ASSIGNMENT-1.2

2303A52187

TASK 0: Install & Configure GitHub Copilot (MANDATORY)

Step-by-Step

Open VS Code

Go to Extensions

Search GitHub Copilot

Click Install

Sign in using your GitHub account

Restart VS Code if asked

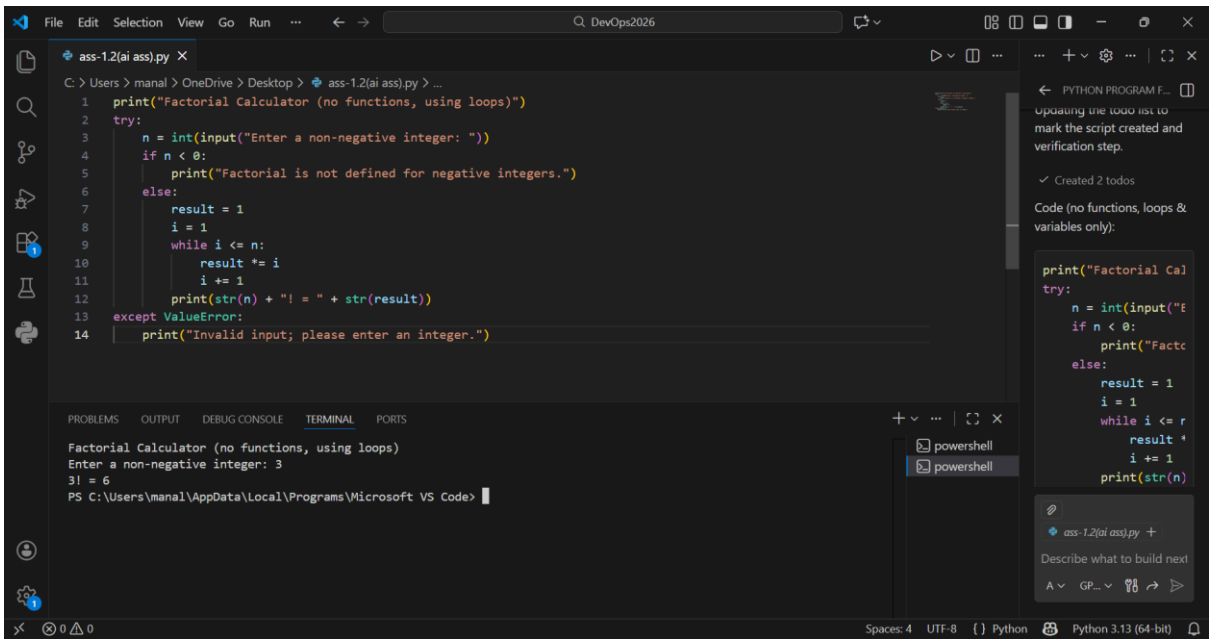
Screenshots to Take

Copilot extension page

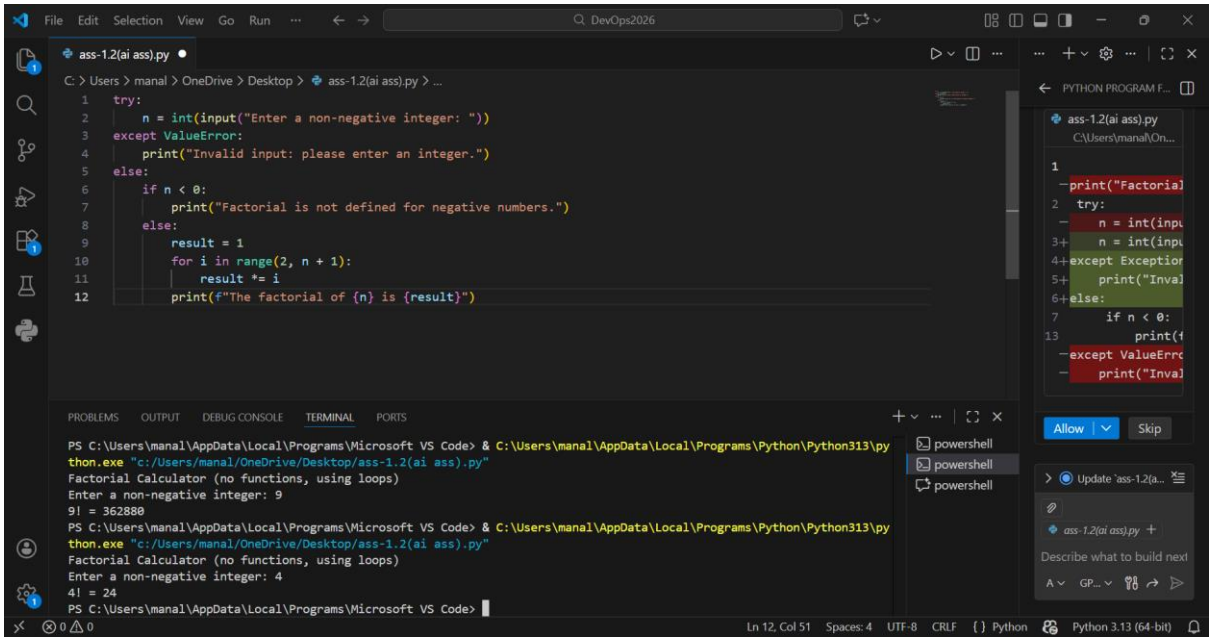
GitHub sign-in success

Copilot enabled in VS Code

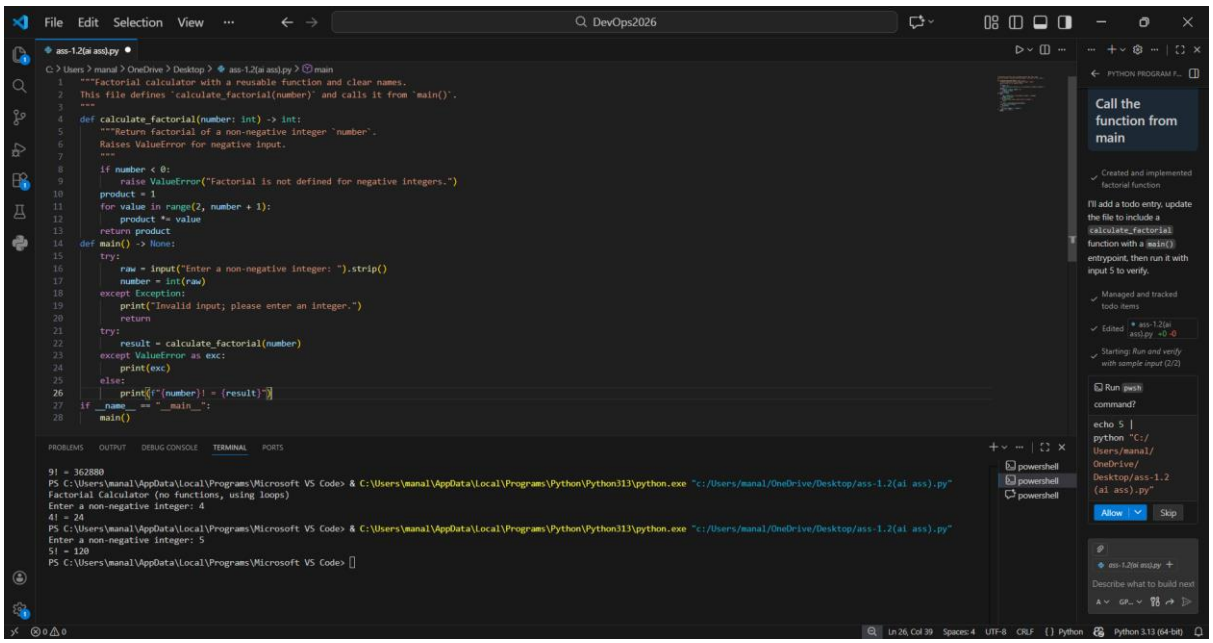
TASK 1: AI-Generated Logic (NO FUNCTIONS)



Task-2: Optimized version



TASK 3: Modular Design (WITH FUNCTIONS)



Task 4: Comparative Analysis – Procedural vs Modular AI Code

Introduction

In this task, we compare two AI-generated Python programs created using GitHub Copilot:

1. A **procedural version** (without functions)
2. A **modular version** (with user-defined functions)

Both programs compute the factorial of a number, but they differ in structure and design. This comparison helps justify design choices during code reviews and understand best practices in AI-assisted coding.

Comparison Table

Criteria	Procedural Code (Without Functions)	Modular Code (With Functions)
Logic Clarity	Logic is written in a single block, making it slightly harder to understand for larger programs	Logic is clearly separated into a function, making it easier to read and follow
Reusability	Cannot be reused easily; code must be rewritten if needed elsewhere	Highly reusable; the function can be called from multiple programs
Ease of Debugging	Debugging is harder since all logic is mixed together	Easier debugging because errors can be isolated inside the function
Suitability for Large Projects	Not suitable for large projects due to poor structure	Very suitable for large projects due to modular design
AI Dependency Risk	Higher risk because AI-generated code may be copied blindly without structure	Lower risk because modular code encourages review and refinement

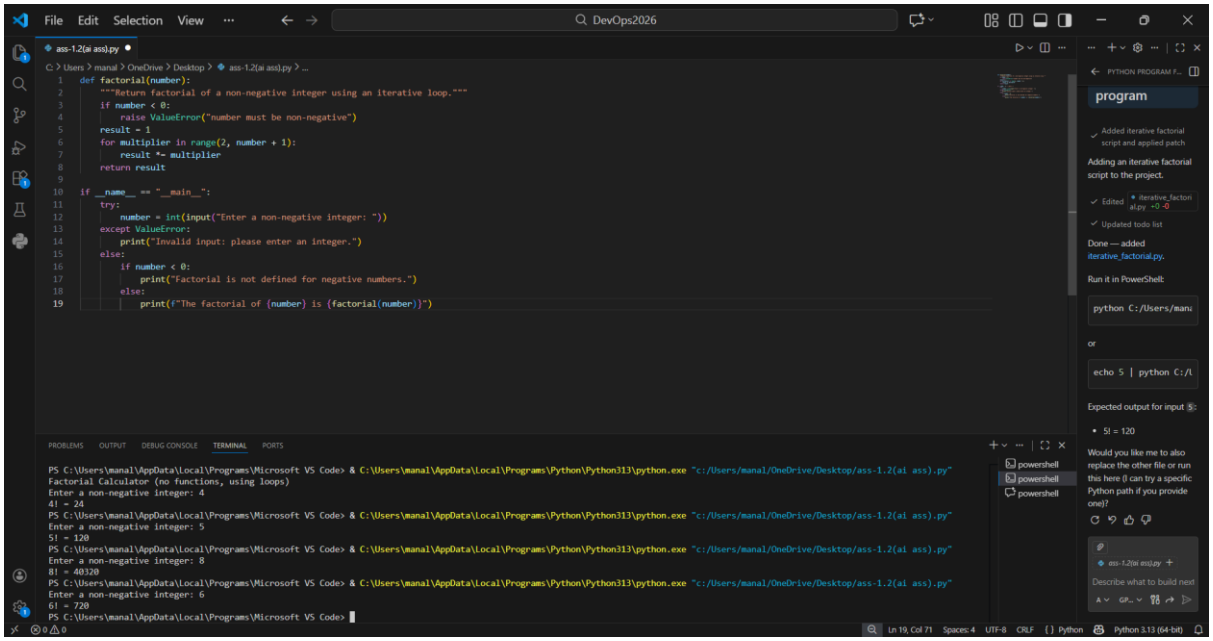
Conclusion

The procedural approach is useful for quick, small programs or beginner-level tasks where speed is more important than structure. However, the modular approach is clearly superior for real-world applications. Function-based programs improve readability, reusability, maintainability, and debugging efficiency.

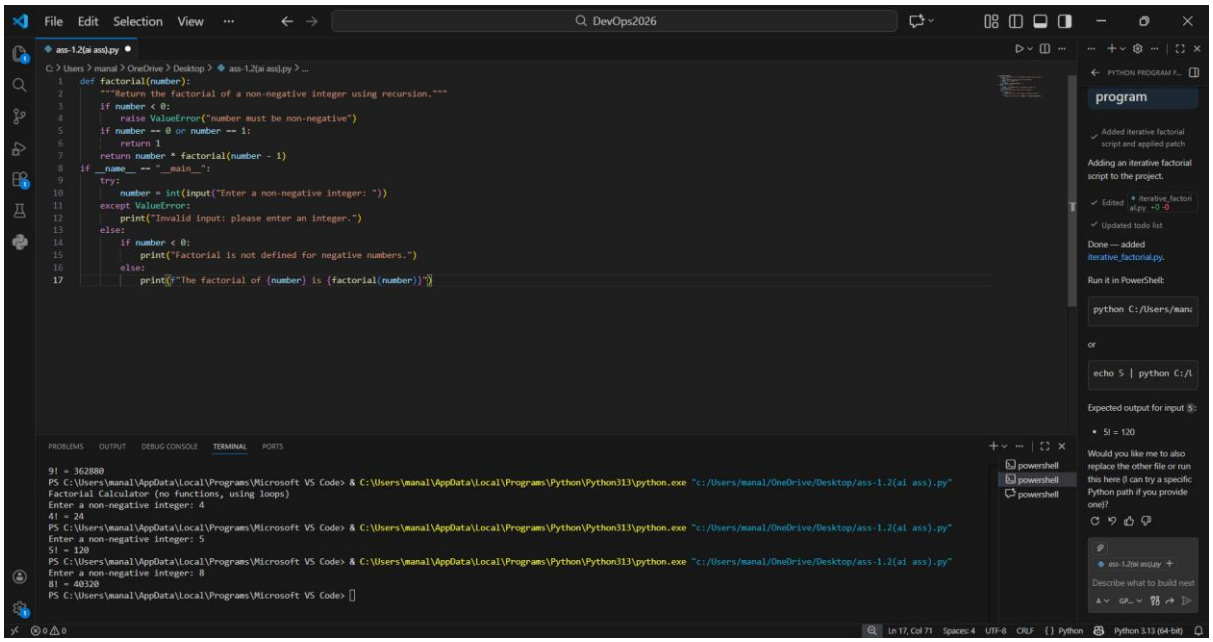
From an AI-assisted coding perspective, modular design also reduces dependency risk because developers are encouraged to understand and validate each function rather than blindly accepting AI-generated output. Therefore, for scalable and professional software development, the modular approach is strongly recommended.

TASK 5: Iterative vs Recursive AI Thinking

Iterative:


File Edit Selection View ···
DevOps2026
ass-1.2(ai ass).py
C: > Users > manal > OneDrive > Desktop > ass-1.2(ai ass).py > ...
def factorial(number):
"""Return factorial of a non-negative integer using an iterative loop."""
if number < 0:
raise ValueError("number must be non-negative")
result = 1
for multiplier in range(2, number + 1):
result *= multiplier
return result
if __name__ == "__main__":
try:
number = int(input("Enter a non-negative integer: "))
except ValueError:
print("Invalid input: please enter an integer.")
else:
if number < 0:
print("Factorial is not defined for negative numbers.")
else:
print(f"The factorial of {number} is {factorial(number)}")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\manal\AppData\Local\Programs\Microsoft VS Code> & C:\Users\manal\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/manal/OneDrive/Desktop/ass-1.2(ai ass).py"
Factorial Calculator (no functions, using loops)
Enter a non-negative integer: 4
4! = 24
PS C:\Users\manal\AppData\Local\Programs\Microsoft VS Code> & C:\Users\manal\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/manal/OneDrive/Desktop/ass-1.2(ai ass).py"
Enter a non-negative integer: 5
5! = 120
PS C:\Users\manal\AppData\Local\Programs\Microsoft VS Code> & C:\Users\manal\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/manal/OneDrive/Desktop/ass-1.2(ai ass).py"
Enter a non-negative integer: 8
8! = 40320
PS C:\Users\manal\AppData\Local\Programs\Microsoft VS Code> & C:\Users\manal\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/manal/OneDrive/Desktop/ass-1.2(ai ass).py"
Enter a non-negative integer: 6
6! = 720
PS C:\Users\manal\AppData\Local\Programs\Microsoft VS Code>
powershell
powershell
powershell
program
Added iterative factorial script and applied patch
Adding an iterative factorial script to the project.
Updated todo list
Done — added iterative_factorial.py.
Run it in PowerShell:
python C:/Users/manal
or
echo 5 | python C:/l
Expected output for input 5:
5! = 120
Would you like me to also replace the other file or run this here (I can try a specific Python path if you provide one)?
Describe what to build next
Ln 19, Col 71 Spaces: 4 UTF-8 CRLF { } Python Python 3.13 (64-bit)

Recursive:


File Edit Selection View ···
DevOps2026
ass-1.2(ai ass).py
C: > Users > manal > OneDrive > Desktop > ass-1.2(ai ass).py > ...
def factorial(number):
"""Return the factorial of a non-negative integer using recursion."""
if number < 0:
raise ValueError("number must be non-negative")
if number == 0 or number == 1:
return 1
return number * factorial(number - 1)
if __name__ == "__main__":
try:
number = int(input("Enter a non-negative integer: "))
except ValueError:
print("Invalid input: please enter an integer.")
else:
if number < 0:
print("Factorial is not defined for negative numbers.")
else:
print(f"The factorial of {number} is {factorial(number)}")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
9! = 362880
PS C:\Users\manal\AppData\Local\Programs\Microsoft VS Code> & C:\Users\manal\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/manal/OneDrive/Desktop/ass-1.2(ai ass).py"
Factorial Calculator (no functions, using loops)
Enter a non-negative integer: 4
4! = 24
PS C:\Users\manal\AppData\Local\Programs\Microsoft VS Code> & C:\Users\manal\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/manal/OneDrive/Desktop/ass-1.2(ai ass).py"
Enter a non-negative integer: 5
5! = 120
PS C:\Users\manal\AppData\Local\Programs\Microsoft VS Code> & C:\Users\manal\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/manal/OneDrive/Desktop/ass-1.2(ai ass).py"
Enter a non-negative integer: 8
8! = 40320
PS C:\Users\manal\AppData\Local\Programs\Microsoft VS Code>
powershell
powershell
powershell
program
Added iterative factorial script and applied patch
Adding an iterative factorial script to the project.
Updated todo list
Done — added iterative_factorial.py.
Run it in PowerShell:
python C:/Users/manal
or
echo 5 | python C:/l
Expected output for input 5:
5! = 120
Would you like me to also replace the other file or run this here (I can try a specific Python path if you provide one)?
Describe what to build next
Ln 17, Col 71 Spaces: 4 UTF-8 CRLF { } Python Python 3.13 (64-bit)