

## 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

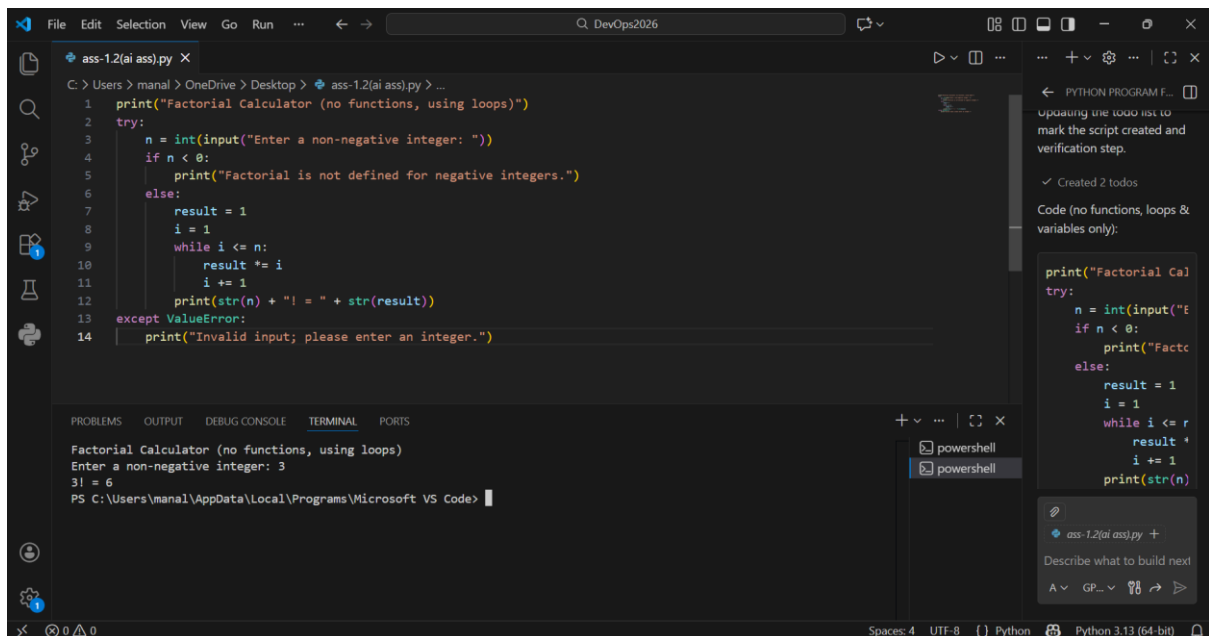
Screenshots to Take

Copilot extension page

GitHub sign-in success

Copilot enabled in VS Code

### TASK 1: AI-Generated Logic (NO FUNCTIONS)



### Task-2: Optimized version

The screenshot shows a Visual Studio Code editor with a Python file named `ass-1.2(ai ass).py`. The code is a simple factorial calculator using loops and exception handling. The terminal shows the program being executed twice: first with input 9 resulting in 362880, and then with input 4 resulting in 24. The right sidebar shows the Python program's structure and a 'Call the function from main' prompt.

```
1 try:
2     n = int(input("Enter a non-negative integer: "))
3 except ValueError:
4     print("Invalid input: please enter an integer.")
5 else:
6     if n < 0:
7         print("Factorial is not defined for negative numbers.")
8     else:
9         result = 1
10        for i in range(2, n + 1):
11            result *= i
12        print(f"The factorial of {n} is {result}")
```

Terminal output:

```
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: 9
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>
```

### TASK 3: Modular Design (WITH FUNCTIONS)

The screenshot shows a Visual Studio Code editor with a Python file named `ass-1.2(ai ass).py`. The code is a modular factorial calculator using functions and exception handling. The terminal shows the program being executed twice: first with input 9 resulting in 362880, and then with input 5 resulting in 120. The right sidebar shows the Python program's structure and a 'Call the function from main' prompt.

```
1 """Factorial calculator with a reusable function and clear names.
2 This file defines 'calculate_factorial(number)' and calls it from 'main()'".
3 """
4 def calculate_factorial(number: int) -> int:
5     """Return factorial of a non-negative integer 'number'.
6     Raises ValueError for negative input.
7     """
8     if number < 0:
9         raise ValueError("Factorial is not defined for negative integers.")
10    product = 1
11    for value in range(2, number + 1):
12        product *= value
13    return product
14 def main() -> None:
15     try:
16         raw = input("Enter a non-negative integer: ").strip()
17         number = int(raw)
18     except Exception:
19         print("Invalid input: please enter an integer.")
20         return
21     try:
22         result = calculate_factorial(number)
23     except ValueError as exc:
24         print(exc)
25     else:
26         print(f"{number}! = {result}")
27 if __name__ == "__main__":
28     main()
```

Terminal output:

```
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: 9
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: 5
5! = 120
PS C:\Users\manal\AppData\Local\Programs\Microsoft VS Code>
```

### 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:**

The screenshot shows a VS Code editor with a file named `ass-1.2(ai ass).py`. The code defines a function `factorial(number)` that calculates the factorial of a non-negative integer using an iterative loop. The function includes error handling for non-negative integers and negative numbers. The `__main__` block prompts the user to enter a non-negative integer and prints the result.

```
1 def factorial(number):
2     """Return factorial of a non-negative integer using an iterative loop."""
3     if number < 0:
4         raise ValueError("number must be non-negative")
5     result = 1
6     for multiplier in range(2, number + 1):
7         result *= multiplier
8     return result
9
10 if __name__ == "__main__":
11     try:
12         number = int(input("Enter a non-negative integer: "))
13     except ValueError:
14         print("Invalid input: please enter an integer.")
15     else:
16         if number < 0:
17             print("Factorial is not defined for negative numbers.")
18         else:
19             print("The factorial of (number) is (factorial(number)).")
```

The terminal output shows the execution of the script for inputs 4, 5, 120, and 8, resulting in factorials of 24, 120, 40320, and 40320 respectively.

On the right, the 'PYTHON PROGRAM' sidebar shows the file `ass-1.2(ai ass).py` is added to the project. The 'Run in PowerShell' button is visible.

## Recursive:

The screenshot shows a VS Code editor with a file named `ass-1.2(ai ass).py`. The code defines a function `factorial(number)` that calculates the factorial of a non-negative integer using recursion. The function includes error handling for non-negative integers and negative numbers. The `__main__` block prompts the user to enter a non-negative integer and prints the result.

```
1 def factorial(number):
2     """Return the factorial of a non-negative integer using recursion."""
3     if number < 0:
4         raise ValueError("number must be non-negative")
5     if number == 0 or number == 1:
6         return 1
7     return number * factorial(number - 1)
8
9 if __name__ == "__main__":
10     try:
11         number = int(input("Enter a non-negative integer: "))
12     except ValueError:
13         print("Invalid input: please enter an integer.")
14     else:
15         if number < 0:
16             print("Factorial is not defined for negative numbers.")
17         else:
18             print("The factorial of (number) is (factorial(number)).")
```

The terminal output shows the execution of the script for inputs 4, 5, 120, and 8, resulting in factorials of 24, 120, 40320, and 40320 respectively.

On the right, the 'PYTHON PROGRAM' sidebar shows the file `ass-1.2(ai ass).py` is added to the project. The 'Run in PowerShell' button is visible.