Assignment-1.5

Name-Arsha vardhini

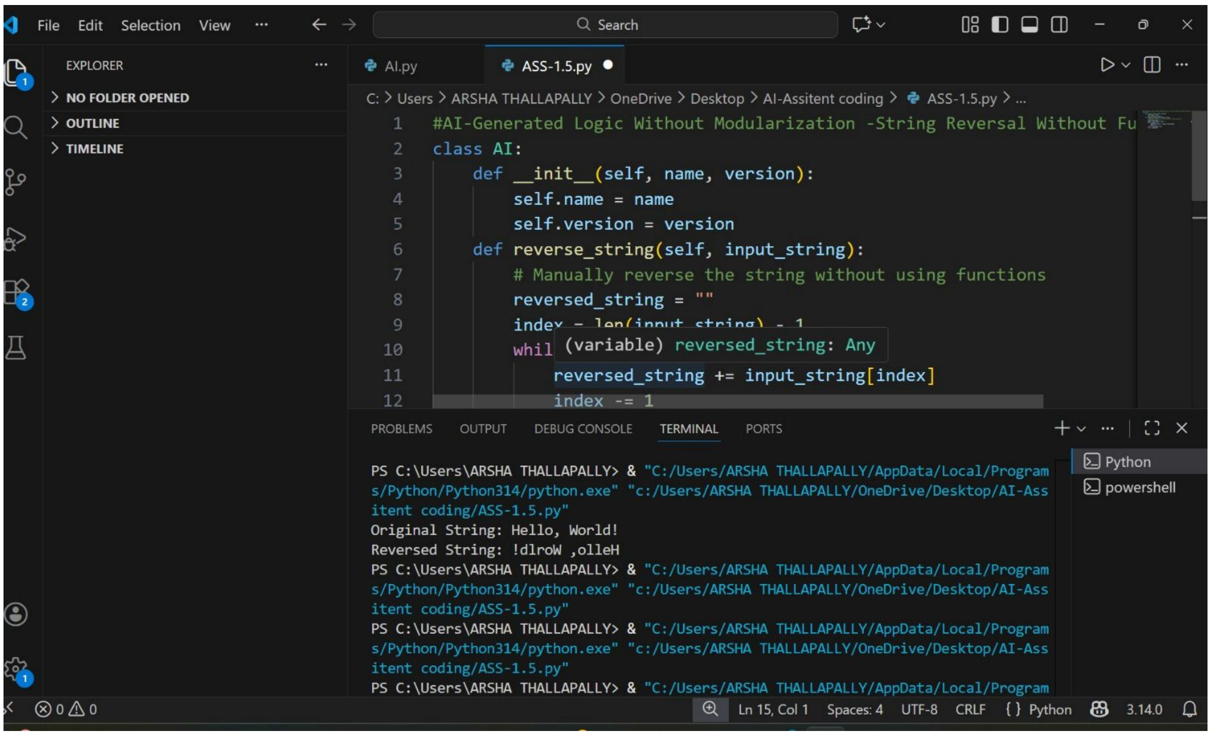
Roll.No:2303A51600

Batch-29

TASK-1

PROMPT: AI-GENERATED LOGIC WITHOUT MODULARIZATION (STRING REVERSAL WITHOUT FUNCTIONS)

CODE:



```python
#AI-Generated Logic Without Modularization -String Reversal Without Fu
class AI:
    def __init__(self, name, version):
        self.name = name
        self.version = version
    def reverse_string(self, input_string):
        # Manually reverse the string without using functions
        reversed_string = ""
        index = len(input_string) - 1
        whil (variable) reversed_string: Any
            reversed_string += input_string[index]
            index -= 1
```

```
PS C:\Users\ARSHA THALLAPALLY> & "C:/Users/ARSHA THALLAPALLY/AppData/Local/Program
s/Python/Python314/python.exe" "c:/Users/ARSHA THALLAPALLY/OneDrive/Desktop/AI-Ass
itent coding/ASS-1.5.py"
Original String: Hello, World!
Reversed String: !dlroW ,olleH
PS C:\Users\ARSHA THALLAPALLY> & "C:/Users/ARSHA THALLAPALLY/AppData/Local/Program
s/Python/Python314/python.exe" "c:/Users/ARSHA THALLAPALLY/OneDrive/Desktop/AI-Ass
itent coding/ASS-1.5.py"
PS C:\Users\ARSHA THALLAPALLY> & "C:/Users/ARSHA THALLAPALLY/AppData/Local/Program
s/Python/Python314/python.exe" "c:/Users/ARSHA THALLAPALLY/OneDrive/Desktop/AI-Ass
itent coding/ASS-1.5.py"
PS C:\Users\ARSHA THALLAPALLY> & "C:/Users/ARSHA THALLAPALLY/AppData/Local/Program
```

OBSERVATION:

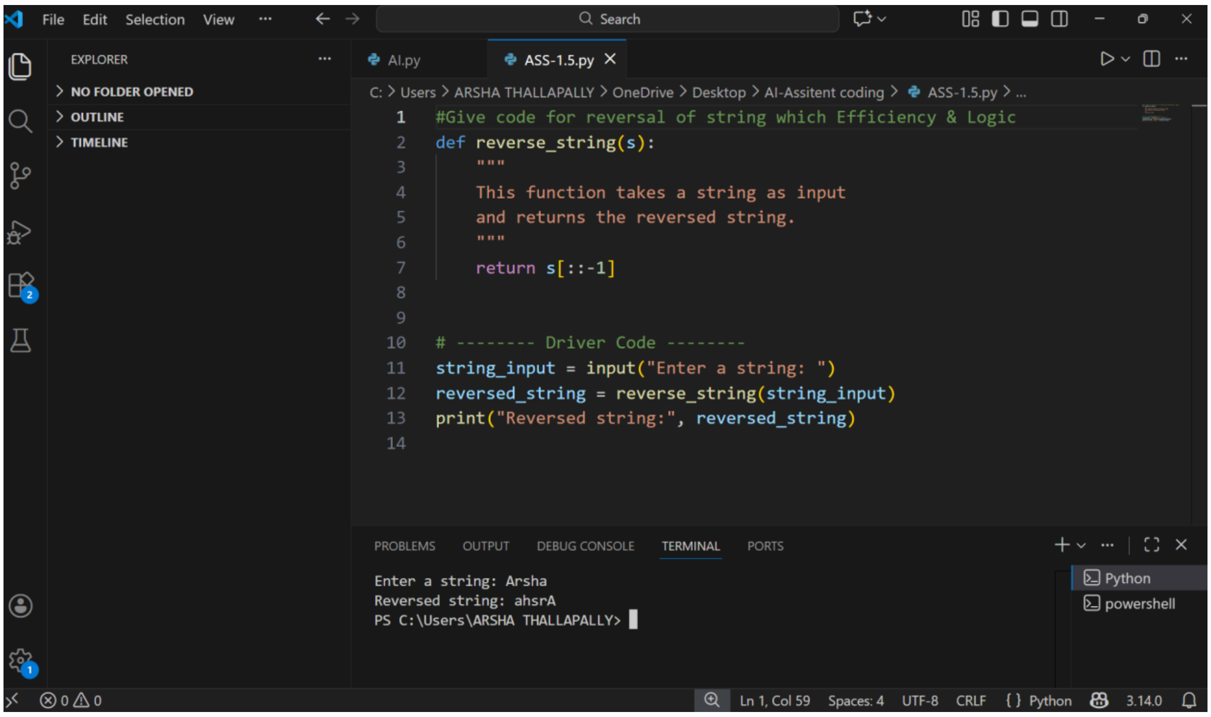
The program successfully reverses the given string using a manual looping approach without built-in reverse functions.
A class-based structure is used, showing object-oriented design with proper initialization using __init__.
The string reversal logic works by iterating from the last index to the

first, appending characters correctly.
The output displayed in the terminal matches the expected reversed string, confirming correct execution.
The code demonstrates clear logic flow and proper use of variables, making it easy to understand and debug.

TASK-2:

PROMPT: Give code for reversal of string which Efficiency & Logic Optimization

CODE:



```python
#Give code for reversal of string which Efficiency & Logic
def reverse_string(s):
    """
    This function takes a string as input
    and returns the reversed string.
    """
    return s[::-1]


# -------- Driver Code --------
string_input = input("Enter a string: ")
reversed_string = reverse_string(string_input)
print("Reversed string:", reversed_string)
```

```
Enter a string: Arsha
Reversed string: ahsrA
PS C:\Users\ARSHA THALLAPALLY>
```

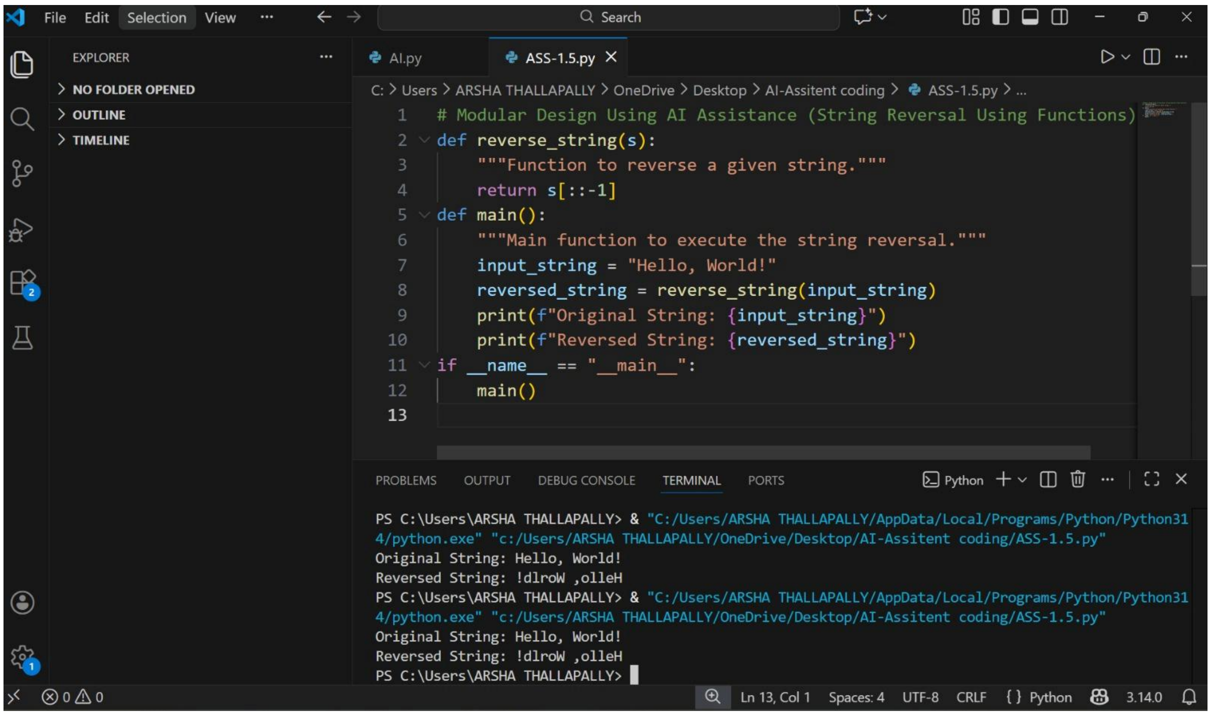
OBSERVATION:

The string reversal is performed using Python slicing, which processes the string from the end to the beginning in a single operation. Since strings are immutable, a new reversed string is created without modifying the original one. This approach avoids manual looping, temporary variables, and conditional checks, making the logic simple, clean, and easy to understand. Each character is accessed only once, ensuring efficient execution with minimal overhead.

TASK:3

PROMPT: Modular Design Using AI Assistance (String Reversal Using Functions)

CODE:



```python
# Modular Design Using AI Assistance (String Reversal Using Functions)
def reverse_string(s):
    """Function to reverse a given string."""
    return s[::-1]
def main():
    """Main function to execute the string reversal."""
    input_string = "Hello, World!"
    reversed_string = reverse_string(input_string)
    print(f"Original String: {input_string}")
    print(f"Reversed String: {reversed_string}")
if __name__ == "__main__":
    main()
```

```
PS C:\Users\ARSHA THALLAPALLY> & "C:/Users/ARSHA THALLAPALLY/AppData/Local/Programs/Python/Python31
4/python.exe" "c:/Users/ARSHA THALLAPALLY/OneDrive/Desktop/AI-Assitent coding/ASS-1.5.py"
Original String: Hello, World!
Reversed String: !dlroW ,olleH
PS C:\Users\ARSHA THALLAPALLY> & "C:/Users/ARSHA THALLAPALLY/AppData/Local/Programs/Python/Python31
4/python.exe" "c:/Users/ARSHA THALLAPALLY/OneDrive/Desktop/AI-Assitent coding/ASS-1.5.py"
Original String: Hello, World!
Reversed String: !dlroW ,olleH
PS C:\Users\ARSHA THALLAPALLY>
```

OBSERVATION:

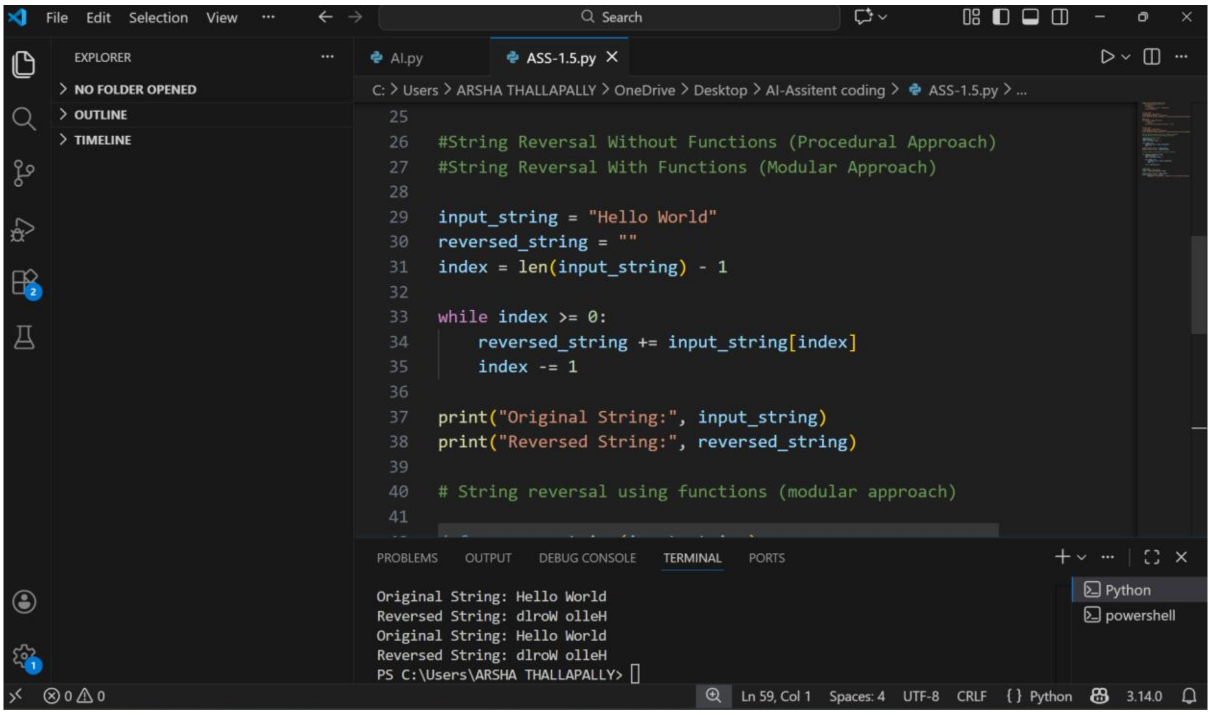
The program uses a separate function to reverse the string, clearly demonstrating modular design.
The function takes the string as input and returns the reversed string, keeping the logic well-structured.
The main part of the code handles only input and output, improving

readability.
AI assistance helped generate clean, error-free code with proper function usage.
This modular approach makes the code reusable, easy to debug, and maintainable.

TASK-4

prompt: Comparative Analysis – Procedural vs Modular Approach (With vs Without Functions)

Code:



```python

#String Reversal Without Functions (Procedural Approach)
#String Reversal With Functions (Modular Approach)

input_string = "Hello World"
reversed_string = ""
index = len(input_string) - 1

while index >= 0:
    reversed_string += input_string[index]
    index -= 1

print("Original String:", input_string)
print("Reversed String:", reversed_string)

# String reversal using functions (modular approach)

```

```
Original String: Hello World
Reversed String: dlroW olleH
Original String: Hello World
Reversed String: dlroW olleH
PS C:\Users\ARSHA THALLAPALLY>
```

OBSERVATION:

Code Clarity: Procedural code mixes everything and is harder to read, while modular code with functions is cleaner and organized.

Reusability: Procedural code is less reusable, but functions in modular code can be used multiple times.

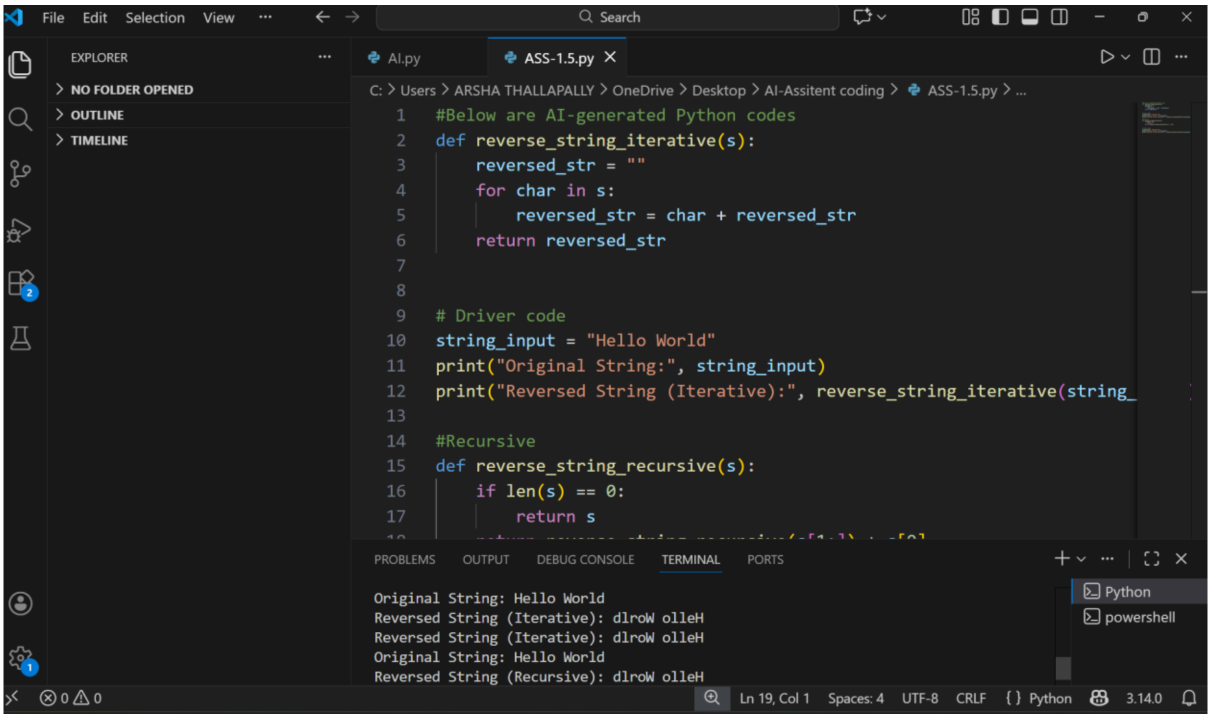
Debugging Ease: Procedural code is harder to debug, whereas modular code allows testing and fixing parts independently.

Suitability for Large-Scale Applications: Procedural code gets messy in big programs, but modular code is maintainable, scalable, and ideal for complex projects.

Task5:

Prompt: AI-generated Python codes Iterative vs recursion

Code:


File Edit Selection View ···
Search
EXPLORER
NO FOLDER OPENED
OUTLINE
TIMELINE
AI.py
ASS-1.5.py
C: > Users > ARSHA THALLAPALLY > OneDrive > Desktop > AI-Assitent coding > ASS-1.5.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
powershell
Ln 19, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.14.0

```python
#Below are AI-generated Python codes
def reverse_string_iterative(s):
    reversed_str = ""
    for char in s:
        reversed_str = char + reversed_str
    return reversed_str


# Driver code
string_input = "Hello World"
print("Original String:", string_input)
print("Reversed String (Iterative):", reverse_string_iterative(string_

#Recursive
def reverse_string_recursive(s):
    if len(s) == 0:
        return s
```

```
Original String: Hello World
Reversed String (Iterative): dlroW olleH
Reversed String (Iterative): dlroW olleH
Original String: Hello World
Reversed String (Recursive): dlroW olleH
```

Observation:

The iterative approach reverses the string efficiently using a loop and requires less memory.
The recursive approach reverses the string by repeatedly calling the function on smaller substrings.
Both methods produce the same correct reversed output for the given

input string.
The iterative method is faster and more suitable for large strings.
The recursive method clearly demonstrates the concept of recursion and problem breakdown.