**ASSIGNMENT-12**

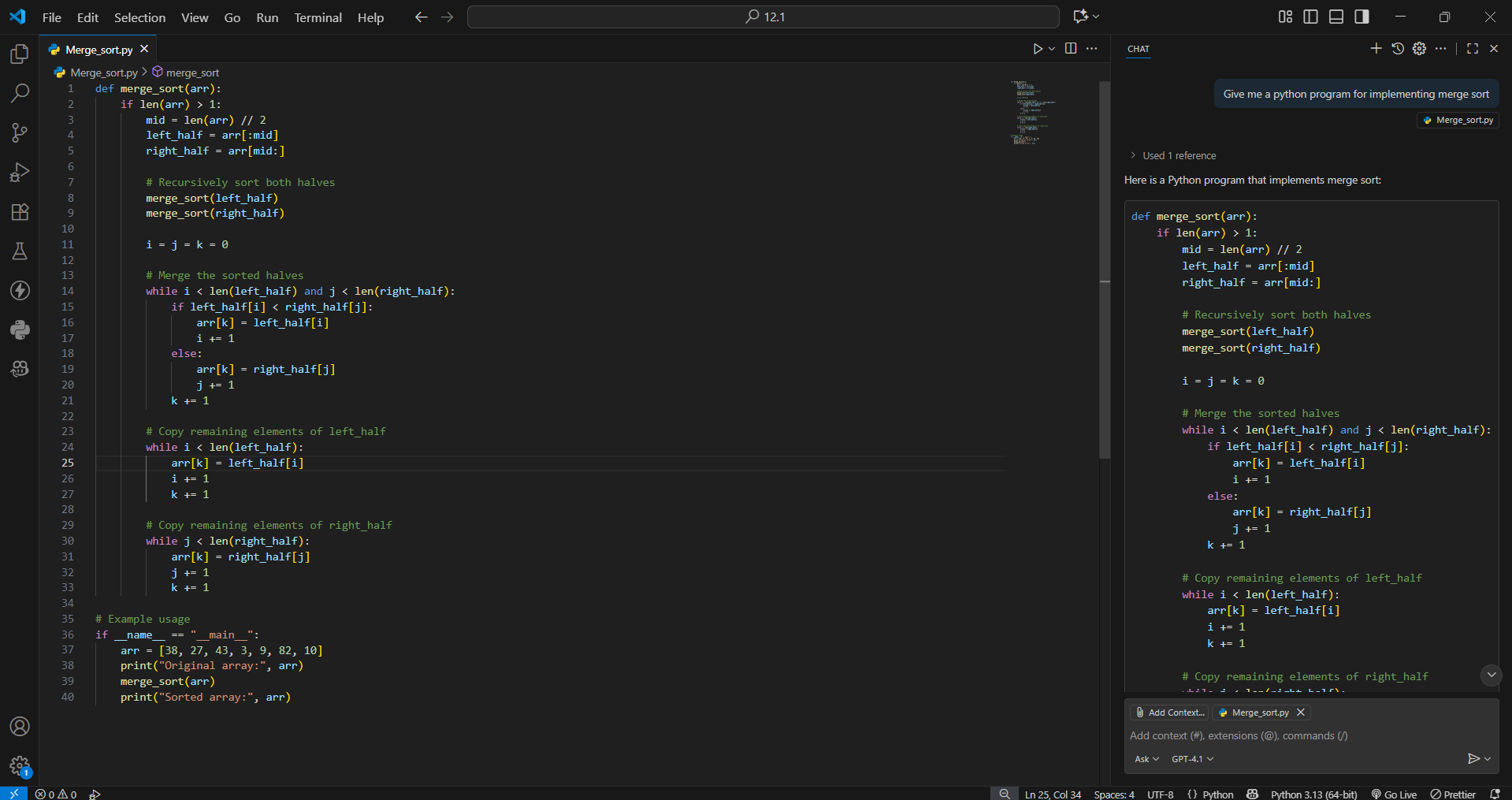
**HNO:2503A52L16  
  
Lab 12: Algorithms with AI Assistance – Sorting, Searching, and Optimizing Algorithms**

**Task Description #1 (Sorting – Merge Sort Implementation):**

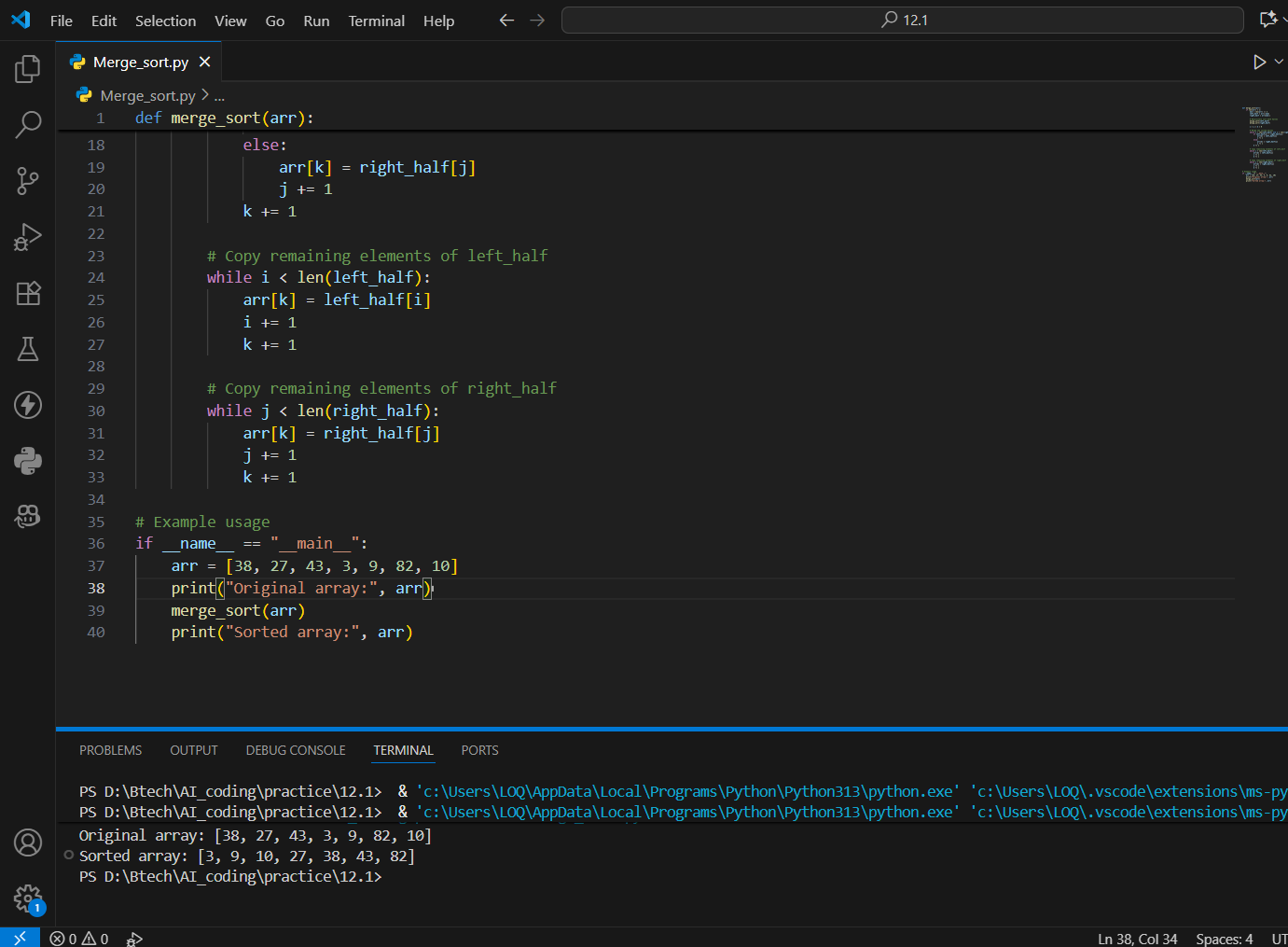
**Task**: Use AI to generate a Python program that implements the Merge Sort algorithm.

**Prompt:** **Give me a python program for implementing merge sort**

**Code:**

****

**Output:**

****

**Observation:**

This program implements the Merge Sort algorithm in Python.

**Function:**

merge\_sort(arr)

**Description:**

The merge\_sort function recursively divides the input list into halves,

sorts each half, and merges them back together in ascending order.

**Time Complexity:**

- Best Case: O(n log n)

- Average Case: O(n log n)

- Worst Case: O(n log n)

**Space Complexity:**

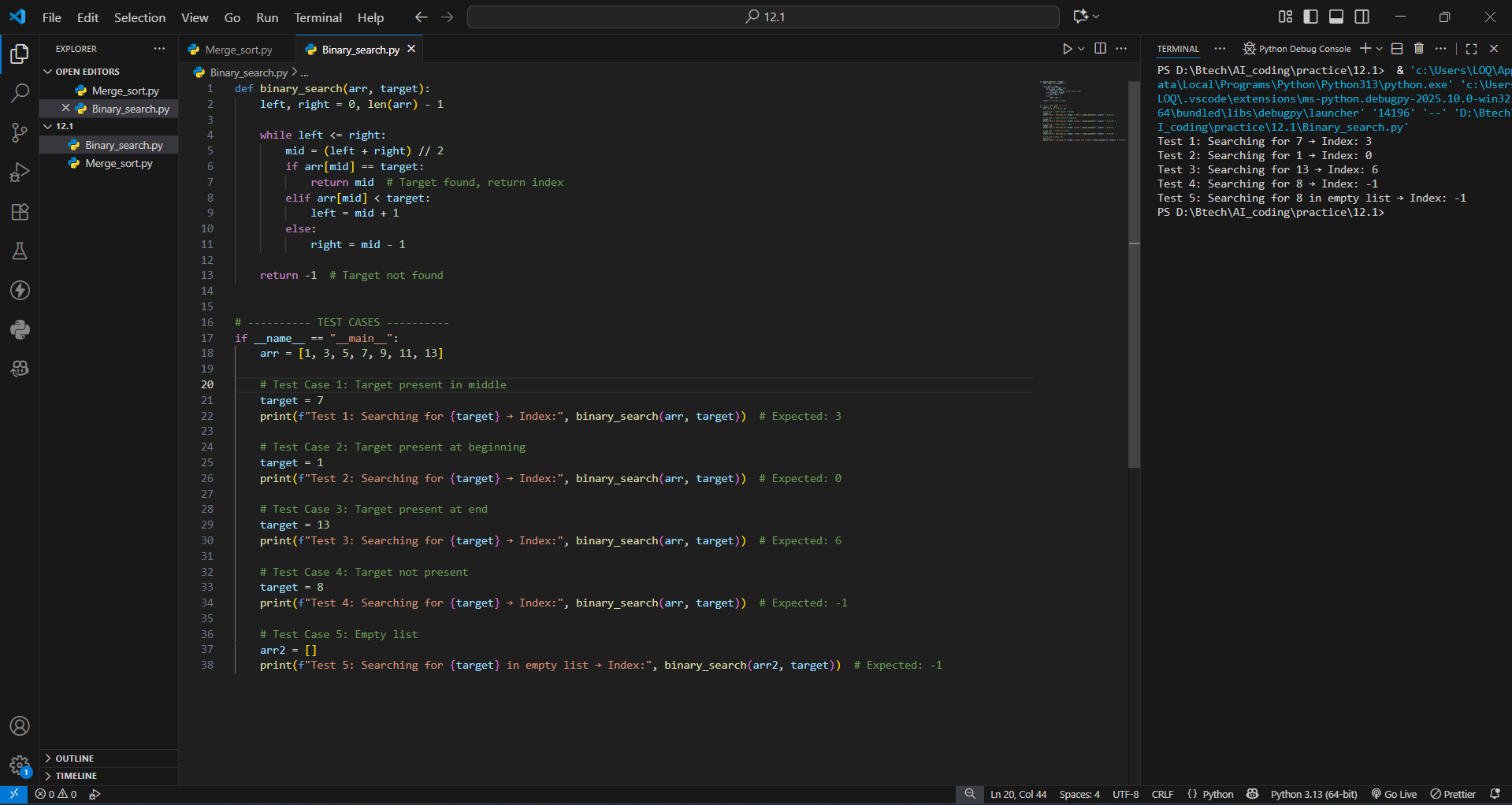
- O(n) (due to the temporary arrays created during merging)

**Task Description #2 (Searching – Binary Search with AI Optimization)**

**Task:** Use AI to create a binary search function that finds a target element in a sorted list.

**Prompt:** give me the python program to implement binary search which finds a target element

**Code & Output:**

****

**Binary Search Implementation**

This program implements the Binary Search algorithm in Python.

**Function:**

binary\_search(arr, target)

**Description:**

The binary\_search function searches for a target element within

a sorted list by repeatedly dividing the search interval in half.

**Time Complexity:**

Best Case: O(1)

Average Case: O(log n)

Worst Case: O(log n)

**Space Complexity:**

O(1) — iterative implementation uses constant extra space.

**Task Description #3** (Real-Time Application – Inventory Management System)

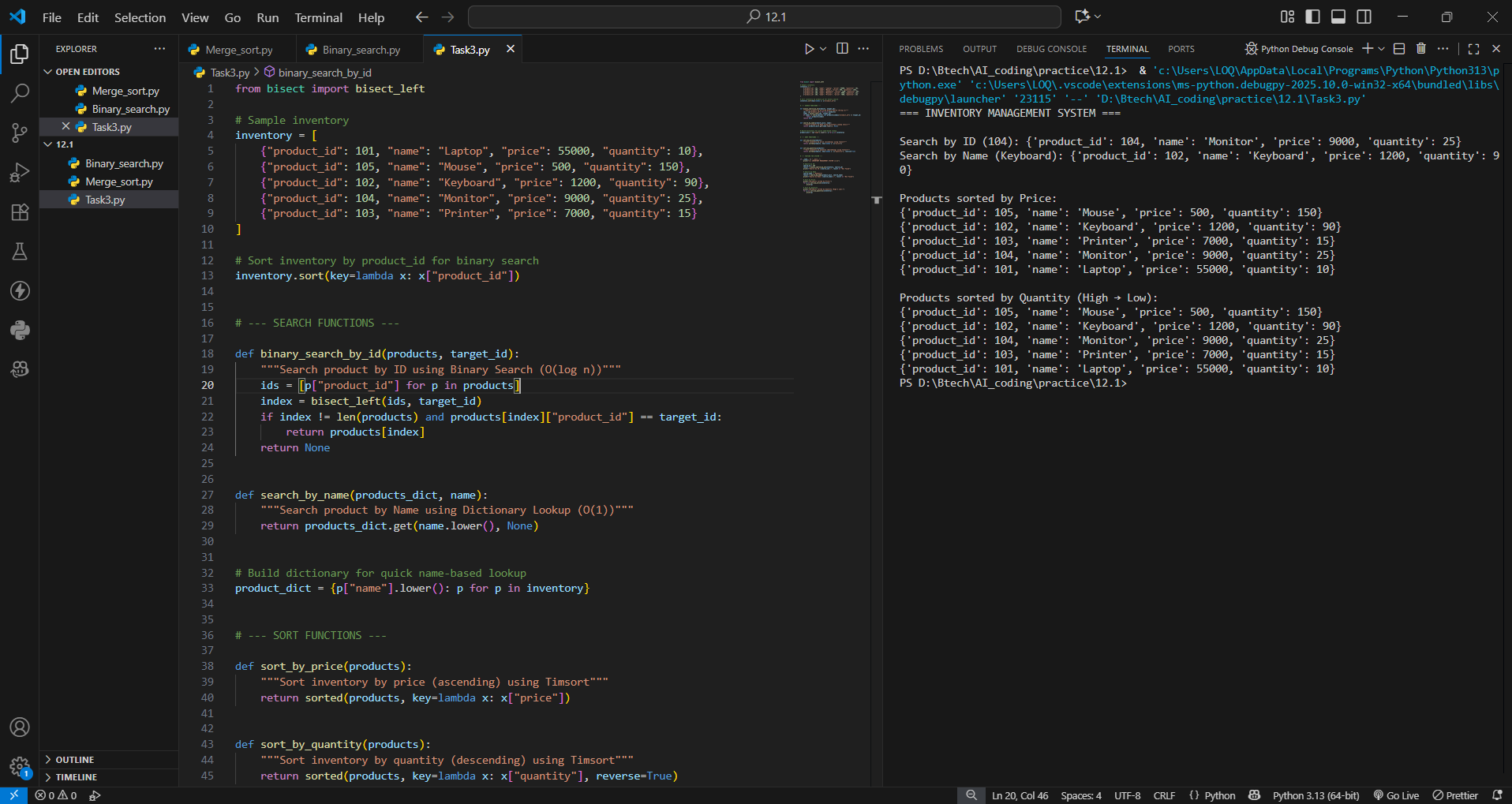
**Scenario**: A retail store’s inventory system contains thousands of products, each with attributes like product ID, name, price, and stock quantity. Store staff need to:  
1. Quickly search for a product by ID or name.  
2. Sort products by price or quantity for stock analysis.

**Task:** **Use AI to suggest the most efficient search and sort algorithms for this use case**

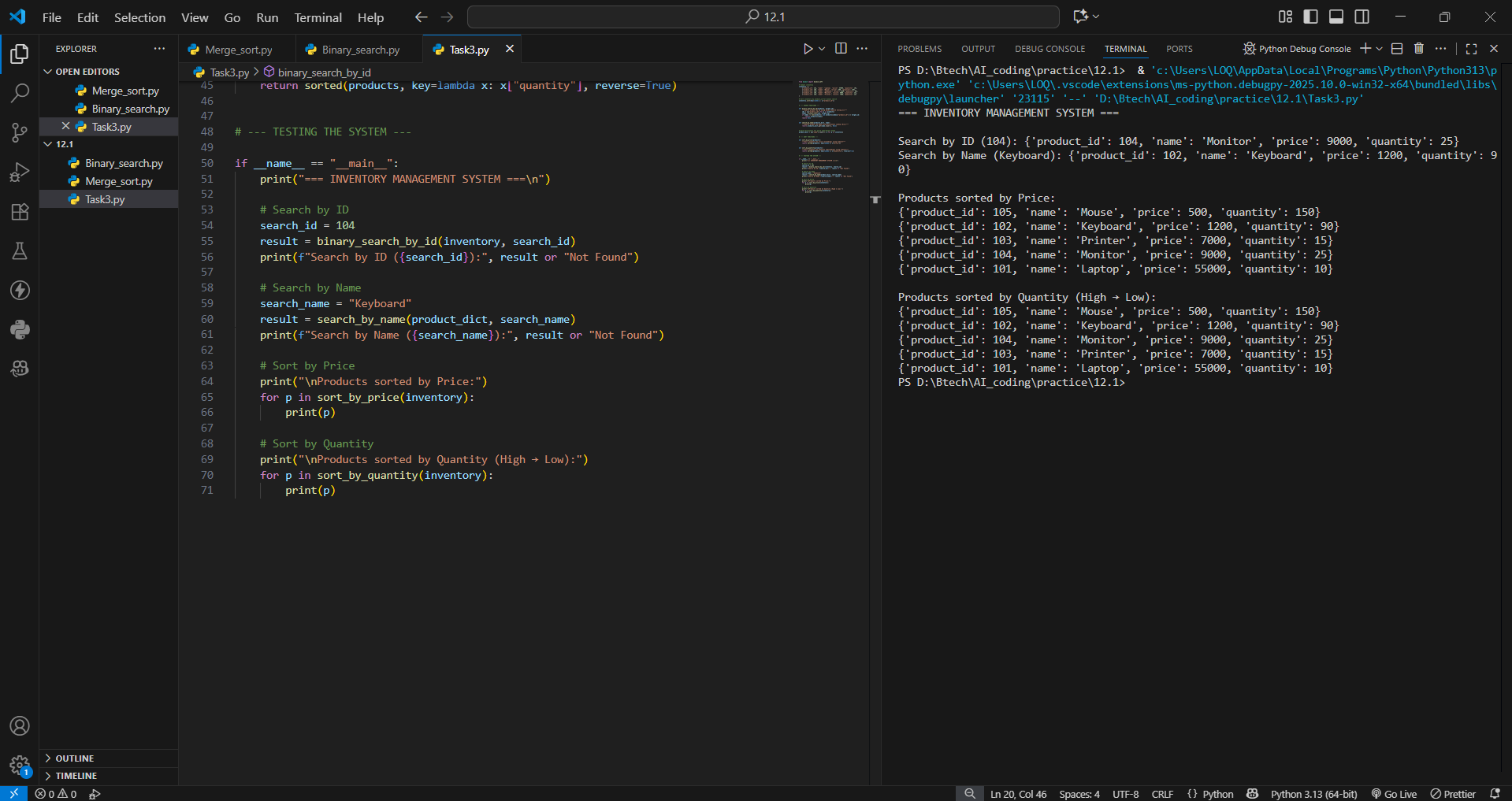
**Prompt:**

"Write a Python program for a real-time inventory management system for a retail store. Each product should have a product ID, name, price, and stock quantity. The program should allow searching for products by ID using binary search, searching by name using a dictionary lookup, and sorting products by price or quantity using Python’s built-in sorting (Timsort). Include comments, test cases, and explanations of why each algorithm was chosen based on performance and dataset size."

**Code & Output:**

****

**Testing:**

****

**Inventory Management System**

**Features:**

1. Search product by ID or Name.

2. Sort products by Price or Quantity.

**Algorithms Used:**

- Binary Search for Product ID (O(log n))

- Dictionary Lookup for Product Name (O(1))

- Timsort for Sorting (Python's built-in sorted(), O(n log n))

**Observation:**

The implemented Inventory Management System demonstrates how efficient algorithm selection enhances real-time data handling in retail applications.  
By combining **Binary Search**, **Dictionary Lookup**, and **Timsort**, the system achieves optimal performance for searching and sorting large product datasets.  
The approach ensures quick access to product information, smooth stock analysis, and reliable performance even as the inventory grows.  
Thus, the chosen algorithms effectively balance **speed**, **accuracy**, and **scalability**, meeting the performance needs of modern retail environments.