Assignment - 1

Name: CH.Harshini

Roll Number: 2303A51185

Batch - 03

AI Assisted Coding

07-01-2026

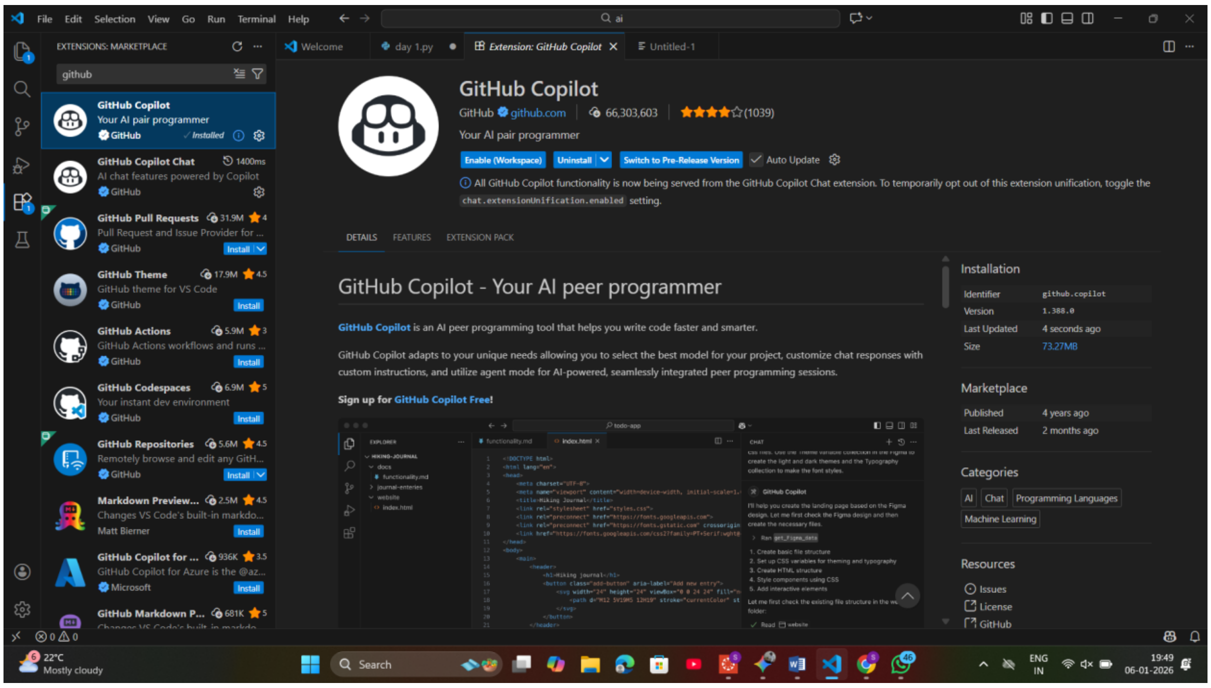
Task 0: Environment Setup:-

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

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

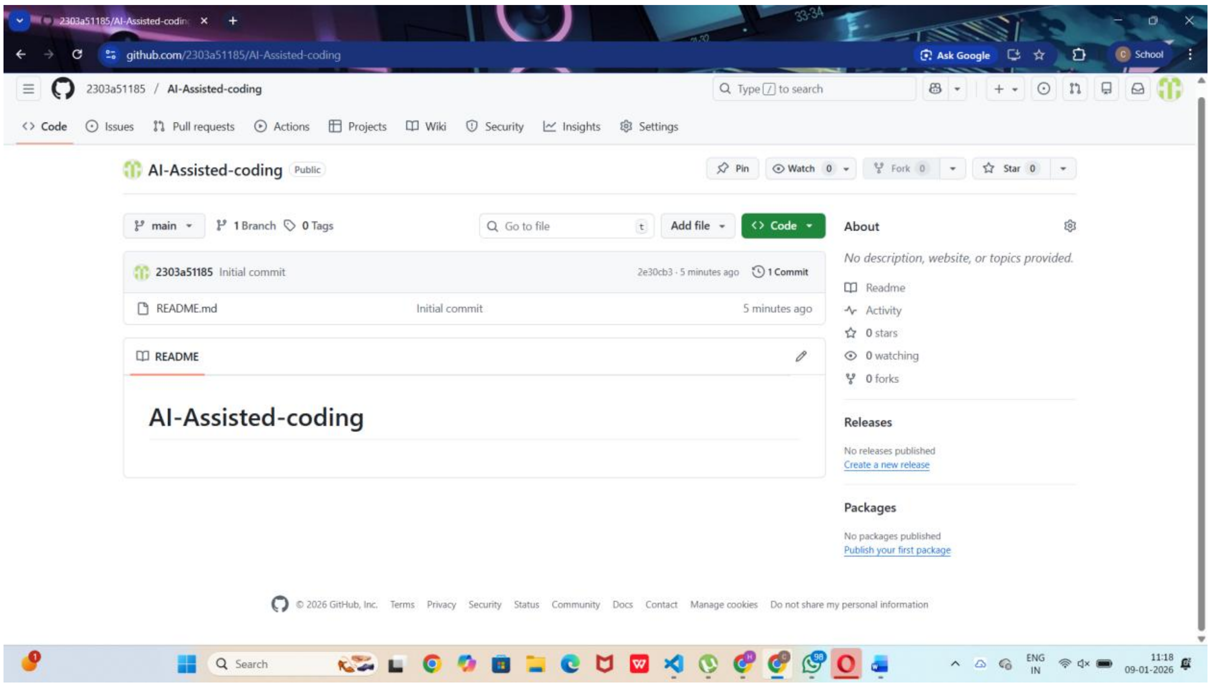
Categories

AI Chat Programming Languages Machine Learning

Resources

Issues License GitHub

22°C Mostly cloudy 19:49 06-01-2026



2303a51185 Initial commit 2e30cb3 · 5 minutes ago 1 Commit

README.md Initial commit 5 minutes ago

README

AI-Assisted-coding

About

No description, website, or topics provided.

Readme
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

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

AI-Generated Logic Without Modularization (Factorial without Functions)

• Scenario

You are building a small command-line utility for a startup intern onboarding task. The program is simple and must be written quickly without modular design.

• Task Description

Use GitHub Copilot to generate a Python program that computes a mathematical product-based value (factorial-like logic) directly in the main execution flow, without using any user-defined functions.

• Constraint:

➢ Do not define any custom function

➢ Logic must be implemented using loops and variables only

• Expected Deliverables

➢ A working Python program generated with Copilot assistance

➢ Screenshot(s) showing:

➢ The prompt you typed

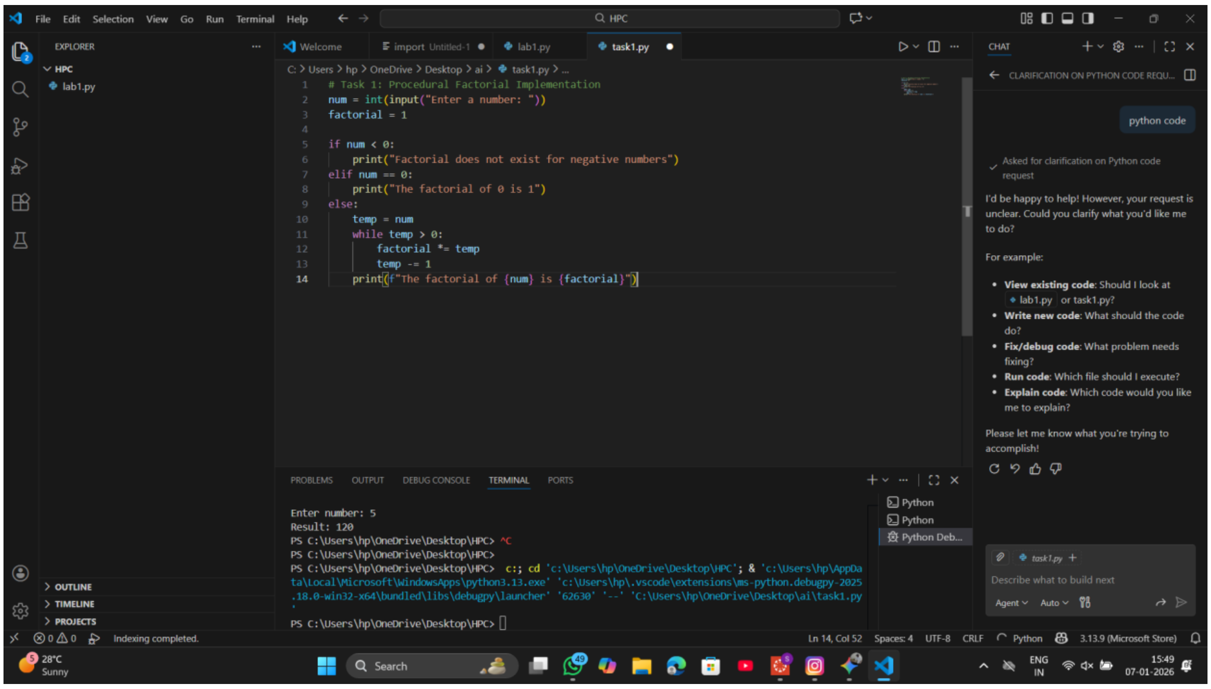
➢ Copilot's suggestions

➢ Sample input/output screenshots

➢ Brief reflection (5–6 lines):

➢ How helpful was Copilot for a beginner?

➢ Did it follow best practices automatically?



```python
# Task 1: Procedural Factorial Implementation
num = int(input("Enter a number: "))
factorial = 1

if num < 0:
    print("Factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    temp = num
    while temp > 0:
        factorial *= temp
        temp -= 1
    print(f"The factorial of {num} is {factorial}")
```

python code

Asked for clarification on Python code request

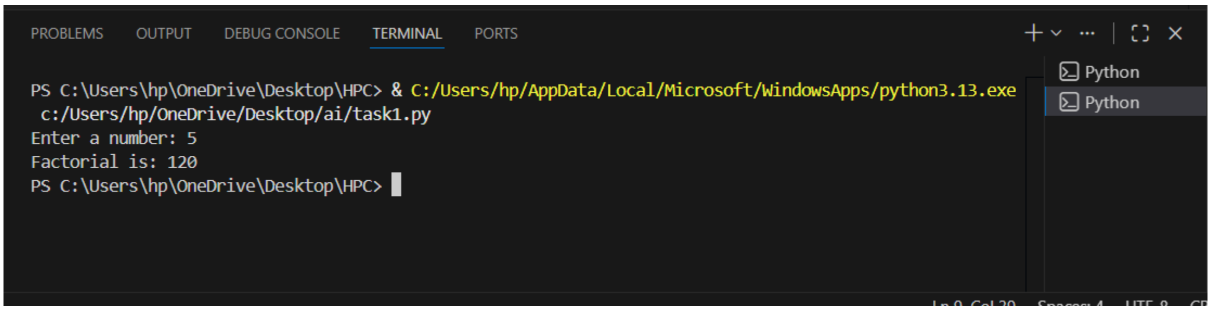
I'd be happy to help! However, your request is unclear. Could you clarify what you'd like me to do?

For example:

- **View existing code**: Should I look at lab1.py or task1.py?
- **Write new code**: What should the code do?
- **Fix/debug code**: What problem needs fixing?
- **Run code**: Which file should I execute?
- **Explain code**: Which code would you like me to explain?

Please let me know what you're trying to accomplish!

```
Enter number: 5
Result: 120
PS C:\Users\hp\OneDrive\Desktop\HPC> ^C
PS C:\Users\hp\OneDrive\Desktop\HPC>
PS C:\Users\hp\OneDrive\Desktop\HPC>  c:; cd 'c:\Users\hp\OneDrive\Desktop\HPC'; & 'c:\Users\hp\AppDa
ta\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\hp\.vscode\extensions\ms-python.debugpy-2025
.18.0-win32-x64\bundled\libs\debugpy\launcher' '62630' '--' 'C:\Users\hp\OneDrive\Desktop\ai\task1.py
'
PS C:\Users\hp\OneDrive\Desktop\HPC> 
```



```
PS C:\Users\hp\OneDrive\Desktop\HPC> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.13.exe
 c:/Users/hp/OneDrive/Desktop/ai/task1.py
Enter a number: 5
Factorial is: 120
PS C:\Users\hp\OneDrive\Desktop\HPC> ▌
```

Task 2: AI Code Optimization:-

AI Code Optimization & Cleanup (Improving Efficiency)

❖ Scenario

Your team lead asks you to review AI-generated code before committing it to a shared repository.

❖ Task Description

Analyze the code generated in Task 1 and use Copilot again to:

➢ Reduce unnecessary variables

➢ Improve loop clarity

➢ Enhance readability and efficiency

Hint:

Prompt Copilot with phrases like

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

❖ Expected Deliverables

➢ Original AI-generated code

➢ Optimized version of the same code

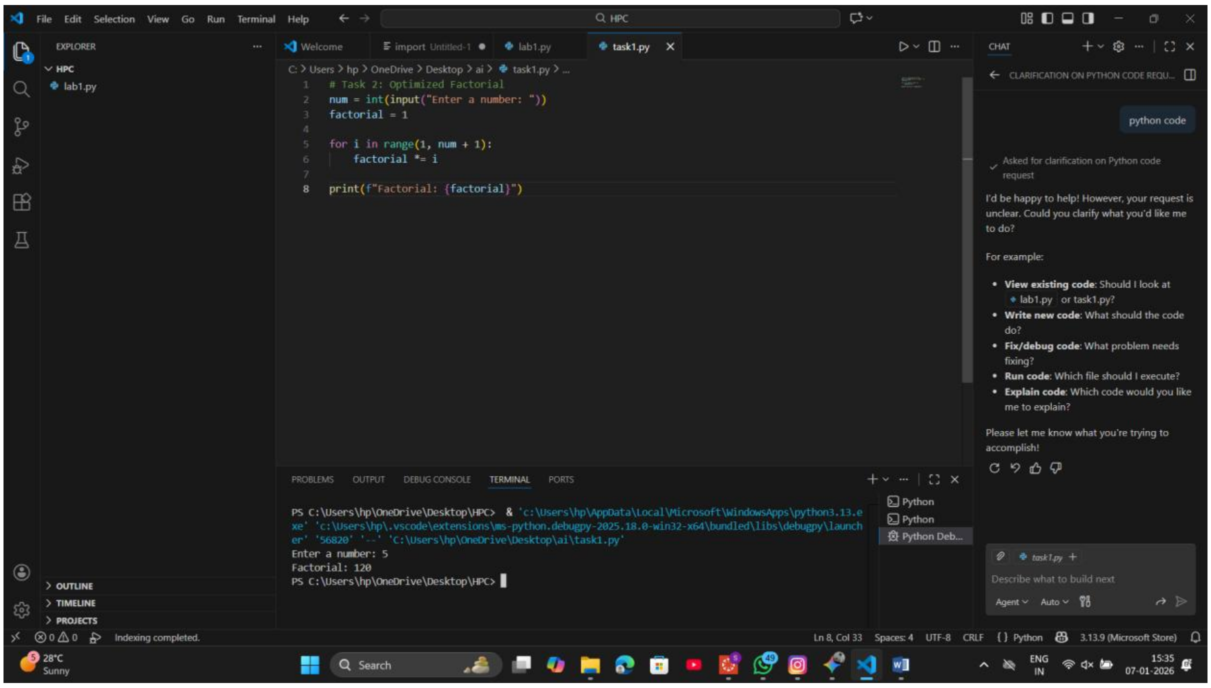
➢ Side-by-side comparison

➢ Written explanation:

▪ What was improved?

▪ Why the new version is better (readability, performance,

maintainability.


File Edit Selection View Go Run Terminal Help
HPC
EXPLORER
HPC
lab1.py
Welcome
import Untitled-1
lab1.py
task1.py
C: > Users > hp > OneDrive > Desktop > ai > task1.py > ...
OUTLINE
TIMELINE
PROJECTS
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Python
Python Deb...
CHAT
CLARIFICATION ON PYTHON CODE REQU...
Describe what to build next
Agent Auto
Indexing completed.
Ln 8, Col 33 Spaces: 4 UTF-8 CRLF {} Python 3.13.9 (Microsoft Store)
28°C Sunny
Search
ENG IN
15:35 07-01-2026

```python
# Task 2: Optimized Factorial
num = int(input("Enter a number: "))
factorial = 1

for i in range(1, num + 1):
    factorial *= i

print(f"Factorial: {factorial}")
```

```
PS C:\Users\hp\OneDrive\Desktop\HPC>  & 'c:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.13.e
xe' 'c:\Users\hp\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launch
er' '56820' '--' 'C:\Users\hp\OneDrive\Desktop\ai\task1.py'
Enter a number: 5
Factorial: 120
PS C:\Users\hp\OneDrive\Desktop\HPC>
```

python code

✓ Asked for clarification on Python code request

I'd be happy to help! However, your request is unclear. Could you clarify what you'd like me to do?

For example:

- **View existing code**: Should I look at lab1.py or task1.py?
- **Write new code**: What should the code do?
- **Fix/debug code**: What problem needs fixing?
- **Run code**: Which file should I execute?
- **Explain code**: Which code would you like me to explain?

Please let me know what you're trying to accomplish!

Task 3: Modular Design Using AI Assistance (Factorial with Functions)

❖ Scenario

The same logic now needs to be reused in multiple scripts.

❖ Task Description

Use GitHub Copilot to generate a modular version of the program by:

➢ Creating a user-defined function

➢ Calling the function from the main block

❖ Constraints

➢ Use meaningful function and variable names

➢ Include inline comments (preferably suggested by Copilot)

❖ Expected Deliverables

➢ AI-assisted function-based program

➢ Screenshots showing:

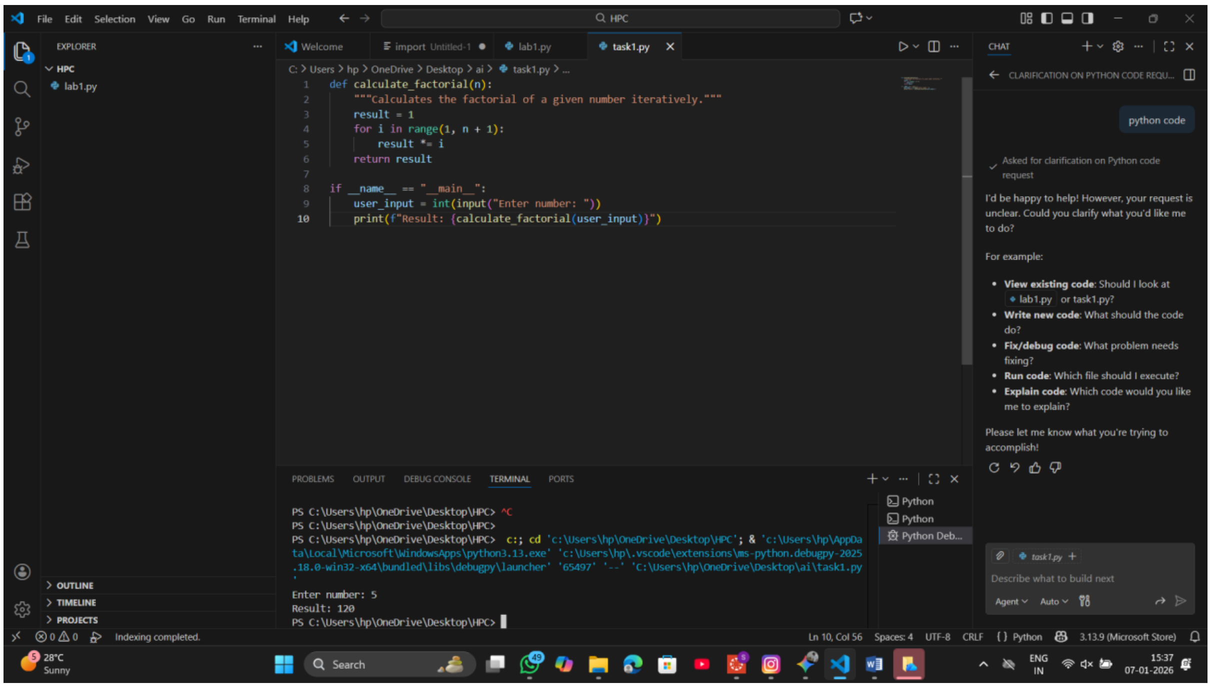
o Prompt evolution

o Copilot-generated function logic

➢ Sample inputs/outputs

➢ Short note:

o How modularity improves reusability.


File Edit Selection View Go Run Terminal Help
HPC
EXPLORER
HPC
lab1.py
Welcome
import Untitled-1
lab1.py
task1.py
C: > Users > hp > OneDrive > Desktop > ai > task1.py > ...
OUTLINE
TIMELINE
PROJECTS
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Python
Python Deb...
CHAT
CLARIFICATION ON PYTHON CODE REQU...
task1.py
Describe what to build next
Agent Auto
Indexing completed.
Ln 10, Col 56 Spaces: 4 UTF-8 CRLF {} Python 3.13.9 (Microsoft Store)
28°C Sunny
Search
ENG IN
15:37 07-01-2026

```python
def calculate_factorial(n):
    """Calculates the factorial of a given number iteratively."""
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

if __name__ == "__main__":
    user_input = int(input("Enter number: "))
    print(f"Result: {calculate_factorial(user_input)}")
```

```
PS C:\Users\hp\OneDrive\Desktop\HPC> ^C
PS C:\Users\hp\OneDrive\Desktop\HPC>
PS C:\Users\hp\OneDrive\Desktop\HPC>  c:; cd 'c:\Users\hp\OneDrive\Desktop\HPC'; & 'c:\Users\hp\AppDa
ta\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\hp\.vscode\extensions\ms-python.debugpy-2025
.18.0-win32-x64\bundled\libs\debugpy\launcher' '65497' '--' 'C:\Users\hp\OneDrive\Desktop\ai\task1.py
'
Enter number: 5
Result: 120
PS C:\Users\hp\OneDrive\Desktop\HPC> 
```

python code

✓ Asked for clarification on Python code request

I'd be happy to help! However, your request is unclear. Could you clarify what you'd like me to do?

For example:

- **View existing code**: Should I look at lab1.py or task1.py?
- **Write new code**: What should the code do?
- **Fix/debug code**: What problem needs fixing?
- **Run code**: Which file should I execute?
- **Explain code**: Which code would you like me to explain?

Please let me know what you're trying to accomplish!

Task 4: Comparative Analysis:-

Comparative Analysis – Procedural vs Modular AI Code (With vs Without Functions)

❖ Scenario

As part of a code review meeting, you are asked to justify design choices.

❖ Task Description

Compare the non-function and function-based Copilot-generated programs on the following criteria:

➢ Logic clarity

➢ Reusability

➢ Debugging ease

➢ Suitability for large projects

➢ AI dependency risk

❖ Expected Deliverables

Choose one:

➢ A comparison table

OR

➢ A short technical report (300–400 words).

Criteria	**Procedural (Task 1 & 2)**	**Modular (Task 3)**
Logic Clarity	Linear and straightforward for very small tasks but becomes "spaghetti code" as complexity grows.	High clarity; the mathematical logic is isolated from the input/output logic.
Reusability	None. To use the logic elsewhere, the code must be manually copied and pasted.	High. The function can be imported into other Python files or called multiple times in one script.
Debugging Ease	Difficult. Errors in logic are mixed with errors in user input handling.	Simple. You can test the function with specific values

Criteria	Procedural (Task 1 & 2)	Modular (Task 3)
		(Unit Testing) to ensure the math is correct.
Project Suitability	Suitable only for small, one-off scripts or prototypes.	Essential for enterprise-level, large-scale software development.
AI Dependency Risk	High. AI might generate redundant variables or inefficient loops in long scripts.	Low. AI is highly specialized and accurate when asked to write specific, single-purpose functions.

Task 5: Iterative vs Recursive Thinking:-

: AI-Generated Iterative vs Recursive Thinking

❖ Scenario

Your mentor wants to test how well AI understands different

computational paradigms.

❖ Task Description

Prompt Copilot to generate:

An iterative version of the logic

A recursive version of the same logic

❖ Constraints

Both implementations must produce identical outputs

Students must not manually write the code first

❖ Expected Deliverables

Two AI-generated implementations

Execution flow explanation (in your own words)

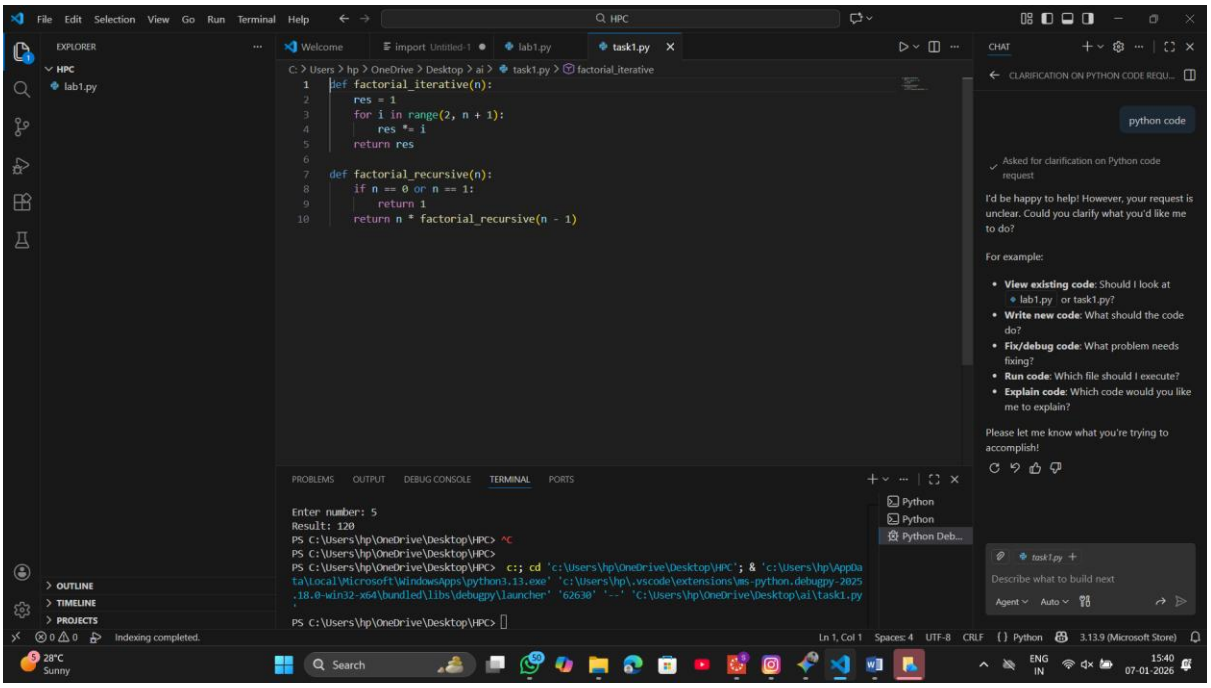
Comparison covering:

➢ Readability

➢ Stack usage

➢ **Performance implications**

➢ **When recursion is not recommended.**



```python
def factorial_iterative(n):
    res = 1
    for i in range(2, n + 1):
        res *= i
    return res

def factorial_recursive(n):
    if n == 0 or n == 1:
        return 1
    return n * factorial_recursive(n - 1)
```

```
Enter number: 5
Result: 120
PS C:\Users\hp\OneDrive\Desktop\HPC> ^C
PS C:\Users\hp\OneDrive\Desktop\HPC>
PS C:\Users\hp\OneDrive\Desktop\HPC>  c:; cd 'c:\Users\hp\OneDrive\Desktop\HPC'; & 'c:\Users\hp\AppDa
ta\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\hp\.vscode\extensions\ms-python.debugpy-2025
.18.0-win32-x64\bundled\libs\debugpy\launcher' '62630' '--' 'C:\Users\hp\OneDrive\Desktop\ai\task1.py
'
PS C:\Users\hp\OneDrive\Desktop\HPC> 
```

python code

Asked for clarification on Python code request

I'd be happy to help! However, your request is unclear. Could you clarify what you'd like me to do?

For example:

- **View existing code**: Should I look at lab1.py or task1.py?
- **Write new code**: What should the code do?
- **Fix/debug code**: What problem needs fixing?
- **Run code**: Which file should I execute?
- **Explain code**: Which code would you like me to explain?

Please let me know what you're trying to accomplish!

Assignment – 1.5

Name:CH.Harshini

Roll Number: 2303A51185

Batch - 03

AI Assisted Coding

09-01-2026

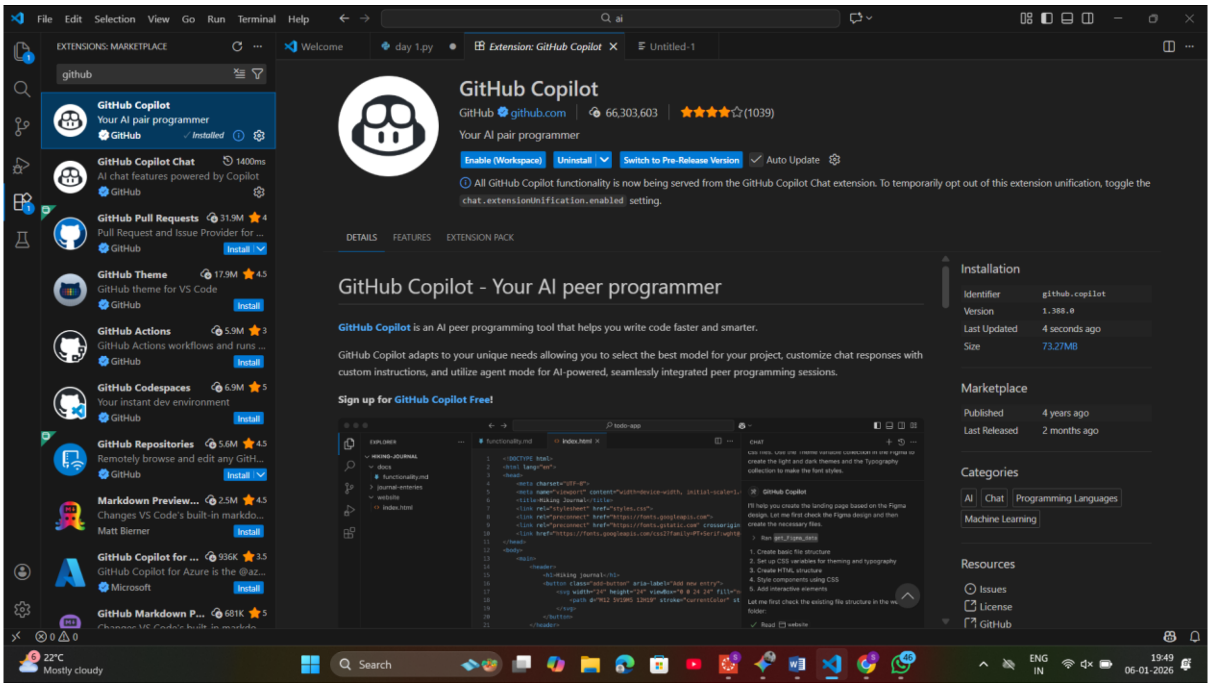
Task 0: Environment Setup:-

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.**


EXTENSIONS: MARKETPLACE

github

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

GitHub Copilot

GitHub github.com | 66,303,603 | ★★★★☆(1039)

Your AI pair programmer

Enable (Workspace) | Uninstall | Switch to Pre-Release Version | ✓ Auto Update

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.

DETAILS FEATURES EXTENSION PACK

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

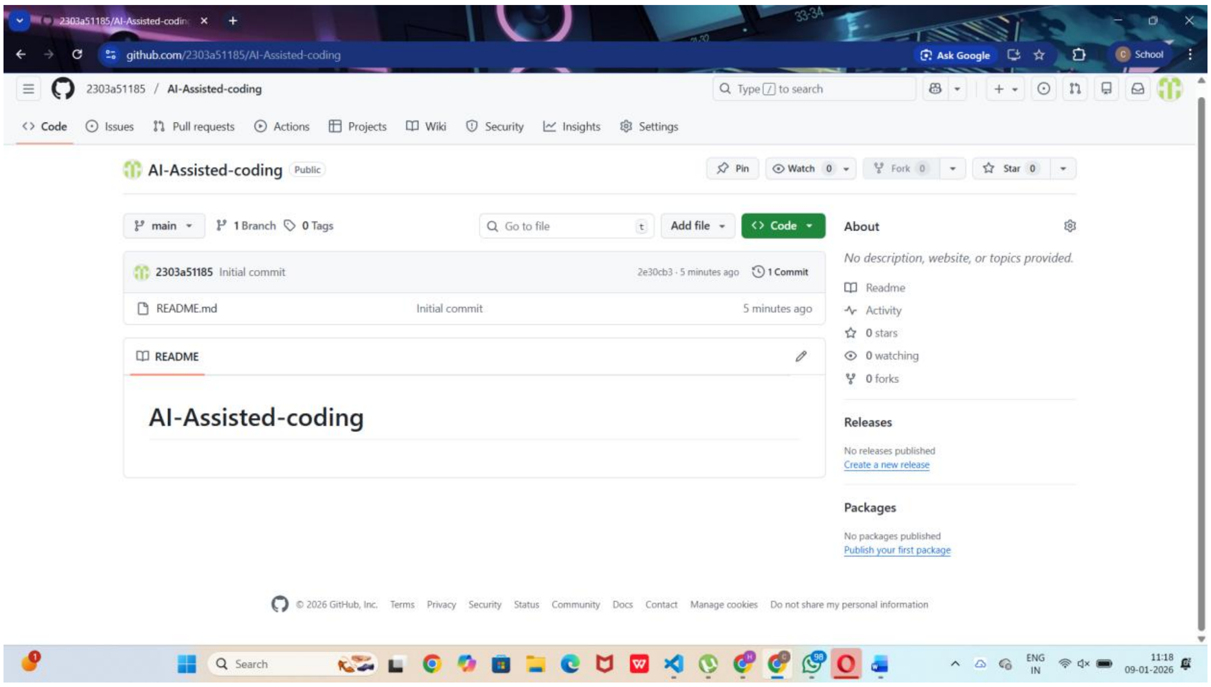
Categories

AI | Chat | Programming Languages | Machine Learning

Resources

Issues
License
GitHub

File Edit Selection View Go Run Terminal Help
Welcome
day 1.py
Extension: GitHub Copilot
Untitled-1
22°C Mostly cloudy
Search
19:49
06-01-2026


github.com/2303a51185/AI-Assisted-coding
2303a51185 / AI-Assisted-coding
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings

AI-Assisted-coding Public

main · 1 Branch · 0 Tags

2303a51185 Initial commit 2e30cb3 · 5 minutes ago 1 Commit

README.md Initial commit 5 minutes ago

README

AI-Assisted-coding

About

No description, website, or topics provided.

Readme
Activity
0 stars
0 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

© 2026 GitHub, Inc. Terms Privacy Security Status Community Docs Contact Manage cookies Do not share my personal information

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

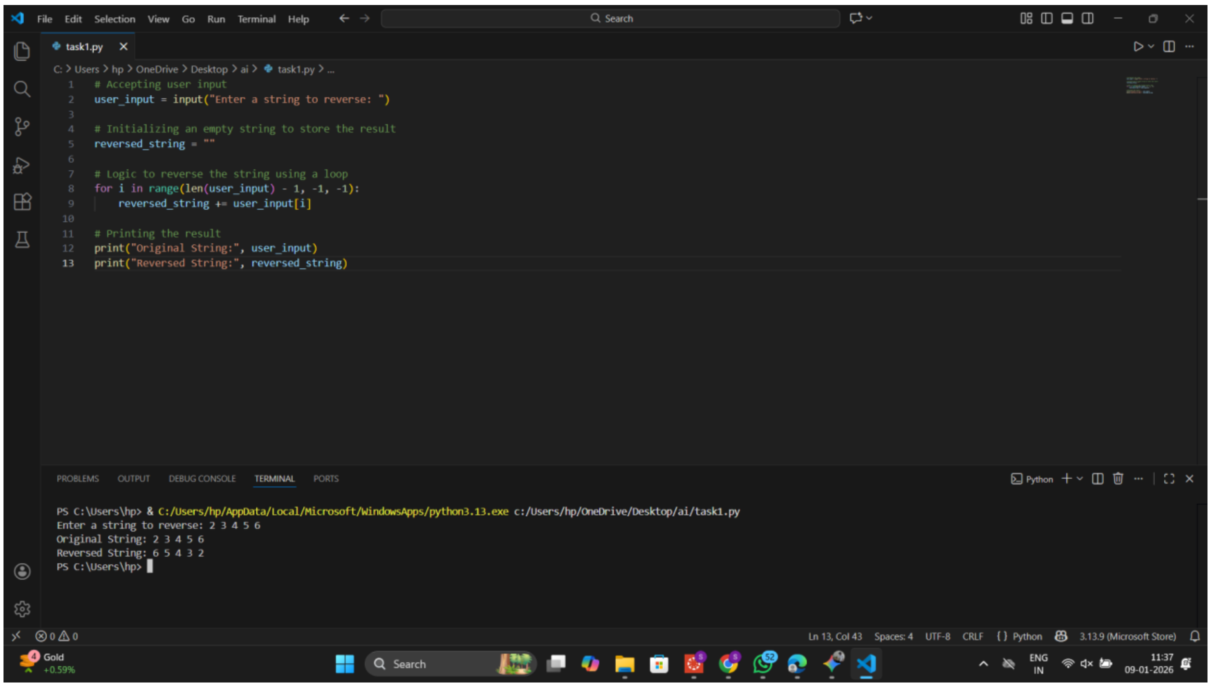
➢ Does not use any user-defined functions

❖ Expected Output

➢ Correct reversed string

➢ Screenshots showing Copilot-generated code suggestions

➢ Sample inputs and outputs



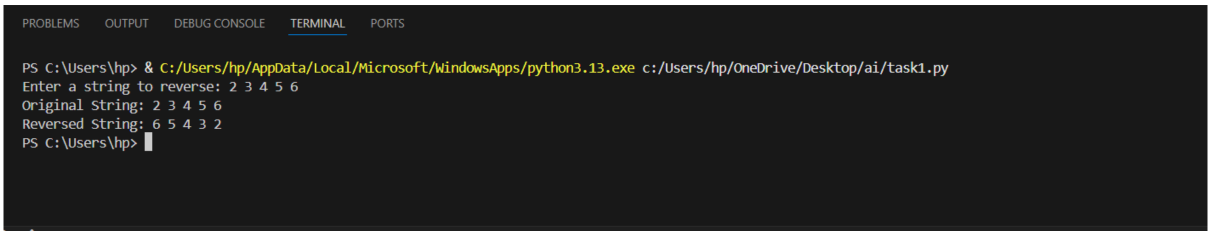
```python
# Accepting user input
user_input = input("Enter a string to reverse: ")

# 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 & 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:

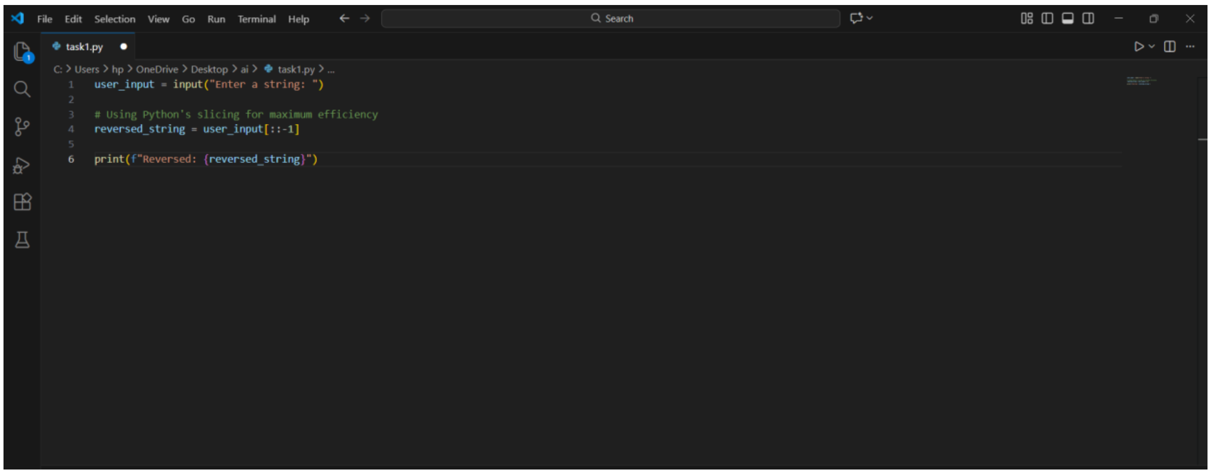
Prompt Copilot with phrases like

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

❖ Expected Output

➢ Original and optimized code versions

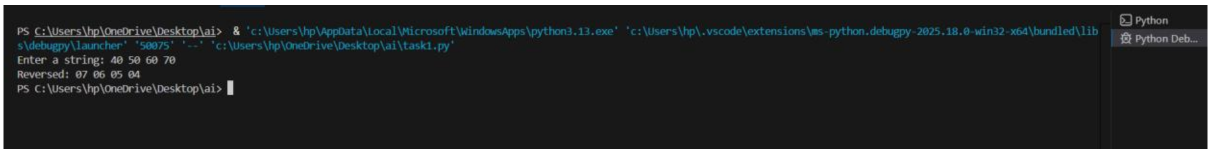
➢ Explanation of how the improvements reduce time complexity



```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

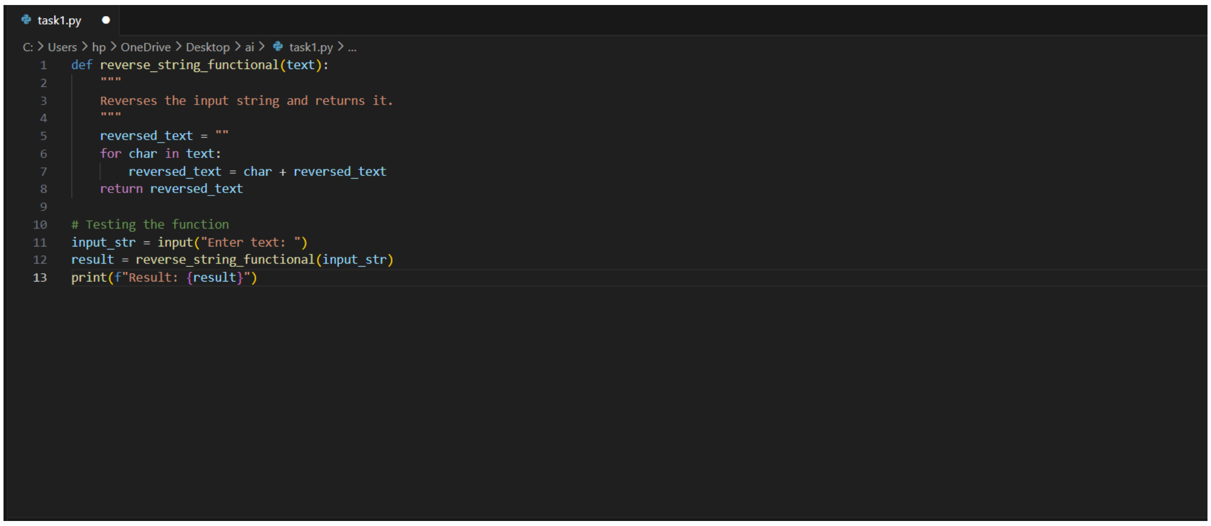
➢ Includes meaningful comments (AI-assisted)

❖ Expected Output

➢ Correct function-based implementation

➢ Screenshots documenting Copilot’s function generation

➢ Sample test cases and outputs

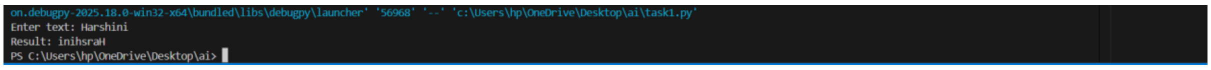


task1.py

C: > Users > hp > OneDrive > Desktop > ai > task1.py > ...

```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}")
```



```
on.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '56968' '--' 'c:\Users\hp\OneDrive\Desktop\ai\task1.py'
Enter text: Harshini
Result: inihsraH
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

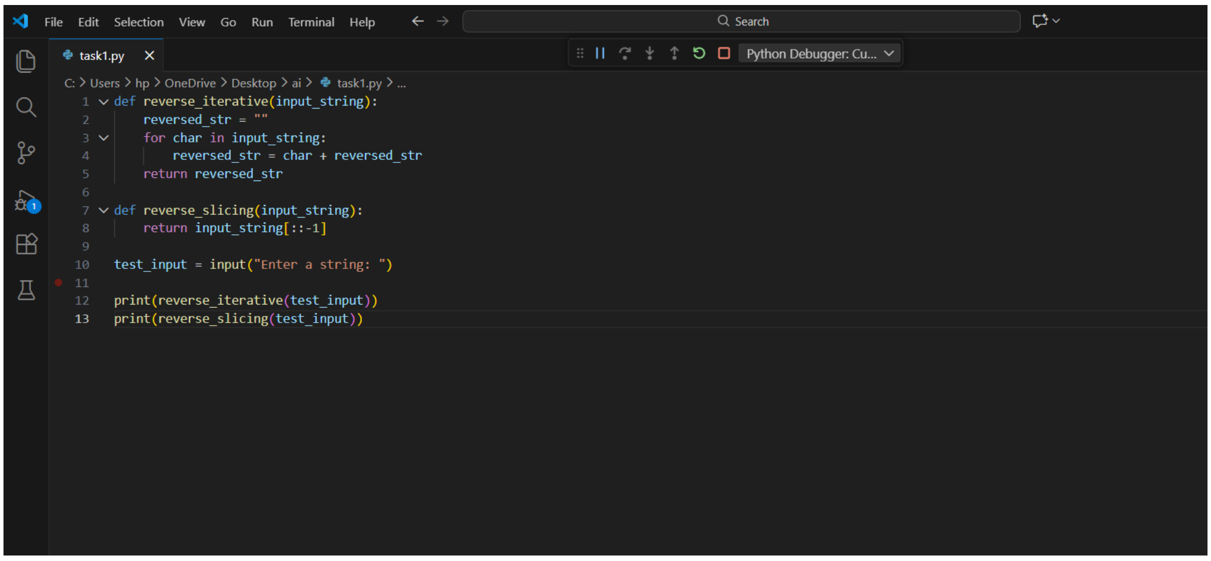
➢ Comparison discussing:

▪ Execution flow

▪ Time complexity

▪ Performance for large inputs

▪ When each approach is appropriate.

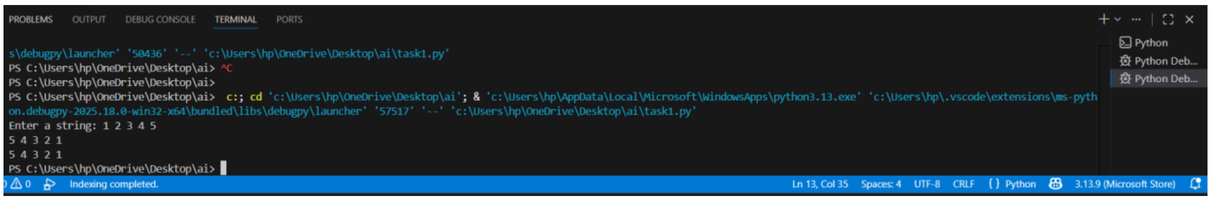


```python
def reverse_iterative(input_string):
    reversed_str = ""
    for char in input_string:
        reversed_str = char + reversed_str
    return reversed_str

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>
```