AI Assisted Coding

Assignment 1

Name: Vivek

Hallticket:2303A52009

Task-01

Task Description

Use GitHub Copilot to generate a Python program that:

➢ Reverses a given string

➢ Accepts user input

➢ Implements the logic directly in the main code

➢ Does not use any user-defined functions

Prompt Given: # write a python code to reverse a string which should take input from user and don't use any user defined functions.

```
# Optional: Add a loop to allow multiple reversals
while True:
    choice = input("Do you want to reverse another string? (yes/no): ")
    if choice.lower() != 'yes':
        break
    a = input("Enter a string: ")
    reversed_string = ""
    for char in a:
        reversed_string = char + reversed_string
    print("Reversed string is:", reversed_string)
```

Output:

```
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> python -u "c:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding\Lab_02.py"
Do you want to reverse another string? (yes/no): yes
Enter a string: Projector
Reversed string is: rotcejorP
```

Justification:

Task-02

Task Description
Examine the Copilot-generated code from Task 1 and improve it by:

➢ Removing unnecessary variables

➢ Simplifying loop or indexing logic

➢ Improving readability

➢ Use Copilot prompts like:

▪ "Simplify this string reversal code"

▪ "Improve readability and efficiency

Prompt Given:
Generate code by removing unnecessary variables, simplifying loops and indexing logic, and improving overall readability and efficiency without changing the output

Original Version Code:

```
while True:
    choice = input("Do you want to reverse another string? (yes/no): ")
    if choice.lower() != 'yes':
        break
    a = input("Enter a string: ")
    reversed_string = ""
    for char in a:
        reversed_string = char + reversed_string
    print("Reversed string is:", reversed_string)
```

Simplified version Code :

```
    a = input("Enter a string: ")
    reversed_string = ""
    for char in a:
        reversed_string = char + reversed_string
    print("Reversed string is:", reversed_string)
```

Output:

```
Enter a string: Python
Reversed string is: nohtyP
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
```

Justification:

Task-03

Task Description

Use GitHub Copilot to generate a function-based Python program that:

➢ Uses a user-defined function to reverse a string

➢ Returns the reversed string

➢ Includes meaningful comments (AI-assisted)

Prompt Given:

Create a Python function that reverses a given string and returns the result. Keep the logic simple.

Code:

```
def reverse_string(s):
    return s[::-1]

a = input("Enter a string: ")
print("Reversed string is:", reverse_string(a))
```

Output:

```
Enter a string: Python Prgm
Reversed string is: mgrP nohtyP
Enter a string:
```

Justification:

Task 04
Task Description
Compare the Copilot-generated programs:
➢ Without functions (Task 1)
➢ With functions (Task 3)
Analyze them based on:
➢ Code clarity
➢ Reusability
➢ Debugging ease
➢ Suitability for large-scale applications
Prompt Given:
Compare the two Python programs below: one written without functions and one using user-defined functions.
Analyze and clearly explain the differences in terms of code clarity, reusability, ease of debugging,
and suitability for large-scale applications. Keep the comparison structured and easy to understand output in tabular form.

```
# Comparison of String Reversal Approaches

comparison_table = """
╔═══════════════════════╦═══════════════════════════╦═══════════════════════════════════╗
║ Aspect                ║ Without Functions         ║ With User-Defined Functions       ║
╠═══════════════════════╬═══════════════════════════╬═══════════════════════════════════╣
║ Code Clarity          ║ Linear, easy to follow    ║ Clear intent, self-documenting    ║
║                       ║ for simple tasks          ║ through function name             ║
╠═══════════════════════╬═══════════════════════════╬═══════════════════════════════════╣
║ Reusability           ║ Code duplication if used  ║ Single definition, multiple       ║
║                       ║ multiple times            ║ calls reduce redundancy           ║
╠═══════════════════════╬═══════════════════════════╬═══════════════════════════════════╣
║ Ease of Debugging     ║ Harder to isolate bugs    ║ Easier to test and debug          ║
║                       ║ in larger codebases       ║ specific functionality            ║
╠═══════════════════════╬═══════════════════════════╬═══════════════════════════════════╣
║ Maintainability       ║ Changes needed in         ║ Update logic in one place         ║
║                       ║ multiple locations        ║ affects all usages                ║
╠═══════════════════════╬═══════════════════════════╬═══════════════════════════════════╣
║ Testability           ║ Difficult to unit test    ║ Easy to test with different       ║
║                       ║ isolated logic            ║ inputs systematically             ║
╠═══════════════════════╬═══════════════════════════╬═══════════════════════════════════╣
║ Large-Scale Apps      ║ Not suitable; leads to    ║ Highly suitable; promotes         ║
║                       ║ spaghetti code            ║ modularity and scalability        ║
╠═══════════════════════╬═══════════════════════════╬═══════════════════════════════════╣
║ Memory Efficiency     ║ Code repeated in memory   ║ Single definition in memory       ║
║                       ║ multiple times            ║ with multiple references          ║
╚═══════════════════════╩═══════════════════════════╩═══════════════════════════════════╝
"""

print(comparison_table)

# Summary
print("\n✓ Recommendation: Use user-defined functions for production code")
print("✓ Use without functions: Learning, prototyping, or very simple scripts")
```

Justification :

Task-05:

Given Prompt:

Generate two Python implementations to reverse a string, one using a loop and one using slicing or built-in methods, and then compare them based on execution flow, time complexity, performance on large inputs, and when each approach should be used.

Generate code

Add Context...

```python
# Implementation 1: Using a loop
def reverse_with_loop(s):
    reversed_string = ""
    for char in s:
        reversed_string = char + reversed_string
    return reversed_string

# Implementation 2: Using slicing
def reverse_with_slicing(s):
    return s[::-1]

# Test both implementations
test_string = "Hello, World!"
print("Original string:", test_string)
print("Reversed (Loop):", reverse_with_loop(test_string))
print("Reversed (Slicing):", reverse_with_slicing(test_string))

# Performance comparison on large input
large_string = "a" * 100000

# Time loop approach
start = time.time()
for _ in range(100):
    reverse_with_loop(large_string)
loop_time = time.time() - start

# Time slicing approach
start = time.time()
for _ in range(100):
    reverse_with_slicing(large_string)
slicing_time = time.time() - start

print("\n" + "="*60)
print("PERFORMANCE COMPARISON (100,000 character string, 100 iterations)")
print("="*60)
```

Output:

```
Reversed (Loop): !dlroW ,olleH
Reversed (Slicing): !dlroW ,olleH

============================================================
PERFORMANCE COMPARISON (100,000 character string, 100 iterations)
============================================================
Loop approach:    36.9229 seconds
Slicing approach: 0.0137 seconds
Slicing is 2689.5x faster

============================================================
COMPARISON TABLE
============================================================
```

Aspect	Loop Approach	Slicing Approach
Time Complexity (string concat)	$O(n^2)$ (string immutable)	O(n) (optimized)
Space Complexity	O(n)	O(n)
Readability	Verbose	Concise, Pythonic
Large Inputs	Slow, inefficient	Fast, optimized
Educational Value	Great for learning	Practical, modern
When to Use	Learning/Teaching Understanding string mechanics	Production code Real-world apps Performance-critical

Justification: