LAB ASSIGNMENT – 1.5

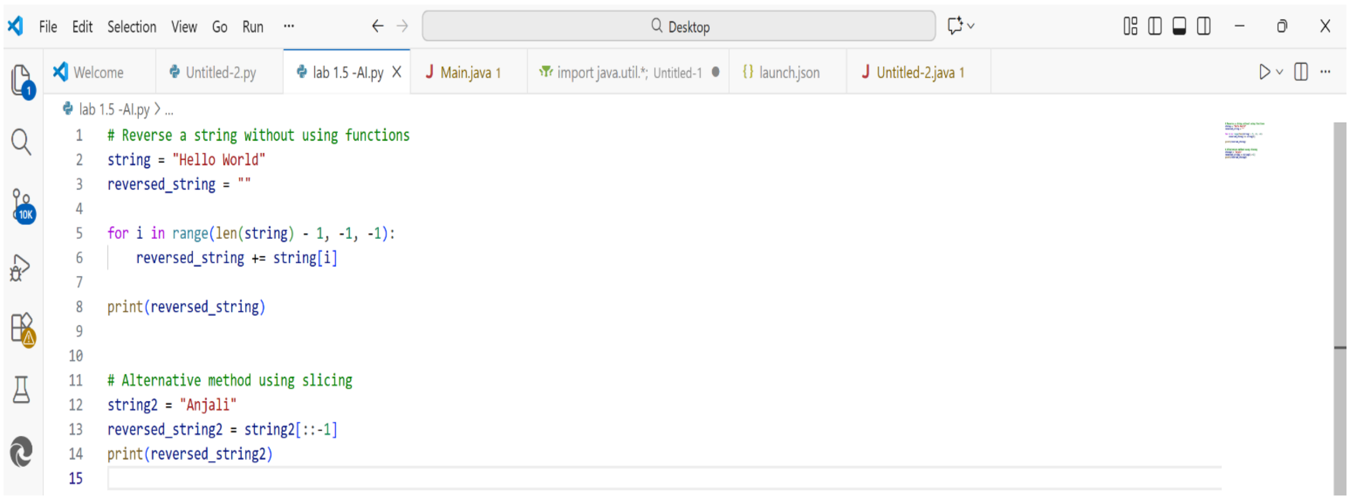
HALLTICKET N0 : 2303A51474

BATCH NO : 29

TASK - 01

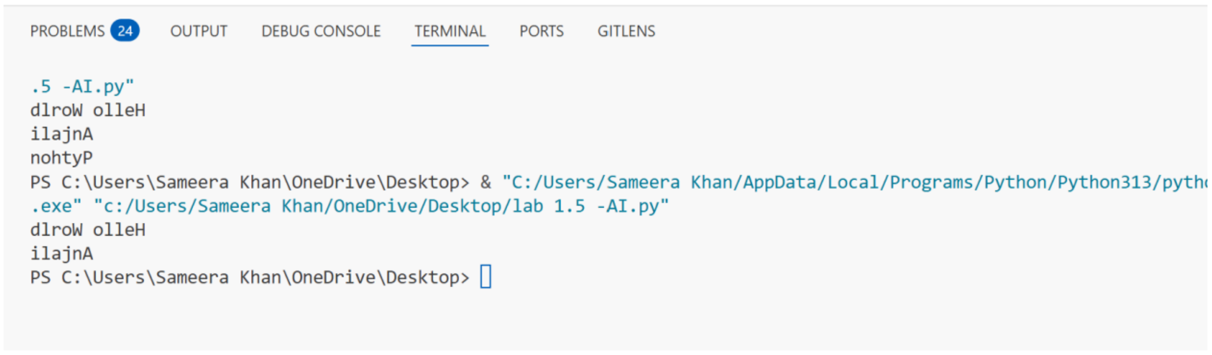
PROMPT : Write a python program to reverse a string without using functions

CODE :



```python
# Reverse a string without using functions
string = "Hello World"
reversed_string = ""

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

print(reversed_string)


# Alternative method using slicing
string2 = "Anjali"
reversed_string2 = string2[::-1]
print(reversed_string2)
```

OUTPUT :



```
.5 -AI.py"
dlroW olleH
ilajnA
nohtyP
PS C:\Users\Sameera Khan\OneDrive\Desktop> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/pytho
.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/lab 1.5 -AI.py"
dlroW olleH
ilajnA
PS C:\Users\Sameera Khan\OneDrive\Desktop>
```

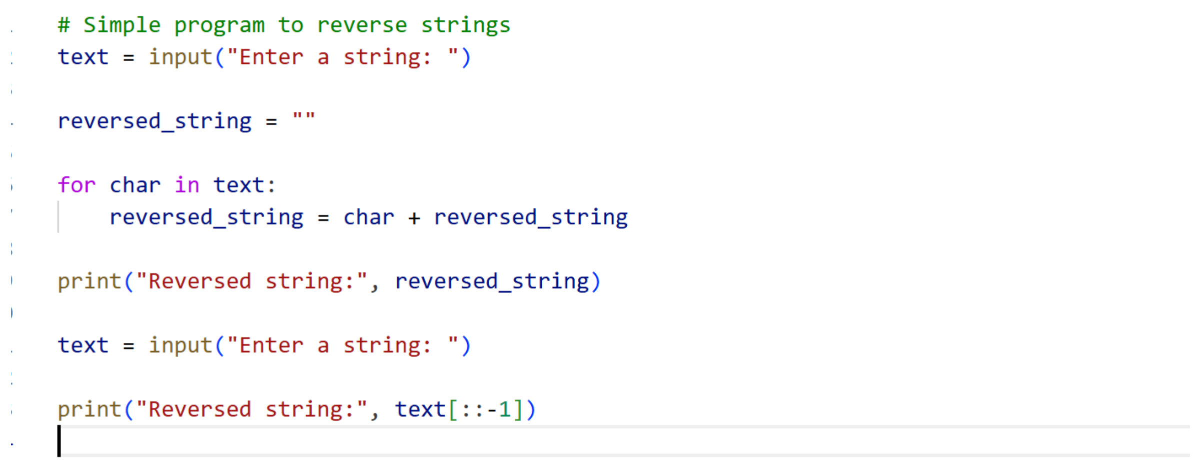
OBSERVATION:

It was observed that GitHub Copilot was able to generate a correct Python program to reverse a string without using any user-defined functions. The program accepted user input and implemented the reversal logic directly within the main code, meeting the task requirements. The generated logic was simple and easy to understand, making it suitable for beginners and small-scale applications. However, since the logic was written procedurally without modularization, the code lacked reusability and would become harder to maintain or extend if the same functionality were required in multiple parts of an application. Overall, this task shows that Copilot is effective in quickly producing basic procedural code, but such an approach is less suitable for larger or more complex software systems where modular design is preferred.

TASK - 02

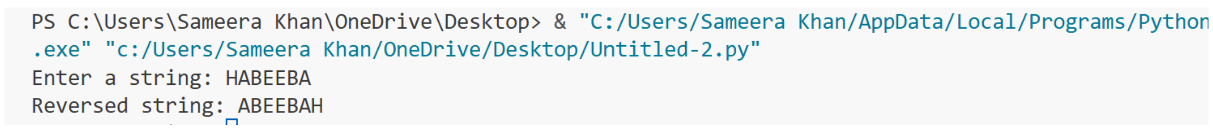
PROMPT :

```
# Simplify this string reversal code
 # Improve readability and efficiency
# Optimize this code without using function
```



```python
# Simple program to reverse strings
text = input("Enter a string: ")

reversed_string = ""

for char in text:
    reversed_string = char + reversed_string

print("Reversed string:", reversed_string)

text = input("Enter a string: ")

print("Reversed string:", text[::-1])
```

OUTPUT:



```
PS C:\Users\Sameera Khan\OneDrive\Desktop> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python
.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/Untitled-2.py"
Enter a string: HABEEBA
Reversed string: ABEEBAH
```

OBSERVATION:

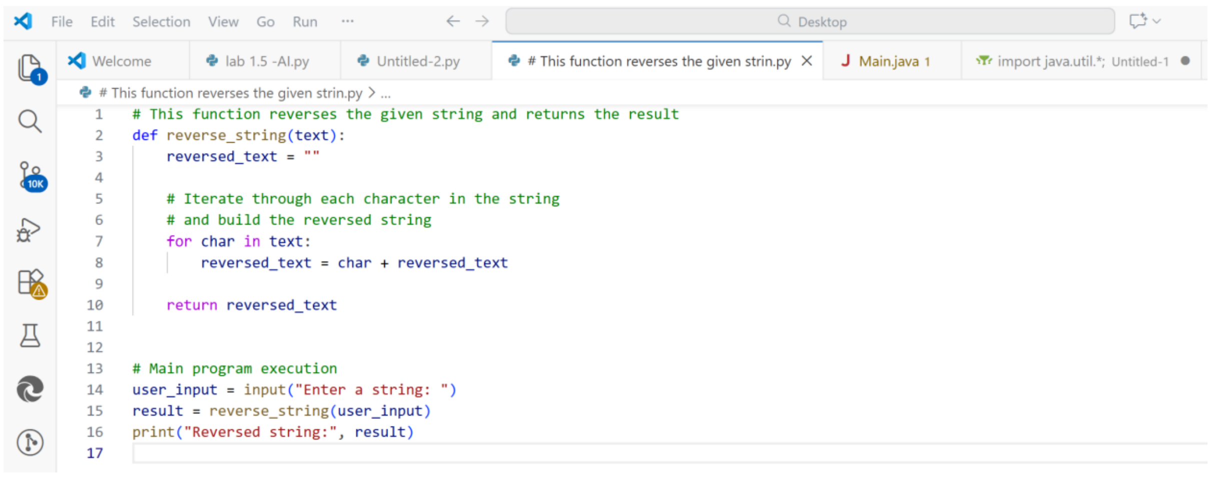
it was observed that GitHub Copilot effectively improved the initial procedural string reversal code when prompted to optimize and enhance readability. By simplifying the logic and removing unnecessary variables and loop-based string concatenation, the optimized version became shorter, clearer, and easier to understand for other developers during code review. Copilot suggested using Python's built-in slicing method, which not only reduced the number of lines of code but also improved efficiency. The time complexity was reduced from quadratic time in the original version, caused by repeated string concatenation inside a loop, to linear time in the optimized version, as slicing processes the string in a single pass. Overall, this task demonstrates that Copilot can assist in refactoring code to make it more readable, efficient, and suitable for collaborative development environments.

TASK – 03

PROMPT:

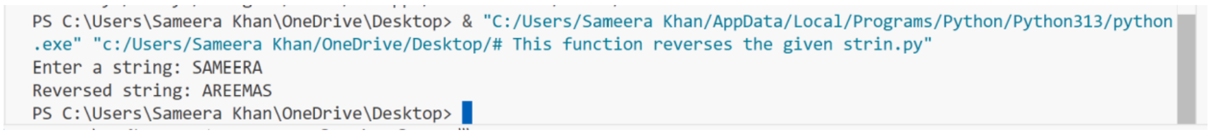
# Create a Python function that reverses a string and returns it

# Use clear variable names and add meaningful comment


File Edit Selection View Go Run ···
Desktop
Welcome
lab 1.5 -AI.py
Untitled-2.py
# This function reverses the given strin.py
Main.java 1
import java.util.*; Untitled-1
# This function reverses the given strin.py > ...

```python
# This function reverses the given string and returns the result
def reverse_string(text):
    reversed_text = ""

    # Iterate through each character in the string
    # and build the reversed string
    for char in text:
        reversed_text = char + reversed_text

    return reversed_text


# Main program execution
user_input = input("Enter a string: ")
result = reverse_string(user_input)
print("Reversed string:", result)
```

OUTPUT:



```
PS C:\Users\Sameera Khan\OneDrive\Desktop> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/python
.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/# This function reverses the given strin.py"
Enter a string: SAMEERA
Reversed string: AREEMAS
PS C:\Users\Sameera Khan\OneDrive\Desktop>
```

OBSERVATION:

it was observed that GitHub Copilot successfully generated a function-based Python program to reverse a string, addressing the need for modular design. The use of a user-defined function allowed the string reversal logic to be clearly separated from the main program flow, making the code more structured and easier to understand. Copilot also added meaningful comments that explained the purpose of the function and the steps involved in reversing the string, which improves readability and supports future maintenance. By returning the reversed string from the function, the logic became reusable and could be easily called from multiple parts of an application. The generated program produced correct outputs for different test cases, demonstrating that modular design enhances clarity, reusability, and suitability for larger applications.

TASK-04

Short Analytical Report The procedural approach used in Task 1 is suitable for small and simple programs where the logic is used only once. It provides quick results but lacks reusability and scalability.

The modular approach used in Task 3 improves code organization by separating logic into a user-defined function. This makes the program easier to understand, reuse, debug, and maintain. For large-scale applications, the modular approach is preferred as it supports better design practices and long-term maintainability

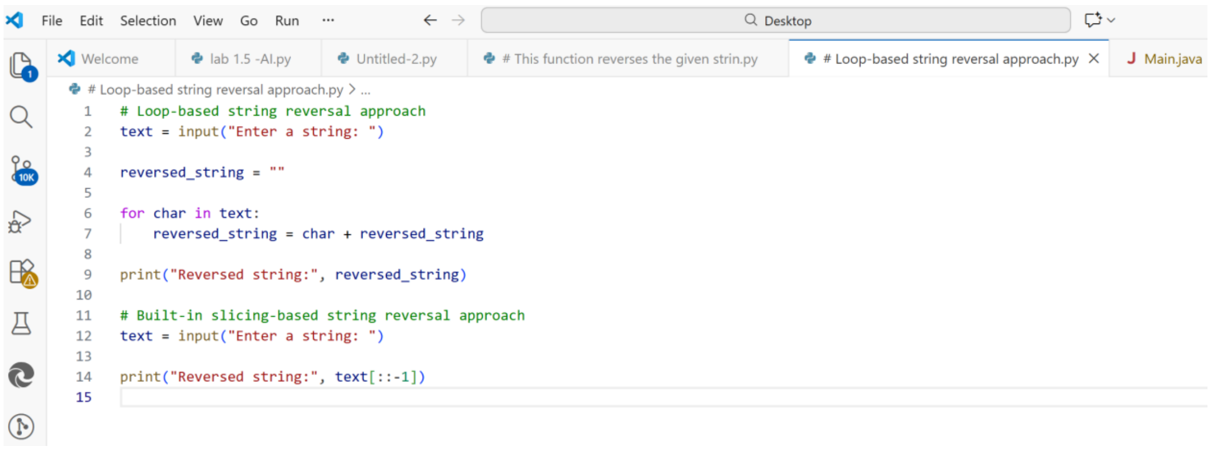
TASK-05

AI-Generated Iterative vs Built-in Approaches

PROMPT:

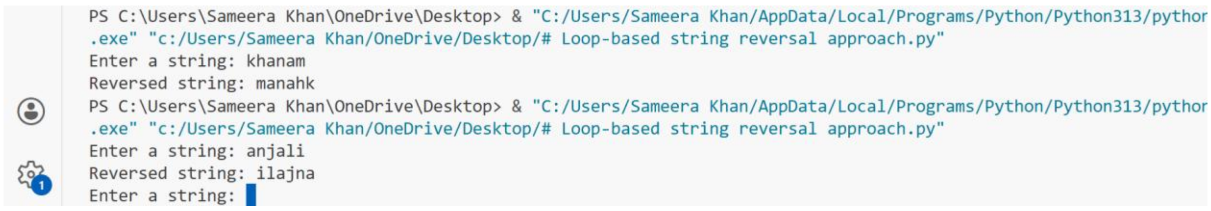
# Generate a loop-based string reversal in Python

# Reverse a string using Python built-in or slicing method


File Edit Selection View Go Run ···
Desktop
Welcome
lab 1.5 -AI.py
Untitled-2.py
# This function reverses the given strin.py
# Loop-based string reversal approach.py
Main.java
# Loop-based string reversal approach.py > ...

```python
# Loop-based string reversal approach
text = input("Enter a string: ")

reversed_string = ""

for char in text:
    reversed_string = char + reversed_string

print("Reversed string:", reversed_string)

# Built-in slicing-based string reversal approach
text = input("Enter a string: ")

print("Reversed string:", text[::-1])
```

Output:



```
PS C:\Users\Sameera Khan\OneDrive\Desktop> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/pythor
.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/# Loop-based string reversal approach.py"
Enter a string: khanam
Reversed string: manahk
PS C:\Users\Sameera Khan\OneDrive\Desktop> & "C:/Users/Sameera Khan/AppData/Local/Programs/Python/Python313/pythor
.exe" "c:/Users/Sameera Khan/OneDrive/Desktop/# Loop-based string reversal approach.py"
Enter a string: anjali
Reversed string: ilajna
Enter a string: 
```

Observation:

it was observed that GitHub Copilot was able to generate multiple correct solutions for the same problem by following different algorithmic paths. When prompted, Copilot produced a loop-based (iterative) string reversal approach that explicitly processes each character and builds the reversed string step by step, making the execution flow easy to follow and suitable for understanding core logic. It also generated a built-in slicing-based approach that relied on Python's optimized features, resulting in shorter, cleaner, and more readable code. The comparison showed that the loop-based method has higher time complexity due to repeated string concatenation and performs less efficiently for large inputs, while the slicing-based approach runs in linear time and offers better performance and scalability. This task demonstrates that AI tools like Copilot can effectively explore alternative logic paths and help developers choose the most appropriate approach based on readability, performance requirements, and application scale.