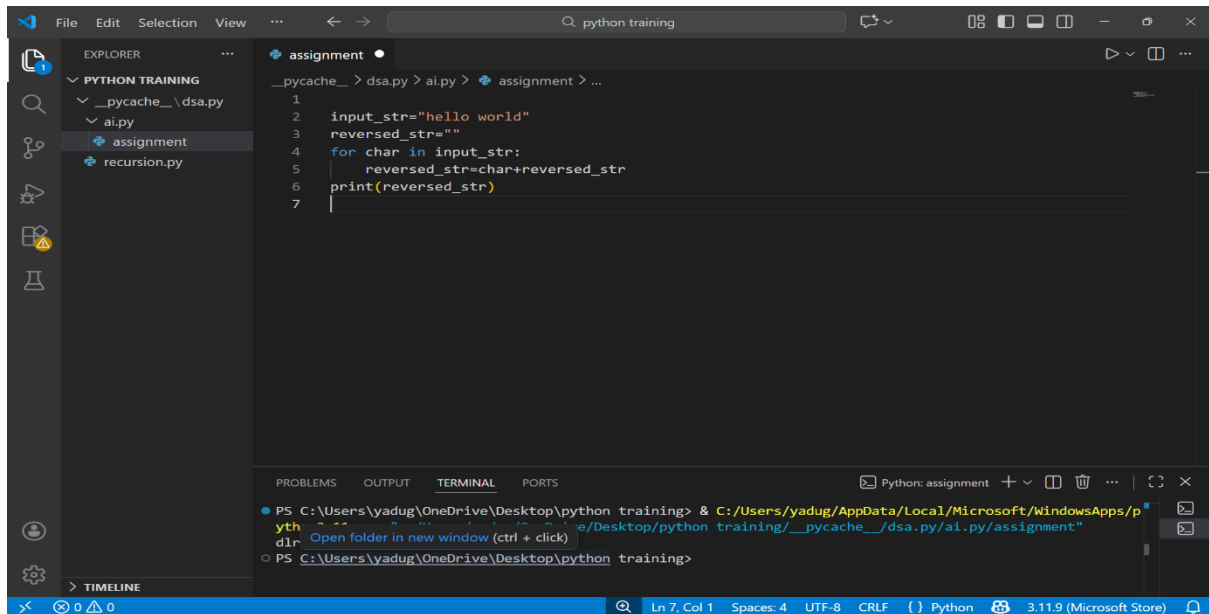


Assignment-1.5

Name: K.Likhith

RollNo:2303A51331

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



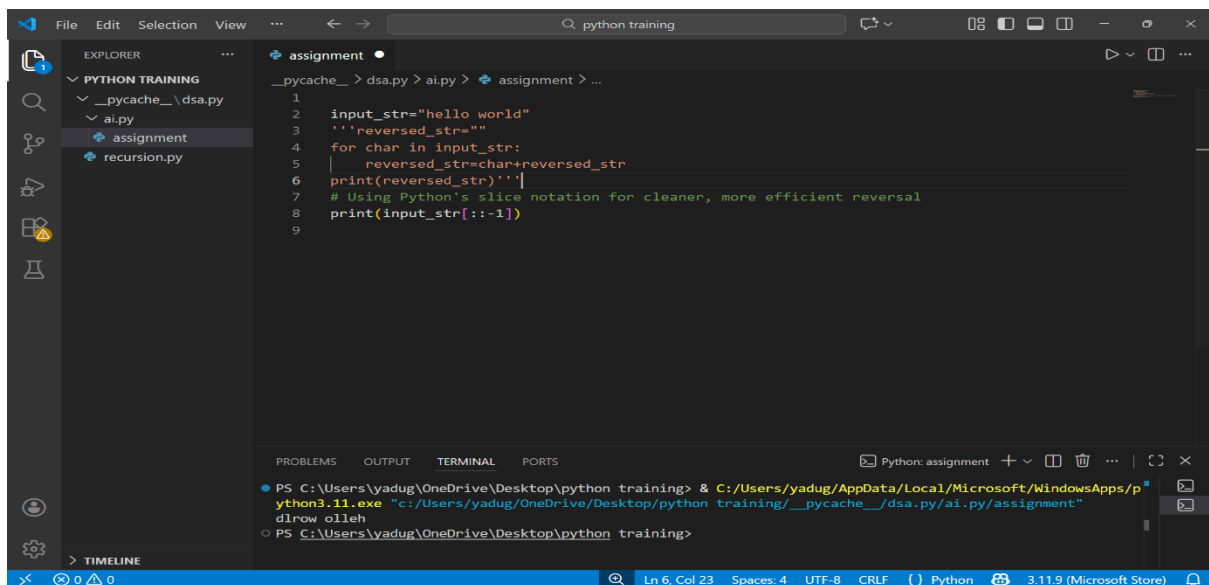
The screenshot shows a Visual Studio Code editor window with a file explorer on the left and a code editor in the center. The file explorer shows a project named 'PYTHON TRAINING' with subfolders '__pycache__' and 'ai.py'. The 'ai.py' file is open, showing a Python script for string reversal. The script uses a loop to build the reversed string. The terminal at the bottom shows the command to run the script, which outputs 'dlrow olleh'.

```
1 input_str="hello world"
2 reversed_str=""
3 for char in input_str:
4     reversed_str=char+reversed_str
5 print(reversed_str)
```

Terminal output:

```
PS C:\Users\yadug\OneDrive\Desktop\python training> & C:/Users/yadug/AppData/Local/Microsoft/WindowsApps/python3.11.exe "C:/Users/yadug/OneDrive/Desktop/python training/__pycache__/dsa.py/ai.py/assignment"
dlrow olleh
```

TASK-2 Efficiency & Logic Optimization (Readability Improvement)



The screenshot shows a Visual Studio Code editor window with a file explorer on the left and a code editor in the center. The file explorer shows a project named 'PYTHON TRAINING' with subfolders '__pycache__' and 'ai.py'. The 'ai.py' file is open, showing a Python script for string reversal. The script uses Python's slice notation for string reversal. The terminal at the bottom shows the command to run the script, which outputs 'dlrow olleh'.

```
1 input_str="hello world"
2 reversed_str=""
3 for char in input_str:
4     reversed_str=char+reversed_str
5 print(reversed_str)
6 # Using Python's slice notation for cleaner, more efficient reversal
7 print(input_str[::-1])
```

Terminal output:

```
PS C:\Users\yadug\OneDrive\Desktop\python training> & C:/Users/yadug/AppData/Local/Microsoft/WindowsApps/python3.11.exe "C:/Users/yadug/OneDrive/Desktop/python training/__pycache__/dsa.py/ai.py/assignment"
dlrow olleh
```

- Explanation: `[::-1]` tells Python:
 - Start from end

- Move backwards
- Step = -1
- Python internally reverses the string in **one pass**.

Time Complexity

- Each character is accessed **once**.
- No repeated copying like the loop version.

Time Complexity: $O(n)$

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

Before optimization

```

1 def strreverse(s):
2     return s[::-1]
3 str1 = "Hello, World!"
4 print(strreverse(str1))
5 Output: ldlrW ,olleH

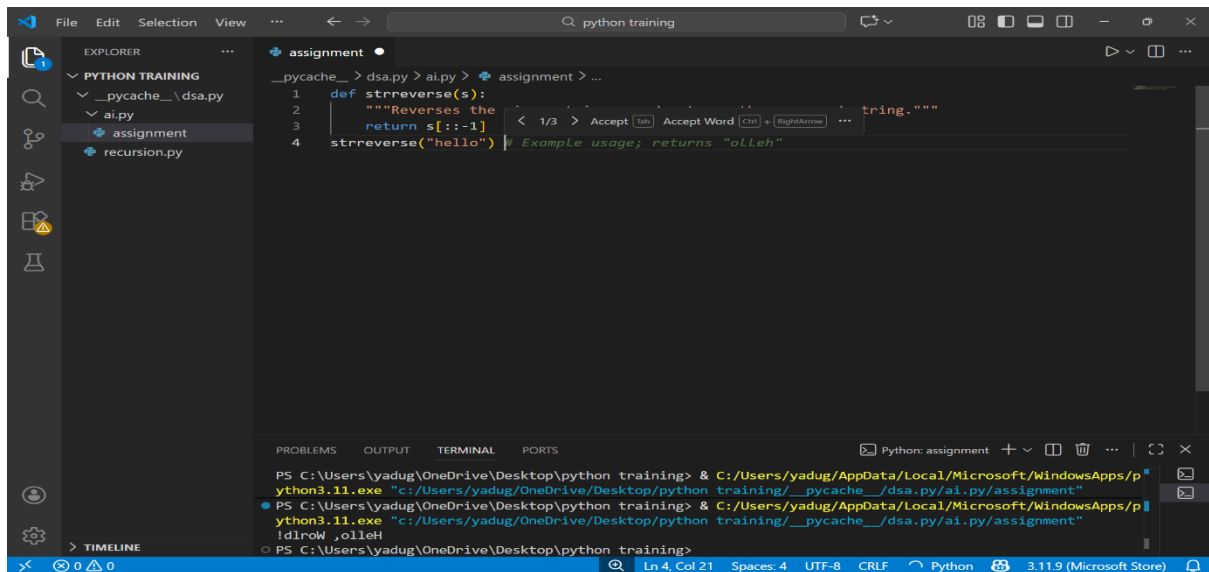
```

```

PS C:\Users\yadug\OneDrive\Desktop\python training> & C:/Users/yadug/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/yadug/OneDrive/Desktop/python training/_pycache_/dsa.py/ai.py/assignment"
PS C:\Users\yadug\OneDrive\Desktop\python training> & C:/Users/yadug/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/yadug/OneDrive/Desktop/python training/_pycache_/dsa.py/ai.py/assignment"
ldlrW ,olleH
PS C:\Users\yadug\OneDrive\Desktop\python training>

```

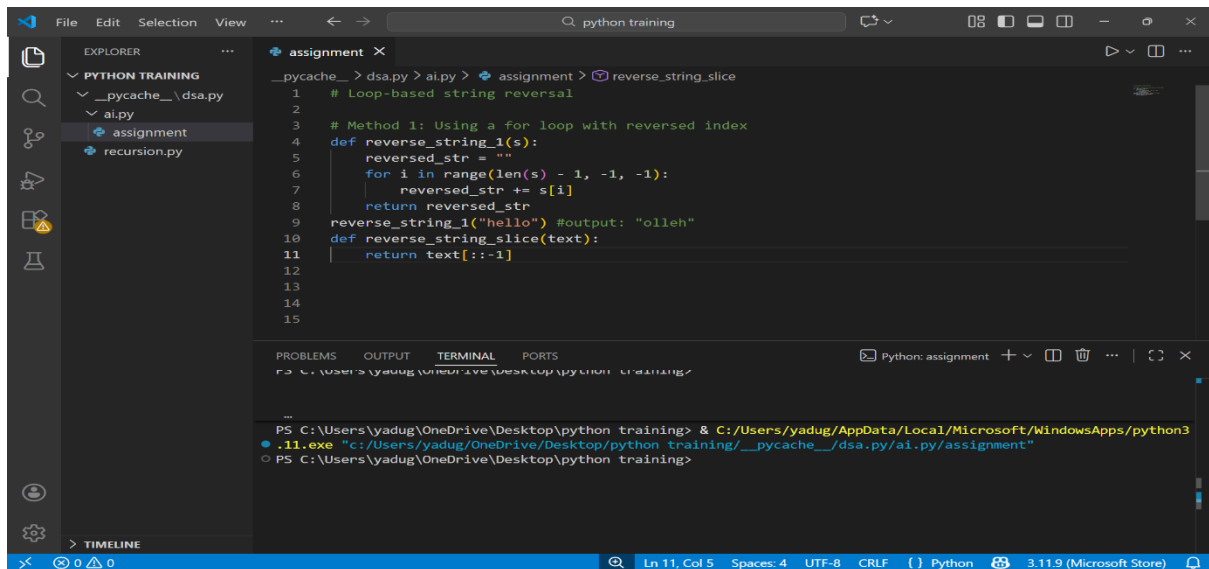
After optimization using copilot



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

Criteria	Without Functions (Procedural)	With Functions (Modular)
Code Clarity	Logic mixed with execution, harder to read in large files	Clear separation of logic and execution
Reusability	Code must be rewritten for every use	Function can be reused anywhere
Debugging Ease	Bugs are harder to isolate	Easy to debug by testing the function alone
Scalability	Poor for large programs	Highly suitable for large applications
Maintenance	Changes must be made in multiple places	Changes made once in the function
Testing	Manual testing only	Supports unit testing easily
Readability	Acceptable for small scripts	Excellent for professional codebases
Copilot Assistance	Limited suggestions	Strong AI support for docstrings & logic

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



The screenshot shows a Visual Studio Code editor window with a Python file named `assignment.py`. The file contains two functions for reversing a string: `reverse_string_1` (loop-based) and `reverse_string_slice` (slicing-based). The `reverse_string_slice` function is highlighted. Below the editor, a terminal window shows the command `python3 .\11.exe` being executed, which runs the script. The output of the script is `olleh`, which is the reverse of the input string `hello`.

```
__pycache__ > dsa.py > ai.py > assignment > reverse_string_slice
1 # Loop-based string reversal
2
3 # Method 1: Using a for loop with reversed index
4 def reverse_string_1(s):
5     reversed_str = ""
6     for i in range(len(s) - 1, -1, -1):
7         reversed_str += s[i]
8     return reversed_str
9 reverse_string_1("hello") #output: "olleh"
10 def reverse_string_slice(text):
11     return text[::-1]
12
13
14
15
```

PROBLEMS OUTPUT TERMINAL PORTS Python: assignment

PS C:\Users\yadug\OneDrive\Desktop\python training> & C:/Users/yadug/AppData/Local/Microsoft/WindowsApps/python3
● .11.exe "c:/Users/yadug/OneDrive/Desktop/python training/__pycache__/dsa.py/ai.py/assignment"
○ PS C:\Users\yadug\OneDrive\Desktop\python training>

Explanation:

Both approaches correctly reverse strings.

Slicing-based reversal is superior in performance, readability, and scalability.