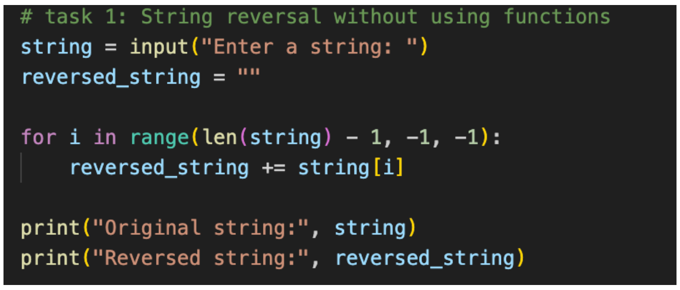
ASSIGNMENT-1.5

NAME: CH.MANMITH VARDHAN

2303A51321

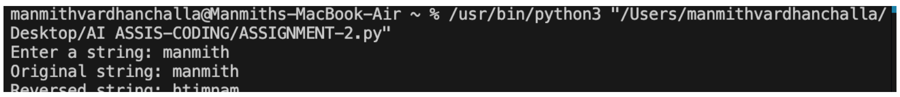
BATCH-19

Task – 1: AI-Generated Logic Without Modularization (String Reversal Without Functions)



```python
# task 1: String reversal without using functions
string = input("Enter a string: ")
reversed_string = ""

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

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

Output:

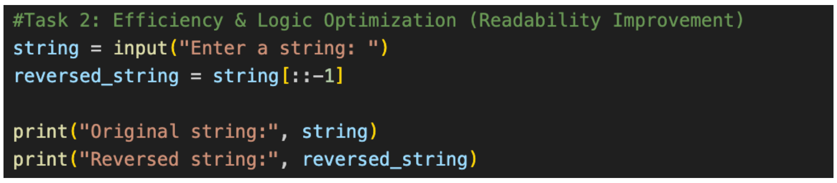


```
manmithvardhanchalla@Manmiths-MacBook-Air ~ % /usr/bin/python3 "/Users/manmithvardhanchalla/
Desktop/AI ASSIS-CODING/ASSIGNMENT-2.py"
Enter a string: manmith
Original string: manmith
Reversed string: htimnam
```

Justification:

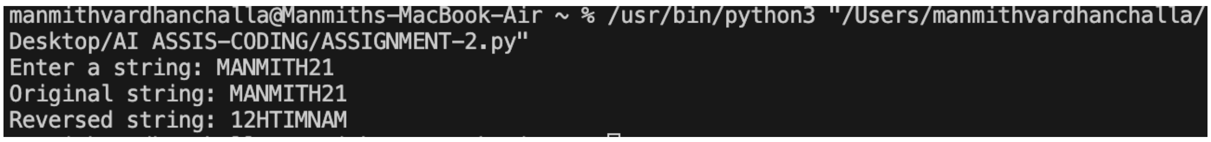
The given program reverses a string without using functions by iterating through each character and adding it to the beginning of a new string. This logic correctly produces the reversed output as shown. The approach is simple and easy to understand for small programs. However, the code is not reusable since the logic is written directly in the main block. Debugging and maintenance become difficult if the program grows. Using functions would improve readability, reusability, and suitability for larger applications.

Task -2: Efficiency & Logic Optimization (Readability Improvement)



```python
#Task 2: Efficiency & Logic Optimization (Readability Improvement)
string = input("Enter a string: ")
reversed_string = string[::-1]

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

Output:

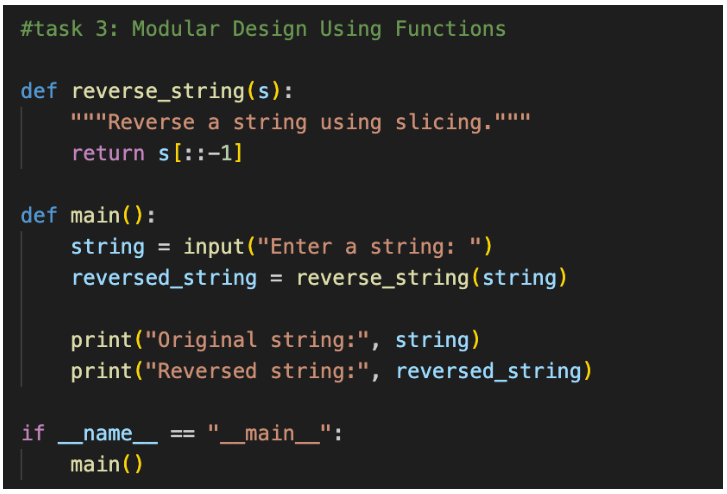


```
manmithvardhanchalla@Manmiths-MacBook-Air ~ % /usr/bin/python3 "/Users/manmithvardhanchalla/
Desktop/AI ASSIS-CODING/ASSIGNMENT-2.py"
Enter a string: MANMITH21
Original string: MANMITH21
Reversed string: 12HTIMNAM
```

Justification:

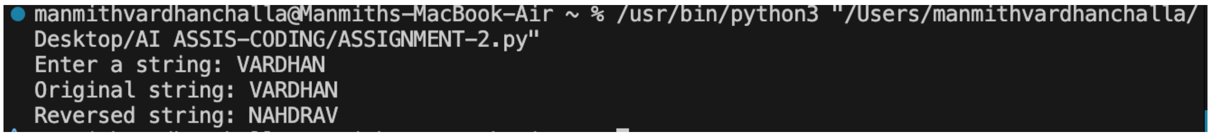
The manual approach reverses the string using a loop by adding each character to the beginning of a new string. The slicing approach reverses the string using Python's built-in slicing [::-1], which is shorter and more readable. Both methods produce the same correct output. The manual method helps in understanding the logic behind string reversal. The slicing method is more efficient and preferred for real-world applications. Hence, slicing is best for performance, while the manual approach is useful for learning purposes.

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



```python
#task 3: Modular Design Using Functions

def reverse_string(s):
    """Reverse a string using slicing."""
    return s[::-1]

def main():
    string = input("Enter a string: ")
    reversed_string = reverse_string(string)

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

if __name__ == "__main__":
    main()
```

Output:



```
● manmithvardhanchalla@Manmiths-MacBook-Air ~ % /usr/bin/python3 "/Users/manmithvardhanchalla/
  Desktop/AI ASSIS-CODING/ASSIGNMENT-2.py"
  Enter a string: VARDHAN
  Original string: VARDHAN
  Reversed string: NAHDRAV
```

Explanation:

The function-based approach reverses the string using a reusable function, making the code clean and well-structured. The procedural approach performs string reversal directly in the main program without functions. Both approaches produce the same correct output. The function-based method is easier to reuse and maintain in larger programs. Debugging is simpler in the modular approach compared to procedural code. Hence, the function-based approach is more suitable for real-world applications, while the procedural approach is good for small tasks.

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

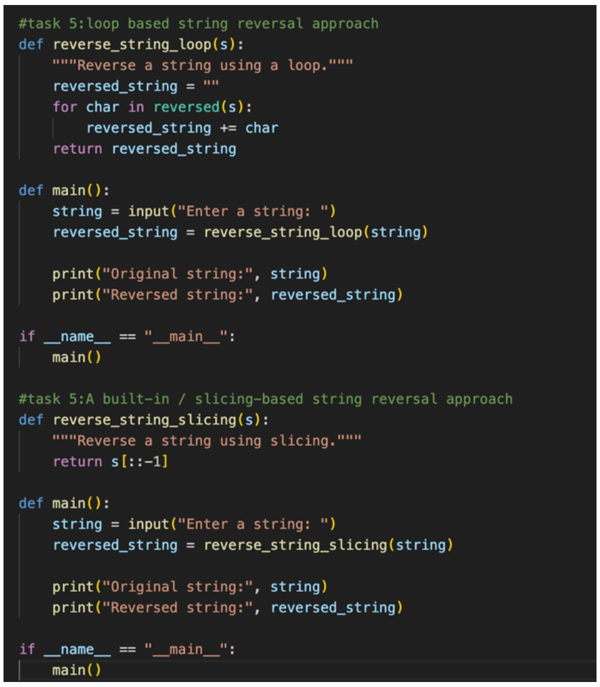
In this task, two string reversal programs generated using GitHub Copilot are compared:

- Without functions (Procedural approach)

- With functions (Modular approach)

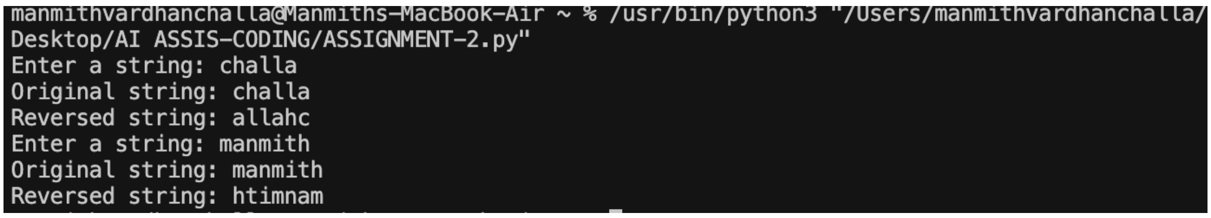
Aspect	Without Functions	With Functions
Code clarity	Hard to understand when code increases	Easy to read and understand
Reusability	Cannot reuse code	Function can be reused
Debugging	Difficult	Easy
Maintainability	Hard to modify	Easy to modify
Large applications	Not suitable	Suitable

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



```python
#task 5:loop based string reversal approach
def reverse_string_loop(s):
    """Reverse a string using a loop."""
    reversed_string = ""
    for char in reversed(s):
        reversed_string += char
    return reversed_string

def main():
    string = input("Enter a string: ")
    reversed_string = reverse_string_loop(string)

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

if __name__ == "__main__":
    main()

#task 5:A built-in / slicing-based string reversal approach
def reverse_string_slicing(s):
    """Reverse a string using slicing."""
    return s[::-1]

def main():
    string = input("Enter a string: ")
    reversed_string = reverse_string_slicing(string)

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

if __name__ == "__main__":
    main()
```

Output:



```
manmithvardhanchalla@Manmiths-MacBook-Air ~ % /usr/bin/python3 "/Users/manmithvardhanchalla/
Desktop/AI ASSIS-CODING/ASSIGNMENT-2.py"
Enter a string: challa
Original string: challa
Reversed string: allahc
Enter a string: manmith
Original string: manmith
Reversed string: htimnam
```

Explanation:

Both iterative and recursive approaches generate the same Fibonacci sequence correctly. The iterative approach uses a loop and runs faster with less memory usage. The recursive approach follows a mathematical definition and is easier to understand conceptually. However, recursion involves repeated function calls, which increases time and space usage. For large input values, the iterative method is more efficient and reliable. Therefore, iteration is preferred for performance, while recursion is useful for learning and clarity.