**Minimax** Algorithm

This code implements the **Minimax algorithm**, which is used in decision-making for two-player games where one player aims to maximize the score and the other aims to minimize it.

1. **Minimax Function**: The minimax function recursively explores the game tree and returns the optimal value for the current player. The function takes four parameters:
   * depth: The current depth in the game tree.
   * nodeIndex: The index of the current node.
   * max\_player: A boolean indicating whether it's the maximizing player's turn (True) or the minimizing player's turn (False).
   * values: A list of values at the leaf nodes of the game tree.
2. **Base Case**: The base case occurs when depth == 3, which means we've reached the leaf nodes of the game tree. The function returns the value of the current node from the values list.
3. **Maximizing Player**: If it's the maximizing player's turn (max\_player = True), the function iterates over the child nodes (by exploring two possible moves) and chooses the maximum value from all the possible moves.
4. **Minimizing Player**: If it's the minimizing player's turn (max\_player = False), the function iterates over the child nodes and chooses the minimum value from all the possible moves.
5. **Execution**: The function starts with the root node and recursively explores the tree, considering all possible moves by both players and selecting the optimal one.