Implementing TIC TAC TOE game using minimax algorithm and alpha beta pruning method

Game starts at main:

1. Frist an input is shown to human player (user) by intro() function
2. Then computer starts the competition by randomly choosing one of 1,3,7,9 cells. The reason is due to : playing the corner gives the opponent the smallest choice of squares which must be played to avoid losing.(https://en.wikipedia.org/wiki/Tic-tac-toe#Strategy)
3. Game loop is going to start working till board become full which means no one won so it's a draw.
4. In the loop first winner is going to be checked every time so if either of X or O is won loop(game) be ended.
5. If no one is still won players play by they turns.
6. If variable turn is True it means it's computer's turn. Otherwise it's users turn.(Almost every True/False is from the eyes of computer)
7. If it's user turns it will takes the number of cell user what's to put it's symbol(o) to and then assigns character 'O' to the wanted cell.
8. If it's computer's turn cause it's no longer first move else part of computer_plays() executes. Computer wanna maximize it's children values so act to much a like max_value() function and based on the most value from children chooses the best move it can make(best child to move to). Indexes of best child to choose is gonna be saved in best_move variable and then character 'X' is assigned to this cell.
9. X_win(state) : function for checking if x is winning
10. O_win(state):function to check if o is winning(which never happens☺)

11. board_full:checks if board is full
12. board_empty:checks if board is empty
13. empty_space:checks how many empty spaces are remained.
14. alpha beta pruning minimax as it's said in book and class slides, is recursive and it's base cases are when either of two players is winning or it's tie .Simply when X is winning -> 1
When O is winning-> -1
When it's tie -> 0
But in source code you can see that I multiply this number to number of empty cells before this move so maybe it can be executed faster!(maybe I'm not sure about it.)
15. in alpha beta if the child of parent which is passed trough inputs of function is minimizing max_min is False,otherwise it's True. So either min_value or max_value is called for the state (which is a child)
16. max_value and min_value are implemented just like the sudocode in slides which an initial value is assigned to v which later on should be minimized or maximized so it's -,+ infinity then it will be compared to max or min of given state's children so at the end the min value or max value of all children be assigned to it .
16 . for pruning we check v with alpha (a) and beta (b)
In min_value parents wanna maximize so v is compared with b which is max best option till then and if it's equal to or smaller than it we prune other children.
17. In max_value parents wanna minimize so v is compared with a which is min best option till then and if it's equal to or larger than it we prune other children.

18.in min_value a should be updated with each child which is not pruned . and in max_value b should be updated with each child which is not pruned .