Assignment - 1

Name: Sambaraju Vignesh

Roll Number: 2303A51217

Batch - 04

AI Assisted Coding

07-01-2026

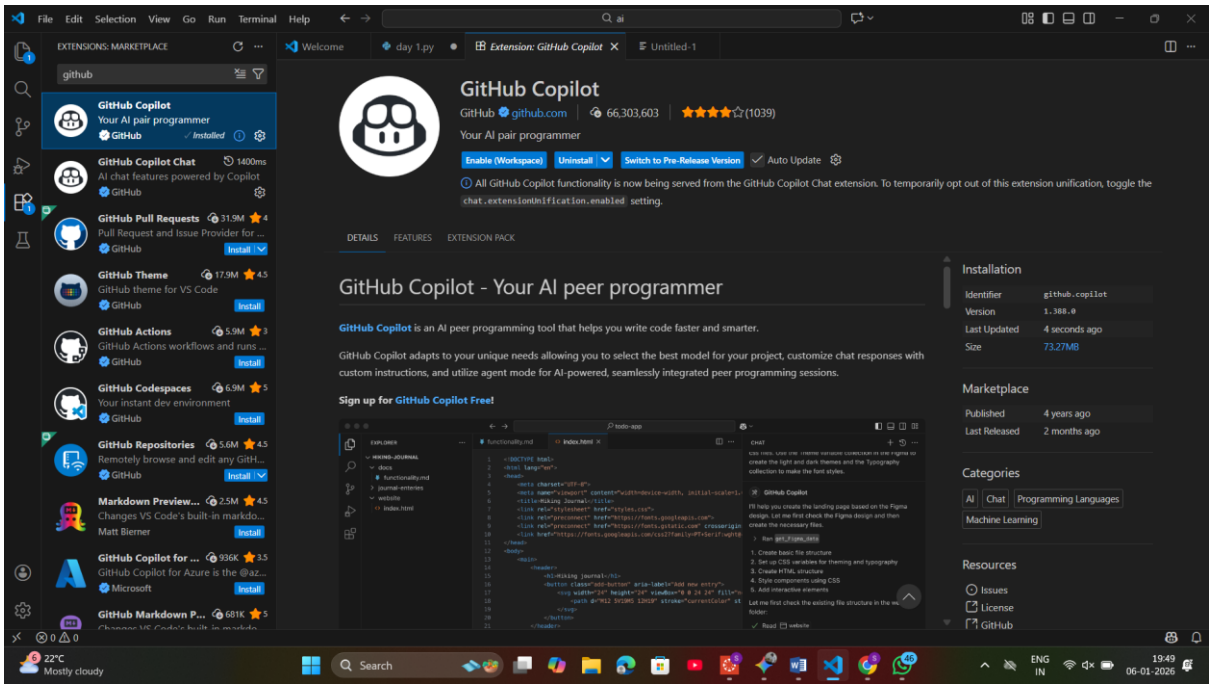
Task 0: Environment Setup:-

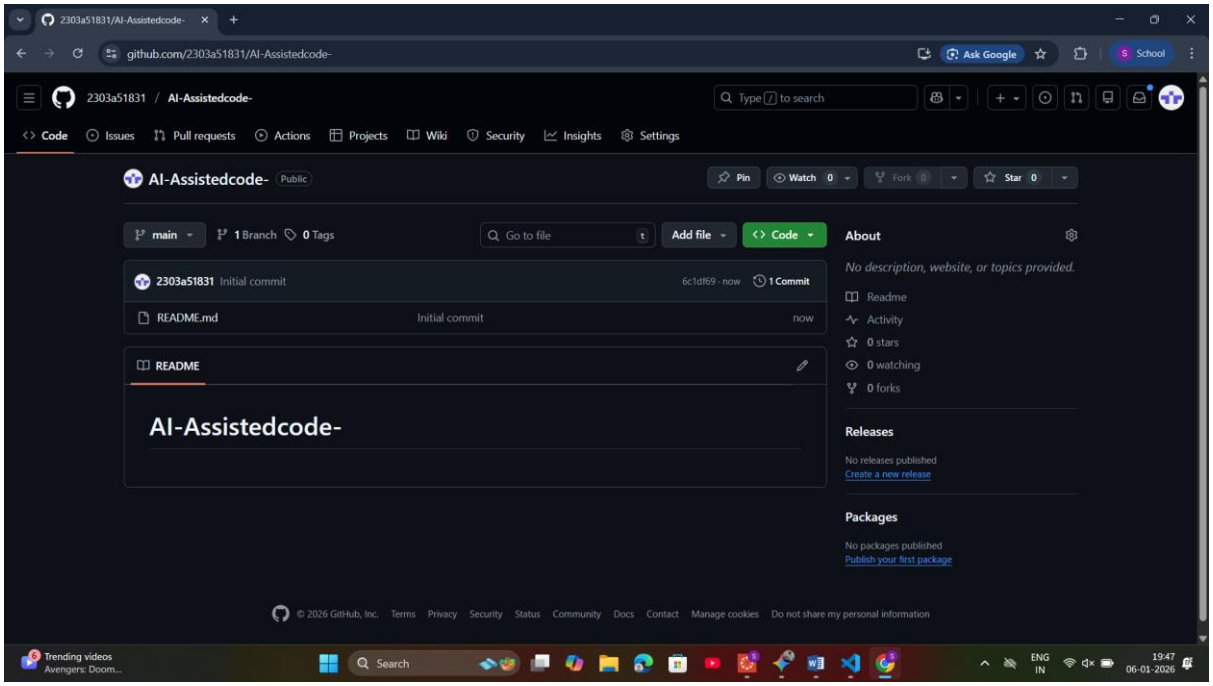
Task 0

● 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.





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

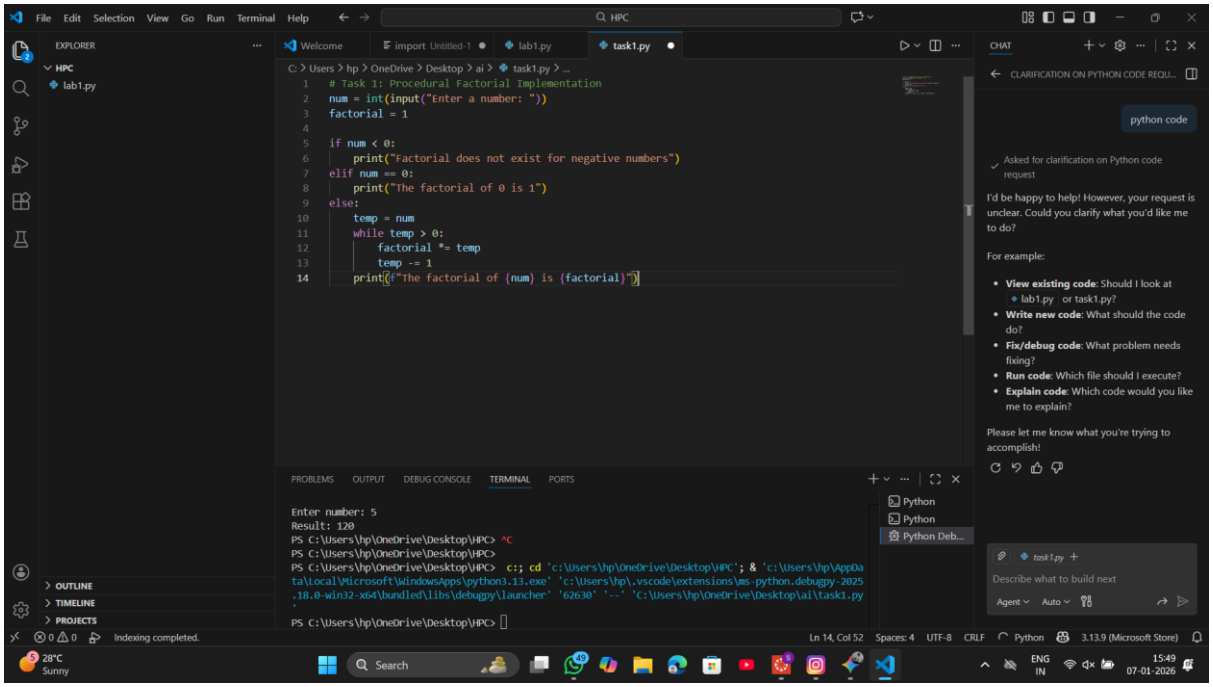
➢ Copilot’s suggestions

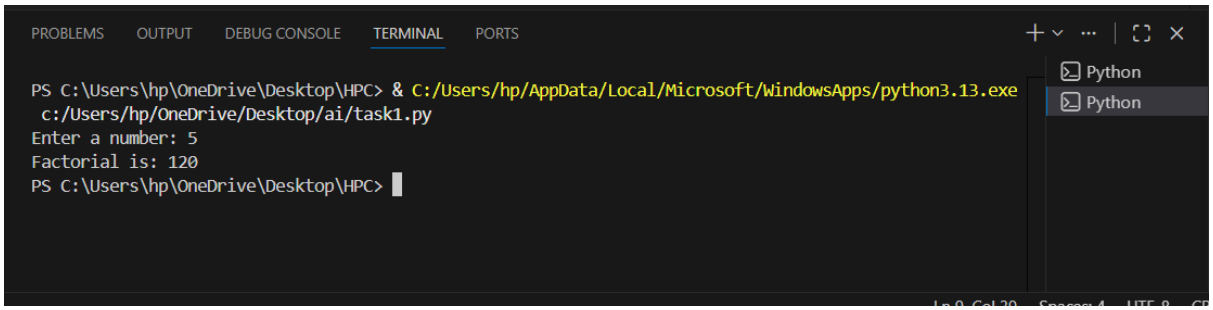
➢ Sample input/output screenshots

➢ Brief reflection (5–6 lines):

➢ How helpful was Copilot for a beginner?

➢ Did it follow best practices automatically?





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**

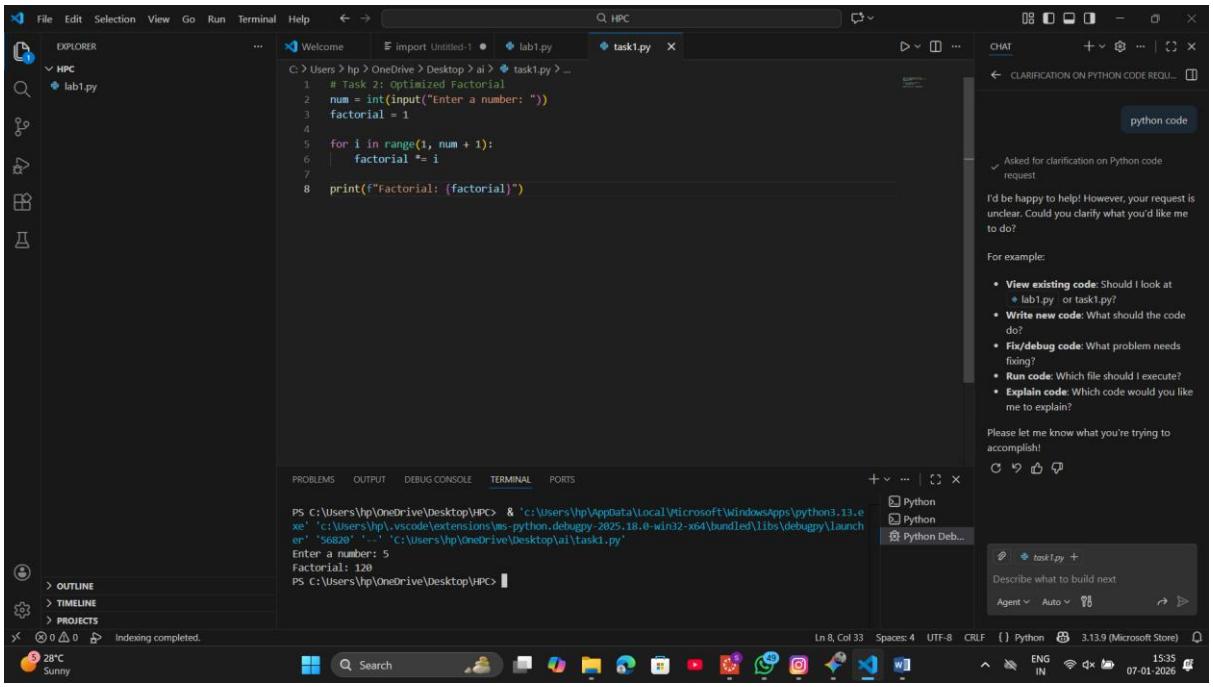
➢ **Optimized version of the same code**

➢ **Side-by-side comparison**

➢ **Written explanation:**

▪ **What was improved?**

▪ **Why the new version is better (readability, performance, maintainability.**



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:**

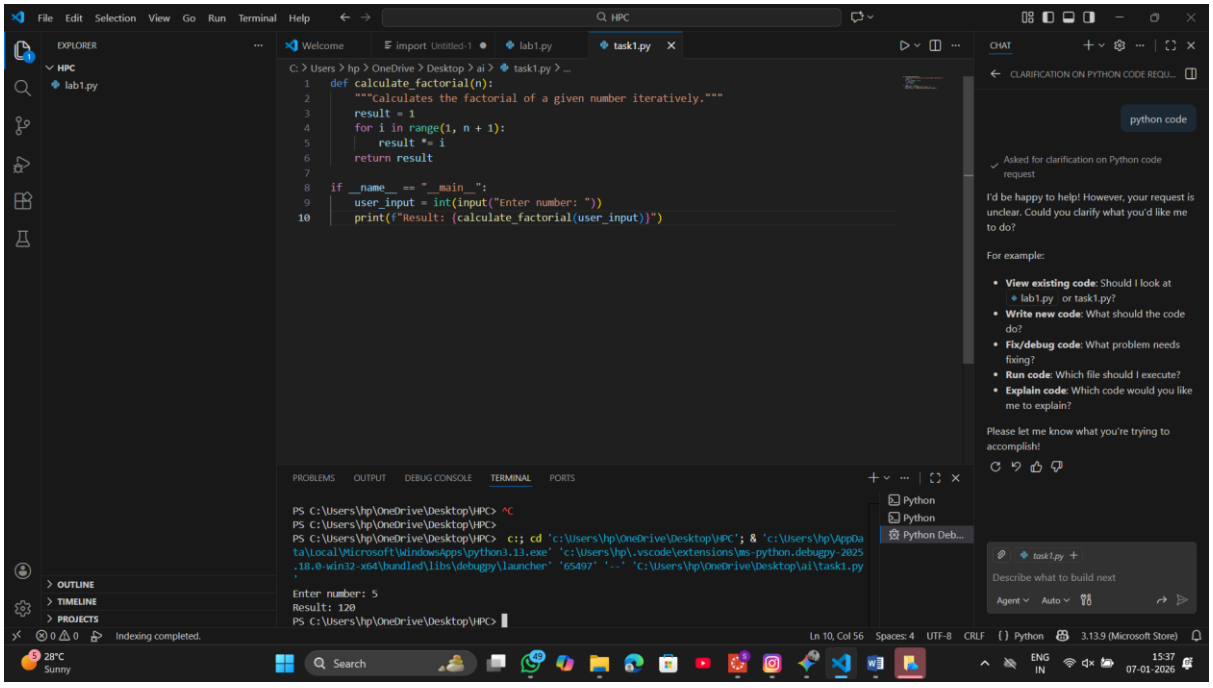
o Prompt evolution

o Copilot-generated function logic

➢ **Sample inputs/outputs**

➢ **Short note:**

o How modularity improves reusability.



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 (Unit Testing) to ensure the math is correct.

Criteria	Procedural (Task 1 & 2)	Modular (Task 3)
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)

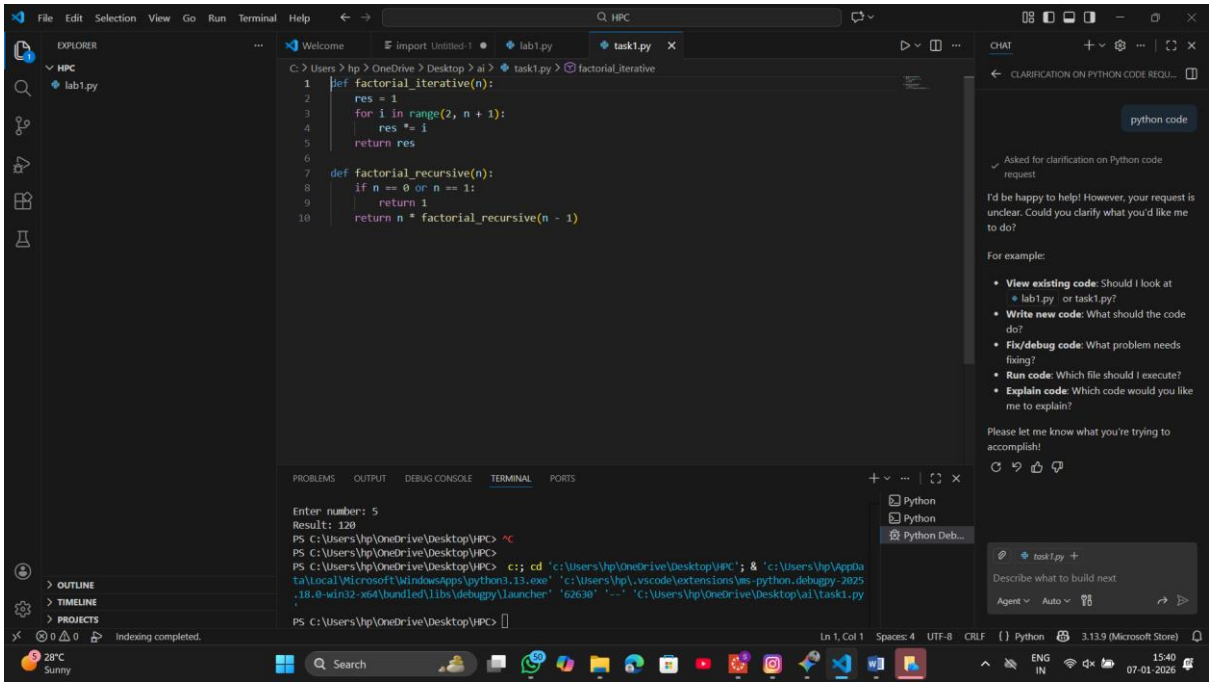
Comparison covering:

➢ Readability

➢ Stack usage

➢ Performance implications

➢ When recursion is not recommended.


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 > factorial_iterative
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)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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\AppData\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
Python
Python Deb...
OUTLINE
TIMELINE
PROJECTS
Indexing completed.
CHAT
CLARIFICATION ON PYTHON CODE REQU...
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!
task1.py
Describe what to build next
Agent
Auto
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.13.9 (Microsoft Store)
28°C
Sunny
Search
ENG
IN
15:40
07-01-2026