ASSIGNMENT-01

NAME: Varsha Sri Chukka

BATCH:50

HALL TICKET no.: 2303A52494

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 Used

Write a Python program to reverse a string taken as user input. Do not use functions and write logic in main code

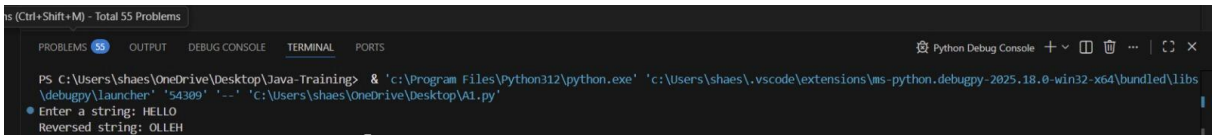
#CODE

```python
user_input = input("Enter a string: ")
reversed_string = ""

for i in range(len(user_input) - 1, -1, -1):
    reversed_string += user_input[i]

print("Reversed string:", reversed_string)
```

#OUTPUT



#Explanation

This prompt clearly instructs Copilot to avoid user-defined functions and implement logic directly. It helps demonstrate procedural programming and allows evaluation of Copilot's basic logic generation capability.

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

Prompt Used

Simplify this string reversal code. Improve readability and efficiency.

#CODE

```
user_input = input("Enter a string: ")
reversed_string = user_input[::-1]

print("Reversed string:", reversed_string)

```

#OUTPUT

```
ython.debugpy-2025.18.0-win32-x64\bundled\l
Enter a string: HEYHI
Reversed string: IHYEH
```

#Explanation

The prompt explicitly asks Copilot to optimize and simplify the code. This encourages the AI to use Python’s built-in features, resulting in cleaner, shorter, and more readable code.

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 Used

Create a Python function to reverse a string. Return the reversed string and add meaningful comments

#CODE

```
def reverse_string(input_string):
    """Reverses the given string."""
    return input_string[::-1]

# Get user input
user_input = input("Enter a string to reverse: ")

# Call the function and print the result
reversed_string = reverse_string(user_input)
print("Reversed string:", reversed_string)
```

#OUTPUT

```
Enter a string to reverse: acceptance
Reversed string: ecnatpecca
```

#Explanation

This prompt encourages modular programming by requesting a function-based solution. Including comments helps evaluate Copilot's ability to generate readable and well-documented 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)

Analyze them based on:

➢ Code clarity

➢ Reusability

➢ Debugging ease

➢ Suitability for large-scale applications

❖ Expected Output

Comparison table or short analytical report.

#ANSWER

In this task, the Copilot-generated program without functions (Task 1) and the program with functions (Task 3) are compared to evaluate their design quality. The comparison focuses on important software engineering aspects such as code clarity, reusability, ease of debugging, and suitability for large-scale applications.

The procedural approach places all logic in the main program, which is simpler for small tasks but becomes harder to manage as the program grows. In contrast, the function-based approach organizes logic into reusable modules, making the code easier to read, maintain, and extend. This analysis helps justify why modular programming is preferred in real-world and large-scale software development.

Expected Output:

Comparison Table

Aspect	Without Functions (Task 1)	With Functions (Task 3)
Code Clarity	Logic is mixed in main code, reducing readability	Clear separation of logic improves
Reusability	Code cannot be reused easily	Function can be reused multiple times
Debugging Ease	Harder to debug due to lack of structure	Easier to debug individual functions
Suitability for Large-Scale Applications	Highly suitable for large applications	Highly suitable for large applications

#Explanation

This comparison demonstrates that modular, function-based programs improve maintainability and scalability, making them more suitable for professional software development.

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

Note: Report should be submitted as a word document for all tasks in a

single document with prompts, comments & code explanation, and output

and if required, screenshots.

Prompt 1 (Loop-Based / Iterative String Reversal)

Write a Python program to reverse a string using a loop. Take user input and print the reversed string.

#CODE

```python
# Get user input
user_string = input("Enter a string: ")

# Reverse the string using a loop
reversed_string = ""
for char in user_string:
    reversed_string = char + reversed_string

# Print the result
print("Reversed string:", reversed_string)
```

#OUTPUT

```
Enter a string: whatever
Reversed string: revetahw
```

Prompt 2 (Built-in / Slicing-Based String Reversal)

Write a Python program to reverse a string using Python slicing.Take user input and print the reversed string.

#CODE

```python
# Take user input
user_string = input("Enter a string: ")

# Reverse the string using slicing
reversed_string = user_string[::-1]

# Print the reversed string
print(f"Reversed string: {reversed_string}")
```

#OUTPUT

```
Enter a string: whether
Reversed string: rehtehw
```

#Explanation

These prompts were chosen to explicitly request two different algorithmic approaches. By separating the prompts, Copilot is guided to generate distinct logic paths, enabling performance and readability comparison.