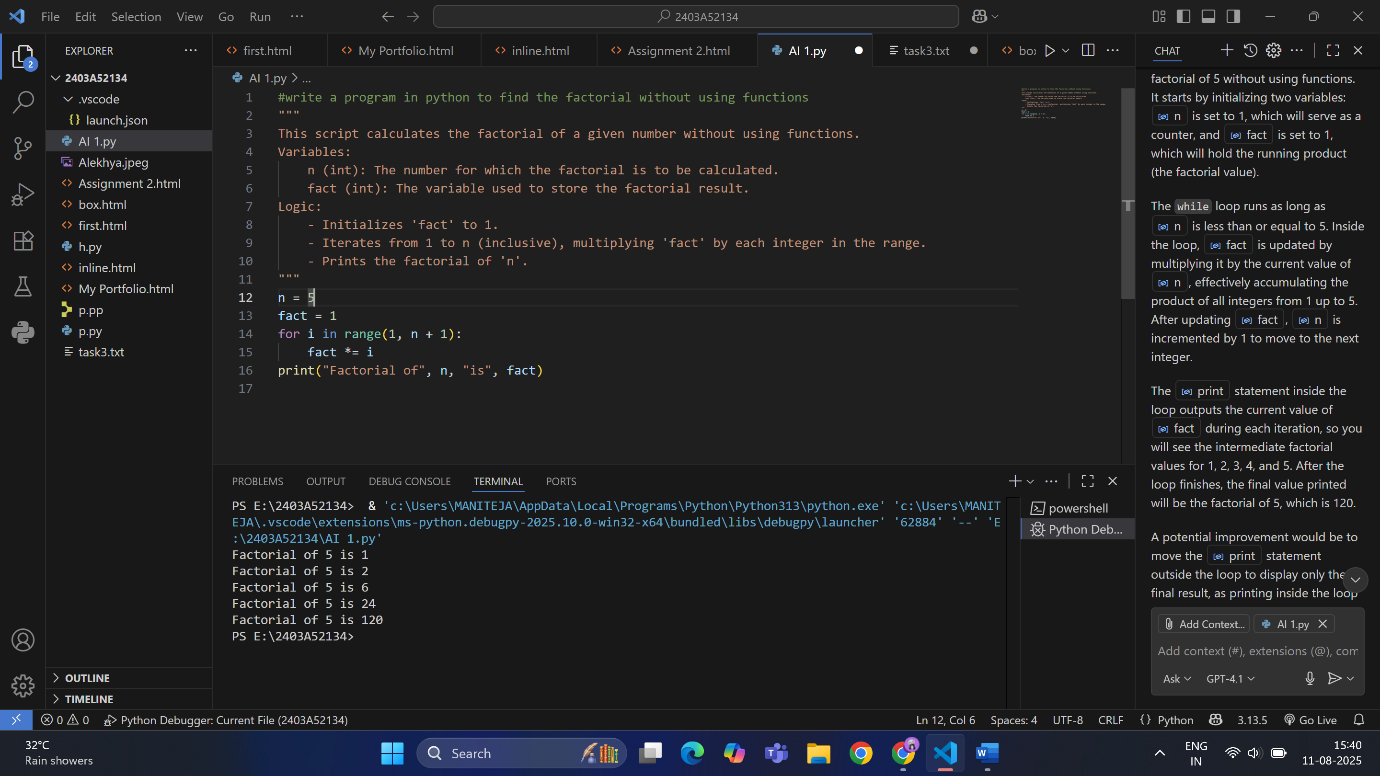
HTNO:2403A52134

Assignment-1

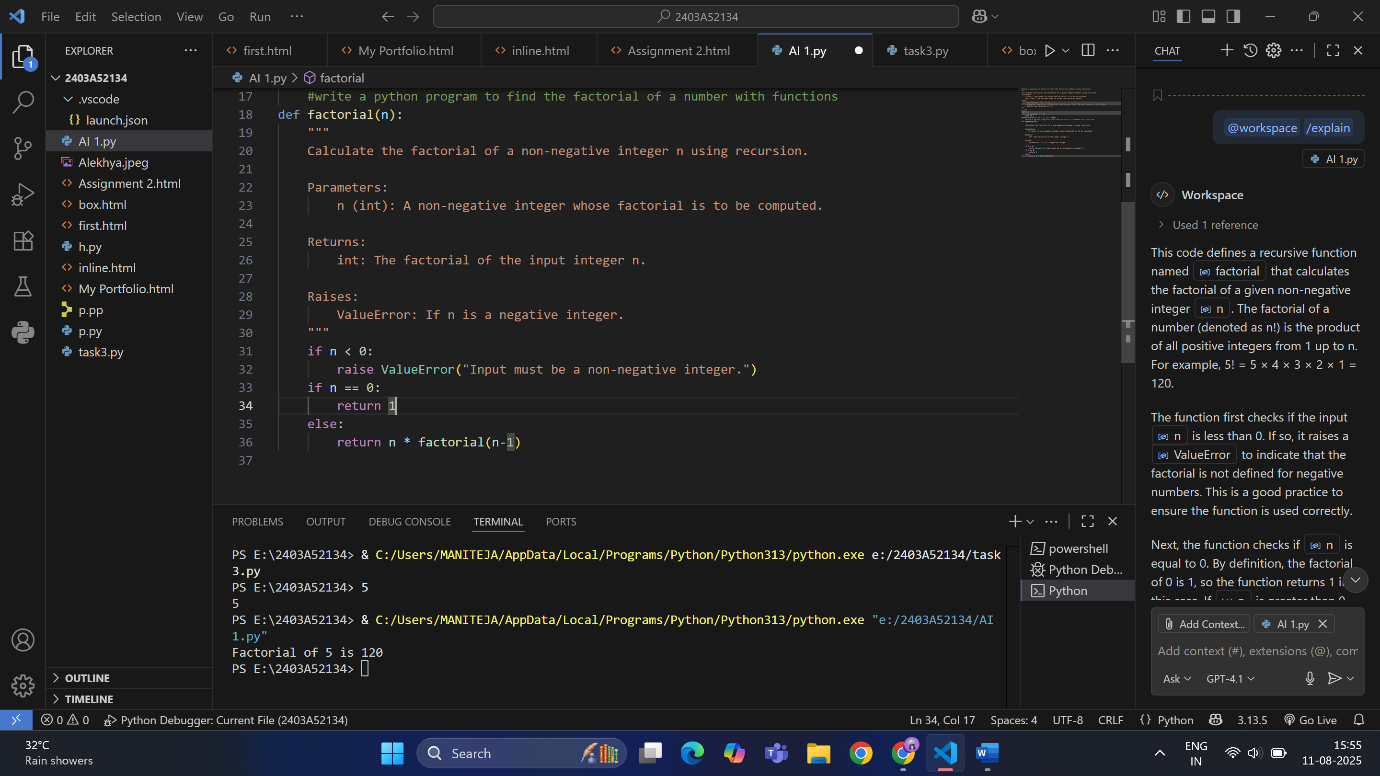
Task-1:

To find factorial of a number without using functions.

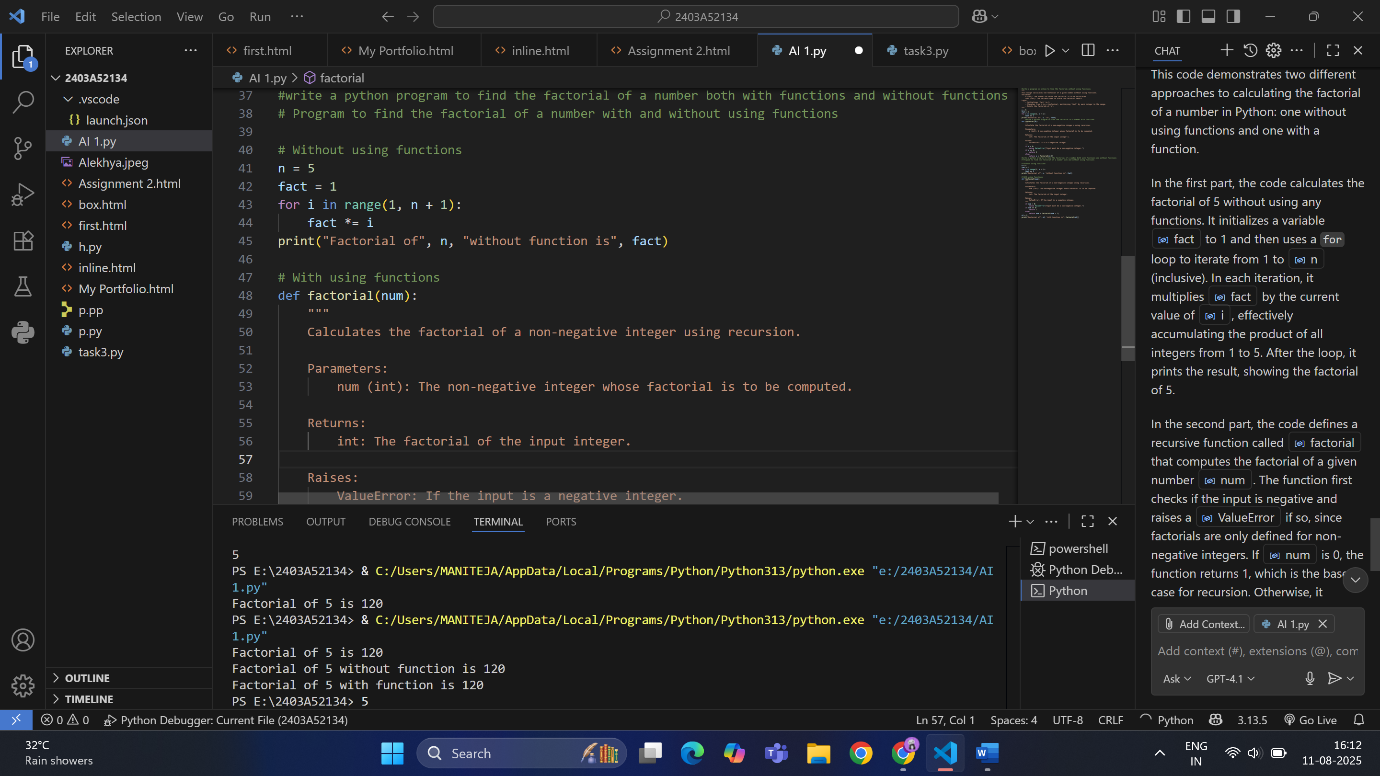


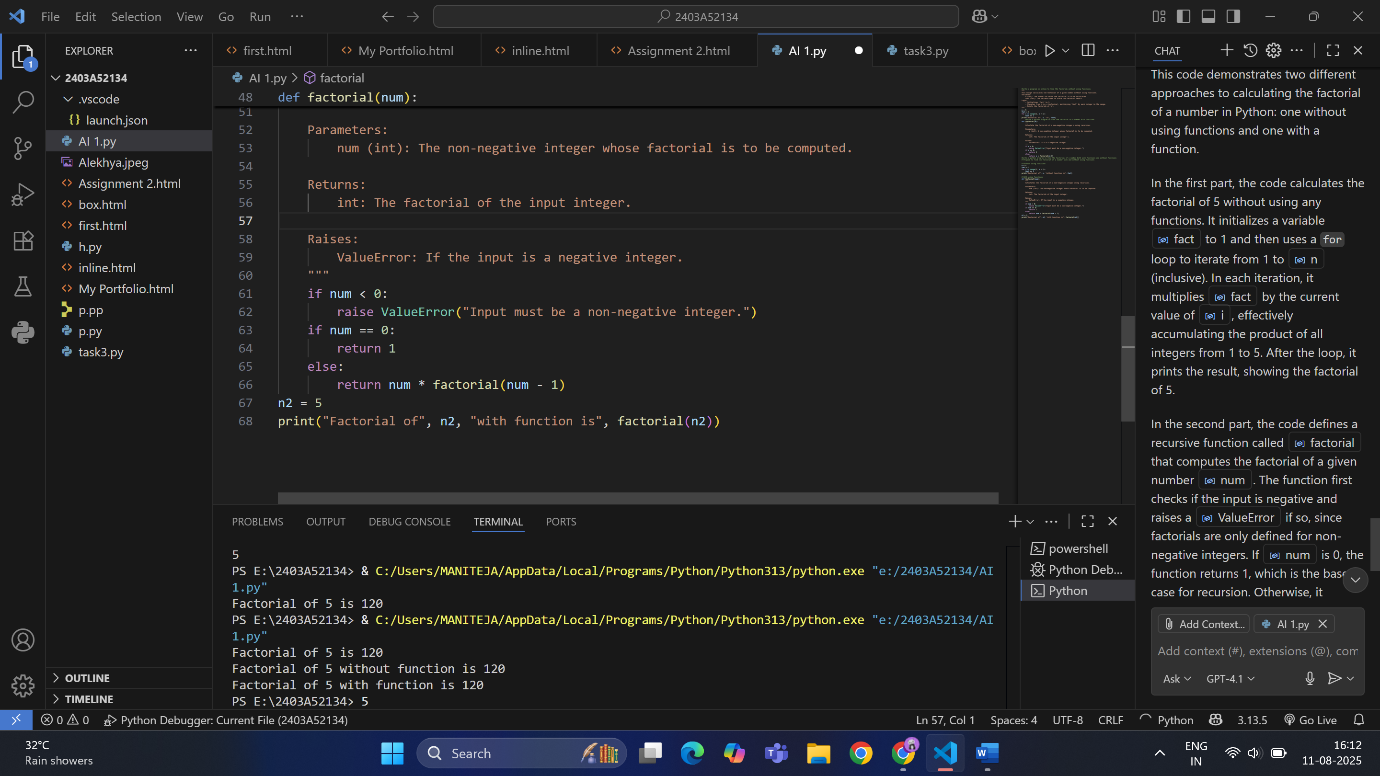
Task-3:

To find factorial of a number with functions.



Task-4:





Differences:

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  and fact.

Logic: Iteratively multiplies numbers from 1 to num.

Output: Prints the result directly after calculation.

2. With Using a Function

Approach: Uses a function named factorial.

Two Versions:

Iterative Version:

Uses a for loop inside the function.

Returns the result after the loop.

Recursive Version:

Calls itself with n-1 until n is 0.

Returns 1 when n is 0 (base case).

Returns n \* factorial(n-1) otherwise.

Output: The function is called and its result is printed.

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).

 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.