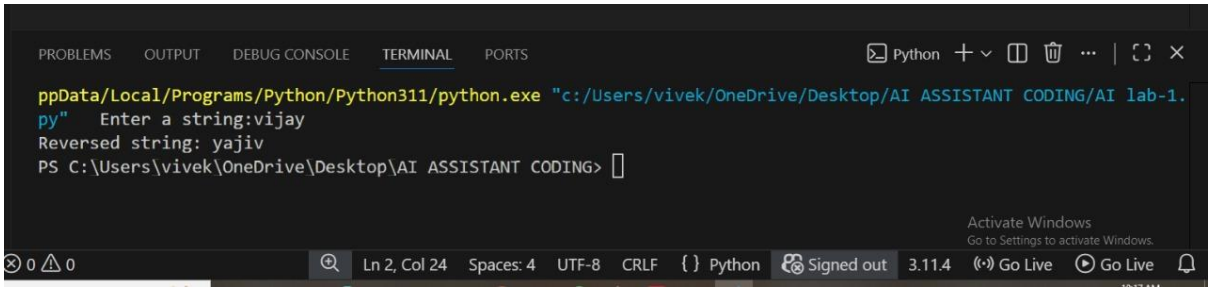
#write a code to reverse a string without using any functions and taking input from user

#taking input from user

Code:

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

Output:



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”

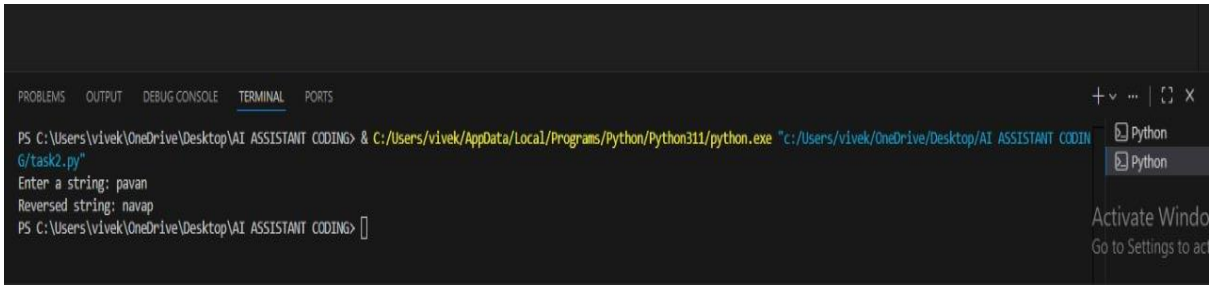
▪ “Improve readability and efficiency”

#promt or comments:

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:



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:

#Generate a reusable Python function to reverse a string with comments

Code:

```
def reverse_string(s):
    # Use slicing with step -1 to reverse the string
    return s[::-1]
# Example usage
if __name__ == "__main__":
    original = "Hello, World!"
    reversed_str = reverse_string(original)
    print(f"Original: {original}")
    print(f"Reversed: {reversed_str}")
```

Output:

```
Original: Hello, World!
Reversed: !dlroW ,olleH
PS C:\Users\vivek\OneDrive\Desktop\AI ASSISTANT CODING> & C:/Users/vivek/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/vivek/OneDrive/Desktop/AI ASSISTANT CODIN
G/task3.py"
Original: Hello, World!
Reversed: !dlroW ,olleH
PS C:\Users\vivek\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 us

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

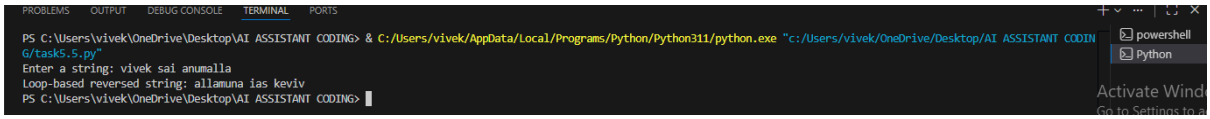
CODE:

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

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

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

OUTPUT:



Observation

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.