Assignment - 1

Name: Gampa Rithika

Roll Number: 2303A51190

Batch - 03

AI Assisted Coding

09-01-2026

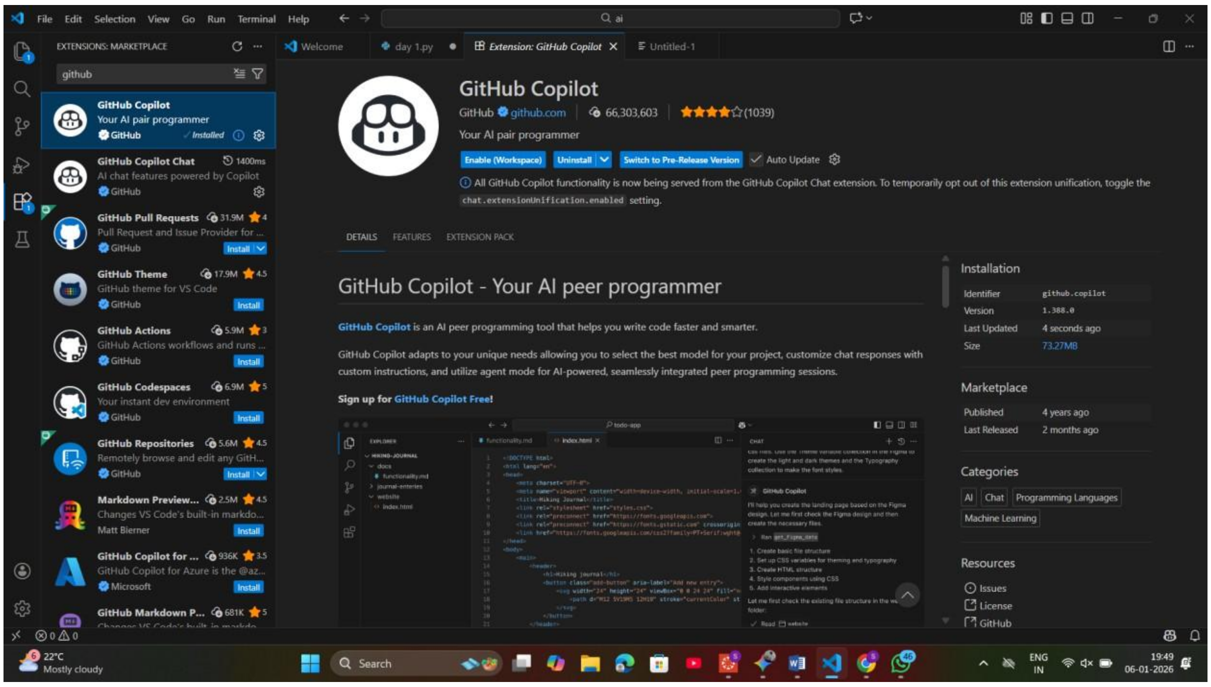
Task 0: Environment Setup:-

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.



GitHub Copilot — Your AI pair programmer — GitHub — Installed
GitHub Copilot Chat — 1400ms — AI chat features powered by Copilot — GitHub
GitHub Pull Requests — 31.9M — 4 — Pull Request and Issue Provider for ... — GitHub — Install
GitHub Theme — 17.9M — 4.5 — GitHub theme for VS Code — GitHub — Install
GitHub Actions — 5.9M — 3 — GitHub Actions workflows and runs ... — GitHub — Install
GitHub Codespaces — 6.9M — 5 — Your instant dev environment — GitHub — Install
GitHub Repositories — 5.6M — 4.5 — Remotely browse and edit any GitH... — GitHub — Install
Markdown Preview... — 2.5M — 4.5 — Changes VS Code's built-in markdo... — Matt Bierner — Install
GitHub Copilot for ... — 936K — 3.5 — GitHub Copilot for Azure is the @az... — Microsoft — Install
GitHub Markdown P... — 681K — 5

GitHub Copilot
GitHub github.com | 66,303,603 | (1039)
Your AI pair programmer

All GitHub Copilot functionality is now being served from the GitHub Copilot Chat extension. To temporarily opt out of this extension unification, toggle the `chat.extensionUnification.enabled` setting.

GitHub Copilot - Your AI peer programmer

GitHub Copilot is an AI peer programming tool that helps you write code faster and smarter.

GitHub Copilot adapts to your unique needs allowing you to select the best model for your project, customize chat responses with custom instructions, and utilize agent mode for AI-powered, seamlessly integrated peer programming sessions.

Sign up for GitHub Copilot Free!

Installation

Identifier	github.copilot
Version	1.388.0
Last Updated	4 seconds ago
Size	73.27MB

Marketplace

Published	4 years ago
Last Released	2 months ago

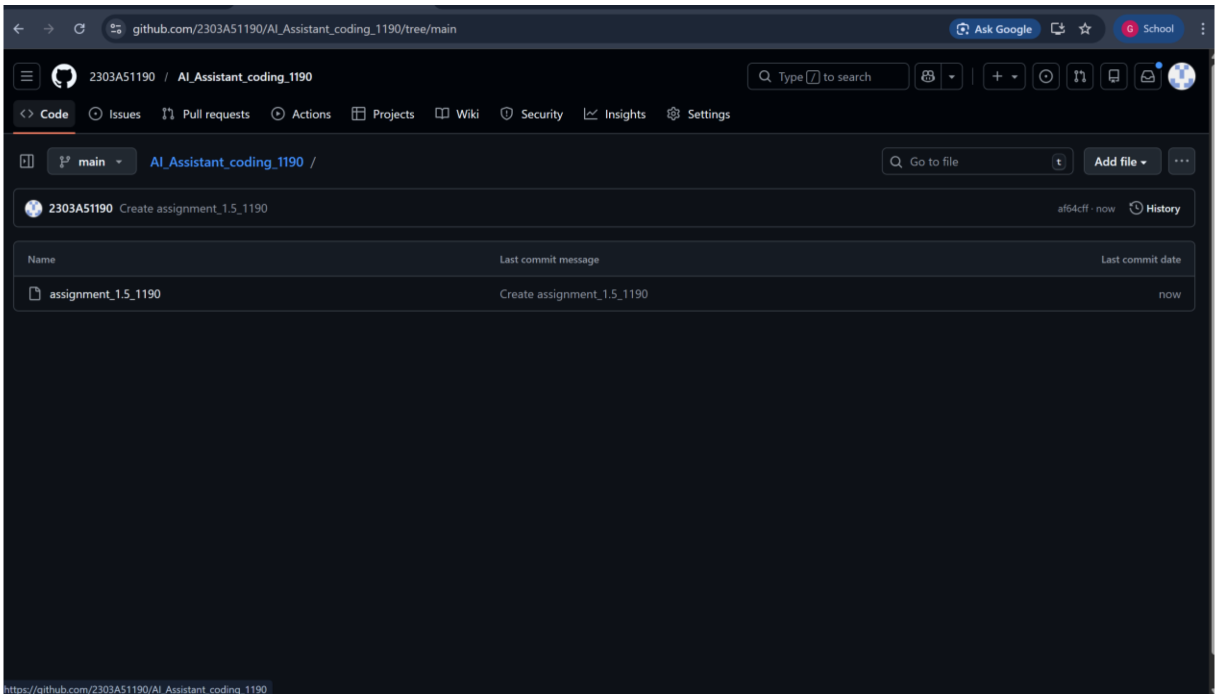
Categories

AI Chat Programming Languages Machine Learning

Resources

Issues
License
GitHub

22°C Mostly cloudy
19:49 06-01-2026


github.com/2303A51190/AI_Assistant_coding_1190/tree/main

2303A51190 / AI_Assistant_coding_1190

main AI_Assistant_coding_1190 /

2303A51190 Create assignment_1.5_1190 af64cff · now History

Name	Last commit message	Last commit date
assignment_1.5_1190	Create assignment_1.5_1190	now

https://github.com/2303A51190/AI_Assistant_coding_1190

Task 1: Non-Modular Logic (Factorial):-

: 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

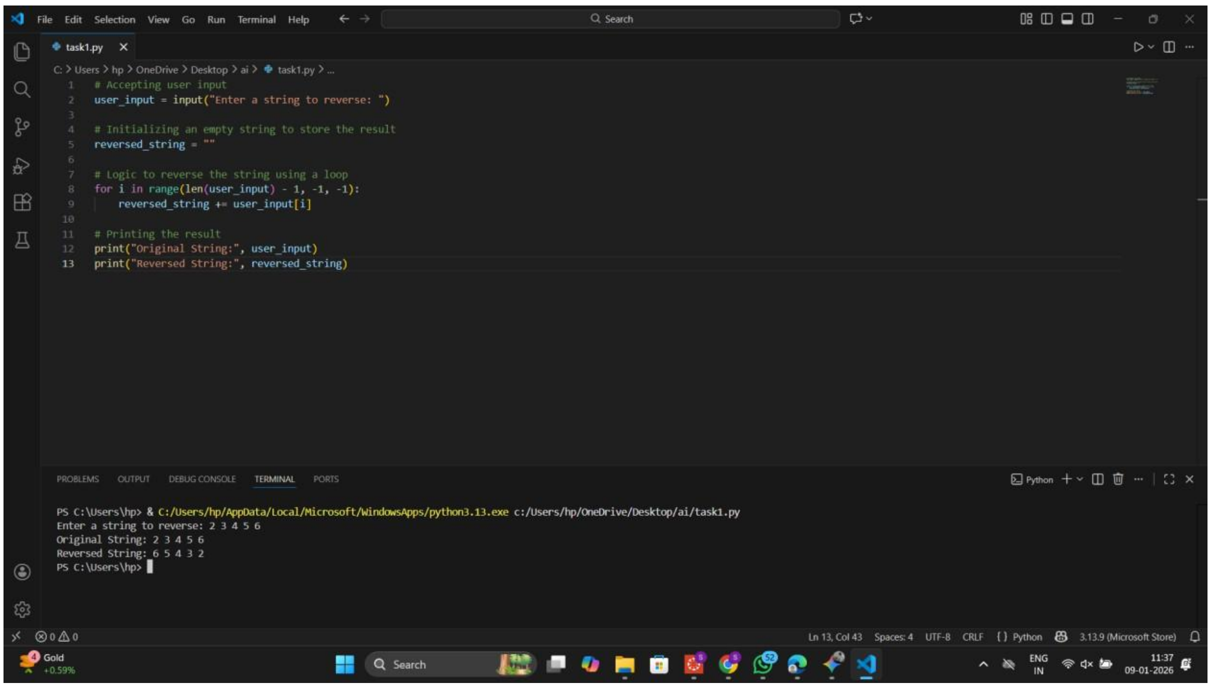
➢ Does not use any user-defined functions

❖ Expected Output

➢ Correct reversed string

➢ Screenshots showing Copilot-generated code suggestions

➢ Sample inputs and outputs



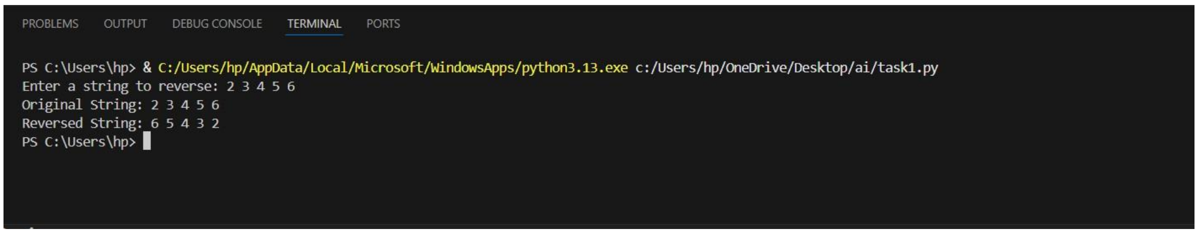
```python
# Accepting user input
user_input = input("Enter a string to reverse: ")

# Initializing an empty string to store the result
reversed_string = ""

# Logic to reverse the string using a loop
for i in range(len(user_input) - 1, -1, -1):
    reversed_string += user_input[i]

# Printing the result
print("Original String:", user_input)
print("Reversed String:", reversed_string)
```

```
PS C:\Users\hp> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/hp/OneDrive/Desktop/ai/task1.py
Enter a string to reverse: 2 3 4 5 6
Original String: 2 3 4 5 6
Reversed String: 6 5 4 3 2
PS C:\Users\hp> 
```



```
PS C:\Users\hp> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/hp/OneDrive/Desktop/ai/task1.py
Enter a string to reverse: 2 3 4 5 6
Original String: 2 3 4 5 6
Reversed String: 6 5 4 3 2
PS C:\Users\hp> 
```

Task 2: AI Code Optimization:-

Efficiency s 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:

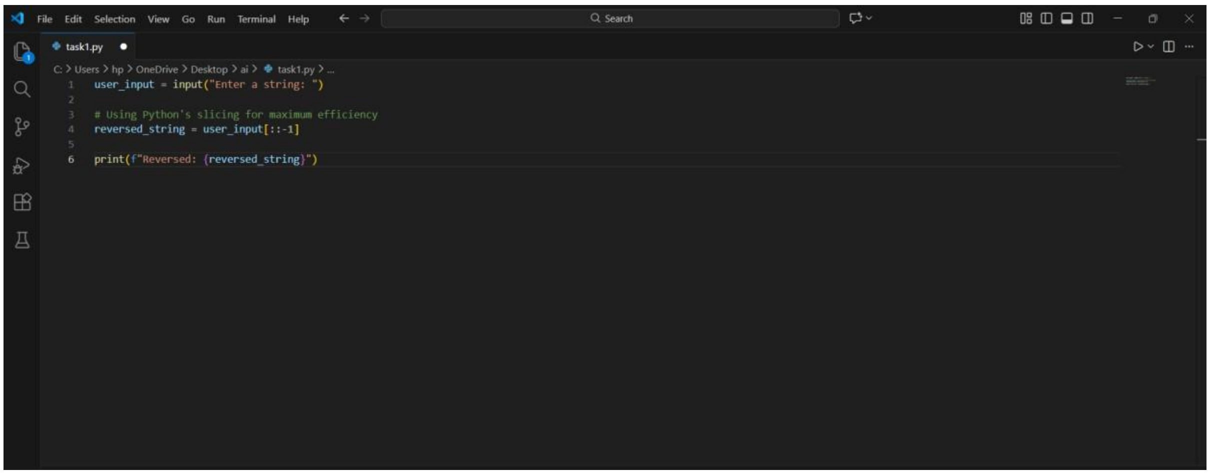
Prompt Copilot with phrases like

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

❖ Expected Output

➢ Original and optimized code versions

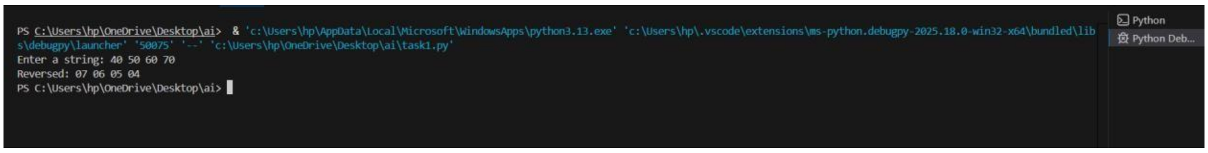
➢ Explanation of how the improvements reduce time complexity



```python
user_input = input("Enter a string: ")

# Using Python's slicing for maximum efficiency
reversed_string = user_input[::-1]

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



```
PS C:\Users\hp\OneDrive\Desktop\ai>  & 'c:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\hp\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\lib
s\debugpy\launcher' '50075' '--' 'c:\Users\hp\OneDrive\Desktop\ai\task1.py'
Enter a string: 40 50 60 70
Reversed: 07 06 05 04
PS C:\Users\hp\OneDrive\Desktop\ai> 
```

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

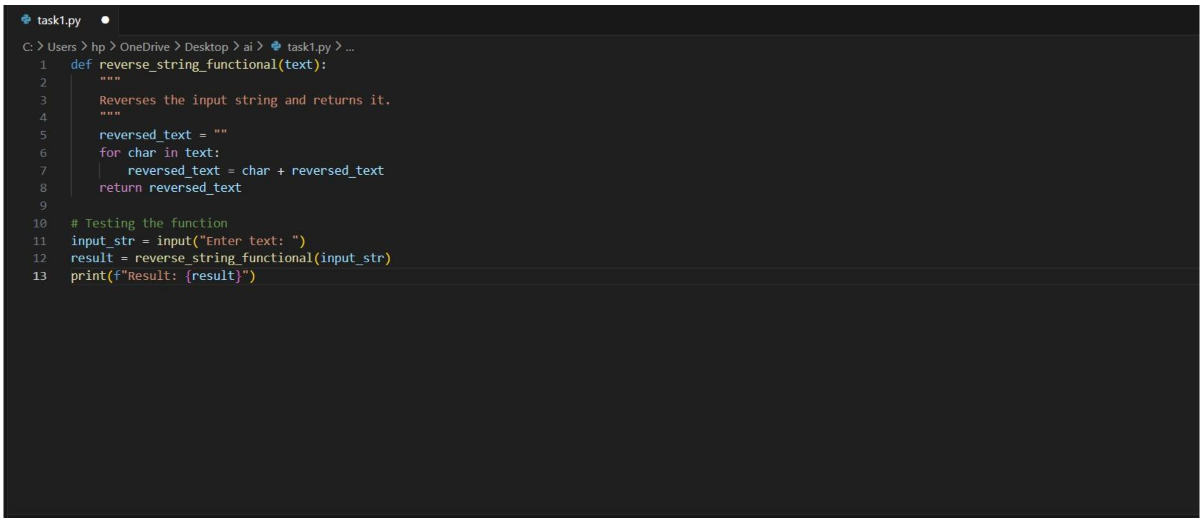
➢ Includes meaningful comments (AI-assisted)

❖ Expected Output

➢ Correct function-based implementation

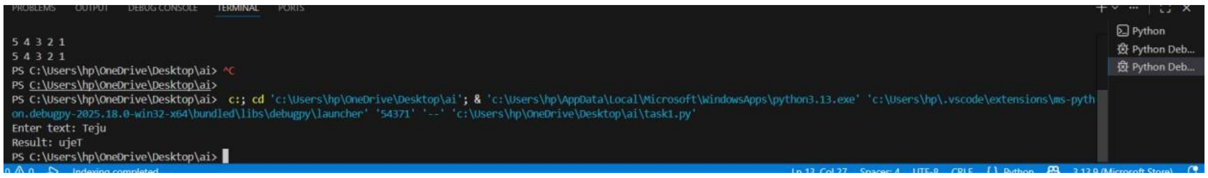
➢ Screenshots documenting Copilot’s function generation

➢ Sample test cases and outputs



```python
def reverse_string_functional(text):
    """
    Reverses the input string and returns it.
    """
    reversed_text = ""
    for char in text:
        reversed_text = char + reversed_text
    return reversed_text

# Testing the function
input_str = input("Enter text: ")
result = reverse_string_functional(input_str)
print(f"Result: {result}")
```



```
5 4 3 2 1
5 4 3 2 1
PS C:\Users\hp\OneDrive\Desktop\ai> ^C
PS C:\Users\hp\OneDrive\Desktop\ai>
PS C:\Users\hp\OneDrive\Desktop\ai>  c:; cd 'c:\Users\hp\OneDrive\Desktop\ai'; & 'c:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\hp\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '54371' '--' 'c:\Users\hp\OneDrive\Desktop\ai\task1.py'
Enter text: Teju
Result: ujeT
PS C:\Users\hp\OneDrive\Desktop\ai>
```

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

Feature	Procedural (Without Functions)	Modular (With Functions)
Code Clarity	Easy for tiny scripts; messy for large ones.	Very high; logic is isolated and named.
Reusability	Must copy-paste code to use it again.	Can be called anywhere in the app.
Debugging	Harder to isolate where an error occurs.	Easy to unit test the specific function.
Scalability	Not suitable for large applications.	Essential for professional development.

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

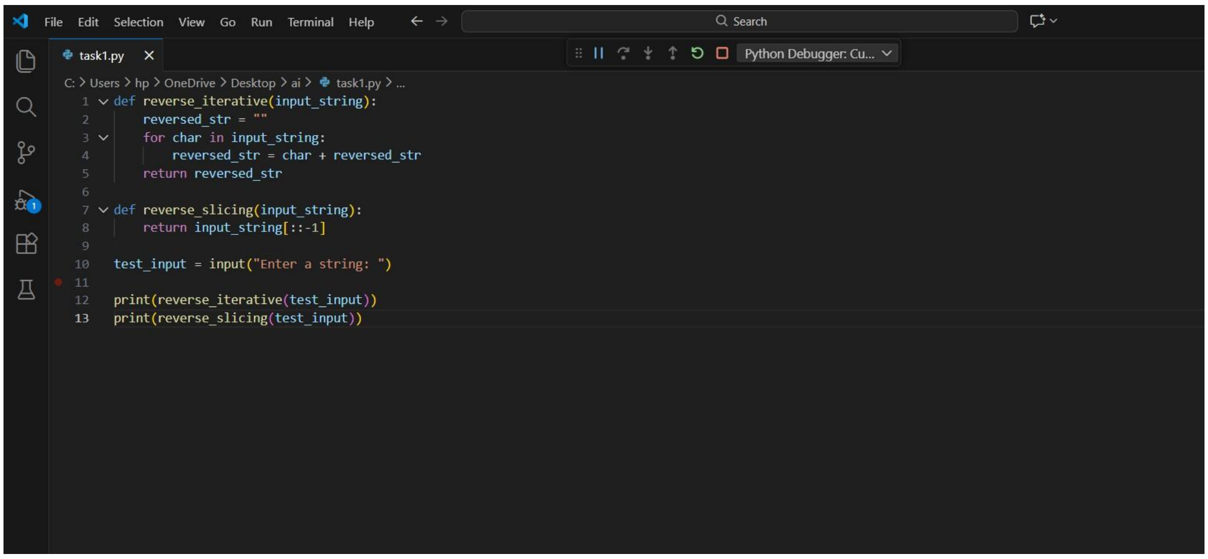
➢ Comparison discussing:

▪ Execution flow

▪ Time complexity

▪ Performance for large inputs

▪ When each approach is appropriate.

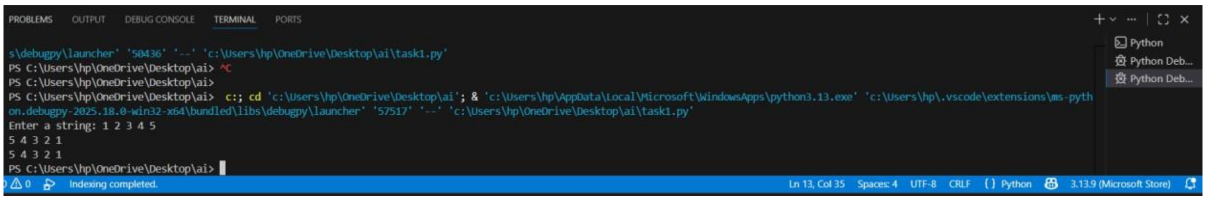

File Edit Selection View Go Run Terminal Help
Search
task1.py
Python Debugger: Cu...
C: > Users > hp > OneDrive > Desktop > ai > task1.py > ...

```python
def reverse_iterative(input_string):
    reversed_str = ""
    for char in input_string:
        reversed_str = char + reversed_str
    return reversed_str

def reverse_slicing(input_string):
    return input_string[::-1]

test_input = input("Enter a string: ")

print(reverse_iterative(test_input))
print(reverse_slicing(test_input))
```



```
s\debugpy\launcher' '50436' '--' 'c:\Users\hp\OneDrive\Desktop\ai\task1.py'
PS C:\Users\hp\OneDrive\Desktop\ai> ^C
PS C:\Users\hp\OneDrive\Desktop\ai>
PS C:\Users\hp\OneDrive\Desktop\ai>  c:; cd 'c:\Users\hp\OneDrive\Desktop\ai'; & 'c:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\hp\.vscode\extensions\ms-pyth
on.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '57517' '--' 'c:\Users\hp\OneDrive\Desktop\ai\task1.py'
Enter a string: 1 2 3 4 5
5 4 3 2 1
5 4 3 2 1
PS C:\Users\hp\OneDrive\Desktop\ai> 
```

0 ⚠ 0 Indexing completed.
Ln 13, Col 35 Spaces: 4 UTF-8 CRLF {} Python 3.13.9 (Microsoft Store)