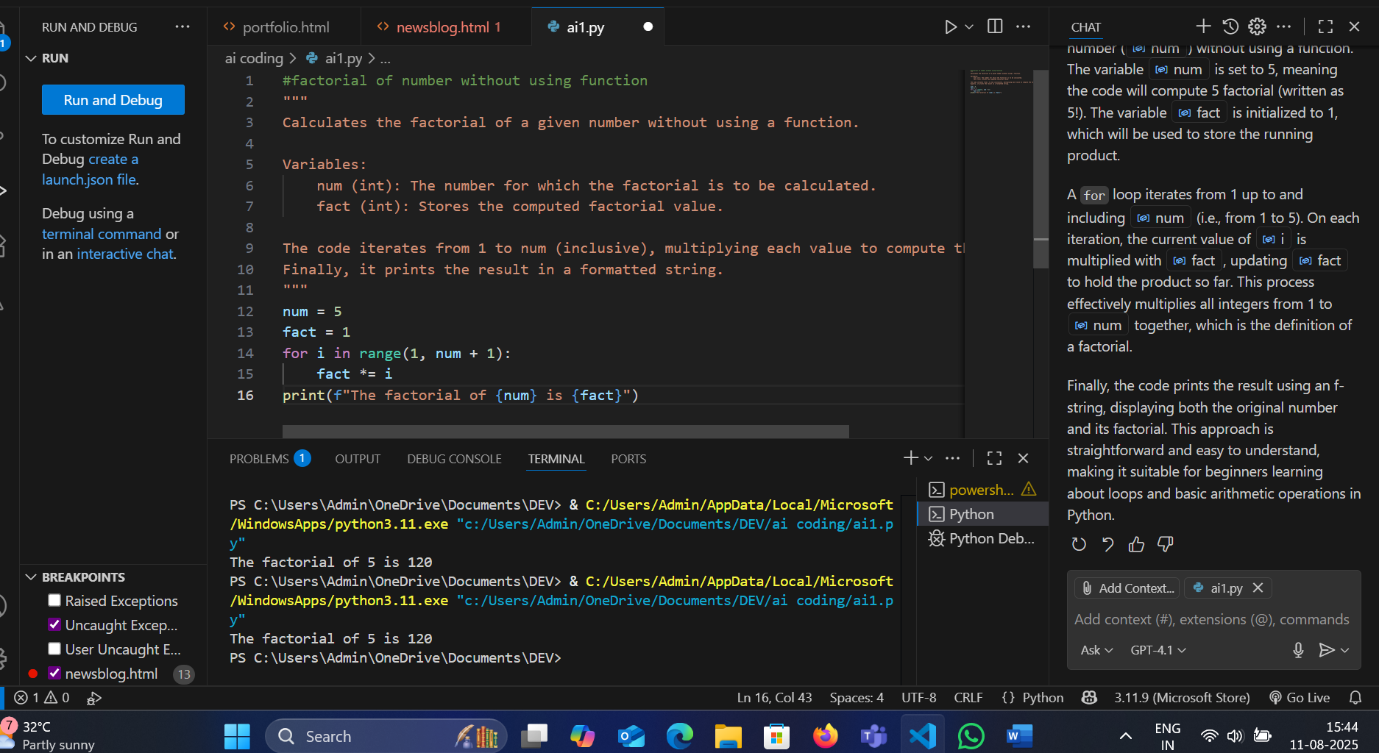
**Assingment-1**

**Python using AI Factorial of Number**

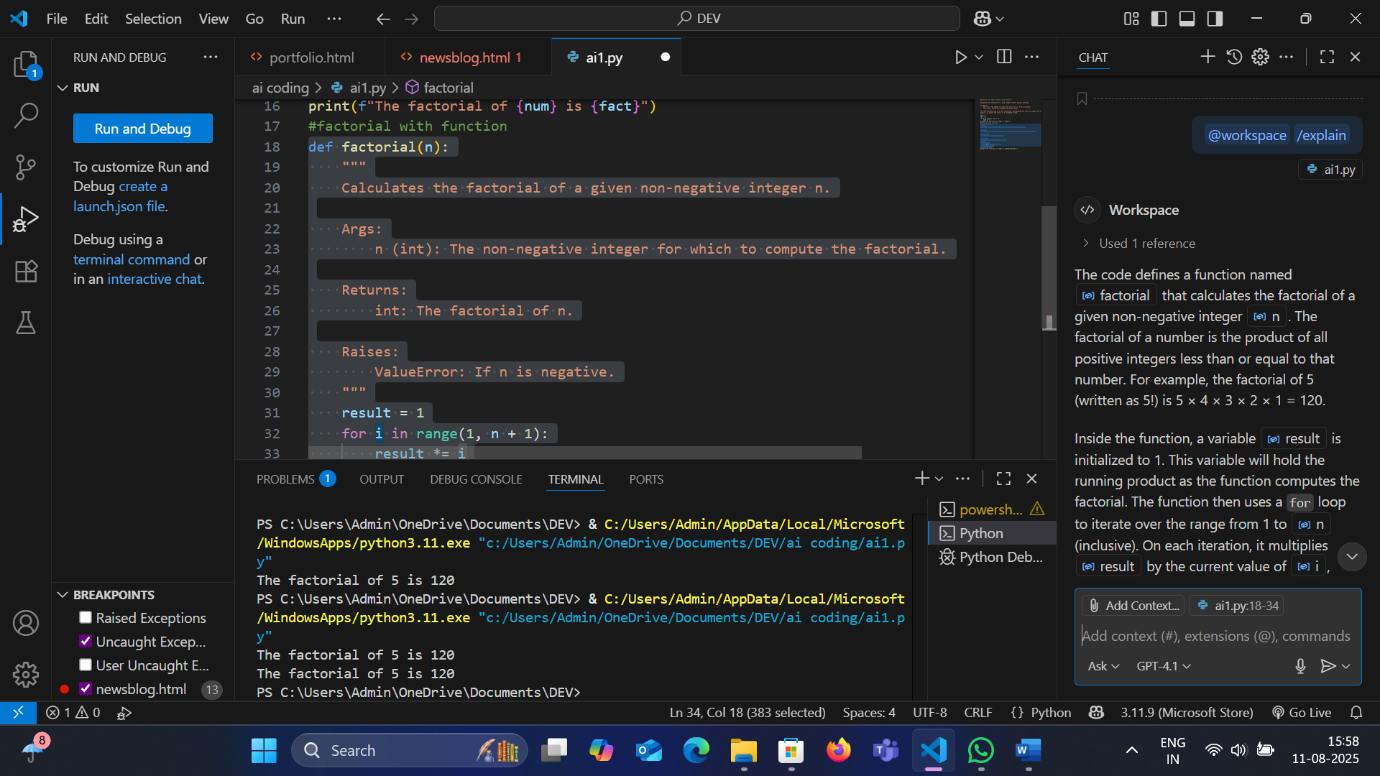
* **Task-1**

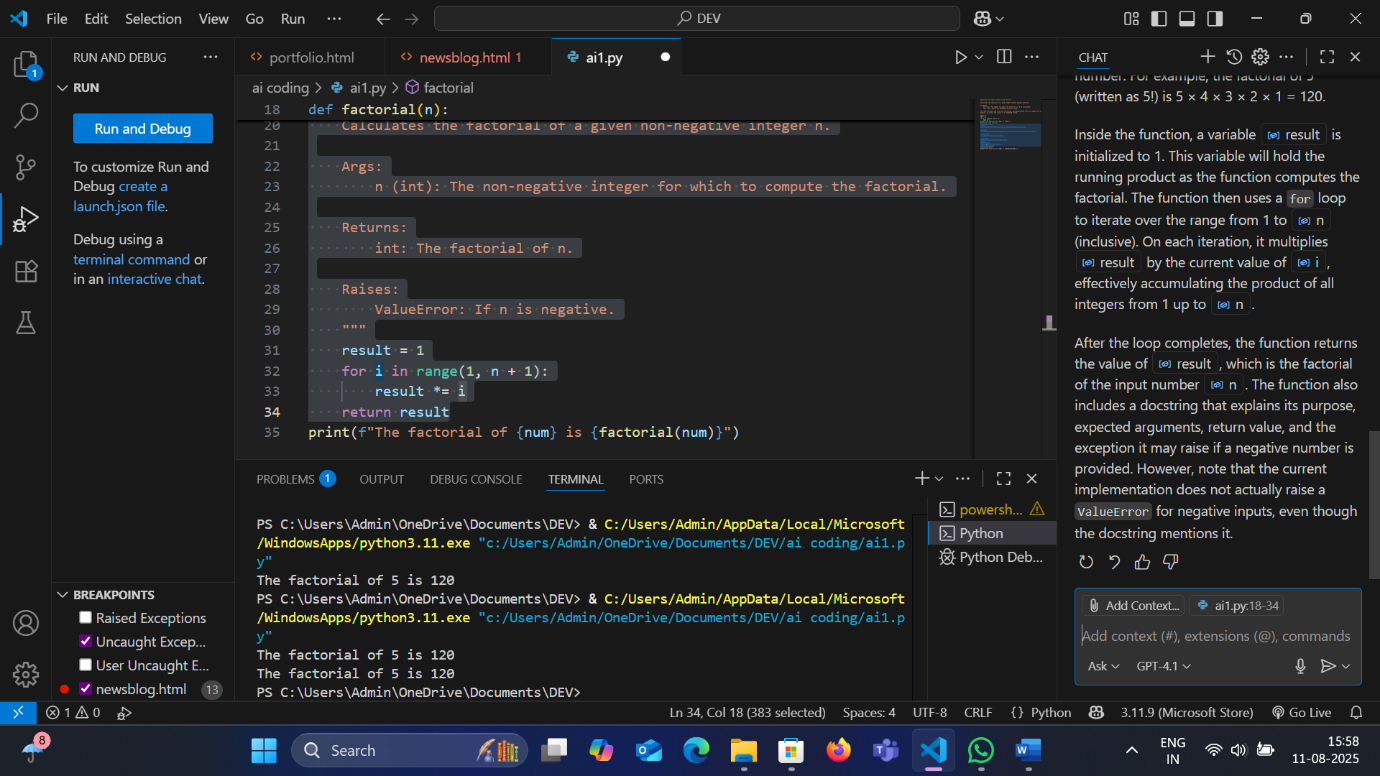
**Without using the function**

****

* **Task-3**

**With Functions**

****

****

* **Task 4**

**Here are the key differences between the two factorial implementations in your file:**

**1. Without Using a Function**

* **Approach: Uses a simple for loop directly in the main code.**
* **Variables: Uses**[**num**](vscode-file://vscode-app/c:/Users/Admin/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**and**[**fact**](vscode-file://vscode-app/c:/Users/Admin/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**.**
* **Logic: Iteratively multiplies numbers from 1 to**[**num**](vscode-file://vscode-app/c:/Users/Admin/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**.**
* **Output: Prints the result directly after calculation.**

**2. With Using a Function**

* **Approach: Uses a function named**[**factorial**](vscode-file://vscode-app/c:/Users/Admin/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**.**
* **Two Versions:**
  + **Iterative Version:**
    - **Uses a for loop inside the function.**
    - **Returns the result after the loop.**
  + **Recursive Version:**
    - **Calls itself with**[**n-1**](vscode-file://vscode-app/c:/Users/Admin/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**until**[**n**](vscode-file://vscode-app/c:/Users/Admin/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**is 0.**
    - **Returns 1 when**[**n**](vscode-file://vscode-app/c:/Users/Admin/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**is 0 (base case).**
    - **Returns**[**n \* factorial(n-1)**](vscode-file://vscode-app/c:/Users/Admin/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**otherwise.**
* **Output: The function is called and its result is printed.**

**Summary Table**

| **Aspect** | **Without Function** | **With Function (Iterative)** | **With Function (Recursive)** |
| --- | --- | --- | --- |
| **Structure** | **Inline code** | **Function definition** | **Function definition** |
| **Approach** | **Iterative (for loop)** | **Iterative (for loop)** | **Recursive** |
| **Reusability** | **Not reusable** | **Reusable** | **Reusable** |
| **Readability** | **Simple, direct** | **Modular, clear** | **Elegant, concise** |
| **Base Case** | **N/A** | **N/A** | **Yes (**[**n == 0**](vscode-file://vscode-app/c:/Users/Admin/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**)** |

**Both function versions improve code reusability and clarity, while the recursive version demonstrates a different programming technique.**

* **Task 5**

**Here are the main differences between recursion and iteration (as shown in your factorial examples):**

**Recursion**

* **Definition: A function calls itself to solve smaller instances of the same problem.**
* **Example:**

**def factorial(n):**

**if n == 0:**

**return 1**

**else:**

**return n \* factorial(n-1)**

* **Base Case: Must have a condition to stop recursion (e.g.,**[**if n == 0**](vscode-file://vscode-app/c:/Users/Admin/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**).**
* **Memory Usage: Uses more memory due to function call stack.**
* **Readability: Can be more elegant and closer to mathematical definitions.**
* **Performance: May be slower and can cause stack overflow for large inputs.**

**Iteration**

* **Definition: Uses loops (for, while) to repeat operations.**
* **Example**

**def factorial(n):**

**result = 1**

**for i in range(1, n + 1):**

**result \*= i**

**return result**

* **Base Case: Handled by loop conditions.**
* **Memory Usage: More efficient, uses less memory (no extra call stack).**
* **Readability: Sometimes more straightforward, especially for simple tasks.**
* **Performance: Generally faster and safer for large inputs.**

**Summary:**

* **Use recursion for problems that are naturally recursive or when code clarity is more important than performance.**
* **Use iteration for better performance and memory efficiency, especially with large datasets.**