Artificial Intelligence &Machine Learning Lab (PCCCS495)

**CONSTRUCTION OF TIC-TAC-TOE**

Department of Computer Science and Engineering(CSE)

## **Submitted by:**

## **Anushka Maji(12021002002202,Roll-139)**

## **Sagnik Sinha(12021002002182,Roll-119)**

|  |  |  |
| --- | --- | --- |
|  | INDEX |  |
| Contents |  | Pages |
| ABSTRACT |  | 3 |
| ACKNOWLEDGEMENT |  | 4 |
| INTRODUCTION |  | 5 |
| CODE |  | 7 |
| RESULT |  | 11 |
| CONCLUSION |  | 16 |
| REFERENCE |  | 16 |

# ABSRACT

A Tic-Tac-Toe is a 2 player game, who take turns marking the spaces in a 3x3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.

tictactoe.py is the python implementation of the game. There are two modes to play:

* Single Player (Against Computer)
* Double Players

The double players mode iteratively takes input from both the players, while making sure, if anyone has won or not.

The single player mode uses Minimax algorithm to make the computer unbeatable. Even if the player plays the most optimal move every time, the end result would be at-most a draw. Every time you make a move, the computer plays automatically.

*Key words: Tic-tac-toe, single player, double player, Minimax algorithm*.

# ACKNOWLEDGEMENT

I would want to convey my heartfelt gratitude to Prof. Sanchita Ghosh, my mentor, for her invaluable advice and assistance in completing my project. I would also want to thank the Institute of Engineering & Management for accepting my project in my desired field of expertise.

# INTRODUCTION

Playing Tic Tac Toe game in childhood was one of our best time pass task. It was so much funny. In childhood, we play it on paper but today we will build this game and will play on computer.

* We have here by constructed a board that have 9 cells with 3x3 (row x col).
* It is a two player game in which each player assigned with a marker symbol that is ‘X’ and ‘O’.
* During the first turn, the player marks the symbol X (Marker symbol corresponding to the player) to a cell among the available cells, and the second player will mark O (Second player’s symbol) to a cell among the available cells.
* The game continues until it reaches either one of the conditions:

o When one column, row or diagonal has X, the player assigned with X wins else if this state arrived for O, the player O wins.

* If the board contains no free cell left and none of the above conditions arrived, the Game ends with a Draw.

**Our goal to write the code**:

tictactoe.py is the python implementation of the game. There are two modes to play:

* Single Player (Against Computer)
* Double Players

The double players mode iteratively takes input from both the players, while making sure, if anyone has won or not.

The single player mode uses Minimax algorithm to make the computer unbeatable. Even if the player plays the most optimal move every time, the end result would be at-most a draw. Every time you make a move, the computer plays automatically.

Minimax is a decision rule used in decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case (maximum loss) scenario. Originally formulated for two-player zero-sum game theory, covering both the cases where players take alternate

moves and those where they make simultaneous moves, it has also been extended to more complex games and to general decision making in the presence of uncertainty.

# CODE

Here is the general outline of the process:

* ConstBoard: it is where the bode is been constructer and all the input of the user aswell as the computer is shown.
* User1Turn, User2Turn: it is the function bulid for the inputs from the user in the game.
* Minimax: it is the algorithm used by the computer to play the game. Minimax is a decision rule used in decision theory, game theory, statistics and philosophy for minimizing the possible loss for a worst case (maximum loss) scenario. Originally formulated for two-player zero-sum game theory, covering both the cases where players take alternate moves and those where they make simultaneous moves, it has also been extended to more complex games and to general decision making in the presence of uncertainty.
* CompTurn: it is the fuction for the computer to make the move.
* Analyzeboard: It is the function which shows the conditions for wining and if any player satisfy any of the conditions then he/ she/ it is proclaimed as winner. The conditions are: [[0, 1, 2], [3, 4, 5], [6, 7, 8], [0, 3, 6], [1, 4, 7], [2, 5, 8], [0, 4, 8], [2, 4, 6]]

# Tic-Tac-Toe using Mini-Max Algorithm

*""" """*

# This function is used to draw the board's current state every time the user turn arrives.

def ConstBoard(board):

print("Current State Of Board : \n\n"); for i in range(0, 9):

if ((i > 0) and (i % 3) == 0):

print("\n");

if (board[i] == 0): print("- ", end=" ");

if (board[i] == 1): print("O ", end=" ");

if (board[i] == -1):

print("X ", end=" "); print("\n\n");

# This function takes the user move as input and make

the required changes on the board. def User1Turn(board):

pos = input("Enter X's position from [1...9]: "); pos = int(pos);

if (board[pos - 1] != 0):

print("Wrong Move!!!"); exit(0);

board[pos - 1] = -1;

def User2Turn(board):

pos = input("Enter O's position from [1...9]: "); pos = int(pos);

if (board[pos - 1] != 0):

print("Wrong Move!!!"); exit(0);

board[pos - 1] = 1;

# MinMax function.

def minimax(board, player): x = analyzeboard(board); if (x != 0):

return (x \* player); pos = -1;

value = -2;

for i in range(0, 9): if (board[i] == 0):

board[i] = player;

score = -minimax(board, (player \* -1)); if (score > value):

value = score; pos = i;

board[i] = 0;

if (pos == -1): return 0;

return value;

# This function makes the computer's move using minmax algorithm.

def CompTurn(board): pos = -1;

value = -2;

for i in range(0, 9): if (board[i] == 0):

board[i] = 1;

score = -minimax(board, -1); board[i] = 0;

if (score > value): value = score; pos = i;

board[pos] = 1;

# This function is used to analyze a game. def analyzeboard(board):

cb = [[0, 1, 2], [3, 4, 5], [6, 7, 8], [0, 3, 6],

[1, 4, 7], [2, 5, 8], [0, 4, 8], [2, 4, 6]];

for i in range(0, 8):

if (board[cb[i][0]] != 0 and board[cb[i][0]] == board[cb[i][1]]

and

board[cb[i][0]] == board[cb[i][2]]): return board[cb[i][2]];

return 0;

# Main Function. def main():

choice = input("Enter 1 for single player, 2 for multiplayer: ");

choice = int(choice);

# The broad is considered in the form of a single dimentional array.

# One player moves 1 and other move -1. board = [0, 0, 0, 0, 0, 0, 0, 0, 0];

if (choice == 1):

print("Computer : O Vs. You : X");

player = input("Enter to play 1(st) or 2(nd)

:");

player = int(player); for i in range(0, 9):

if (analyzeboard(board) != 0): break;

if ((i + player) % 2 == 0): CompTurn(board);

else:

ConstBoard(board); User1Turn(board);

else:

for i in range(0, 9):

if (analyzeboard(board) != 0): break;

if ((i) % 2 == 0):

ConstBoard(board); User1Turn(board);

else:

ConstBoard(board); User2Turn(board);

x = analyzeboard(board); if (x == 0):

ConstBoard(board); print("Draw!!!")

if (x == -1):

ConstBoard(board);

print("X Wins!!! Y Loose !!!") if (x == 1):

ConstBoard(board);

print("X Loose!!! O Wins !!!!")

# # main()

# #

# RESULT

OUTPUT FOR SINGLE PLAYER GAME:

"C:\Users\Ganapati Cemicals\AppData\Local\Microsoft\WindowsApps\python3.10.exe" "C:\Users\Ganapati Cemicals\PycharmProjects\pythonProject\tictacto.py"

Enter 1 for single player, 2 for multiplayer: 1 Computer : O Vs. You : X

Enter to play 1(st) or 2(nd) :1 Current State Of Board :

- - -

- - -

- - -

Enter X's position from [1...9]: 1 Current State Of Board :

X - -

* O -

- - -

Enter X's position from [1...9]: 9 Current State Of Board :

X O -

* O -
* - X

Enter X's position from [1...9]: 8 Current State Of Board :

X O -

* O - O X X

Enter X's position from [1...9]: 3 Current State Of Board :

X O X

* O O O X X

Enter X's position from [1...9]: 4 Current State Of Board :

X O X X O O O X X

Draw!!!

Process finished with exit code 0

OUTPUT FOR DOUBLE PLAYER GAME:

"C:\Users\Ganapati Cemicals\AppData\Local\Microsoft\WindowsApps\python3.10.exe" "C:\Users\Ganapati Cemicals\PycharmProjects\pythonProject\tictacto.py"

Enter 1 for single player, 2 for multiplayer: 2 Current State Of Board :

- - -

- - -

- - -

Enter X's position from [1...9]: 1 Current State Of Board :

X - -

- - -

- - -

Enter O's position from [1...9]: 2 Current State Of Board :

X O -

- - -

- - -

Enter X's position from [1...9]: 4 Current State Of Board :

X O -

X - -

- - -

Enter O's position from [1...9]: 3 Current State Of Board :

X O O

X - -

- - -

Enter X's position from [1...9]: 7 Current State Of Board :

X O O

X - -

X - -

X Wins!!! Y Loose !!!

Process finished with exit code 0

# CONCLUSION

We will here by conclude that we have successfully completed making tic-tac- toe and run it in our computer. We made two types of game one single player other the double player game. We first constructed our board and asked is it a single player or double player. In case of double player both the marks player will be human input otherwise single player the interesting part of the code where the computer plays with the human. The computer uses minimax algorithm which makes it unbeatable even if the human uses the optimal moves every time they may end up with drawing the game.

* GITHUB

# REFERENCES

* realpython.com/tic-tac-toe-ai-python
* [geeksforgeeks.org](http://www.geeksforgeeks.org/)
* [www.simplifiedpython.net](http://www.simplifiedpython.net/)