This program looked over ticktack toe moves and eliminated poor choices using minmax. I did this by using 1 for a win, 0 for a tie, and a -1 for a loss.

There were two players X and O, b represented a unfilled space. First player X is the maximizing player, O moves second and is the minimizing player.

The way I had it set up, was for each move, X would search through all possible moves it could make on the board, and recursively look through all additional moves. Then it recursively call min and max moves (with moves for sets of 3's) and add up all the scores for each tree. Then it would make the move with the highest score. Min player makes the move with the lowest score.

Alpha beta pruning was applied by ignoring all moves that were suboptimal. If max player moves first, then the min player must move second, and vice versa. However in this case the max player always moves first. This means the min player will always try to find the tree with the minimum values and vice versa, therefor I implemented a algorithm to attempt to figure it out, thought it's not the most efficient.

To note, the leaf was when a player won or board was filled up (tie).

I didn't use the database much for this one, besides for alpha beta pruning, as it took longer to run and was more intensive than iterative calls, at least for a 3 by 3 game.

```
board = [
    [ 'x', 'o', 'x' ],
    [ 'o', 'o', 'x' ],
    [ 'b', 'b', 'b' ]
]
```

```
The Optimal Move is :
ROW: 2  COL: 2
```