AI ASSISTANT CODING

Name: J Sai Harini

Ht no: 2303A52426

Batch: 35

Q.No. Question Expected

Time

to

complete

1

Lab 1: Environment Setup – GitHub Copilot and VS Code Integration +

Understanding AI-assisted Coding Workflow

Lab Objectives:

❖ To install and configure GitHub Copilot in Visual Studio Code.

Week1 -

Monday

❖ To explore AI-assisted code generation using GitHub Copilot.

❖ To analyze the accuracy and effectiveness of Copilot's code suggestions.

❖ To understand prompt-based programming using comments and code

context

Lab Outcomes (LOs):

After completing this lab, students will be able to:

❖ Set up GitHub Copilot in VS Code successfully.

❖ Use inline comments and context to generate code with Copilot.

❖ Evaluate AI-generated code for correctness and readability.

❖ Compare code suggestions based on different prompts and programming

styles.

Task 0

❖ Install and configure GitHub Copilot in VS Code. Take screenshots of each

step.

Expected Output

❖ Install and configure GitHub Copilot in VS Code. Take screenshots of each

step.

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

Functions)

❖ Scenario

You are developing a basic text-processing utility for a messaging

application.

❖ 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

❖ Expected Output

➢ Correct reversed string

➢ Screenshots showing Copilot-generated code suggestions

➢ Sample inputs and outputs

Task 2: Efficiency & Logic Optimization (Readability Improvement)

❖ Scenario

The code will be reviewed by other developers.

❖ 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”

Hint:

Prompt Copilot with phrases like

“optimize this code”, “simplify logic”, or “make it more readable”

❖ Expected Output

➢ Original and optimized code versions

➢ Explanation of how the improvements reduce time complexity

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

❖ Scenario

The string reversal logic is needed in multiple parts of an application.

❖ 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)

❖ Expected Output

➢ Correct function-based implementation

➢ Screenshots documenting Copilot's function generation

➢ Sample test cases and outputs

Task 4: Comparative Analysis – Procedural vs Modular Approach (With vs

Without Functions)

❖ Scenario

You are asked to justify design choices during a code review.

❖ 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

❖ Expected Output

Comparison table or short analytical report

Task 5: AI-Generated Iterative vs Recursive Fibonacci Approaches (Different

Algorithmic Approaches to String Reversal)

❖ Scenario

Your mentor wants to evaluate how AI handles alternative logic paths.

❖ Task Description

Prompt GitHub Copilot to generate:

➢ A loop-based string reversal approach

➢ A built-in / slicing-based string reversal approach

❖ Expected Output

➢ Two correct implementations

➢ Comparison discussing:

▪ Execution flow

▪ Time complexity

▪ Performance for large inputs

▪ When each approach is appropriate

Note: Report should be submitted as a word document for all tasks in a

single document with prompts, comments & code explanation, and output

and if required, screenshots.

Solutions:

```
# Python program to reverse a string


user_input = input("Enter a string to reverse: ")
print(f"Original: {user_input}\nReversed: {user_input[::-1]}")


# Python program to reverse a string using a user-defined function
# Demonstrates function-based approach with proper documentation


def reverse_string(text):
    """
    Reverses the given string using Python's slicing technique.

    Args:
        text (str): The input string to be reversed
```

```
    Returns:
        str: The reversed string
    """
    return text[::-1]

# Main program execution
if __name__ == "__main__":
    # Prompt user for input
    user_input = input("Enter a string to reverse: ")

    # Call the function to get reversed string
    result = reverse_string(user_input)

    # Display the results
    print(f"Original: {user_input}")
    print(f"Reversed: {result}")
```

1. CODE CLARITY

Without Functions (Task 1)

```python
# Python program to reverse a string

user_input = input("Enter a string to reverse: ")
print(f"Original: {user_input}\nReversed: {user_input[::-1]}")
```

Observations:

- ✓ Extremely concise (4 lines)
- ✓ Easy to understand at a glance
- ✗ Mixes input/output logic with core functionality
- ✗ No documentation of what the reversal does

With Functions (Task 3)

```python
def reverse_string(text):
    """
    Reverses the given string using Python's slicing technique.
    ...
    """
    return text[::-1]

if __name__ == "__main__":
```

```
    user_input = input("Enter a string to reverse: ")
    result = reverse_string(user_input)
    print(f"Original: {user_input}\nReversed: {result}")
```

Observations:

- ✓ Clear separation of concerns
- ✓ Comprehensive docstring documentation
- ✓ Explicit function purpose and behavior
- ✗ More verbose (27 lines with documentation)
- ✓ Follows Python best practices with `if __name__ == "__main__":`

Winner: Task 3 - Better documentation and separation of concerns make intent clearer.

---

2. REUSABILITY

Without Functions (Task 1)

Reusability Score: 1/5

- + Cannot reuse the reversal logic in other parts of the program

- + Cannot import and use in other modules

- + Must duplicate code if reversal is needed elsewhere

- + Tightly coupled with input/output operations

Example Problem:

```python
# To reverse multiple strings, must repeat the logic
string1_reversed = string1[::-1]
string2_reversed = string2[::-1]
string3_reversed = string3[::-1]
```

With Functions (Task 3)

Reusability Score: 5/5

- ✓ Function can be imported into other modules

- ✓ Can reverse any number of strings without duplication

- ✓ Logic is isolated and independent

- ✓ Can be used in different contexts (APIs, GUIs, batch processing)

Example Benefit:

```python
from reverse_string_function import reverse_string

strings = ["hello", "world", "python"]
reversed_strings = [reverse_string(s) for s in strings]
# Result: ['olleh', 'dlrow', 'nohtyp']
```

Winner: Task 3 - Dramatically superior for code reuse and modularity.

---

3. DEBUGGING EASE

Without Functions (Task 1)

Debugging Score: 2/5

- + Hard to isolate which part has issues
- + No clear entry/exit points for testing
- + Cannot debug the reversal logic independently
- + Changes require modifying the entire script

Challenges:

```python
# Is the problem in input handling or reversal logic?
print(f"Original: {user_input}\nReversed: {user_input[::-1]}")
# Difficult to pinpoint issues
```

With Functions (Task 3)

Debugging Score: 5/5

- ✓ Can test the function independently
- ✓ Can add breakpoints specifically in the function
- ✓ Unit testing is straightforward
- ✓ Can isolate bugs to specific sections

Example Testing:

```python
# Easy to test the function directly
assert reverse_string("hello") == "olleh"
assert reverse_string("world") == "dlrow"
assert reverse_string("") == ""
```

Winner: Task 3 - Function-based design enables systematic debugging and testing.

---

4. SUITABILITY FOR LARGE-SCALE APPLICATIONS

Without Functions (Task 1)

Large-Scale Suitability: 1/5

- + Not scalable to larger applications
- + No code organization or structure
- + Impossible to maintain in teams
- + Cannot be part of a larger project
- + No testing framework compatibility

Why It Fails:

- Single-purpose scripts only
- Cannot integrate with frameworks
- No separation of business logic from I/O
- Violates Single Responsibility Principle

With Functions (Task 3)

Large-Scale Suitability: 5/5

- ✓ Easily integrates into larger projects
- ✓ Follows software engineering principles
- ✓ Compatible with unit testing frameworks (pytest, unittest)
- ✓ Can be part of modules and packages
- ✓ Supports code organization and architectural patterns

Enterprise-Ready Features:

```python
# Can be used in web frameworks
from flask import Flask
from reverse_string_function import reverse_string

app = Flask(__name__)

@app.route('/reverse/<text>')
def api_reverse(text):
    return {"reversed": reverse_string(text)}

# Can be tested with pytest
```

```
def test_reverse_string():
    assert reverse_string("test") == "tset"

# Can be integrated into larger modules
# from utils.string_operations import reverse_string
```

Winner: Task 3 - Essential for professional, scalable software development.

SUMMARY COMPARISON TABLE

Criterion	Task 1 (No Functions)	Task 3 (With Functions)
Lines of Code	4	27
Code Clarity	★★★ (Concise)	★★★★★ (Well-documented)
Reusability	★ (None)	★★★★★ (Excellent)
Debugging Ease	★★ (Difficult)	★★★★★ (Easy)

| **Large-Scale Apps** | ★ (Not suitable) | ★★★★★ (Ideal) |

| **Testing** | + (Not testable) | ✓ (Fully testable) |

| **Maintenance** | + (Difficult) | ✓ (Easy) |

CONCLUSION

Use Task 1 (No Functions) When:

- Writing quick, throwaway scripts
- Learning Python basics
- One-time utility tasks
- Minimal complexity required

Use Task 3 (With Functions) When:

- Building production code
- Working in teams
- Planning to extend the application
- Requiring code quality and maintainability
- Needing unit tests
- Integrating into larger projects

Best Practice Recommendation: **Task 3 is the professional standard.** Even for simple programs, using functions establishes good habits and makes code enterprise-ready. The minimal overhead of function definition pays dividends in reusability, testing, and maintenance.

```
# Python program demonstrating different string reversal approaches
# Compares loop-based vs built-in/slicing-based methods

# ============================================================
# APPROACH 1: LOOP-BASED STRING REVERSAL
# ============================================================

def reverse_string_loop(text):
    """
    Reverses a string using a manual loop approach.
    Iterates through the string backwards and builds a new string.
```

```
    Args:
        text (str): The input string to be reversed

    Returns:
        str: The reversed string
    """
    reversed_text = ""
    for i in range(len(text) - 1, -1, -1):
        reversed_text += text[i]
    return reversed_text

def reverse_string_loop_alt(text):
    """
    Alternative loop-based approach using a for-each loop.
    Converts string to list, then iterates in reverse.

    Args:
        text (str): The input string to be reversed

    Returns:
        str: The reversed string
```

```
    """
    reversed_text = ""
    for char in reversed(text):
        reversed_text += char
    return reversed_text

#
==========================================================
==========================
# APPROACH 2: BUILT-IN / SLICING-BASED STRING REVERSAL
#
==========================================================
==========================

def reverse_string_slicing(text):
    """
    Reverses a string using Python's slicing technique.
    Most efficient and Pythonic approach.

    Args:
        text (str): The input string to be reversed

    Returns:
```

```
        str: The reversed string
    """
    return text[::-1]

def reverse_string_reversed_builtin(text):
    """
    Reverses a string using the built-in reversed() function.
    Returns an iterator, so needs to be joined.

    Args:
        text (str): The input string to be reversed

    Returns:
        str: The reversed string
    """
    return "".join(reversed(text))

# ============================================================
# MAIN PROGRAM - DEMONSTRATION AND COMPARISON
```

```
#
==========================================================
==========================

if __name__ == "__main__":
    # Test string
    test_string = input("Enter a string to reverse: ")

    print("\n" + "=" * 60)
    print("STRING REVERSAL APPROACHES COMPARISON")
    print("=" * 60)
    print(f"\nOriginal String: '{test_string}'")
    print("-" * 60)

    # LOOP-BASED APPROACHES
    print("\n1. LOOP-BASED APPROACHES:")
    print("-" * 60)

    # Approach 1a: Manual index-based loop
    result1 = reverse_string_loop(test_string)
    print(f"\n   Loop with Index (range):")
    print(f"    Code: for i in range(len(text) - 1, -1, -1): ...")
```

```
print(f"   Result: '{result1}'")

# Approach 1b: Using reversed() function with loop
result2 = reverse_string_loop_alt(test_string)
print(f"\n   Loop with reversed() function:")
print(f"   Code: for char in reversed(text): ...")
print(f"   Result: '{result2}'")

# BUILT-IN / SLICING APPROACHES
print("\n\n2. BUILT-IN / SLICING APPROACHES:")
print("-" * 60)

# Approach 2a: Slicing (Most Pythonic)
result3 = reverse_string_slicing(test_string)
print(f"\n   Slicing (MOST PYTHONIC):")
print(f"   Code: text[::-1]")
print(f"   Result: '{result3}'")

# Approach 2b: Using reversed() with join()
result4 = reverse_string_reversed_builtin(test_string)
print(f"\n   reversed() + join():")
print(f"   Code: ''.join(reversed(text))")
```

```
    print(f"   Result: '{result4}'")

    # VERIFICATION
    print("\n" + "=" * 60)
    print("VERIFICATION - All methods produce same result:")
    print("=" * 60)
    all_equal = result1 == result2 == result3 == result4
    print(f"All results equal: {all_equal} ✓" if all_equal else
f"Results differ: FAILED ✗")

    # PERFORMANCE COMPARISON SUMMARY
    print("\n" + "=" * 60)
    print("PERFORMANCE & READABILITY COMPARISON")
    print("=" * 60)
    print("""
    Method                  Speed      Readability   Recommenda
tion
    ____________________________________________________________
_____________________

    1. Loop with
Index          Slow       Medium       Learning/Detailed
    2. Loop with
reversed()      Medium     Medium       Educational
```

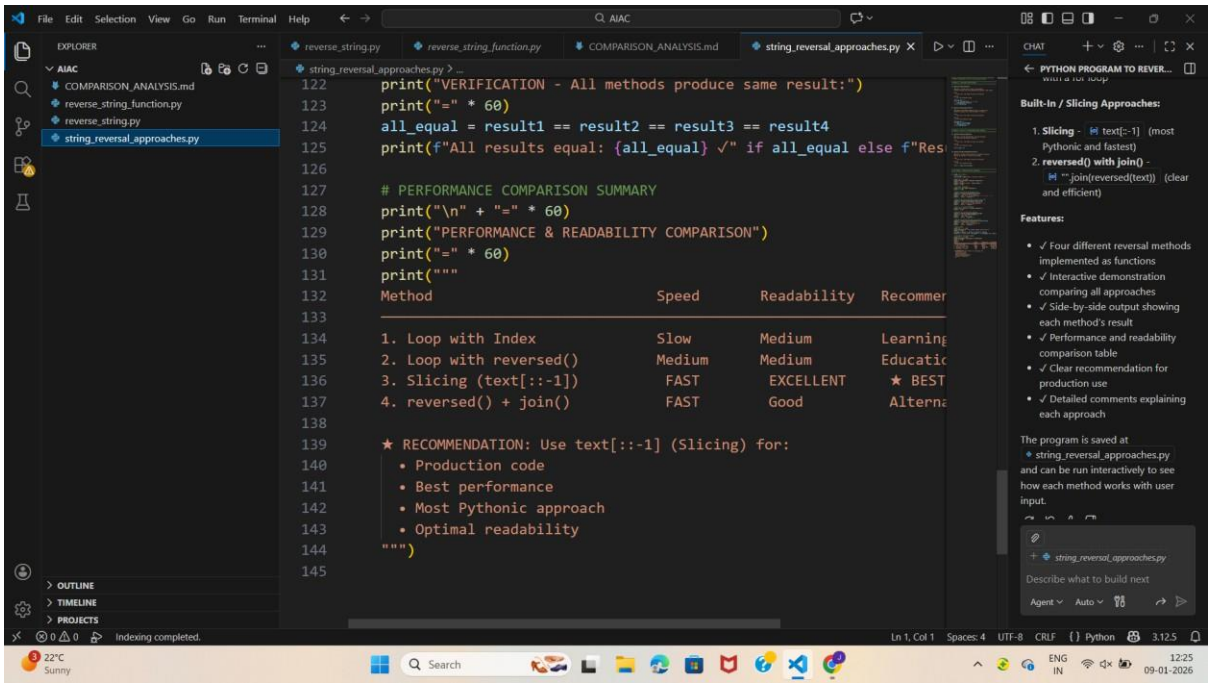
3. Slicing (text[::-1]) FAST EXCELLENT ★ BEST CHOICE

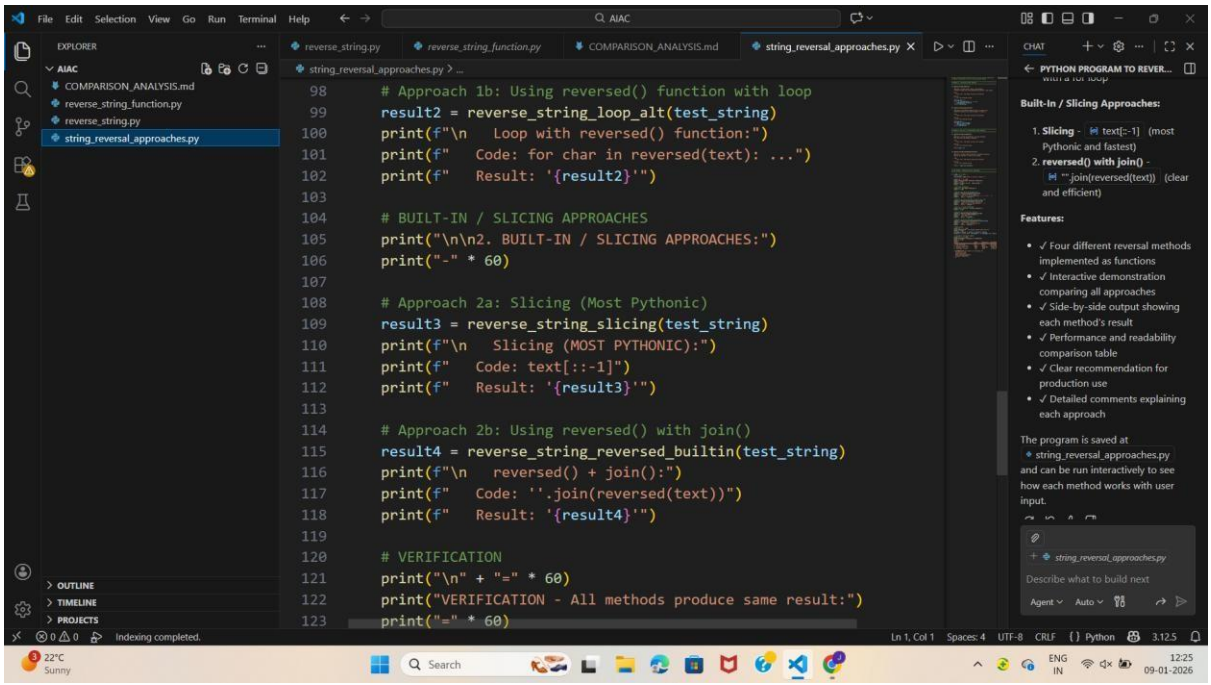
4. reversed() + join() FAST Good Alternative

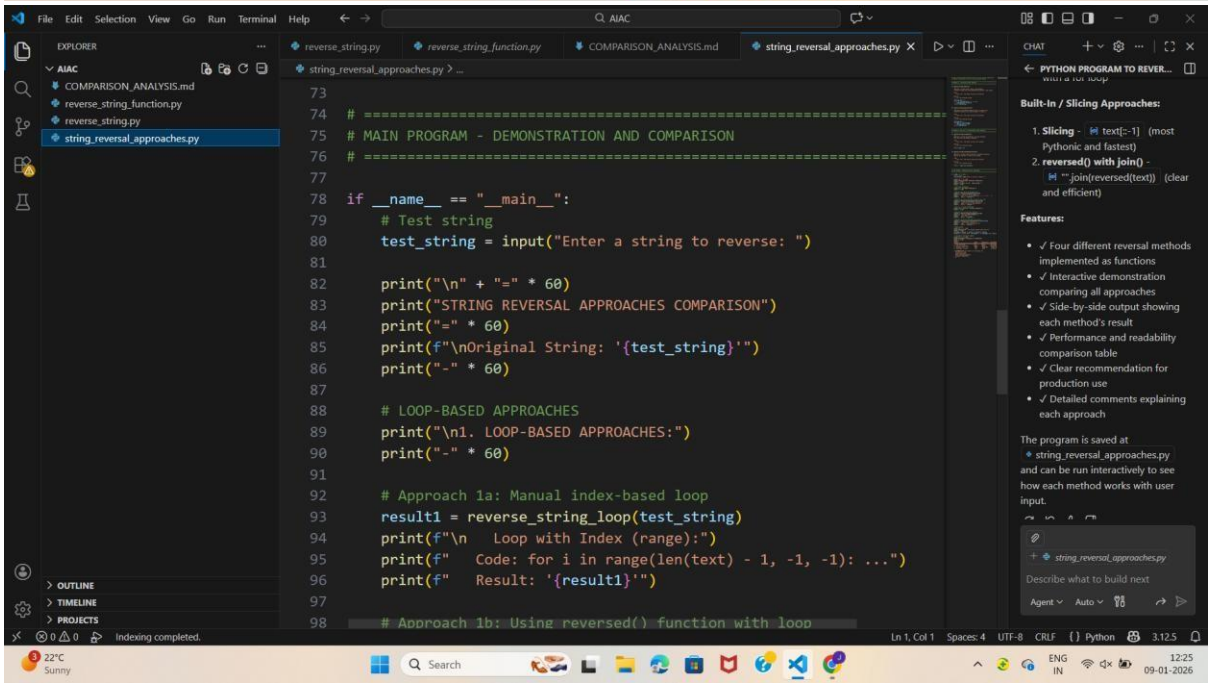
★ RECOMMENDATION: Use text[::-1] (Slicing) for:

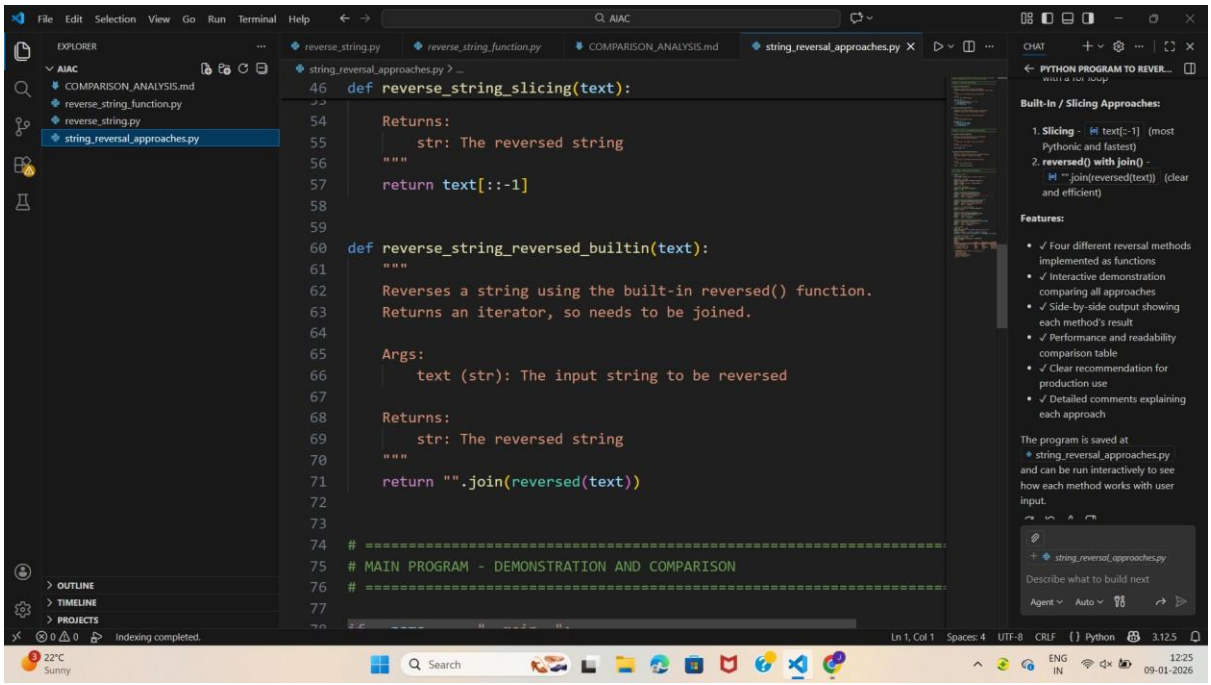
- Production code
- Best performance
- Most Pythonic approach
- Optimal readability

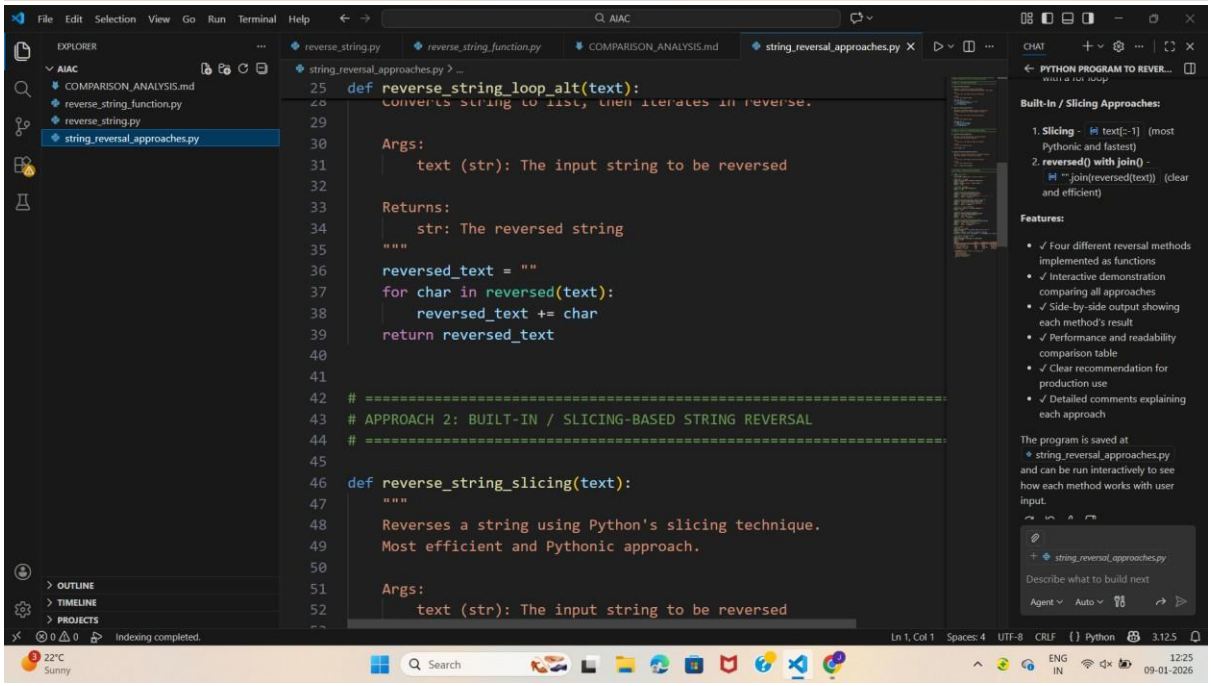
""")

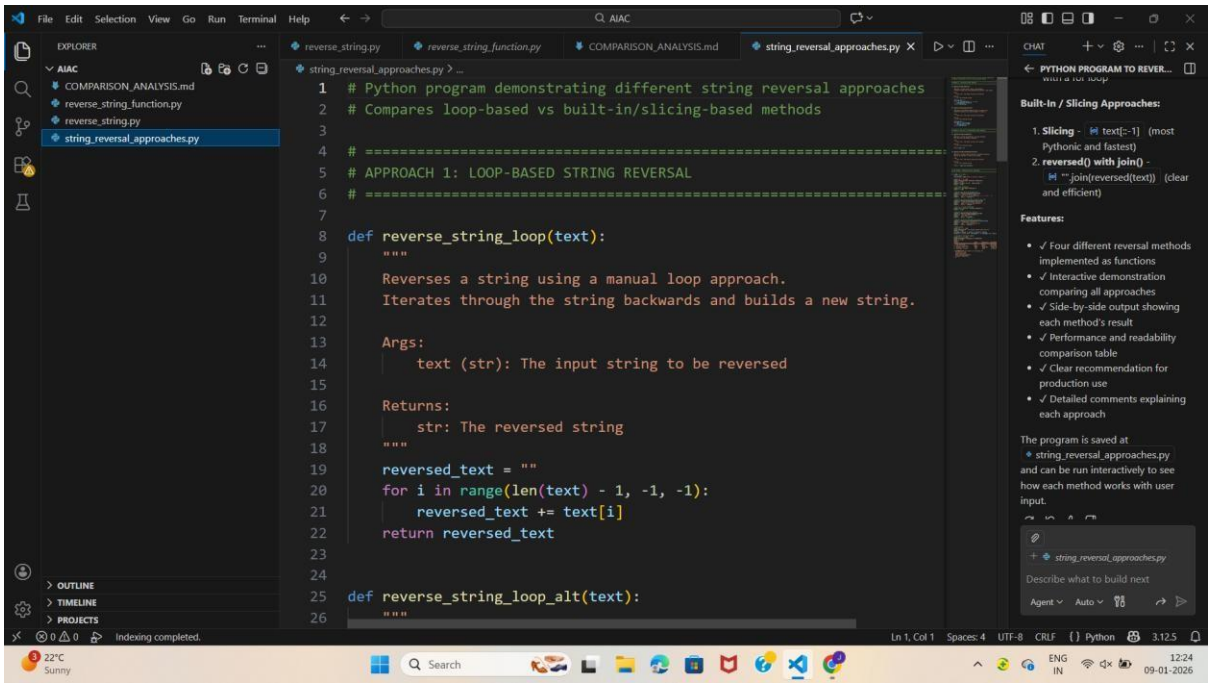

File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
string_reversal_approaches.py > ...
print("VERIFICATION - All methods produce same result:")
print("=" * 60)
all_equal = result1 == result2 == result3 == result4
print(f"All results equal: {all_equal} ✓" if all_equal else f"Res
# PERFORMANCE COMPARISON SUMMARY
print("\n" + "=" * 60)
print("PERFORMANCE & READABILITY COMPARISON")
print("=" * 60)
print("""
Method Speed Readability Recommer
1. Loop with Index Slow Medium Learning
2. Loop with reversed() Medium Medium Educatic
3. Slicing (text[::-1]) FAST EXCELLENT ★ BEST
4. reversed() + join() FAST Good Alterna
★ RECOMMENDATION: Use text[::-1] (Slicing) for:
• Production code
• Best performance
• Most Pythonic approach
• Optimal readability
""")
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.5
CHAT
PYTHON PROGRAM TO REVER...
Built-in / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
string_reversal_approaches.py
Describe what to build next
Agent Auto
22°C Sunny
Search
ENG IN
12:25 09-01-2026

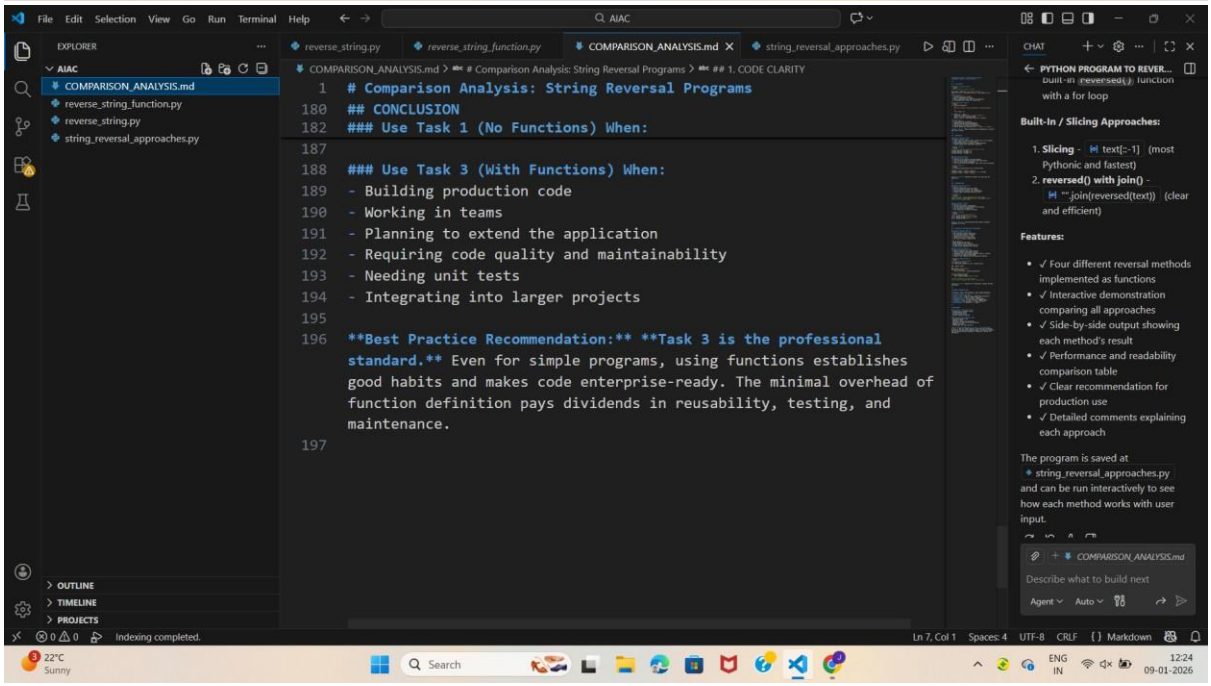

File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
string_reversal_approaches.py > ...
# Approach 1b: Using reversed() function with loop
result2 = reverse_string_loop_alt(test_string)
print(f"\n Loop with reversed() function:")
print(f" Code: for char in reversed(text): ...")
print(f" Result: '{result2}'")
# BUILT-IN / SLICING APPROACHES
print("\n\n2. BUILT-IN / SLICING APPROACHES:")
print("-" * 60)
# Approach 2a: Slicing (Most Pythonic)
result3 = reverse_string_slicing(test_string)
print(f"\n Slicing (MOST PYTHONIC):")
print(f" Code: text[::-1]")
print(f" Result: '{result3}'")
# Approach 2b: Using reversed() with join()
result4 = reverse_string_reversed_builtin(test_string)
print(f"\n reversed() + join():")
print(f" Code: ''.join(reversed(text))")
print(f" Result: '{result4}'")
# VERIFICATION
print("\n" + "=" * 60)
print("VERIFICATION - All methods produce same result:")
print("=" * 60)
CHAT
PYTHON PROGRAM TO REVER...
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
Describe what to build next
Agent Auto
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.12.5
22°C Sunny
Search
ENG IN
12:25 09-01-2026

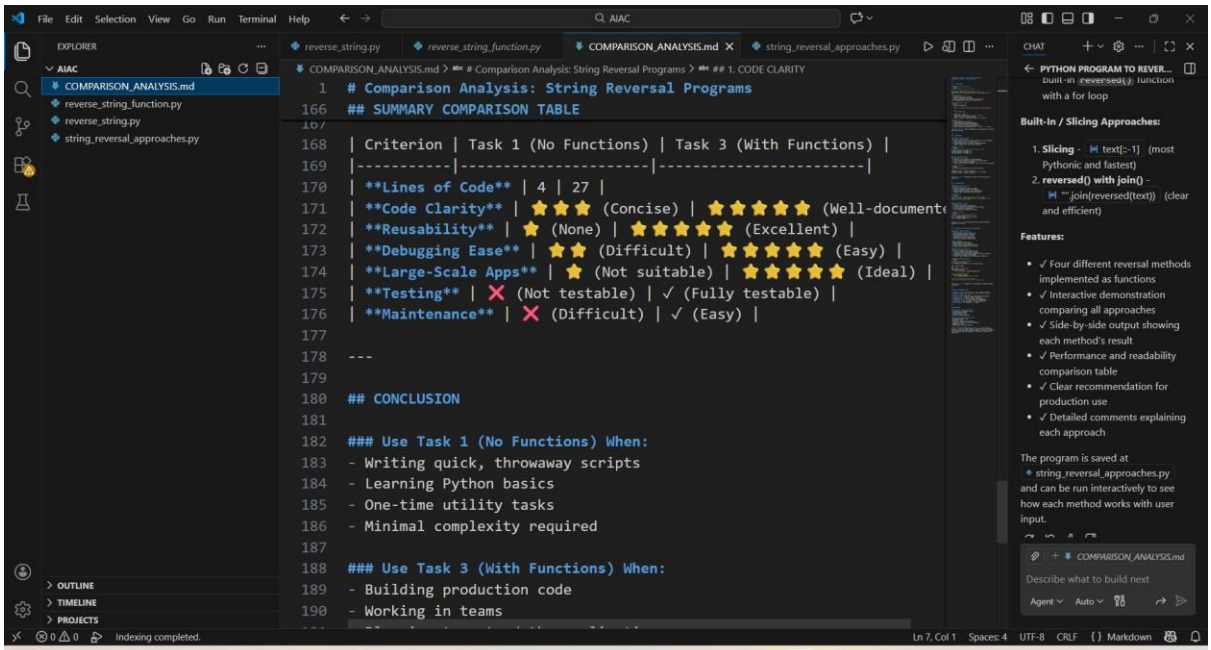

File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
string_reversal_approaches.py > ...
# ==
# MAIN PROGRAM - DEMONSTRATION AND COMPARISON
# ==
if __name__ == "__main__":
# Test string
test_string = input("Enter a string to reverse: ")
print("\n" + "=" * 60)
print("STRING REVERSAL APPROACHES COMPARISON")
print("=" * 60)
print(f"\nOriginal String: '{test_string}'")
print("-" * 60)
# LOOP-BASED APPROACHES
print("\n1. LOOP-BASED APPROACHES:")
print("-" * 60)
# Approach 1a: Manual index-based loop
result1 = reverse_string_loop(test_string)
print(f"\n Loop with Index (range):")
print(f" Code: for i in range(len(text) - 1, -1, -1): ...")
print(f" Result: '{result1}'")
# Approach 1b: Using reversed() function with loop
CHAT
PYTHON PROGRAM TO REVER...
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
Describe what to build next
Agent Auto
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.12.5
22°C Sunny
Search
ENG IN
12:25 09-01-2026

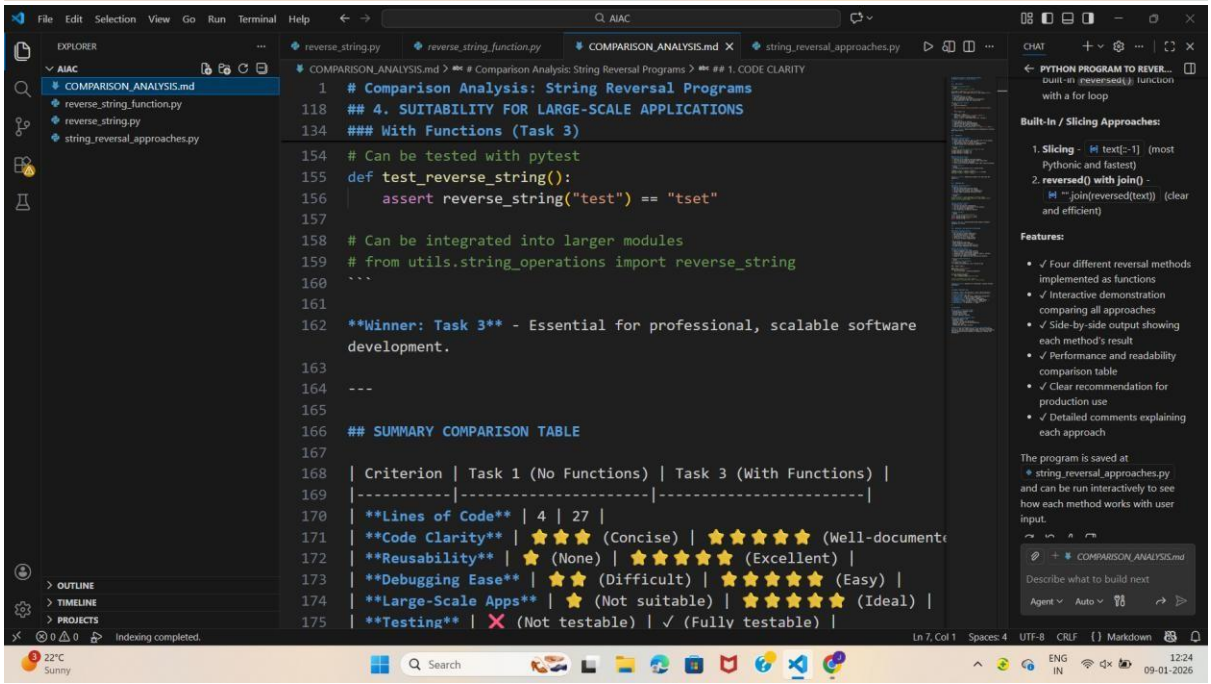

def reverse_string_slicing(text):
Returns:
str: The reversed string
return text[::-1]
def reverse_string_reversed_builtin(text):
Reverses a string using the built-in reversed() function.
Returns an iterator, so needs to be joined.
Args:
text (str): The input string to be reversed
Returns:
str: The reversed string
return "".join(reversed(text))
MAIN PROGRAM - DEMONSTRATION AND COMPARISON
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.

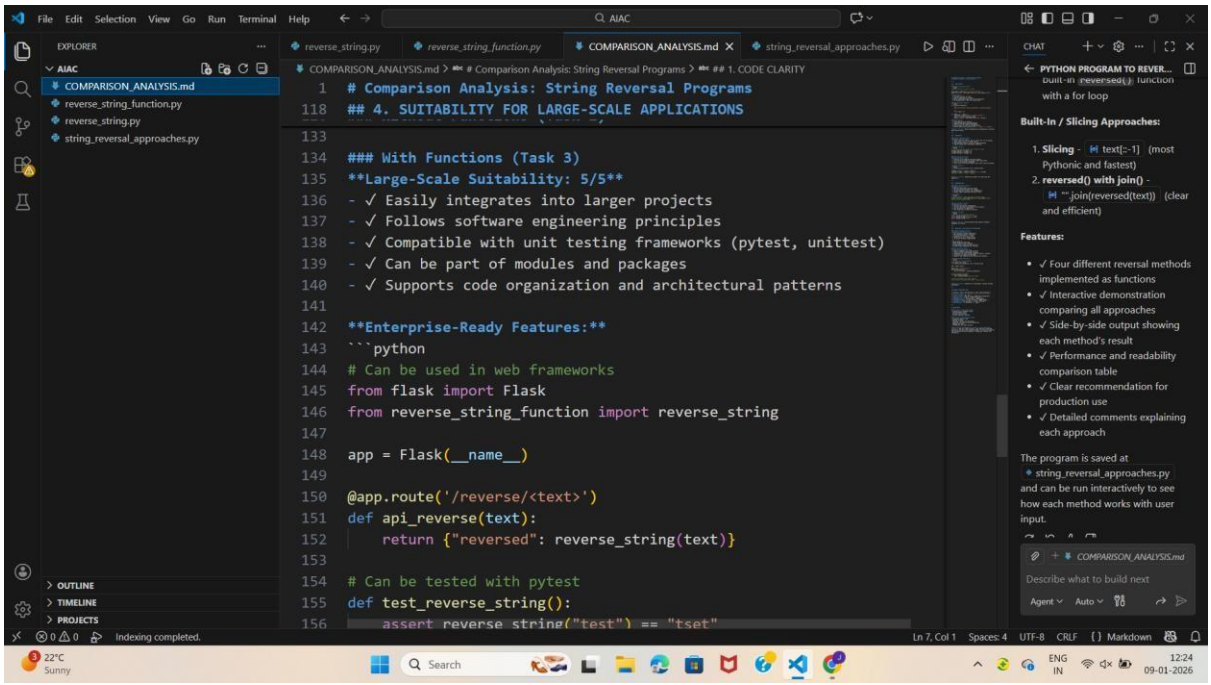

def reverse_string_loop_alt(text):
Args:
text (str): The input string to be reversed
Returns:
str: The reversed string
reversed_text = ""
for char in reversed(text):
reversed_text += char
return reversed_text
# APPROACH 2: BUILT-IN / SLICING-BASED STRING REVERSAL
def reverse_string_slicing(text):
Reverses a string using Python's slicing technique.
Most efficient and Pythonic approach.
Args:
text (str): The input string to be reversed
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.

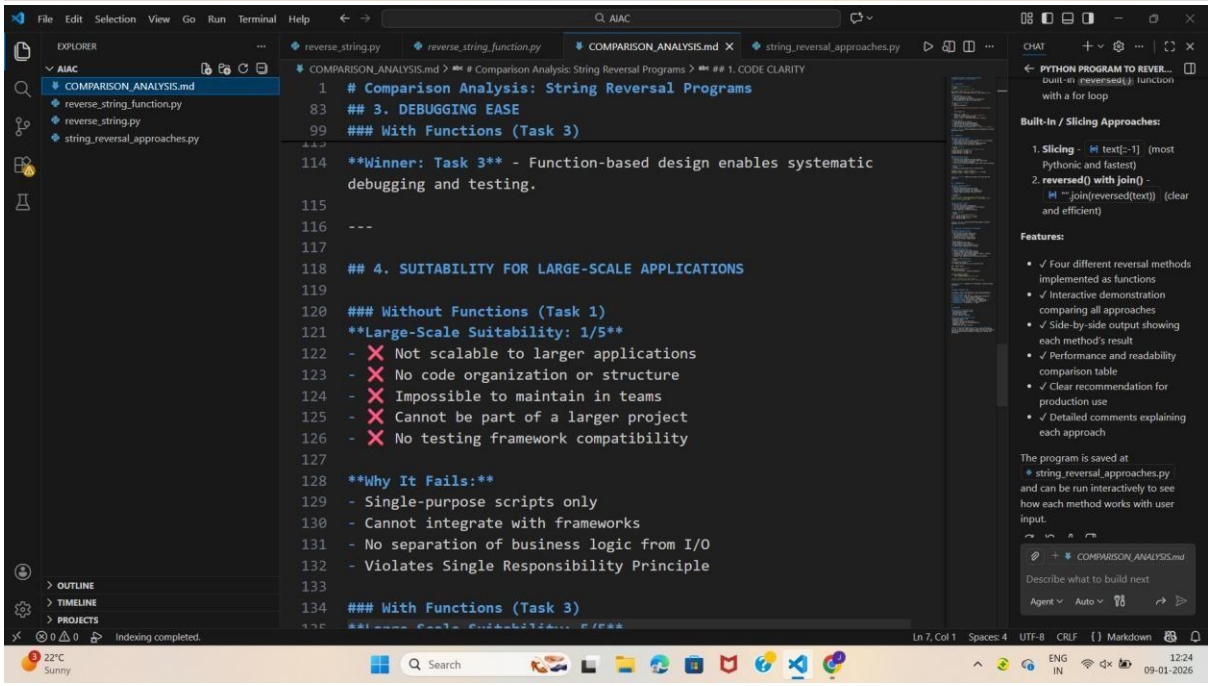

File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
string_reversal_approaches.py > ...
# Python program demonstrating different string reversal approaches
# Compares loop-based vs built-in/slicing-based methods
# ==
# APPROACH 1: LOOP-BASED STRING REVERSAL
# ==
def reverse_string_loop(text):
"""
Reverses a string using a manual loop approach.
Iterates through the string backwards and builds a new string.
Args:
text (str): The input string to be reversed
Returns:
str: The reversed string
"""
reversed_text = ""
for i in range(len(text) - 1, -1, -1):
reversed_text += text[i]
return reversed_text
def reverse_string_loop_alt(text):
"""
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.12.5
CHAT
PYTHON PROGRAM TO REVER...
Built-in / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
string_reversal_approaches.py
Describe what to build next
Agent Auto
22°C Sunny
Search
ENG IN
12:24 09-01-2026

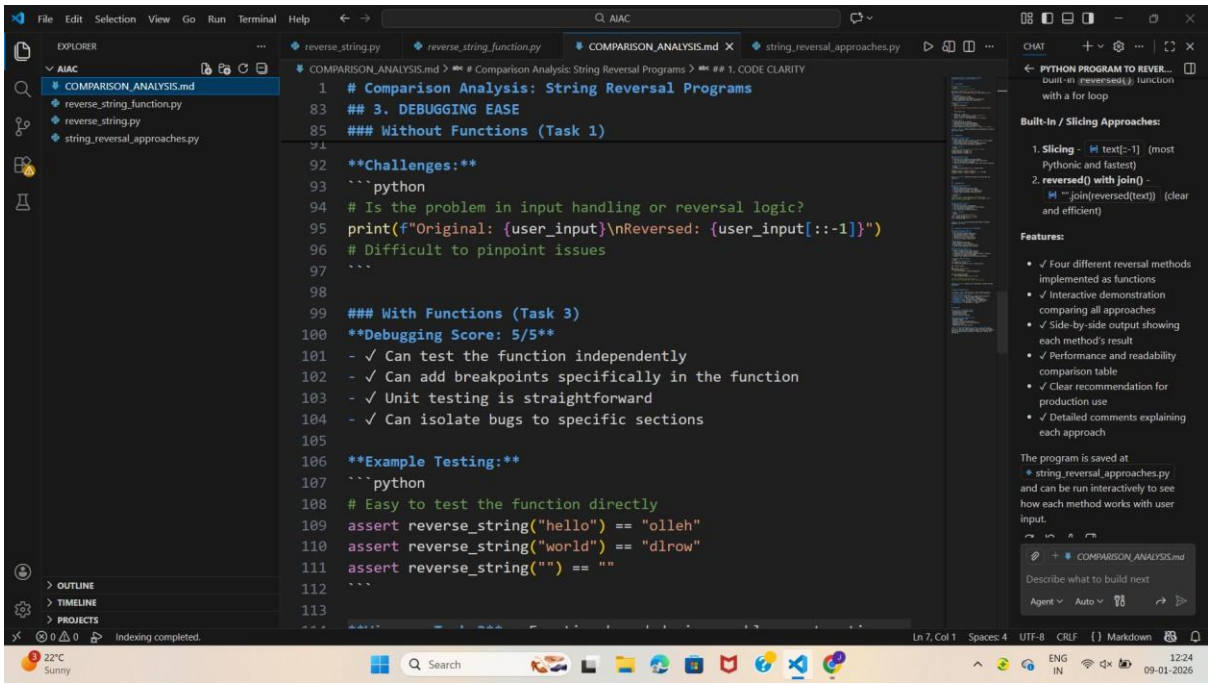

File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
COMPARISON_ANALYSIS.md > # Comparison Analysis: String Reversal Programs > ## 1. CODE CLARITY
Comparison Analysis: String Reversal Programs
CONCLUSION
Use Task 1 (No Functions) When:
Use Task 3 (With Functions) When:
- Building production code
- Working in teams
- Planning to extend the application
- Requiring code quality and maintainability
- Needing unit tests
- Integrating into larger projects
Best Practice Recommendation: **Task 3 is the professional standard.** Even for simple programs, using functions establishes good habits and makes code enterprise-ready. The minimal overhead of function definition pays dividends in reusability, testing, and maintenance.
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 7, Col 1 Spaces: 4 UTF-8 CRLF { } Markdown
CHAT
PYTHON PROGRAM TO REVER...
Built-in / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
COMPARISON_ANALYSIS.md
Describe what to build next
Agent Auto
22°C Sunny
Search
ENG IN
12:24 09-01-2026

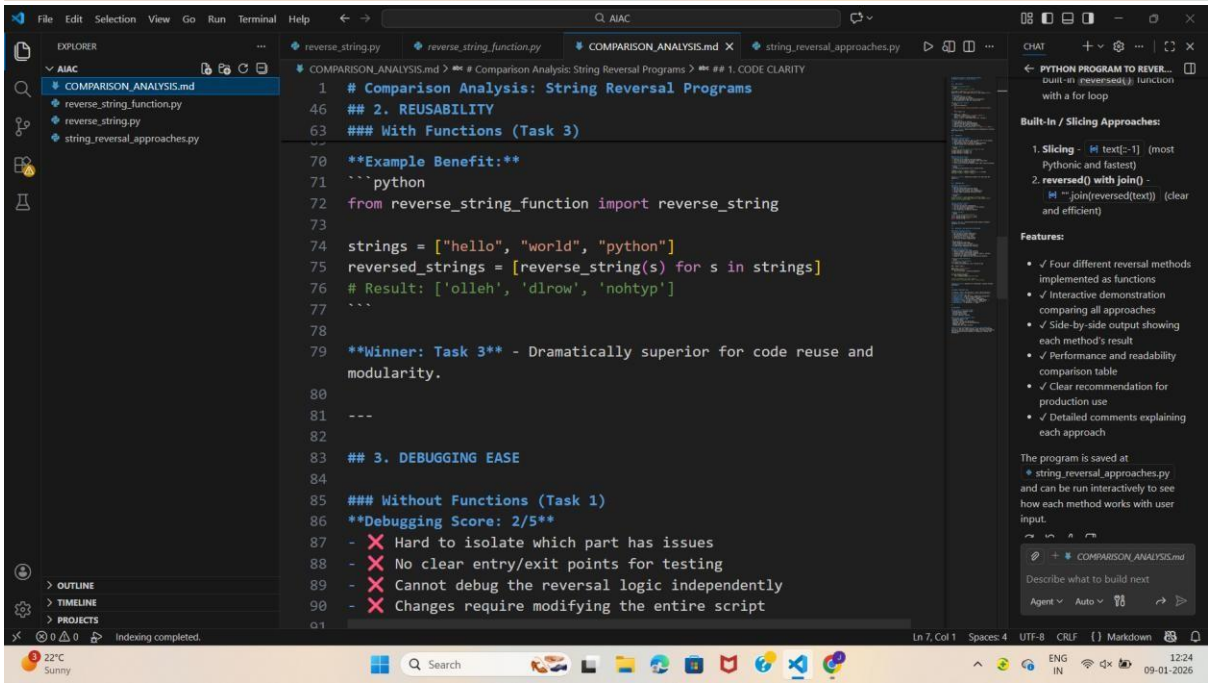

File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
COMPARISON_ANALYSIS.md > # Comparison Analysis: String Reversal Programs > ## 1. CODE CLARITY
1 # Comparison Analysis: String Reversal Programs
166 ## SUMMARY COMPARISON TABLE
168 | Criterion | Task 1 (No Functions) | Task 3 (With Functions) |
169 |----------|----------|----------|
170 | **Lines of Code** | 4 | 27 |
171 | **Code Clarity** | ⭐⭐⭐ (Concise) | ⭐⭐⭐⭐⭐ (Well-documente
172 | **Reusability** | ⭐ (None) | ⭐⭐⭐⭐⭐ (Excellent) |
173 | **Debugging Ease** | ⭐⭐ (Difficult) | ⭐⭐⭐⭐⭐ (Easy) |
174 | **Large-Scale Apps** | ⭐ (Not suitable) | ⭐⭐⭐⭐⭐ (Ideal) |
175 | **Testing** | ❌ (Not testable) | ✓ (Fully testable) |
176 | **Maintenance** | ❌ (Difficult) | ✓ (Easy) |
178 ---
180 ## CONCLUSION
182 ### Use Task 1 (No Functions) When:
183 - Writing quick, throwaway scripts
184 - Learning Python basics
185 - One-time utility tasks
186 - Minimal complexity required
188 ### Use Task 3 (With Functions) When:
189 - Building production code
190 - Working in teams
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 7, Col 1 Spaces: 4 UTF-8 CRLF { } Markdown
CHAT
PYTHON PROGRAM TO REVER...
built-in reversed() function with a for loop
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
COMPARISON_ANALYSIS.md
Describe what to build next
Agent Auto
22°C Sunny
Search
ENG IN
12:24 09-01-2026

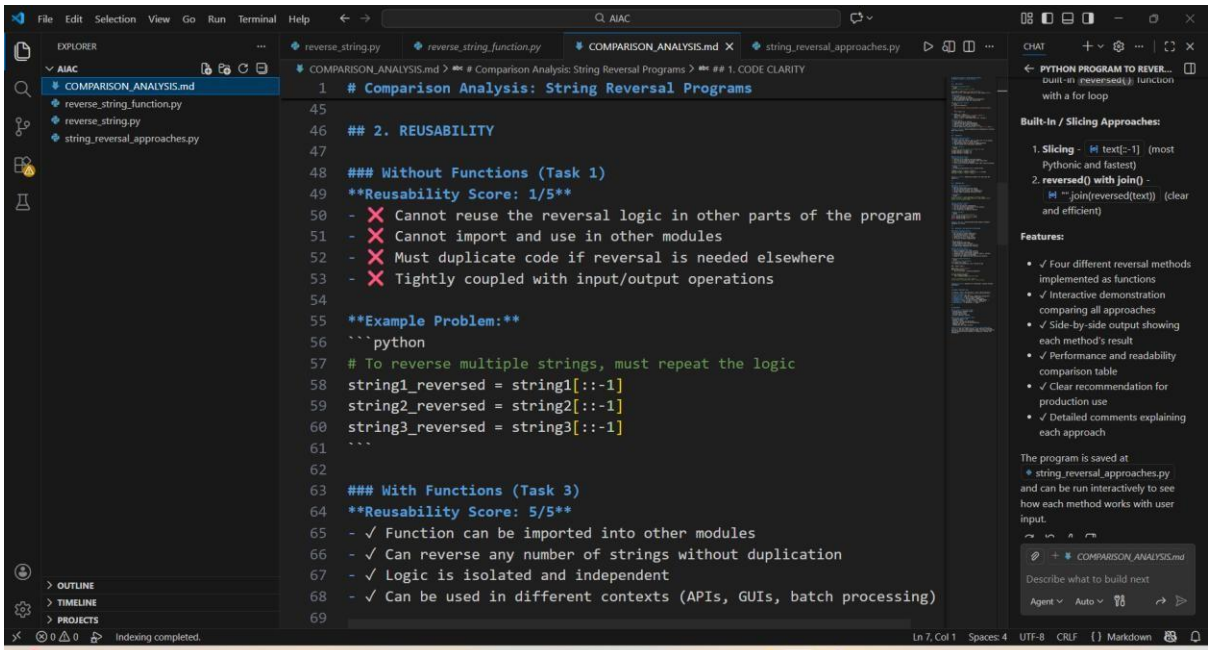

File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
COMPARISON_ANALYSIS.md > # Comparison Analysis: String Reversal Programs > ## 1. CODE CLARITY
1 # Comparison Analysis: String Reversal Programs
118 ## 4. SUITABILITY FOR LARGE-SCALE APPLICATIONS
134 ### With Functions (Task 3)
154 # Can be tested with pytest
155 def test_reverse_string():
156 assert reverse_string("test") == "tset"
158 # Can be integrated into larger modules
159 # from utils.string_operations import reverse_string
160 ```
162 **Winner: Task 3** - Essential for professional, scalable software development.
164 ---
166 ## SUMMARY COMPARISON TABLE
168 | Criterion | Task 1 (No Functions) | Task 3 (With Functions) |
169 |----------|----------|----------|
170 | **Lines of Code** | 4 | 27 |
171 | **Code Clarity** | ⭐⭐⭐ (Concise) | ⭐⭐⭐⭐⭐ (Well-documente
172 | **Reusability** | ⭐ (None) | ⭐⭐⭐⭐⭐ (Excellent) |
173 | **Debugging Ease** | ⭐⭐ (Difficult) | ⭐⭐⭐⭐⭐ (Easy) |
174 | **Large-Scale Apps** | ⭐ (Not suitable) | ⭐⭐⭐⭐⭐ (Ideal) |
175 | **Testing** | ❌ (Not testable) | ✓ (Fully testable) |
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 7, Col 1 Spaces: 4 UTF-8 CRLF { } Markdown
CHAT
PYTHON PROGRAM TO REVER...
built-in reversed() function with a for loop
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
COMPARISON_ANALYSIS.md
Describe what to build next
Agent Auto
22°C Sunny
Search
ENG IN
12:24 09-01-2026

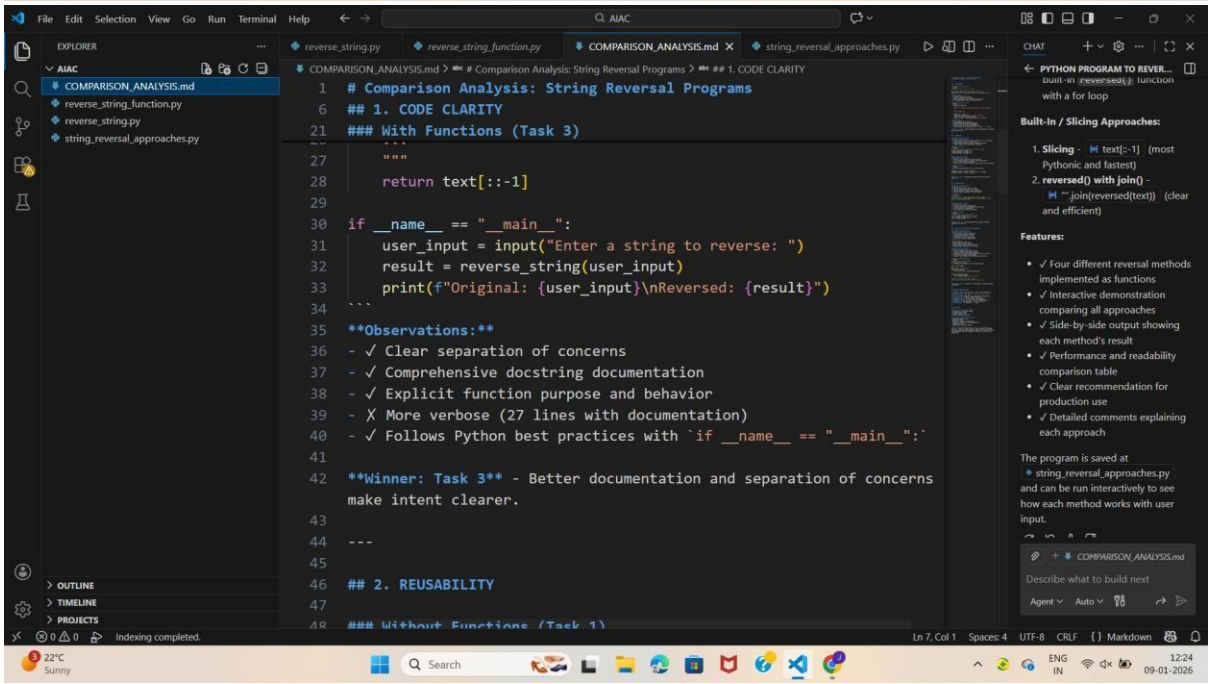

File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
COMPARISON_ANALYSIS.md > # Comparison Analysis: String Reversal Programs > ## 1. CODE CLARITY
1 # Comparison Analysis: String Reversal Programs
118 ## 4. SUITABILITY FOR LARGE-SCALE APPLICATIONS
133
134 ### With Functions (Task 3)
135 **Large-Scale Suitability: 5/5**
136 - ✓ Easily integrates into larger projects
137 - ✓ Follows software engineering principles
138 - ✓ Compatible with unit testing frameworks (pytest, unittest)
139 - ✓ Can be part of modules and packages
140 - ✓ Supports code organization and architectural patterns
141
142 **Enterprise-Ready Features:**
143 ```python
144 # Can be used in web frameworks
145 from flask import Flask
146 from reverse_string_function import reverse_string
147
148 app = Flask(__name__)
149
150 @app.route('/reverse/<text>')
151 def api_reverse(text):
152 return {"reversed": reverse_string(text)}
153
154 # Can be tested with pytest
155 def test_reverse_string():
156 assert reverse_string("test") == "tset"
CHAT
PYTHON PROGRAM TO REVER...
built-in reversed() function with a for loop
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
COMPARISON_ANALYSIS.md
Describe what to build next
Agent Auto
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 7, Col 1 Spaces: 4 UTF-8 CRLF { } Markdown
22°C Sunny
Search
ENG IN
12:24 09-01-2026

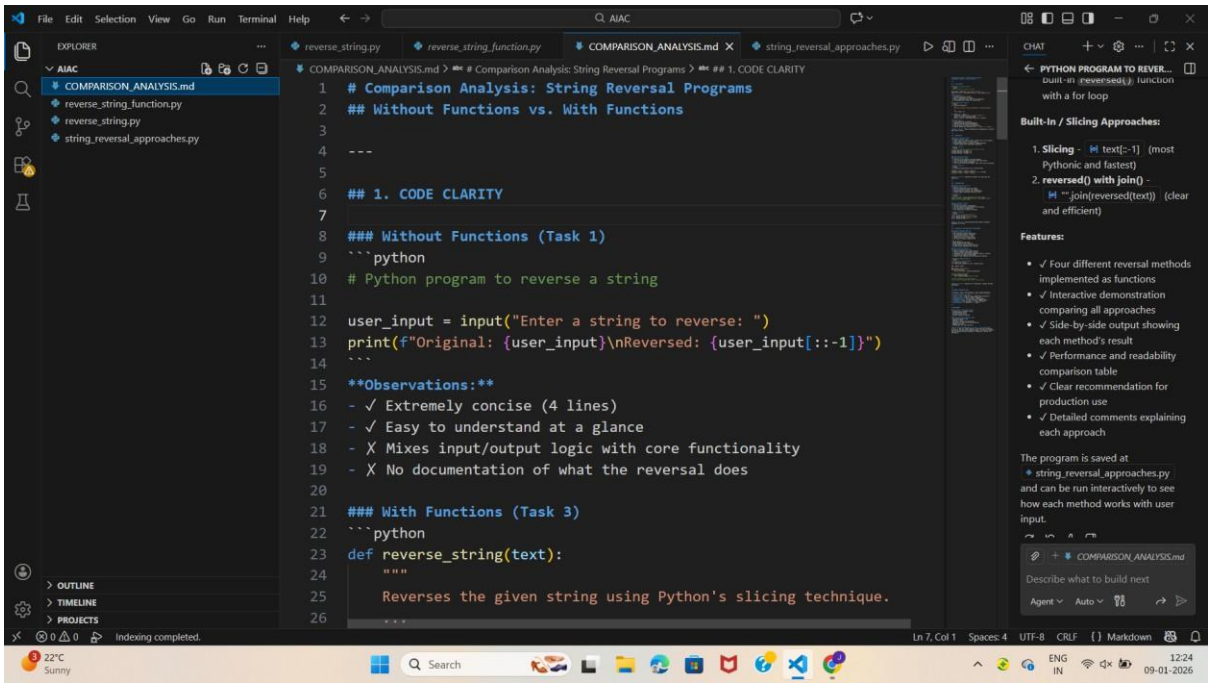

File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
COMPARISON_ANALYSIS.md > # Comparison Analysis: String Reversal Programs > ## 1. CODE CLARITY
1 # Comparison Analysis: String Reversal Programs
83 ## 3. DEBUGGING EASE
99 ### With Functions (Task 3)
114 **Winner: Task 3** - Function-based design enables systematic debugging and testing.
115
116 ---
117
118 ## 4. SUITABILITY FOR LARGE-SCALE APPLICATIONS
119
120 ### Without Functions (Task 1)
121 **Large-Scale Suitability: 1/5**
122 - ❌ Not scalable to larger applications
123 - ❌ No code organization or structure
124 - ❌ Impossible to maintain in teams
125 - ❌ Cannot be part of a larger project
126 - ❌ No testing framework compatibility
127
128 **Why It Fails:**
129 - Single-purpose scripts only
130 - Cannot integrate with frameworks
131 - No separation of business logic from I/O
132 - Violates Single Responsibility Principle
133
134 ### With Functions (Task 3)
CHAT
PYTHON PROGRAM TO REVER...
built-in reversed() function with a for loop
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
COMPARISON_ANALYSIS.md
Describe what to build next
Agent Auto
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 7, Col 1 Spaces: 4 UTF-8 CRLF { } Markdown
22°C Sunny
Search
ENG IN
12:24 09-01-2026

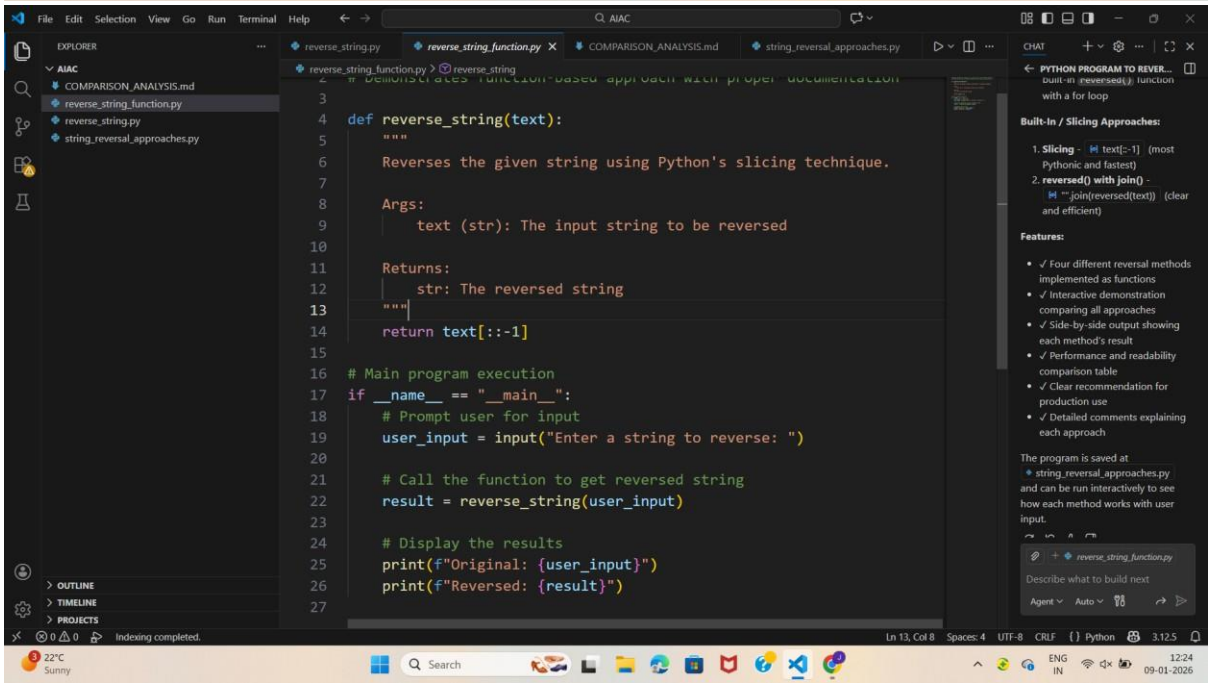

File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
COMPARISON_ANALYSIS.md > # Comparison Analysis: String Reversal Programs > ## 1. CODE CLARITY
1 # Comparison Analysis: String Reversal Programs
83 ## 3. DEBUGGING EASE
85 ### Without Functions (Task 1)
92 **Challenges:**
93 ```python
94 # Is the problem in input handling or reversal logic?
95 print(f"Original: {user_input}\nReversed: {user_input[::-1]}")
96 # Difficult to pinpoint issues
97 ```
99 ### With Functions (Task 3)
100 **Debugging Score: 5/5**
101 - ✓ Can test the function independently
102 - ✓ Can add breakpoints specifically in the function
103 - ✓ Unit testing is straightforward
104 - ✓ Can isolate bugs to specific sections
106 **Example Testing:**
107 ```python
108 # Easy to test the function directly
109 assert reverse_string("hello") == "olleh"
110 assert reverse_string("world") == "dlrow"
111 assert reverse_string("") == ""
112 ```
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 7, Col 1 Spaces: 4 UTF-8 CRLF { } Markdown
CHAT
PYTHON PROGRAM TO REVER...
built-in reversed() function with a for loop
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
COMPARISON_ANALYSIS.md
Describe what to build next
Agent Auto
22°C Sunny
Search
ENG IN
12:24 09-01-2026

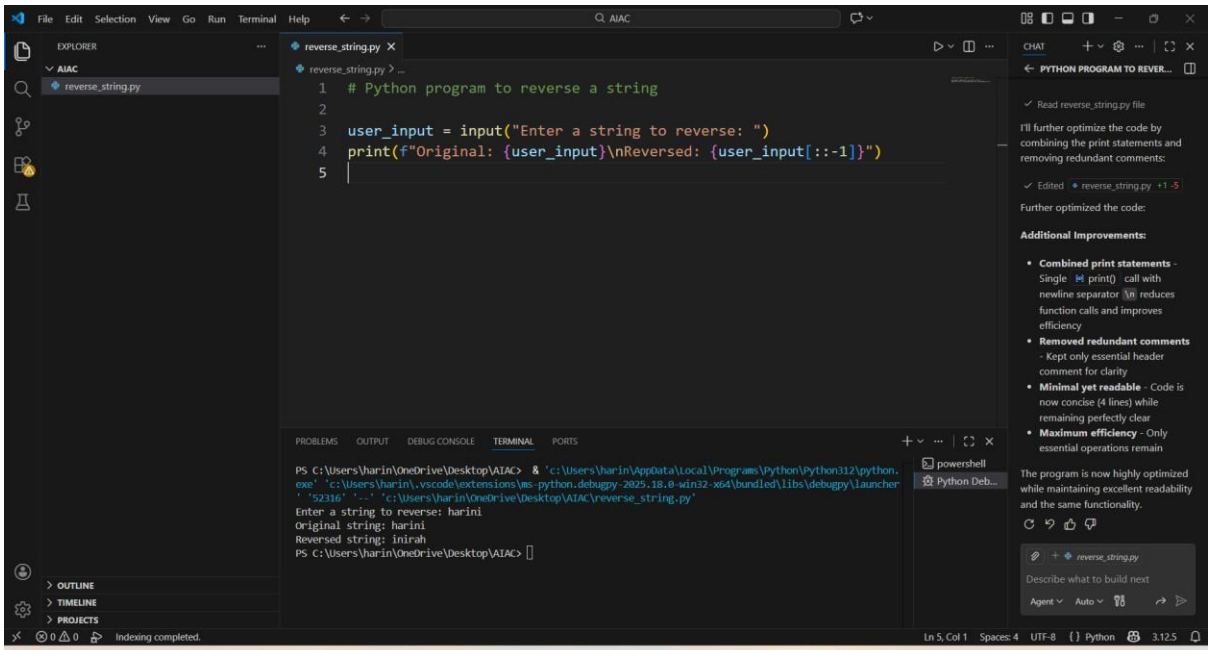

File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
COMPARISON_ANALYSIS.md > # Comparison Analysis: String Reversal Programs > ## 1. CODE CLARITY
1 # Comparison Analysis: String Reversal Programs
46 ## 2. REUSABILITY
63 ### With Functions (Task 3)
70 **Example Benefit:**
71 ```python
72 from reverse_string_function import reverse_string
74 strings = ["hello", "world", "python"]
75 reversed_strings = [reverse_string(s) for s in strings]
76 # Result: ['olleh', 'dlrow', 'nohtyp']
77 ```
79 **Winner: Task 3** - Dramatically superior for code reuse and modularity.
81 ---
83 ## 3. DEBUGGING EASE
85 ### Without Functions (Task 1)
86 **Debugging Score: 2/5**
87 - ❌ Hard to isolate which part has issues
88 - ❌ No clear entry/exit points for testing
89 - ❌ Cannot debug the reversal logic independently
90 - ❌ Changes require modifying the entire script
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 7, Col 1 Spaces: 4 UTF-8 CRLF { } Markdown
CHAT
PYTHON PROGRAM TO REVER...
built-in reversed() function with a for loop
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
COMPARISON_ANALYSIS.md
Describe what to build next
Agent Auto
22°C Sunny
Search
ENG IN
12:24 09-01-2026

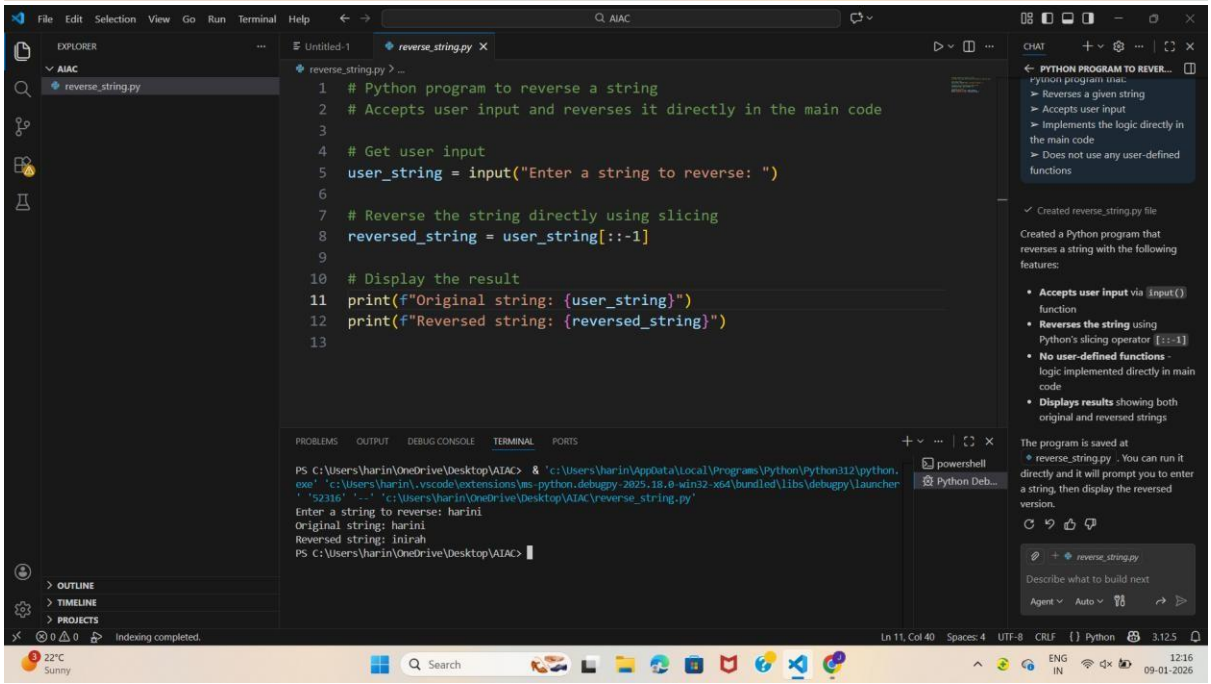

File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
COMPARISON_ANALYSIS.md > # Comparison Analysis: String Reversal Programs > ## 1. CODE CLARITY
1 # Comparison Analysis: String Reversal Programs
45
46 ## 2. REUSABILITY
47
48 ### Without Functions (Task 1)
49 **Reusability Score: 1/5**
50 - ❌ Cannot reuse the reversal logic in other parts of the program
51 - ❌ Cannot import and use in other modules
52 - ❌ Must duplicate code if reversal is needed elsewhere
53 - ❌ Tightly coupled with input/output operations
54
55 **Example Problem:**
56 ```python
57 # To reverse multiple strings, must repeat the logic
58 string1_reversed = string1[::-1]
59 string2_reversed = string2[::-1]
60 string3_reversed = string3[::-1]
61 ```
62
63 ### With Functions (Task 3)
64 **Reusability Score: 5/5**
65 - ✓ Function can be imported into other modules
66 - ✓ Can reverse any number of strings without duplication
67 - ✓ Logic is isolated and independent
68 - ✓ Can be used in different contexts (APIs, GUIs, batch processing)
69
CHAT
PYTHON PROGRAM TO REVER...
built-in reversed() function with a for loop
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
COMPARISON_ANALYSIS.md
Describe what to build next
Agent Auto
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 7, Col 1 Spaces: 4 UTF-8 CRLF { } Markdown
22°C Sunny
Search
ENG IN
12:24 09-01-2026


File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
COMPARISON_ANALYSIS.md > # Comparison Analysis: String Reversal Programs > ## 1. CODE CLARITY
1 # Comparison Analysis: String Reversal Programs
6 ## 1. CODE CLARITY
21 ### With Functions (Task 3)
27 """
28 return text[::-1]
29
30 if __name__ == "__main__":
31 user_input = input("Enter a string to reverse: ")
32 result = reverse_string(user_input)
33 print(f"Original: {user_input}\nReversed: {result}")
34 ```
35 **Observations:**
36 - ✓ Clear separation of concerns
37 - ✓ Comprehensive docstring documentation
38 - ✓ Explicit function purpose and behavior
39 - ✗ More verbose (27 lines with documentation)
40 - ✓ Follows Python best practices with `if __name__ == "__main__":`
41
42 **Winner: Task 3** - Better documentation and separation of concerns make intent clearer.
43
44 ---
45
46 ## 2. REUSABILITY
47
48 ### Without Functions (Task 1)
CHAT
PYTHON PROGRAM TO REVER...
built-in reversed() function with a for loop
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
COMPARISON_ANALYSIS.md
Describe what to build next
Agent Auto
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 7, Col 1 Spaces: 4 UTF-8 CRLF { } Markdown
22°C Sunny
Search
ENG IN
12:24 09-01-2026


File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
COMPARISON_ANALYSIS.md > # Comparison Analysis: String Reversal Programs > ## 1. CODE CLARITY
Comparison Analysis: String Reversal Programs
Without Functions vs. With Functions
---
1. CODE CLARITY
Without Functions (Task 1)
```python
# Python program to reverse a string
user_input = input("Enter a string to reverse: ")
print(f"Original: {user_input}\nReversed: {user_input[::-1]}")
```
Observations:
- ✓ Extremely concise (4 lines)
- ✓ Easy to understand at a glance
- ✗ Mixes input/output logic with core functionality
- ✗ No documentation of what the reversal does
With Functions (Task 3)
```python
def reverse_string(text):
"""
Reverses the given string using Python's slicing technique.
CHAT
PYTHON PROGRAM TO REVER...
built-in reversed() function with a for loop
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
COMPARISON_ANALYSIS.md
Describe what to build next
Agent Auto
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 7, Col 1 Spaces: 4 UTF-8 CRLF { } Markdown
22°C Sunny
Search
ENG IN
12:24 09-01-2026


File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
COMPARISON_ANALYSIS.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py
reverse_string.py
reverse_string_function.py
COMPARISON_ANALYSIS.md
string_reversal_approaches.py
reverse_string_function.py > reverse_string
def reverse_string(text):
"""
Reverses the given string using Python's slicing technique.
Args:
text (str): The input string to be reversed
Returns:
str: The reversed string
"""
return text[::-1]
# Main program execution
if __name__ == "__main__":
# Prompt user for input
user_input = input("Enter a string to reverse: ")
# Call the function to get reversed string
result = reverse_string(user_input)
# Display the results
print(f"Original: {user_input}")
print(f"Reversed: {result}")
CHAT
PYTHON PROGRAM TO REVER...
built-in reversed() function with a for loop
Built-In / Slicing Approaches:
1. Slicing - text[::-1] (most Pythonic and fastest)
2. reversed() with join() - "".join(reversed(text)) (clear and efficient)
Features:
✓ Four different reversal methods implemented as functions
✓ Interactive demonstration comparing all approaches
✓ Side-by-side output showing each method's result
✓ Performance and readability comparison table
✓ Clear recommendation for production use
✓ Detailed comments explaining each approach
The program is saved at string_reversal_approaches.py and can be run interactively to see how each method works with user input.
reverse_string_function.py
Describe what to build next
Agent Auto
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
Ln 13, Col 8 Spaces: 4 UTF-8 CRLF { } Python 3.12.5
22°C Sunny
Search
ENG IN
12:24 09-01-2026
```


File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
reverse_string.py
reverse_string.py
# Python program to reverse a string
user_input = input("Enter a string to reverse: ")
print(f"Original: {user_input}\nReversed: {user_input[::-1]}")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\harin\OneDrive\Desktop\AIAC> & 'c:\Users\harin\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\harin\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '52316' '--' 'c:\Users\harin\OneDrive\Desktop\AIAC\reverse_string.py'
Enter a string to reverse: harini
Original string: harini
Reversed string: inirah
PS C:\Users\harin\OneDrive\Desktop\AIAC>
CHAT
PYTHON PROGRAM TO REVER...
Read reverse_string.py file
I'll further optimize the code by combining the print statements and removing redundant comments:
Edited reverse_string.py +1 -5
Further optimized the code:
Additional Improvements:
Combined print statements - Single print() call with newline separator \n reduces function calls and improves efficiency
Removed redundant comments - Kept only essential header comment for clarity
Minimal yet readable - Code is now concise (4 lines) while remaining perfectly clear
Maximum efficiency - Only essential operations remain
The program is now highly optimized while maintaining excellent readability and the same functionality.
Describe what to build next
Agent Auto
Indexing completed.
Ln 5, Col 1 Spaces: 4 UTF-8 Python 3.12.5
22°C Sunny
Search
ENG IN
12:17 09-01-2026


File Edit Selection View Go Run Terminal Help
AIAC
EXPLORER
AIAC
reverse_string.py
Untitled-1
reverse_string.py
# Python program to reverse a string
# Accepts user input and reverses it directly in the main code
# Get user input
user_string = input("Enter a string to reverse: ")
# Reverse the string directly using slicing
reversed_string = user_string[::-1]
# Display the result
print(f"Original string: {user_string}")
print(f"Reversed string: {reversed_string}")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\harin\OneDrive\Desktop\AIAC> & 'c:\Users\harin\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\harin\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '52316' '--' 'c:\Users\harin\OneDrive\Desktop\AIAC\reverse_string.py'
Enter a string to reverse: harini
Original string: harini
Reversed string: inirah
PS C:\Users\harin\OneDrive\Desktop\AIAC>
CHAT
PYTHON PROGRAM TO REVER...
Reverses a given string
Accepts user input
Implements the logic directly in the main code
Does not use any user-defined functions
Created reverse_string.py file
Created a Python program that reverses a string with the following features:
Accepts user input via input() function
Reverses the string using Python's slicing operator [::-1]
No user-defined functions - logic implemented directly in main code
Displays results showing both original and reversed strings
The program is saved at reverse_string.py . You can run it directly and it will prompt you to enter a string, then display the reversed version.
Describe what to build next
Agent Auto
Indexing completed.
Ln 11, Col 40 Spaces: 4 UTF-8 CRLF Python 3.12.5
22°C Sunny
Search
ENG IN
12:16 09-01-2026