

Lab Assignment 1.5

Environment Setup – GitHub Copilot and VS Code Integration +Understanding AI-assisted Coding Workflow

V. RAVITEJA

2303A51942

Batch-12

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

Prompt: Write a code to reverse a given string without using built-in reverse functions.

Code:

```
code1.py  X  
C:\Users\chari\OneDrive\Desktop\AI Assistant coding> code1.py > reverse_string  
1 # Write a code to reverse a given string without using built-in reverse functions.  
2 def reverse_string(s):  
3     reversed_s = ""  
4     for char in s:  
5         reversed_s = char + reversed_s  
6     return reversed_s  
7 input_string = input("Enter a string: ")  
8 print("Reversed string:", reverse_string(input_string))
```

Output:

```
Enter a string: Hello, World..!  
Reversed string: !..dlrow ,olleH
```

Explanation: This code defines a function `reverse_string` that reverses a given string without using built-in reverse functions. It initializes an empty string `reversed_s` and iterates over each character in the input string `s`. For each character, it prepends the character to `reversed_s`, effectively reversing the order. The function returns the reversed string. The user is prompted to enter a string, and the reversed result is printed to the console.

Task 2: Efficiency & Logic Optimization (Readability Improvement)

Prompt: # Simplify the string reversal and Improve readability and efficiency

Code:

```
# Simplify the string reversal and Improve readability and efficiency
def reverse_string(s):
    return ''.join(reversed(s))
input_string = input("Enter a string: ")
print("Reversed string:", reverse_string(input_string))
```

Output:

```
Enter a string: Hello, World..90
Reversed string (optimal): 09..dlrow ,olleH
```

Explanation: This code helps you reverse any string you type in. It defines a function called `reverse_string` that uses Python's built-in `reversed()` function to flip the order of the characters. The `join()` method then puts these characters back together into a single string. When you run the code, it asks you to enter a string. After you type something and press Enter, it shows you your string, but backwards.

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

Prompt: Write code using functional programming to reverse a string and the function should be not built-in.

Code:

```
# Write code using functional programming to reverse a string and the function should be not built-in
def reverse_string_functional(s):
    return ''.join(map(lambda x: s[len(s)-1 - s.index(x)], s))
input_string = input("Enter a string: ")
print("Reversed string (functional):", reverse_string_functional(input_string))
```

Output:

```
Enter a string: Hello, World..!
Reversed string: !..dlrow ,olleH
```

Explanation: This code lets you reverse a string using a functional programming style. It defines a function that uses `map()` and a little math to pick out each character from the end of your string. The `join()` part puts all those characters together into a new, reversed string. When you run the code, it asks you to type something in. After you press Enter, it shows your string backwards, using this creative approach.

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

Comparision table:

| Aspect | Without Functions (Task 1) | With Functions (Task 3) |
|---------------------|--|---|
| Structure | Uses a simple loop inside a function to reverse the string. | Uses functional programming (<code>map</code> and <code>lambda</code>) inside a function to reverse the string. |
| Readability | Easy to understand; logic is clear and straightforward. | Less readable; the <code>lambda</code> and <code>index</code> logic may confuse beginners. |
| Efficiency | Efficient for most cases; handles repeated characters correctly. | May produce incorrect results for strings with repeated characters due to <code>s.index(x)</code> usage. |
| Functional Approach | Imperative style (step-by-step reversal). | Functional style (mapping and transformation). |
| Output | Correctly reverses any input string. | May not reverse strings with duplicate characters correctly. |

Explanation: The first program (Task 1) is more reliable and readable for reversing strings. The second program (Task 3) uses functional programming but has a logical flaw with repeated characters, making it less robust. For practical use, Task 1's approach is preferred.

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

Prompt-1: A loop-based string reversal approach

Code:

```
28
29 #A loop-based string reversal approach
30 def reverse_string_loop(s):
31     reversed_s = []
32     for char in s:
33         reversed_s.insert(0, char)
34     return ''.join(reversed_s)
35 input_string = input("Enter a string: ")
36 print("Reversed string (loop-based):", reverse_string_loop(input_string))
```

Output:

```
Enter a string: Hi There...!!
```

```
Reversed string (loop-based): !...erehT iH
```

Explanation: This code helps you reverse a string using a loop. It creates an empty list and goes through each character in your input. Each new character is added to the front of the list, so the order gets flipped. After the loop, it joins all the characters together into a single, reversed string. When you run it, you type something in, and it prints your string backwards.

Prompt-2: A built-in / slicing-based string reversal approach

Code:

```
#A built-in / slicing-based string reversal approach
def reverse_string_slicing(s):
    return s[::-1]
input_string = input("Enter a string: ")
print("Reversed string (slicing-based):", reverse_string_slicing(input_string))
```

Output:

```
Enter a string: How are you..?
```

```
Reversed string (slicing-based): ?..uoy era woh
```

Explanation: This code reverses a string in the quickest way possible using Python's slicing feature.

The `s[::-1]` part tells Python to take the string and read it backwards, from end to start. You're asked to type in a string, and the code instantly flips it for you. It then prints out your original string, but in reverse order. This method is simple, fast, and very easy to use.