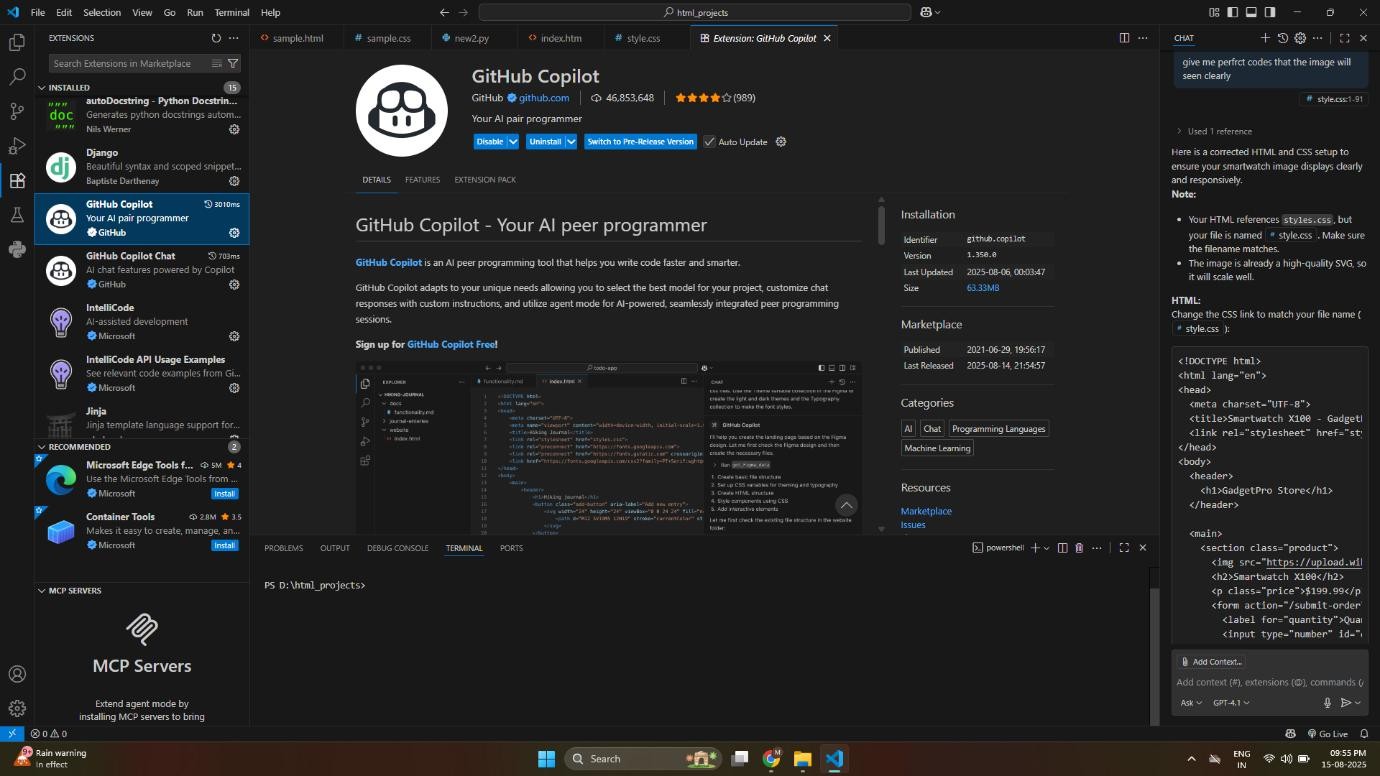
ASSIGNMENT-1

Task: 0

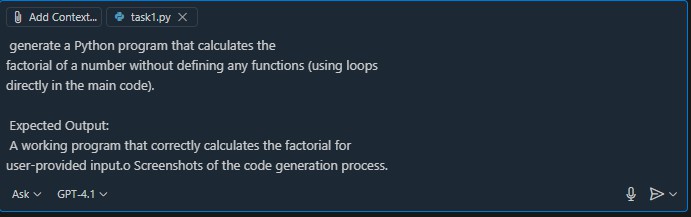
* Install and configure GitHub Copilot in VS Code



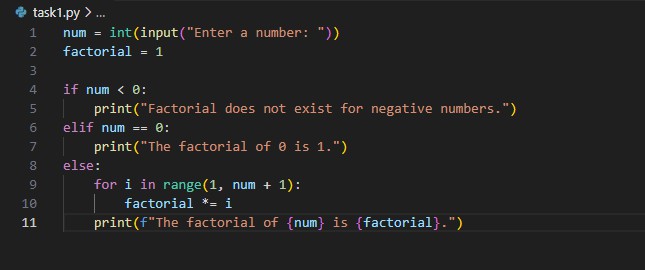
Task1:

* Use GitHub Copilot to generate a Python program that calculates the factorial of a number without defining any functions ( directly in the main code).

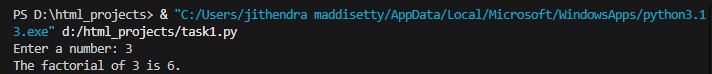
My Prompt:



Code:



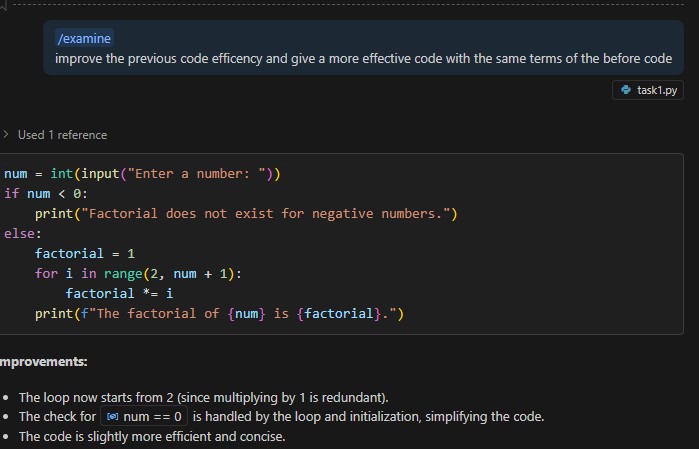
Output:



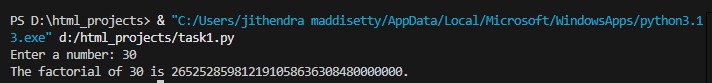
Task2:

* Use GitHub Copilot to generate a Python program that calculates the factorial of a number without defining any functions(using loops directly in the main code)

CODE:



Output:



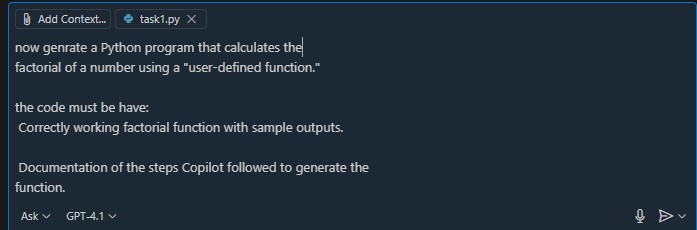
Improvements

* The loop now starts from2(since multiplying by 1 Is redundant).
* The check for[num==0](vscode-file://vscode-app/c:/Users/jithendra%20maddisetty/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)ishandledbytheloopandinitialization,simplifying the code.
* The code is slightly more efficient and concise.

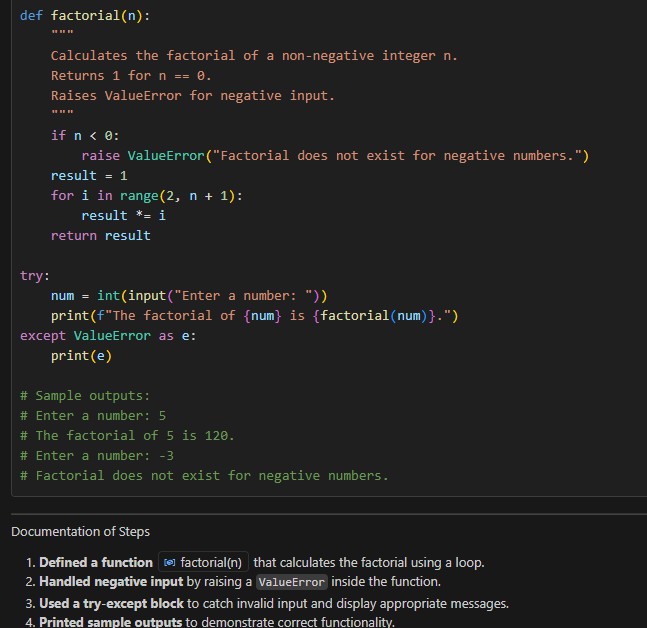
Task3:

Use GitHub Copilot to generate a Python program that calculates the factorial of a number using a user-defined function.

My Prompt:



CODE:



Output:



Documentation of Steps:

Defined a function factorial (n)that calculates the factorial using a loop. Handled negative input by raising a Value Error inside the function.

Used a try- except block to catch invalid input and display appropriate messages. Printedsampleoutputstodemonstratecorrectfunctionalitites

Task4:

Comparative Analysis–With vs Without Functions

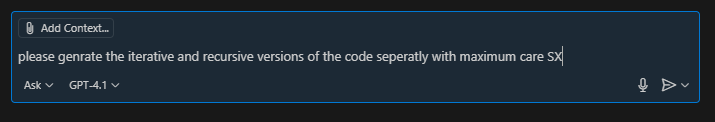
|  |  |  |
| --- | --- | --- |
| Aspect | Without  Functions | With  Functions |
| **Logic** | Logics written directly in the main script ;control  flows sequential | Logic is encapsulated in a separate function(factorial(n)) and invoked when needed. |
| **Reusability** | Cannot be reused without copying the code into another program. | Codeisreusable—call  factorial(n)from any wherein the same or another program |
| **Readabilitys Structure** | All logics’ in one block ;larger programs become harder to follow. | Codeismodular —corelogicis separate from input/output, improving clarity. |
| **ErrorHandling** | Minimal errorhandling,usually tied directly to input validation | Robust errorhandling can be built into the function(e.g., raise ValueError)for better  control. |
| **Execution Flow** | Program executes line-by-line from input to output without modular jumps. | Main section handles input/output,callsthe  function,receivestheresult,  anddisplaysit. |

|  |  |  |
| --- | --- | --- |
| **Maintainability** | Updating logic requires editing it every where it appears | Updating logic in one function automatically updates all uses across the program |
| **Testing** | Logic is hard to test separately from user input/output. | Function can be tested independently without  Involving user inter action |

Task5:

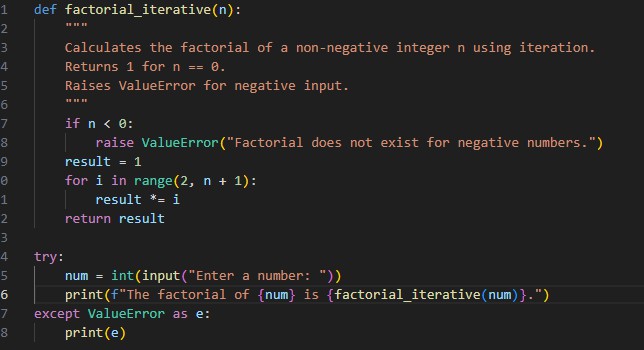
Iterative vs Recursive Factorial

Prompt:

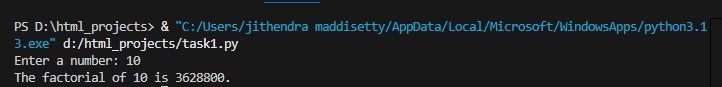


*Iterative:*

Code:

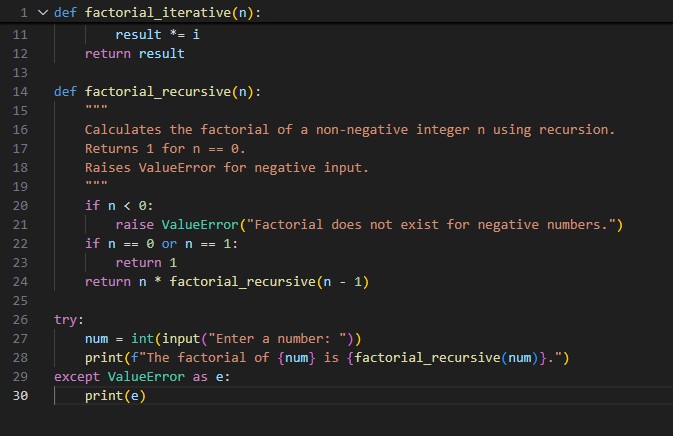


Output:

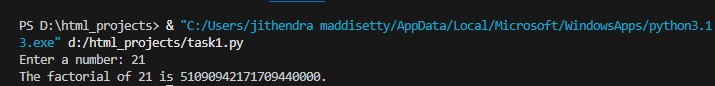


*Recursive:*

Code:



Output:



Comparison

|  |  |  |
| --- | --- | --- |
| Feature | Iterative | Recursive |
| **logic** | Uses a loop to multiply numbers from 2 to n | Callsitselfwithn-1untilbase caser reached |
| **TimeComplexity** | O(n) | O(n) |
| **SpaceComplexity** | O(1) | O(n)(due to calls tack) |
| **MemoryUsage** | Low | Higher |
| **Risk** | Nostackover flow | Risk of Recursion Error for larger |
| **Readability** | Less mathematics l, but straightforward | Matches mathematical definition, more elegant |
| **Performance** | Faster for larger | Slower due to function call overhead |