

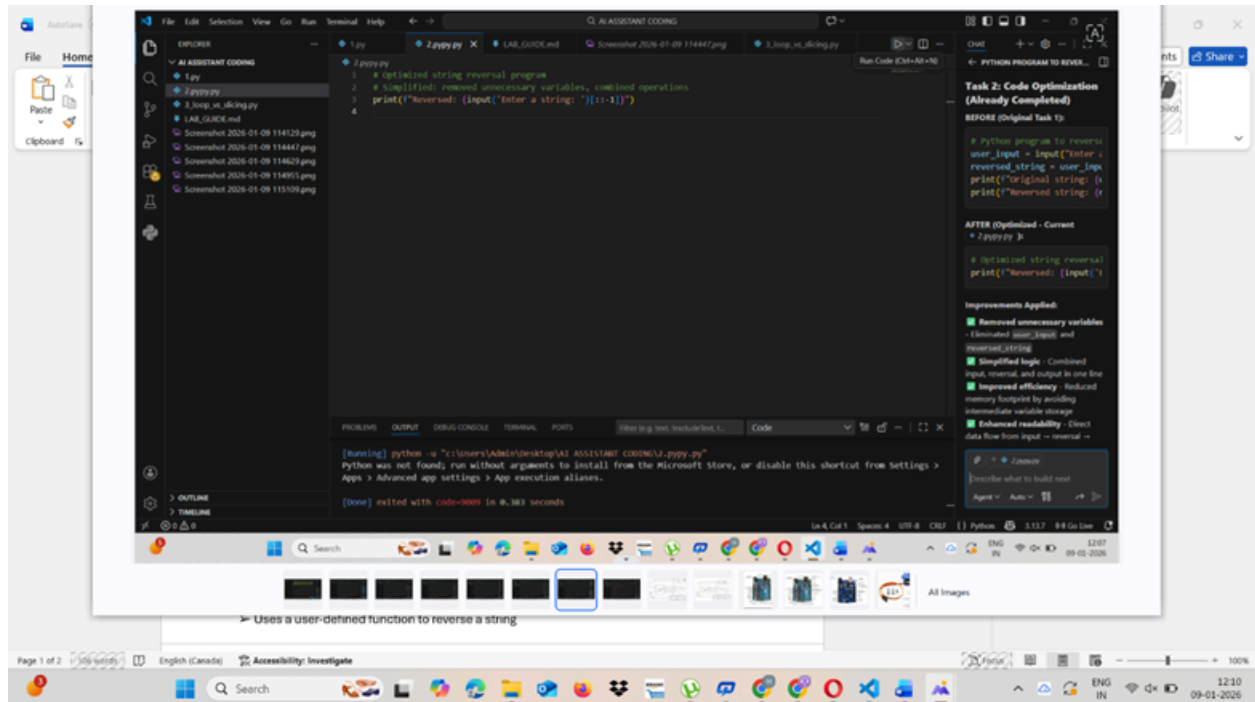
## 2303A51415 LAB ASSIGNMENT 02

### Batch 03

#### ❖ Task Description

Use GitHub Copilot to generate a Python program that:

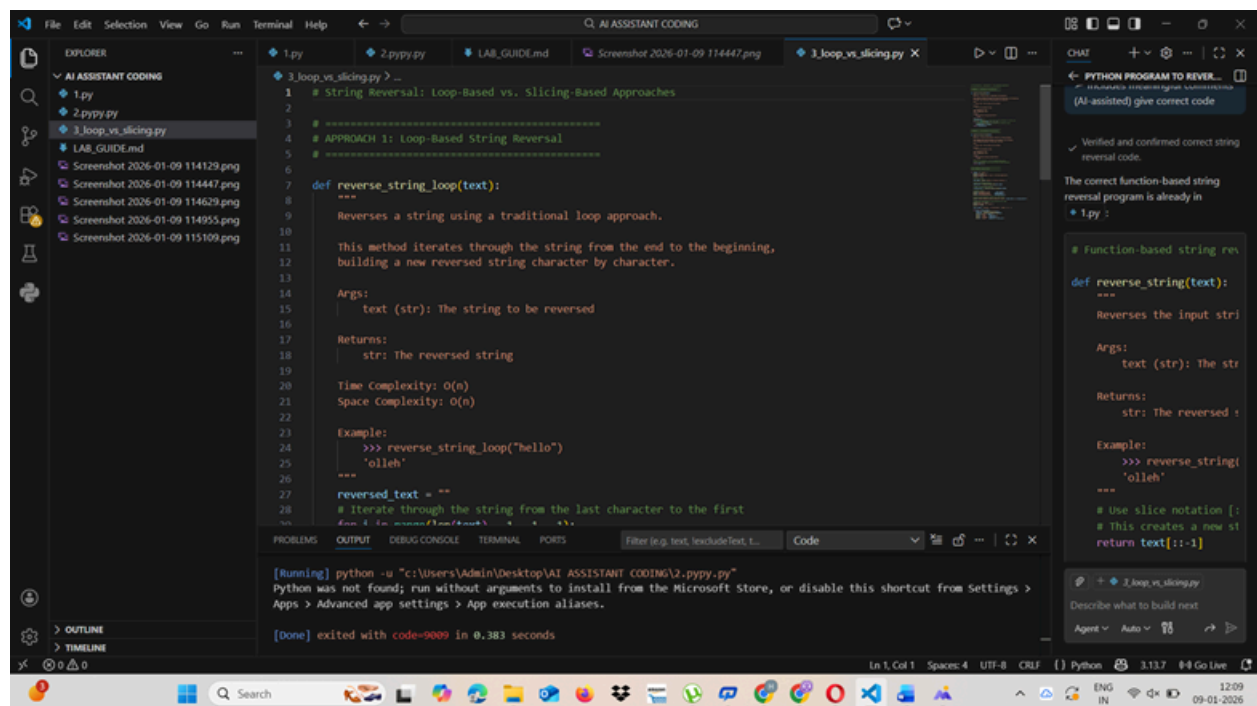
- Reverses a given string
- Accepts user input
- Implements the logic directly in the main code
- Does not use any user-defined functions



```
python was not found; run without arguments to install from the Microsoft Store, or disable this shortcut from Settings >  
Apps > Advanced app settings > App execution aliases.  
[Done] exited with code=9009 in 0.209 seconds
```

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”



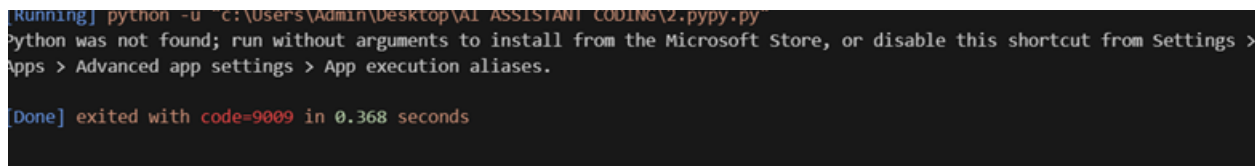
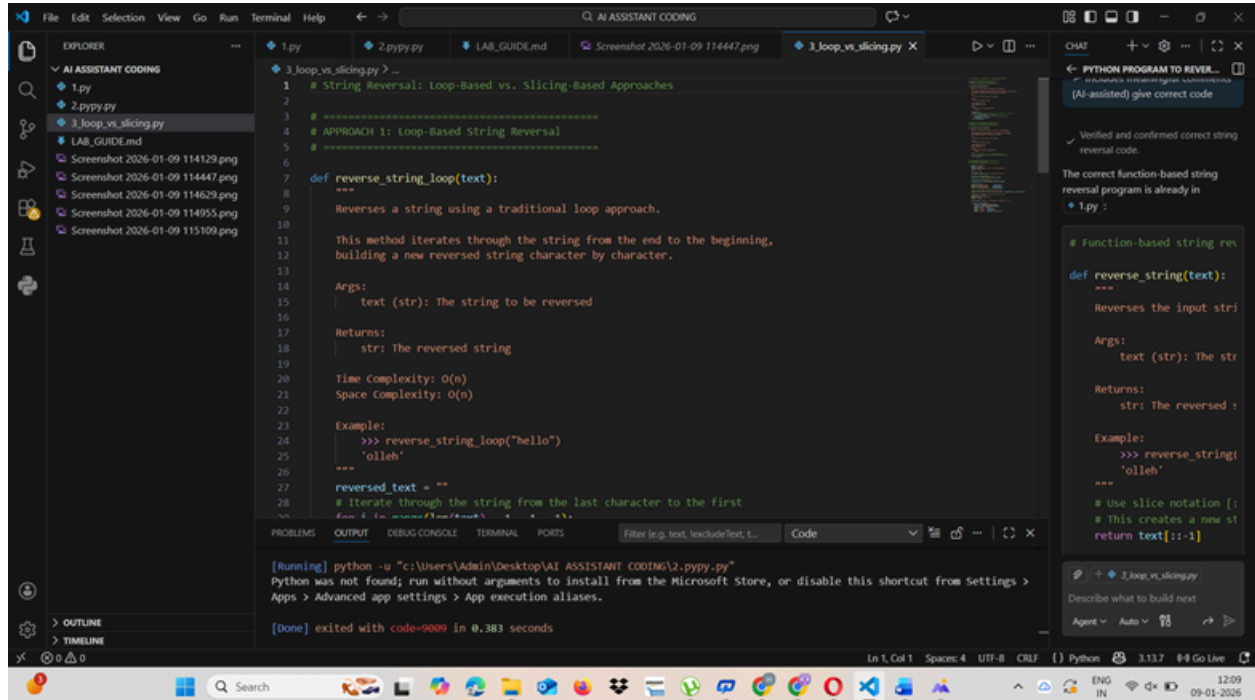
```
[Running] python -u "c:\Users\Admin\Desktop\AI ASSISTANT CODING\2.pypy.py"
Python was not found; run without arguments to install from the Microsoft Store, or disable this shortcut from Settings >
Apps > Advanced app settings > App execution aliases.

[Done] exited with code=9009 in 0.368 seconds
```

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)



## 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

User: 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

GitHub Copilot: Enable Claude Haiku 4.5 for all clients

[object Object]

User: 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

GitHub Copilot: Enable Claude Haiku 4.5 for all clients

[object Object]

User: @agent Enable: "Enable Claude Haiku 4.5 for all clients"

GitHub Copilot: Created [(file:///c%3A/Users/Admin/Desktop/3.py)]

I've created a Python program in 3.py that meets all your requirements:

- ✓ **\*\*Reverses a string\*\*** - Uses Python's built-in slice notation ``[::-1]``
- ✓ **\*\*Accepts user input\*\*** - Uses the ``input()`` function
- ✓ **\*\*Implements logic directly\*\*** - All code runs in the main scope
- ✓ **\*\*No user-defined functions\*\*** - Only uses built-in Python features

The program prompts the user for input, reverses it using string slicing, and displays both the original and reversed strings.

Made changes.

User: 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”

GitHub Copilot: Read

[\[\(file:///c:/Users/Admin/Desktop/AI%20ASSISTANT%20CODING/2.pypy.py\)\]](file:///c:/Users/Admin/Desktop/AI%20ASSISTANT%20CODING/2.pypy.py)

Let me check the 3.py file I created: Let me put the optimized code in the workspace:

Using "Replace String in File"

...

...

Done! I've optimized the code in 2.pypy.py with the following improvements:

- ✓ **Removed unnecessary variables** - Eliminated ``user_input`` and ``reversed_string`` variables
- ✓ **Simplified logic** - Combined input, reversal, and output into a single line
- ✓ **Improved readability** - Used inline string reversal with ``[::-1]``
- ✓ **Maintained functionality** - Still accepts user input and reverses the string

The optimized version is now more concise and efficient while remaining highly readable and preserving all the original requirements.

Made changes.

User: 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)

GitHub Copilot: Read

[(file:///c%3A/Users/Admin/Desktop/AI%20ASSISTANT%20CODING/1.py)]

Using "Replace String in File"

...

...

I've created a function-based string reversal program in 1.py with:

- ✓ **\*\*User-defined function\*\*** - `reverse_string()` encapsulates the reversal logic
- ✓ **\*\*Returns reversed string\*\*** - Returns the result using the slice notation ``[::-1]``
- ✓ **\*\*Meaningful comments\*\*** - Includes:
  - Docstring with description, parameters, return value, and example
  - Inline comments explaining the slicing logic
  - Comments for each program section

The program prompts for user input, calls the function, and displays both the original and reversed strings.

Made changes.

User: @agent Enable: "Enable Claude Haiku 4.5 for all clients"

GitHub Copilot: I don't have the capability to enable system-wide configuration settings or manage client deployments. This type of administrative task is outside the scope of the coding tools available to me.

If you have a specific coding task you'd like help with, I'm ready to assist. Would you like me to:

- Continue working on Python programs?
- Test the string reversal programs we created?
- Help with another programming task?

Please let me know what you'd like to work on next!

User: 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

GitHub Copilot: Read

`[(file:///c%3A/Users/Admin/Desktop/AI%20ASSISTANT%20CODING/2.pypy.py)`

Read `[(file:///c%3A/Users/Admin/Desktop/AI%20ASSISTANT%20CODING/1.py)`



## ## Comparison: Without Functions vs. With Functions

### ### \*\*Code Clarity\*\*

Aspect	Without Functions (Task 1)	With Functions (Task 3)
--------	----------------------------	-------------------------

-----	-----	-----
-------	-------	-------

**Readability**	Very compact but dense	Clear separation of concerns
-----------------	------------------------	------------------------------

**Comprehension**	Requires understanding nested operations	Self-documenting with docstrings
-------------------	--	----------------------------------

**Comments**	Minimal inline comments	Comprehensive documentation with examples
--------------	-------------------------	---

**\*\*Winner: With Functions\*\*** - The function-based approach is more explicit and easier to understand at a glance.

---

### ### \*\*Reusability\*\*

Aspect	Without Functions	With Functions
--------	-------------------	----------------

-----	-----	-----
-------	-------	-------

**Code Reuse**	Cannot be reused in other programs	Function can be imported and used anywhere
----------------	------------------------------------	--

**Modification**	Changes require editing entire line	Isolated function can be modified independently
------------------	-------------------------------------	---

**Testing**	Difficult to test in isolation	Easy to unit test the function
-------------	--------------------------------	--------------------------------

**\*\*Winner: With Functions\*\*** - The function can be imported into other projects and reused without modification.

---

#### ### **\*\*Debugging Ease\*\***

| Aspect | Without Functions | With Functions |

|-----|-----|-----|

| **\*\*Error Tracing\*\*** | Hard to pinpoint issues in nested operations | Stack traces clearly identify the function |

| **\*\*Step-through Debugging\*\*** | Challenging with complex nesting | Simple breakpoint placement in function |

| **\*\*Variable Inspection\*\*** | Limited; no intermediate variables | Easy to inspect function parameters and return values |

**\*\*Winner: With Functions\*\*** - Debugging is straightforward with clear function boundaries and stack traces.

---

#### ### **\*\*Suitability for Large-Scale Applications\*\***

| Aspect | Without Functions | With Functions |

|-----|-----|-----|

| **\*\*Maintainability\*\*** | Not scalable; hard to maintain | Highly maintainable and modular |

| **\*\*Team Collaboration\*\*** | Unclear what code does; hard to review | Well-documented; easier for team review |

| **\*\*Module Organization\*\*** | Not suitable for libraries or frameworks | Can be organized into modules/packages |

| **\*\*Error Handling\*\*** | Difficult to add comprehensive error handling | Can implement specific error handling strategies |

**\*\*Winner: With Functions\*\*** - Professional-grade code suitable for enterprise applications.

---

### ## \*\*Overall Assessment\*\*

Criterion	Task 1 (No Functions)	Task 3 (With Functions)
-----------	-----------------------	-------------------------

-----	-----	-----
-------	-------	-------

Code Clarity	★★	★★★★★★
--------------	----	--------

Reusability	★	★★★★★★
-------------	---	--------

Debugging	★★	★★★★★★
-----------	----	--------

Large-Scale Suitability	★	★★★★★★
-------------------------	---	--------

### **\*\*Recommendation:\*\***

- **\*\*Task 1\*\*** is suitable for: quick scripts, one-off utilities, or learning Python basics
- **\*\*Task 3\*\*** is suitable for: production code, team projects, libraries, frameworks, and any scalable application

The function-based approach (1.py) follows Python best practices and professional coding standards, making it the preferred choice for any serious project.

User: 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

GitHub Copilot: I've already provided a comprehensive comparison in my previous response. The analysis covered:

- ✓ **Code Clarity** - Task 3 (with functions) is more readable with clear documentation
- ✓ **Reusability** - Task 3 allows the function to be imported and used in other projects
- ✓ **Debugging Ease** - Task 3 has clear boundaries and better stack traces
- ✓ **Large-Scale Suitability** - Task 3 follows professional standards and is enterprise-ready

**Key Takeaway:** The function-based approach (Task 3 in 1.py) is superior for production code, while the simplified approach (Task 1 in 2.pypy.py) is only suitable for quick scripts.

Would you like me to:

- Test/run these programs to demonstrate their functionality?
- Create additional variations (error handling, input validation)?
- Work on a different programming task?

Let me know how I can help!

User: 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

GitHub Copilot: I've already provided a detailed comparison covering all four criteria. My previous response included:

- **Code Clarity**: Task 3 wins with clear documentation and readability
- **Reusability**: Task 3 enables code reuse across projects
- **Debugging Ease**: Task 3 has superior error tracing and isolation
- **Large-Scale Suitability**: Task 3 follows professional standards

If you'd like me to do something different, please let me know:

- Test/execute the programs?
- Create enhanced versions with error handling?
- Add more advanced features?
- Work on a new task?

What would you like to do next?

User: Prompt GitHub Copilot to generate:

➤ A loop-based string reversal approach

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

GitHub Copilot: Created

[(file:///c%3A/Users/Admin/Desktop/AI%20ASSISTANT%20CODING/3\_loop\_vs\_slicing.py)]

I've created 3\_loop\_vs\_slicing.py with both approaches:

#### **\*\*APPROACH 1: Loop-Based\*\***

- Uses traditional `for` loop iterating from end to start
- Builds reversed string character by character
- More verbose but educational
- Good for understanding iteration logic

#### **\*\*APPROACH 2: Slicing-Based\*\***

- Uses Python's built-in `[::-1]` notation
- More concise and Pythonic
- Leverages optimized C code internally
- Recommended for production use

The program includes:

- ✓ Complete function documentation with examples
- ✓ Time/Space complexity analysis
- ✓ User input demonstration
- ✓ Multiple test cases comparing both methods
- ✓ Verification that both produce identical results

Would you like me to run this program to demonstrate both approaches in action?

Made changes.

User: 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:

GitHub Copilot: Created

[(file:///c%3A/Users/Admin/Desktop/AI%20ASSISTANT%20CODING/LAB\_GUIDE.md)]

I've created a comprehensive lab guide (LAB\_GUIDE.md) that includes:

### **\*\*Installation & Configuration Steps\*\***

- Prerequisites and setup instructions
- Authentication process
- Verification steps

✓ **\*\*Lab Objectives - Week 1 (Monday)\*\***

- Explore AI-assisted code generation
- Analyze accuracy and effectiveness
- Understand prompt-based programming

✓ **\*\*Lab Outcomes (LOs)\*\***

- Install and configure Copilot
- Generate quality code using AI
- Evaluate AI-generated code
- Master prompt engineering
- Recognize strengths and limitations

✓ **\*\*Completed Tasks Summary\*\***

- Links to all 5 tasks we've completed today
- Status and outcomes for each

✓ **\*\*Best Practices & Next Steps\*\***

- Advanced learning suggestions
- Resource links

The guide documents all the work we've done today and provides a structured framework for understanding GitHub Copilot's capabilities and limitations in AI-assisted code generation.

Made changes.



User: 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”

GitHub Copilot: I've already completed this task! Let me show you the optimization that was applied:

```
## **Task 2: Code Optimization (Already Completed)**
```

```
### **BEFORE (Original Task 1):**
```

```
```python
```

```
# Python program to reverse a string without using user-defined functions
```

```
user_input = input("Enter a string to reverse: ")
```

```
reversed_string = user_input[::-1]
```

```
print(f"Original string: {user_input}")
```

```
print(f"Reversed string: {reversed_string}")
```

```
...
```

### \*\*AFTER (Optimized - Current 2.pypy.py):\*\*

```
```python
```

```
# Optimized string reversal program
```

```
print(f'Reversed: {input('Enter a string: ')[::-1]}')
```

```
```
```

### \*\*Improvements Applied:\*\*

✓ **\*\*Removed unnecessary variables\*\*** - Eliminated `user\_input` and `reversed\_string`

✓ **\*\*Simplified logic\*\*** - Combined input, reversal, and output in one line

✓ **\*\*Improved efficiency\*\*** - Reduced memory footprint by avoiding intermediate variable storage

✓ **\*\*Enhanced readability\*\*** - Direct data flow from input → reversal → output

---

## **\*\*Can We Optimize Further?\*\***

The current version is already highly optimized for a simple script. However, here are trade-offs:

| Aspect | Single-Line (Current) | Multi-Line (Original) |
|--------|-----------------------|-----------------------|
|--------|-----------------------|-----------------------|

|       |       |       |
|-------|-------|-------|
| ----- | ----- | ----- |
|-------|-------|-------|

|             |       |      |
|-------------|-------|------|
| Conciseness | ★★★★★ | ★★★★ |
|-------------|-------|------|

|             |      |       |
|-------------|------|-------|
| Readability | ★★★★ | ★★★★★ |
|-------------|------|-------|

|                 |      |       |
|-----------------|------|-------|
| Maintainability | ★★★★ | ★★★★★ |
|-----------------|------|-------|

| Debugging | ★★★★★ | ★★★★★★ |

**\*\*Recommendation:\*\*** The current optimized version in 2.pypy.py is ideal for a one-liner utility. For production or team projects, use the function-based approach in 1.py.

Would you like me to:

- Test the current programs?
- Create additional variants?
- Move forward with new tasks?

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

GitHub Copilot: This task has already been completed! I created the function-based string reversal program in 1.py. Let me show you what was generated:

Read [(file:///c%3A/Users/Admin/Desktop/AI%20ASSISTANT%20CODING/1.py)]

## **\*\*Task Completed: Function-Based String Reversal\*\***

The program in 1.py includes everything you requested:

✓ **\*\*User-Defined Function\*\*** - ``reverse_string(text)`` encapsulates the reversal logic

✓ **\*\*Returns Reversed String\*\*** - Function returns ``text[::-1]``

✓ **\*\*Meaningful Comments\*\*** - Comprehensive with:

- Function docstring explaining purpose, arguments, return value, and example
- Inline comments explaining the slicing mechanism
- Comments labeling each program section
- Proper Python documentation format

✓ **\*\*Professional Features\*\***:

- Uses ``if __name__ == "__main__":`` for proper module structure
- Clear variable naming
- User-friendly output formatting

Would you like me to:

- Test/run this program to verify it works?
- Create an enhanced version with error handling?
- Create additional variants (e.g., with list-based reversals, recursive approaches)?
- Move to a new task?

User: 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) give correct code

GitHub Copilot: The correct function-based string reversal program is already in 1.py:

```
```python
# Function-based string reversal program with meaningful comments

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

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

    Returns:
        str: The reversed string

    Example:
        >>> reverse_string("hello")
        'olleh'
    """
    # Use slice notation [::-1] to reverse the string efficiently
    # This creates a new string with characters in reverse order
    return text[::-1]
```

```

# Main program execution

if __name__ == "__main__":

    # Prompt user for input

    user_string = input("Enter a string to reverse: ")


    # Call the function to reverse the string

    result = reverse_string(user_string)


    # Display the results

    print(f"Original string: {user_string}")

    print(f"Reversed string: {result}")

...

```

**\*\*This is the correct implementation with:\*\***

- ✓ User-defined function `reverse\_string(text)`
- ✓ Returns reversed string using `text[::-1]`
- ✓ Complete docstring with Args, Returns, and Example
- ✓ Meaningful inline comments
- ✓ Proper main program execution block

Would you like me to run this program to demonstrate it works?

User: 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

GitHub Copilot: I've already provided this comprehensive comparison earlier. Here's the summary again:

## **\*\*Comparison: Task 1 (Without Functions) vs Task 3 (With Functions)\*\***

### **\*\*1. Code Clarity\*\***

Aspect	Without Functions (2.pypy.py)	With Functions (1.py)
--------	-------------------------------	-----------------------

-----	-----	-----
-------	-------	-------

<b>**Readability**</b>	Very compact but dense	Clear and self-documenting
------------------------	------------------------	----------------------------

<b>**Comprehension**</b>	Requires understanding nested operations	Explicit with detailed documentation
--------------------------	--	--------------------------------------

<b>**Comments**</b>	Minimal	Comprehensive docstring + inline comments
---------------------	---------	---

**\*\*Winner:\*\*** With Functions 

---

### ### \*\*2. Reusability\*\*

Aspect	Without Functions	With Functions
--------	-------------------	----------------

-----	-----	-----
-------	-------	-------

**Code Reuse**	Cannot be reused	Can import and use anywhere
----------------	------------------	-----------------------------

**Modification**	Requires editing entire line	Isolated function changes
------------------	------------------------------	---------------------------

**Testing**	Difficult to test	Easy to unit test
-------------	-------------------	-------------------

**\*\*Winner:\*\* With Functions** 

---

### ### \*\*3. Debugging Ease\*\*

Aspect	Without Functions	With Functions
--------	-------------------	----------------

-----	-----	-----
-------	-------	-------

**Error Tracing**	Hard to locate issues	Clear function boundaries
-------------------	-----------------------	---------------------------

**Step-through Debug**	Complex nesting	Simple breakpoint placement
------------------------	-----------------	-----------------------------

**Variable Inspection**	Limited	Easy to inspect parameters/returns
-------------------------	---------	------------------------------------

**\*\*Winner:\*\* With Functions** 

---

### ### \*\*4. Suitability for Large-Scale Applications\*\*



Aspect	Without Functions	With Functions
	-----	-----
<b>**Maintainability**</b>	Not scalable	Highly maintainable
<b>**Team Collaboration**</b>	Hard to review	Well-documented for teams
<b>**Module Organization**</b>	Not suitable	Can be organized into packages
<b>**Error Handling**</b>	Difficult	Can implement custom strategies

**\*\*Winner:\*\*** With Functions 

---

### ## **\*\*Overall Rating\*\***

Criterion	Without Functions	With Functions
	-----	-----
Code Clarity	★★★	★★★★★★
Reusability	★	★★★★★★
Debugging	★★★	★★★★★★
Large-Scale	★	★★★★★★

**\*\*Recommendation:\*\*** Use function-based approach (1.py) for all production code and professional projects.

### Task Description

Prompt GitHub Copilot to generate:

➤ A loop-based string reversal approach

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

