***Simple AI Game***

****

**Submitted to –Miss.Shabnam Mam**

**Submitted by-Rahul Sahu(5)**

**Aditya Shekhar(49)**

**Rajesh Kumar Gupt(51)**

**GithubLink:-** <https://github.com/RahulSahu7957/Tic_Tac_Toe_using_MinMax/blob/master/Tic_Tac_Toe.py>

**Project Report on Tic Tac Toe Game**

**Index**

1.Introduction

2.Project Ideas

3.Refrences

4.Conclusion

**Introduction:-**

In this project we created a Tic Tac Toe game using Min Max algorithm.This is Simple AI game which is easy to play and enjoy.

Python is a high-level, interpreted and general-purpose dynamic programming language that focuses on code readability. The syntax in Python helps the programmers to do coding in fewer steps as compared to other language. The Python is widely used in bigger organizations because of its multiple programming paradigms. They usually involve imperative and object-oriented functional programming. It has a comprehensive and large standard library that has automatic memory management and dynamic features.

Our project is to implement the Tic-Tac-Toe game which will features like single player and two player modes. In single player mode we have used game theory logics like **minimax algorithms** to determine the best move that the computer plays.

The project was tested and it turned out to be **90%** accurate.

Function Used in Program:-

1. Def evaluate()
2. Def wins()
3. Def game over()
4. Def Empty\_cells()
5. Def valid\_move()
6. Def set\_move()
7. Def Minmax()
8. Def clean()
9. Def render()
10. Def ai\_turn()
11. Def human\_turn()
12. Def main()
13. Print()

**Project Ideas**

First of all we need to implement the best move logic for the computer to play in single player mode.As already stated we have used minmax algorithms to determine the best move.Please check the refrences attached with file with the name of **Tic\_Tac\_Toe.py** .

**What is MinMax?**

1. Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
2. Mini-Max algorithm uses recursion to search through the game-tree.

3. Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.

4. The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.

5.if there are two players one is called Maximizer and other is called Minimizer.

6. Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.

7. At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs.

**Project Code:-**

from math import inf as infinity

from random import choice

import platform

import time

from os import system

HUMAN = -1

COMP = +1

board = [

[0, 0, 0],

[0, 0, 0],

[0, 0, 0],

]

def evaluate(state):

"""

Function to heuristic evaluation of state.

:return: +1 if the computer wins; -1 if the human wins; 0 draw

"""

if wins(state, COMP):

score = +1

elif wins(state, HUMAN):

score = -1

else:

score = 0

return score

def wins(state, player):

"""

This function tests if a specific player wins. Possibilities:

\* Three rows [X X X] or [O O O]

\* Three cols [X X X] or [O O O]

\* Two diagonals [X X X] or [O O O]

:return: True if the player wins

"""

win\_state = [

[state[0][0], state[0][1], state[0][2]],

[state[1][0], state[1][1], state[1][2]],

[state[2][0], state[2][1], state[2][2]],

[state[0][0], state[1][0], state[2][0]],

[state[0][1], state[1][1], state[2][1]],

[state[0][2], state[1][2], state[2][2]],

[state[0][0], state[1][1], state[2][2]],

[state[2][0], state[1][1], state[0][2]],

]

if [player, player, player] in win\_state:

return True

else:

return False

def game\_over(state):

"""

This function test if the human or computer wins

:return: True if the human or computer wins

"""

return wins(state, HUMAN) or wins(state, COMP)

def empty\_cells(state):

"""

Each empty cell will be added into cells' list

:return: a list of empty cells

"""

cells = []

for x, row in enumerate(state):

for y, cell in enumerate(row):

if cell == 0:

cells.append([x, y])

return cells

def valid\_move(x, y):

"""

A move is valid if the chosen cell is empty

x: X coordinate

y: Y coordinate

:return: True if the board[x][y] is empty

"""

if [x, y] in empty\_cells(board):

return True

else:

return False

def set\_move(x, y, player):

"""

Set the move on board, if the coordinates are valid

x: X coordinate

y: Y coordinate

player: the current player

"""

if valid\_move(x, y):

board[x][y] = player

return True

else:

return False

def minimax(state, depth, player):

"""

AI function that choice the best move

state: current state of the board

depth: node index in the tree (0 <= depth <= 9),

but never nine in this case (see iaturn() function)

player: an human or a computer

:return: a list with [the best row, best col, best score]

"""

if player == COMP:

best = [-1, -1, -infinity]

else:

best = [-1, -1, +infinity]

if depth == 0 or game\_over(state):

score = evaluate(state)

return [-1, -1, score]

for cell in empty\_cells(state):

x, y = cell[0], cell[1]

state[x][y] = player

score = minimax(state, depth - 1, -player)

state[x][y] = 0

score[0], score[1] = x, y

if player == COMP:

if score[2] > best[2]:

best = score # max value

else:

if score[2] < best[2]:

best = score # min value

return best

def clean():

"""

Clears the console

"""

os\_name = platform.system().lower()

if 'windows' in os\_name:

system('cls')

else:

system('clear')

def render(state, c\_choice, h\_choice):

"""

Print the board on console

:param state: current state of the board

"""

chars = {

-1: h\_choice,

+1: c\_choice,

0: ' '

}

str\_line = '---------------'

print('\n' + str\_line)

for row in state:

for cell in row:

symbol = chars[cell]

print(f'| {symbol} |', end='')

print('\n' + str\_line)

def ai\_turn(c\_choice, h\_choice):

"""

It calls the minimax function if the depth < 9,

else it choices a random coordinate.

c\_choice: computer's choice X or O

h\_choice: human's choice X or O

:return:

"""

depth = len(empty\_cells(board))

if depth == 0 or game\_over(board):

return

clean()

print(f'Computer turn [{c\_choice}]')

render(board, c\_choice, h\_choice)

if depth == 9:

x = choice([0, 1, 2])

y = choice([0, 1, 2])

else:

move = minimax(board, depth, COMP)

x, y = move[0], move[1]

set\_move(x, y, COMP)

time.sleep(1)

def human\_turn(c\_choice, h\_choice):

"""

The Human plays choosing a valid move.

c\_choice: computer's choice X or O

h\_choice: human's choice X or O

:return:

"""

depth = len(empty\_cells(board))

if depth == 0 or game\_over(board):

return

# Dictionary of valid moves

move = -1

moves = {

1: [0, 0], 2: [0, 1], 3: [0, 2],

4: [1, 0], 5: [1, 1], 6: [1, 2],

7: [2, 0], 8: [2, 1], 9: [2, 2],

}

clean()

print(f'Human turn [{h\_choice}]')

render(board, c\_choice, h\_choice)

while move < 1 or move > 9:

try:

move = int(input('Use numpad (1..9): '))

coord = moves[move]

can\_move = set\_move(coord[0], coord[1], HUMAN)

if not can\_move:

print('Bad move')

move = -1

except (EOFError, KeyboardInterrupt):

print('Bye')

exit()

except (KeyError, ValueError):

print('Bad choice')

def main():

"""

Main function that calls all functions

"""

clean()

h\_choice = '' # X or O

c\_choice = '' # X or O

first = '' # if human is the first

# Human chooses X or O to play

while h\_choice != 'O' and h\_choice != 'X':

try:

print('')

h\_choice = input('Choose X or O\nChosen: ').upper()

except (EOFError, KeyboardInterrupt):

print('Bye')

exit()

except (KeyError, ValueError):

print('Bad choice')

# Setting computer's choice

if h\_choice == 'X':

c\_choice = 'O'

else:

c\_choice = 'X'

# Human may starts first

clean()

while first != 'Y' and first != 'N':

try:

first = input('First to start?[y/n]: ').upper()

except (EOFError, KeyboardInterrupt):

print('Bye')

exit()

except (KeyError, ValueError):

print('Bad choice')

# Main loop of this game

while len(empty\_cells(board)) > 0 and not game\_over(board):

if first == 'N':

ai\_turn(c\_choice, h\_choice)

first = ''

human\_turn(c\_choice, h\_choice)

ai\_turn(c\_choice, h\_choice)

# Game over message

if wins(board, HUMAN):

clean()

print(f'Human turn [{h\_choice}]')

render(board, c\_choice, h\_choice)

print('YOU WIN!')

elif wins(board, COMP):

clean()

print(f'Computer turn [{c\_choice}]')

render(board, c\_choice, h\_choice)

print('YOU LOSE!')

else:

clean()

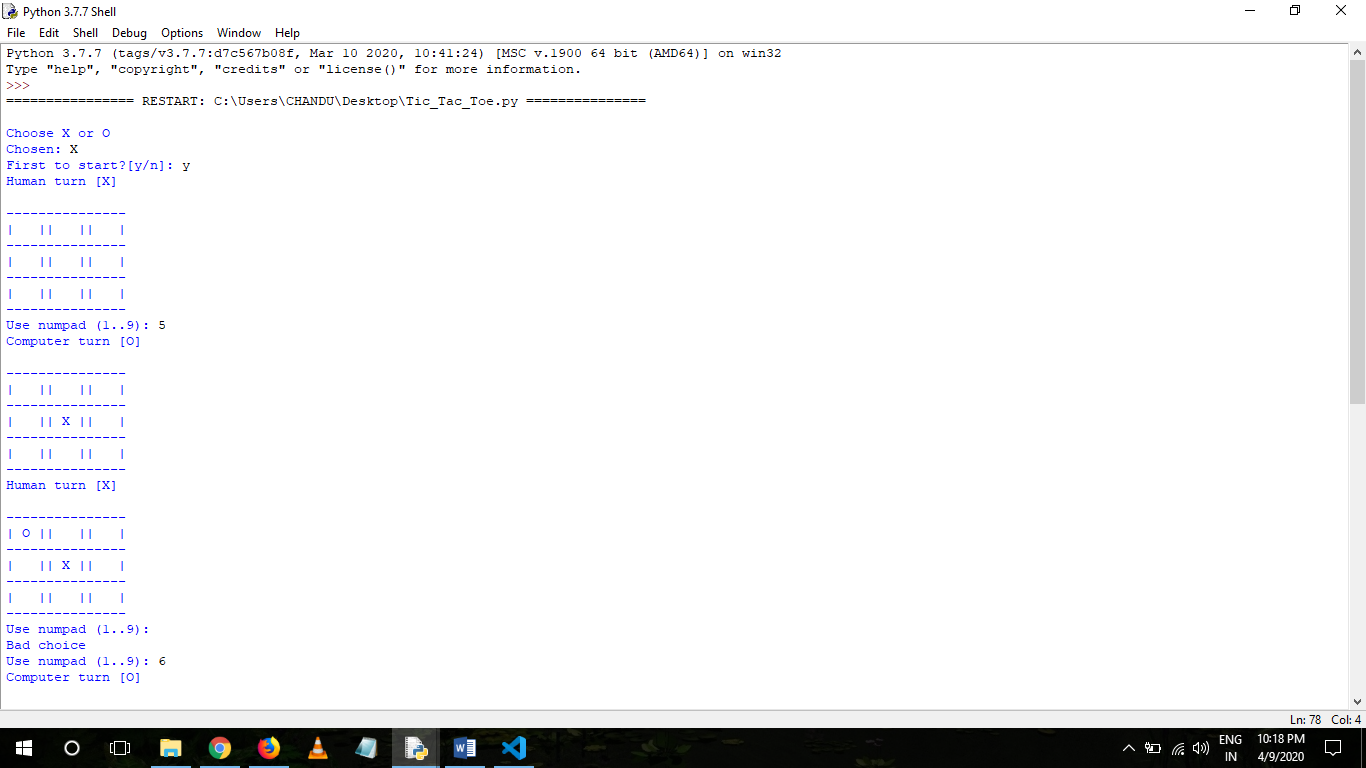
render(board, c\_choice, h\_choice)

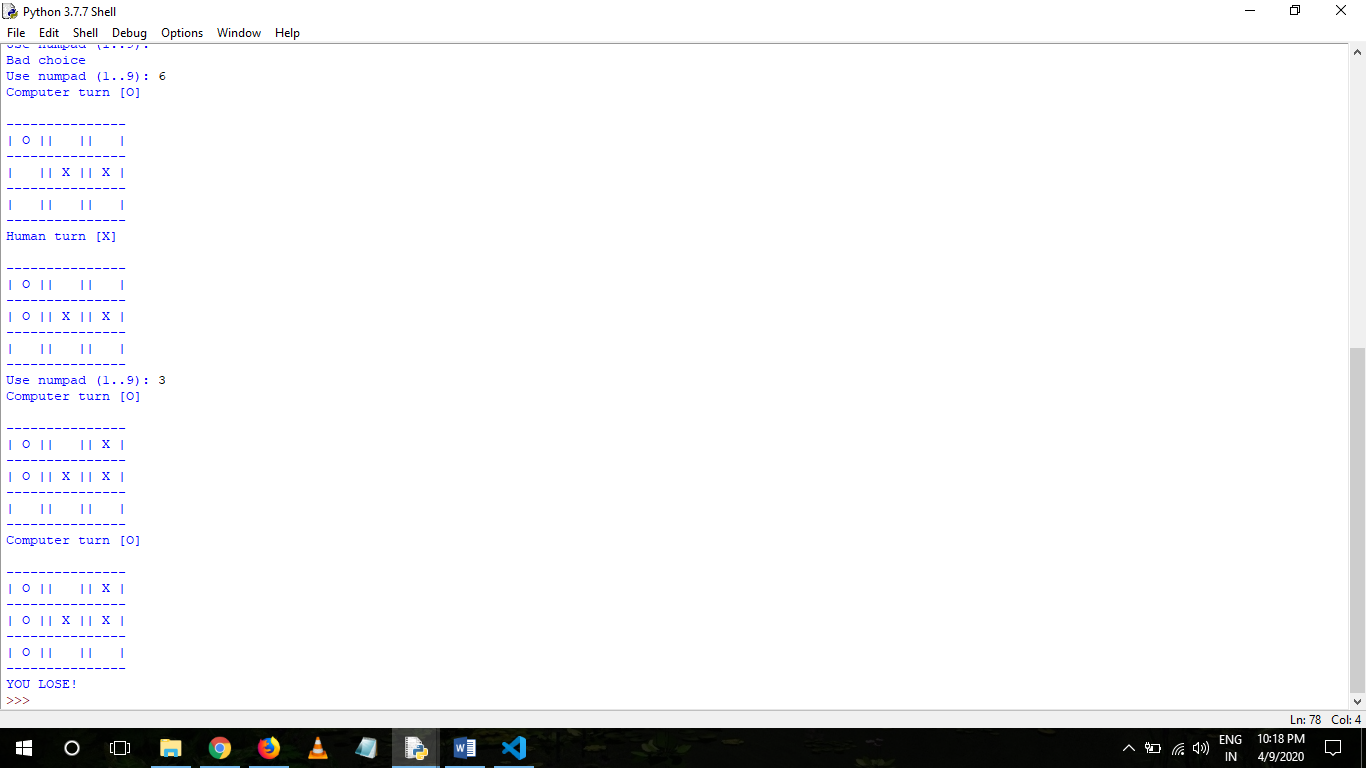
print('DRAW!')

exit()

if \_\_name\_\_ == '\_\_main\_\_':

main()

Snaps. 1. 2.



**Explanation of Output:-**

In the above program when will run it first will ask to choose either X or O.If select X as human player then O will be consider as computer player. Then a 3x3 board will be displayed and will ask to enter number between 1 to 9 as index as per human move,henceforth computer will select any index from 1 to 9 to not let human win this will be done by MinMax algorithm running at backend will find best move for computer.If either by human or computer any player having same move in diagonal,vertical ,horizontal then that player will win or it will be draw.

**References:-**

1.Python Library Documentation

Link : <https://docs.python.org/3/library/tk.html>

2. Documentation of sys module used in the program:

Link: <https://docs.python.org/2/library/sys.html>

3. Documentation of random module used in the program:

Link: <https://docs.python.org/2/library/random.html>

**Conclusion:-**

In the conclusion of this project, I would like to say that Python is a fun and easy programming language and while creating a project like this, it has not just been a good experience but it also helped in the development of my creativity and logical thinking. I would be more than happy to work on other projects in Python because it’s just amazing to work with Python. The program is working and I hope, it’s also bug-free.

**Thank you for your attention**