Alpha-Beta Pruning Implementation in Python

```
def alpha_beta_pruning(depth, node_index,
maximizing_player, values, alpha, beta):
    # Base case: If we reach a leaf node
    if depth == 0 or node_index >= len(values):
        return values[node_index]
    if maximizing_player:
        max_eval = float('-inf')
        # Explore child nodes
        for i in range(2):  # Assuming binary tree
            eval = alpha_beta_pruning(depth - 1, node_index * 2 +
i, False, values, alpha, beta)
            max_eval = max(max_eval, eval)
            alpha = max(alpha, eval)
            if beta <= alpha:
                break  # Beta cut-off
        return max_eval
    else:
        min_eval = float('inf')
        # Explore child nodes
        for i in range(2):  # Assuming binary tree
```

```
            eval = alpha_beta_pruning(depth - 1, node_index * 2 +
i, True, values, alpha, beta)
            min_eval = min(min_eval, eval)
            beta = min(beta, eval)
            if beta <= alpha:
                break  # Alpha cut-off
        return min_eval
# Example usage
if _name_ == "_main_":
    # Example tree with leaf node values
    values = [3, 5, 6, 9, 1, 2, 0, -1]
    depth = 3  # Depth of the tree
    alpha = float('-inf')
    beta = float('inf')
    optimal_value = alpha_beta_pruning(depth, 0, True, values,
alpha, beta)
    print(f"The optimal value is: {optimal_value}")
```