A)Using Minimax:

Assumptions :

1.User's move will be "O" and Computer's move will be "X". 2.If user wins it will return +1 ,if user loses it will return -1 and if game is draw it will return 0. 3.First move will always made by user.

Algorithms : It is Minimax Algorithm and it makes use of DFS tree(Search Space) for finding the best move.

Methodology :

1.User will take first move always. The user can put "O" in any 9 position in board.

2.The computer will take second move and search for best move by expanding the whole search space. The best move of computer will depend on best_val() function by comparing best value with move. Alternatively user and computer call Min_Max() function.User's turn is max and computer's turn is always min.

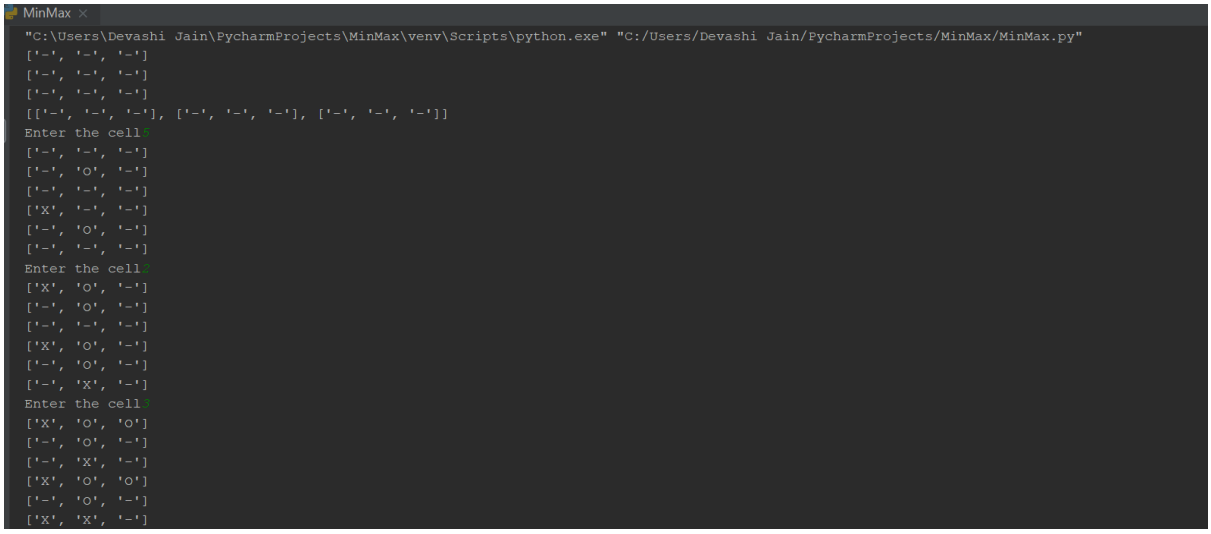
3.If all elements of row are "O" or all elements of column are "O" or if all elements of diagonals are "O" then user will win.

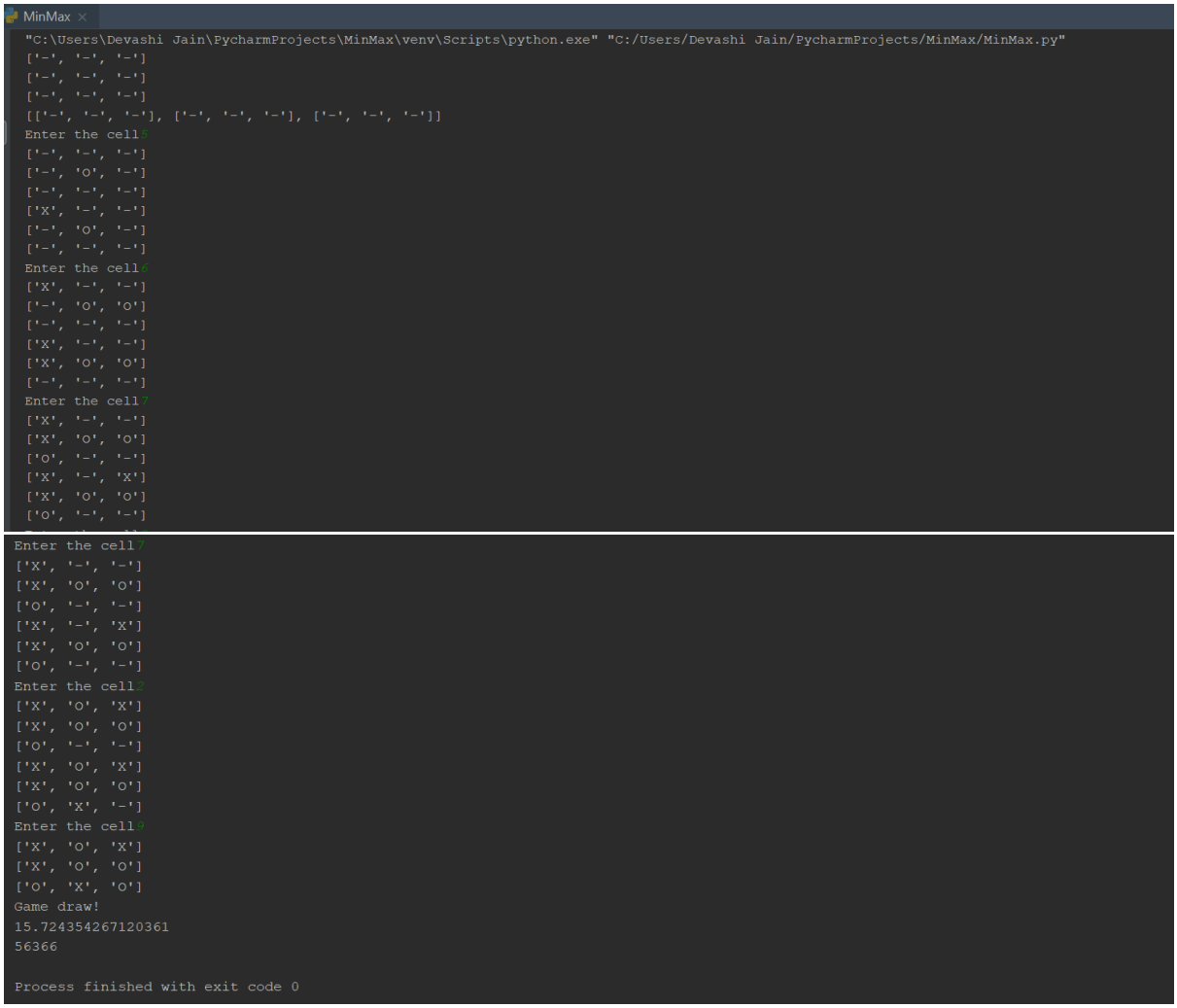
4.If all elements of row are "X" or all elements of column are "X" or if all elements of diagonals are "X" then computer will win.

5.If both the 3 and 4 point not follow then game will be draw.

Observations:

Output is






```
Enter the cell9
['X', '-', '-']
['X', 'O', 'O']
['O', '-', '-']
['X', '-', 'X']
['X', 'O', 'O']
['O', '-', '-']
Enter the cell3
['X', 'O', 'X']
['X', 'O', 'O']
['O', '-', '-']
['X', 'O', 'X']
['X', 'O', 'O']
['O', 'X', '-']
Enter the cell8
['X', 'O', 'X']
['X', 'O', 'O']
['O', 'X', 'O']
Game draw!
15.724354267120361
56366

Process finished with exit code 0
```

B)Using Alpha-Beta Pruning:

Assumptions :

1.User's move will be "O" and Computer's move will be "X". 2.If user wins it will return +1 ,if user loses it will return -1 and if game is draw it will return 0. 3.First move will always made by user.

Algorithms : It is Alpha-Beta Pruning Algorithm and it makes use of DFS tree(Search Space) for finding the best move and prune the unnecessary part.

Methodology :

1.User will take first move always. The user can put "O" in any 9 position in board.

2.The computer will take second move and search for best move by expanding the whole search space. The best move of computer will depend on best_val() function by comparing move with alpha beta values.Alternatively user and computer call Min_Max() function.User's turn is max and computer's turn is always min.Alpha-Beta values are responsible for pruning the search space.

3.If all elements of row are "O" or all elements of column are "O" or if all elements of diagonals are "O" then user will win.

4.If all elements of row are "X" or all elements of column are "X" or if all elements of diagonals are "X" then computer will win.

5.If both the 3 and 4 point not follow then game will be draw.

Observations:

Output is

```
AlphaBeta
"C:\Users\Devashi Jain\PycharmProjects\MinMax\venv\Scripts\python.exe" "C:/Users/Devashi Jain/PycharmProjects/MinMax/AlphaBeta.py"
['-', '-', '-']
['-', '-', '-']
['-', '-', '-']
[['-', '-', '-'], ['-', '-', '-'], ['-', '-', '-']]
Enter the cell5
['-', '-', '-']
['-', 'O', '-']
['-', '-', '-']
['X', '-', '-']
['-', 'O', '-']
['-', '-', '-']
Enter the cell2
['X', 'O', '-']
['-', 'O', '-']
['-', '-', '-']
['X', 'O', '-']
['-', 'O', '-']
['-', 'X', '-']
Enter the cell3
['X', 'O', 'O']
['-', 'O', '-']
['-', 'X', '-']
['X', 'O', 'O']
['-', 'O', '-']
['X', 'X', '-']
```

```
Enter the cell4
['X', 'O', 'O']
['O', 'O', '-']
['X', 'X', '-']
['X', 'O', 'O']
['O', 'O', '-']
['X', 'X', 'X']
You lose
8.915594339370728
6532

Process finished with exit code 0
```

```
AlphaBeta
"C:\Users\Devashi Jain\PycharmProjects\MinMax\venv\Scripts\python.exe" "C:/Users/Devashi Jain/PycharmProjects/MinMax/AlphaBeta.py"
['-', '-', '-']
['-', '-', '-']
['-', '-', '-']
[['-', '-', '-'], ['-', '-', '-'], ['-', '-', '-']]
Enter the cell8
['-', '-', '-']
['-', '-', 'O']
['-', '-', '-']
['-', '-', 'X']
['-', '-', 'O']
['-', '-', '-']
Enter the cell3
['-', 'O', 'X']
['-', '-', 'O']
['-', '-', '-']
['-', 'O', 'X']
['X', '-', 'O']
['-', '-', '-']
Enter the cell1
['O', 'O', 'X']
['X', '-', 'O']
['-', '-', '-']
['O', 'O', 'X']
['X', 'X', 'O']
['-', '-', '-']
```

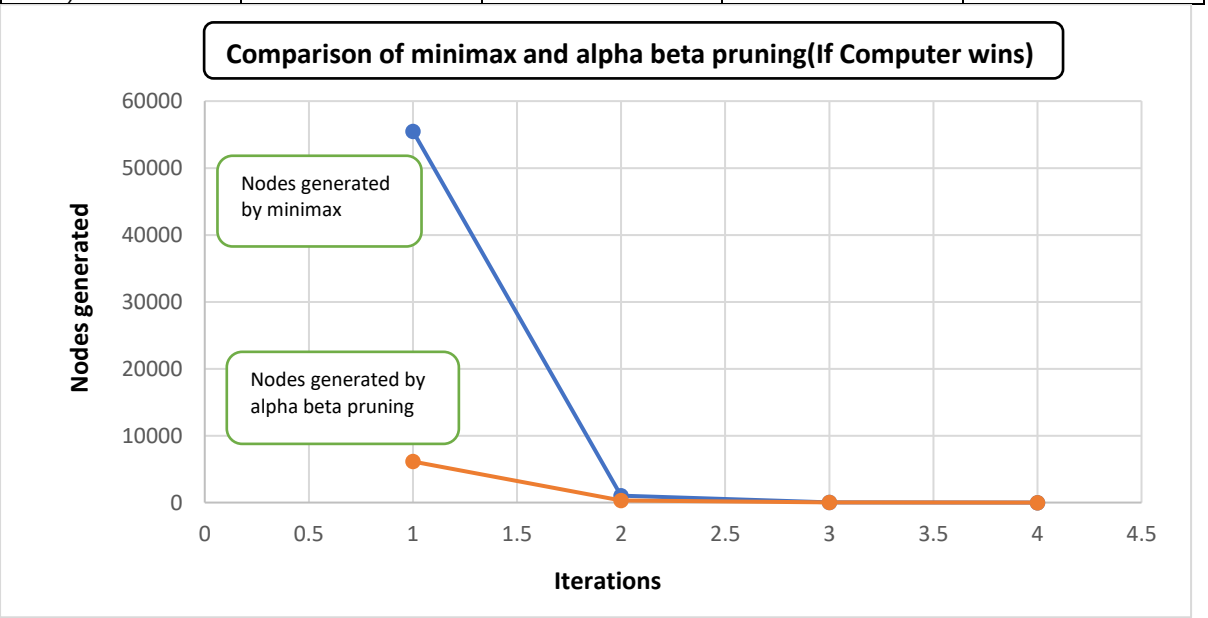
```
Enter the cell1
['O', 'O', 'X']
['X', '-', 'O']
['-', '-', '-']
['O', 'O', 'X']
['X', 'X', 'O']
['-', '-', '-']
Enter the cell7
['O', 'O', 'X']
['X', 'X', 'O']
['O', '-', '-']
['O', 'O', 'X']
['X', 'X', 'O']
['O', 'X', '-']
Enter the cell9
['O', 'O', 'X']
['X', 'X', 'O']
['O', 'X', 'O']
Game draw
14.252618551254272
10370

Process finished with exit code 0
```

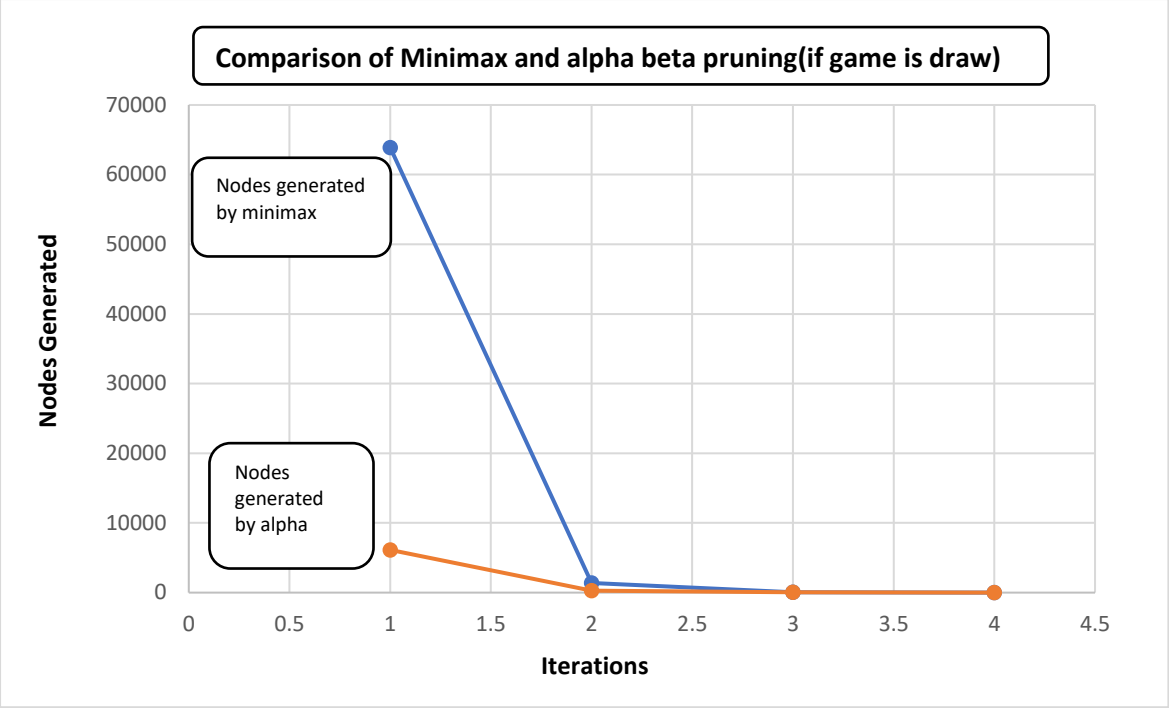
Comparison between in Minimax algorithm and Alpha-Beta Pruning:

Minimax Algorithm: In this it will search all possible for all moves in search state and return move with best value. It takes time in generating output but it provide optimal solution. In our case, Tic-Tac-Toe game it provides optimal solution that is either computer will win or the game will be drawn but the drawback it will search for all moves in search state and return best one, in this it takes time in generating output. Above if computer wins the game then it is taking 11 sec or if match draws then it is taking 15 sec while in case of alpha beta pruning is will take comparatively less time.

Alpha-Beta Pruning: It is specific method. It speed-up the game by cutting off the game tree at a certain depth. It also provide optimal solution with least moves as compare to Minimax. In our example Tic-Tac-Toe game the alpha-beta pruning will prune the unnecessary part of the tree by comparing with alpha beta values and return best move. Above if computer wins it is taking 8 sec or if match draws it is taking 13 sec. It is taking comparatively less time as compared to Minimax.

	Minimax(If computer wins)	Alpha-Beta Pruning(If computer wins)	Minimax(If game is draw)	Alpha-Beta Pruning(If game is draw)
Time	11	8	15	13
Nodes generated(after 1st move)	55496	6130	63896	6130
Nodes generated(after 2nd move)	1048	333	1398	301
Nodes generated(after 3rd move)	40	23	56	36
Nodes generated(after 4th move)	0	0	2	2





Results:

1.In both the algorithms either computer will win or game will be draw.

2.Alpha-Beta Pruning has better performance in terms of time and number of nodes generated