Lab Task Dynamic 4 N- Queen Problem

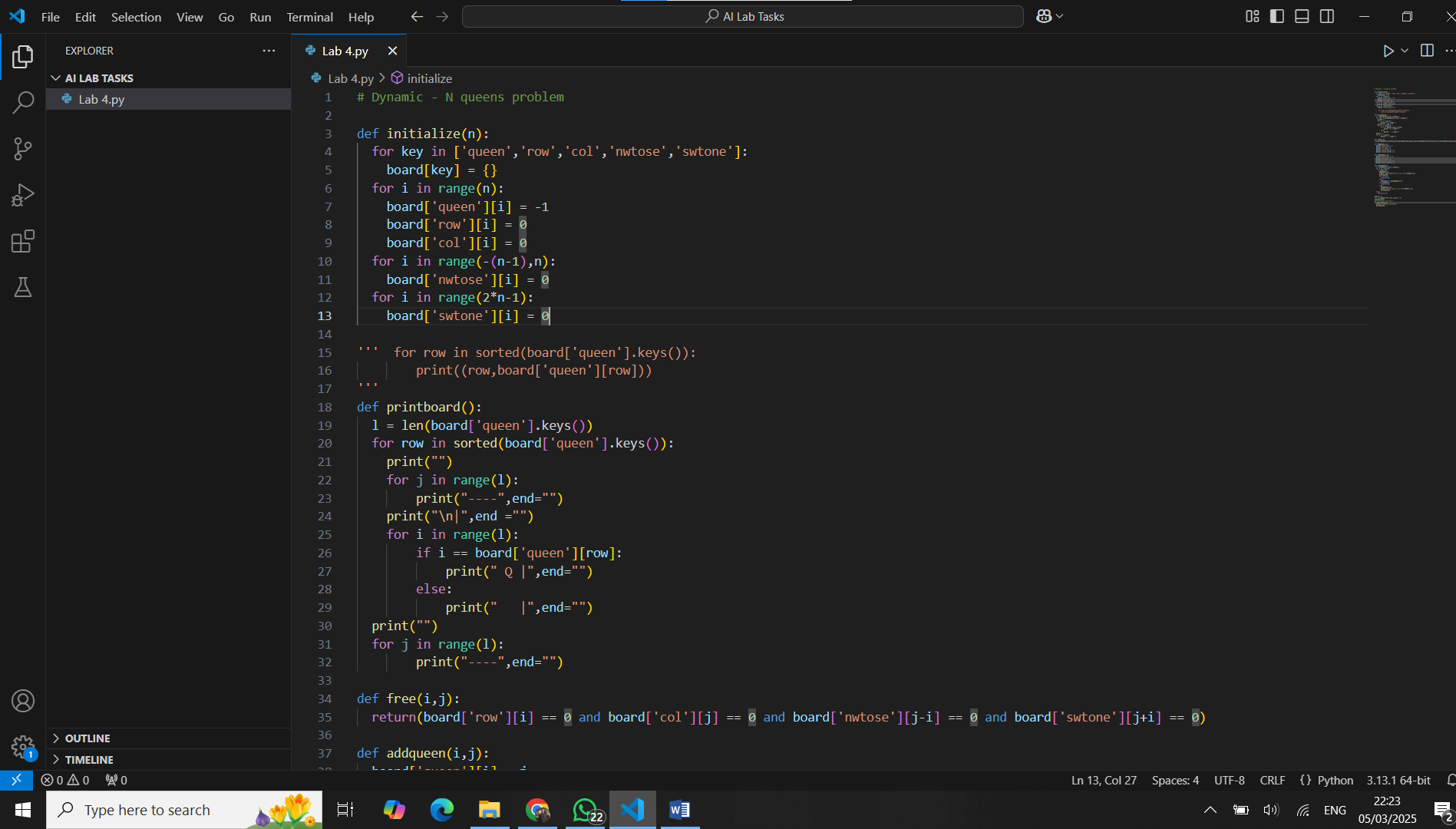
**Explanation of this code……………**

**Step 1: Setting Up the Chessboard**

This code uses a **dictionary** called board with the following keys:

* 'queen': Stores the **column position** of each queen.
* 'row': Keeps track of which rows already have queens.
* 'col': Tracks occupied columns.
* 'nwtose': Diagonal from **Northwest to Southeast**.
* 'swtone': Diagonal from **Southwest to Northeast**.

The **initialize(n**) function sets everything up and marks all spots as empty.



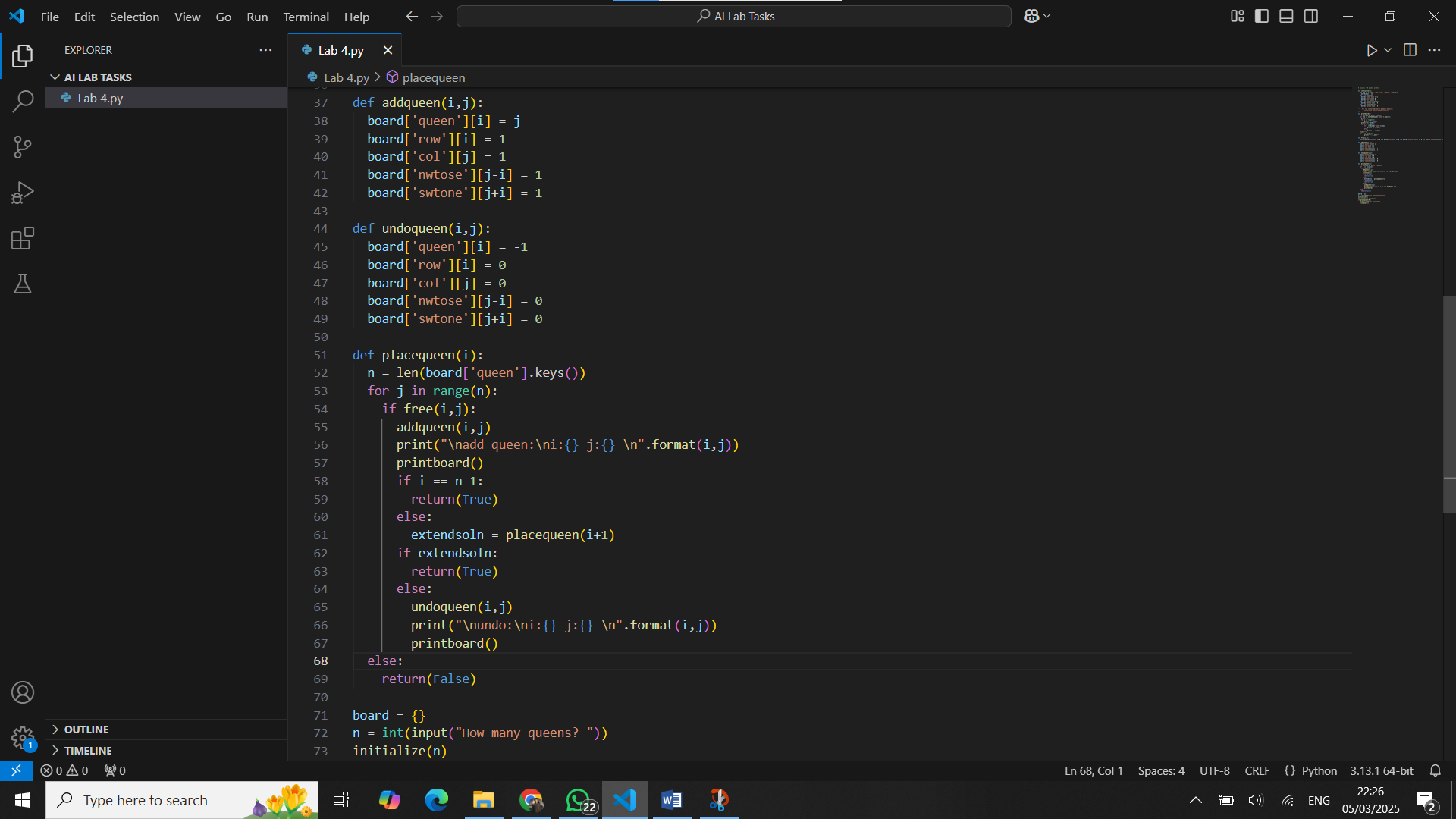
**Step 2: Visualizing the Board**

That's exactly what the **printboard()** function does! It prints out the board with **Q** representing the queens.

**Step 3**: **Checking if a Position is Safe**

If **free(i, j)** returns True, it means **no queen can attack this position**!

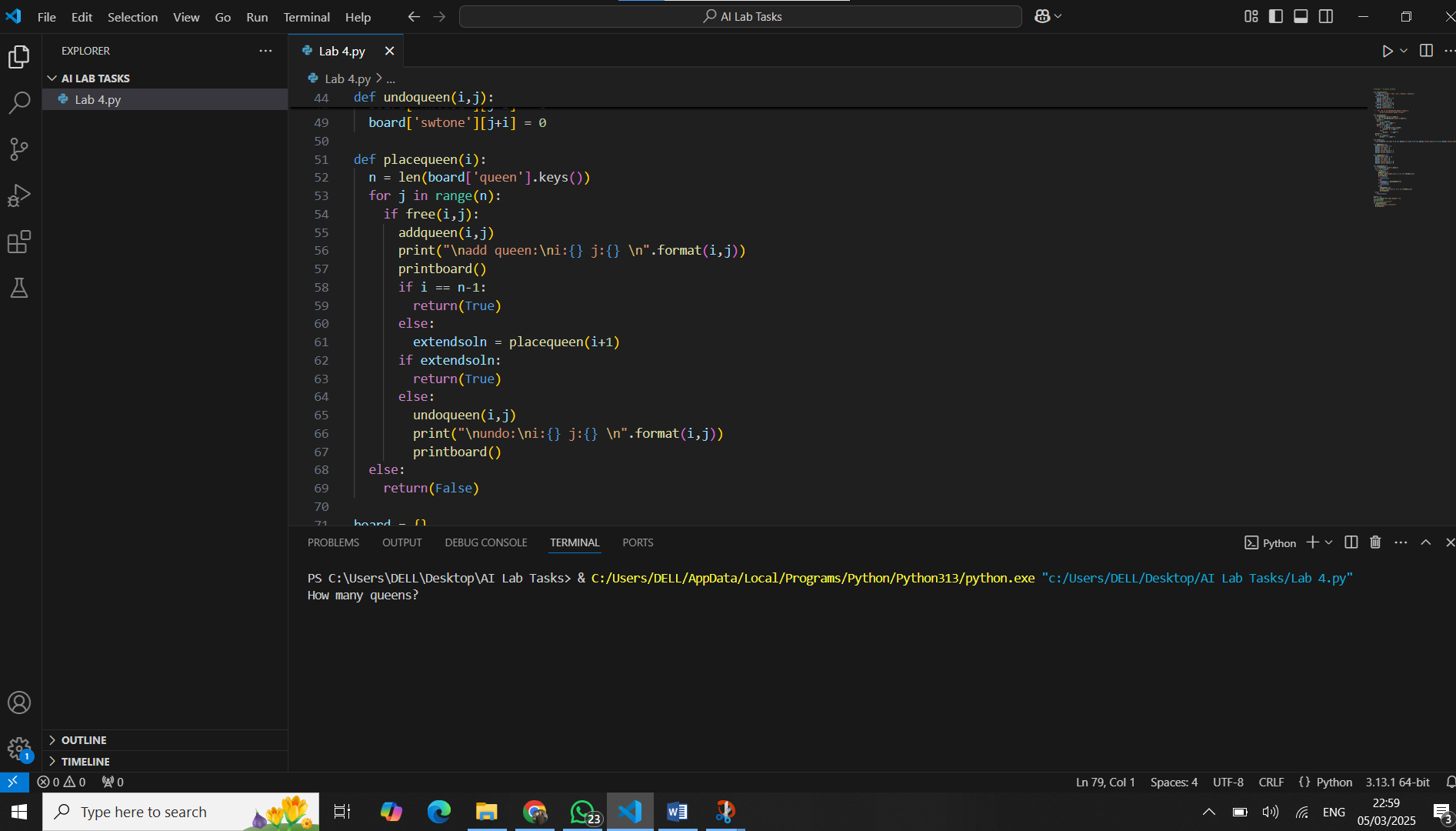
This function goes row by row and prints the board in a neat way!



**Step 4: Placing and Removing Queens**

We need to **place** and **remove** queens dynamically as we search for a solution.

* addqueen(i, j): Puts a queen on the board.
* undoqueen(i, j): Removes a queen if we hit a dead-end.

****

**Step 5: The Backtracking Magic!**

* Finally, we use **backtracking** to explore different ways to place queens. If a solution doesn’t work, we **undo** the last move and try again!

### **Example Run (N = 4)**

