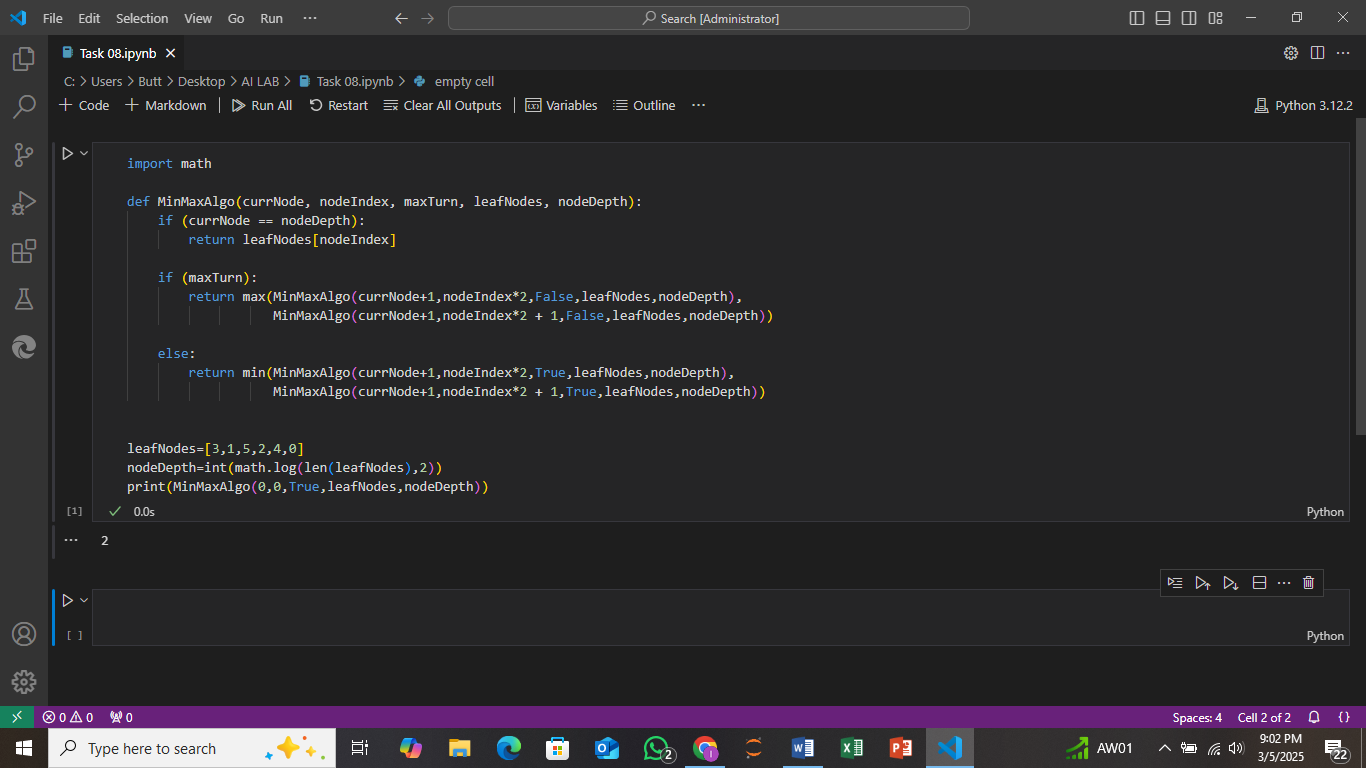
**Task 08:**

**Name: Abdullah Butt Roll Number: 018**



**Explaination:**

This code implements the **Min-Max algorithm** for decision-making in a game tree. Here's a brief explanation of its components:

1. **MinMaxAlgo Function**: This is a recursive function that simulates the Min-Max algorithm. It evaluates a tree of decisions to find the best possible outcome.
   * **currNode**: The current level of the tree (starting from the root).
   * **nodeIndex**: The index of the current node in the current level.
   * **maxTurn**: A boolean flag that indicates if it's the "maximizing" player's turn (True for max, False for min).
   * **leafNodes**: A list of leaf node values representing the final outcomes of the game.
   * **nodeDepth**: The total depth of the tree (calculated from the number of leaf nodes).
2. **Base Case**: If the currNode equals nodeDepth, it means the recursion has reached the leaf nodes, so it returns the value of that leaf node.
3. **Maximizing Player (maxTurn)**: If it's the maximizing player's turn, the function returns the maximum value from its two child nodes (left and right).
4. **Minimizing Player (not maxTurn)**: If it's the minimizing player's turn, the function returns the minimum value from its two child nodes.
5. **Tree Structure**: The game tree is assumed to be a complete binary tree where each node has two children, and the leaf nodes are where the final outcomes are stored.
6. **leafNodes List**: The array represents the leaf node values of the tree, and nodeDepth calculates the height of the tree, which is log2 of the number of leaf nodes.