Logo, company name

Description automatically generated

**Institute of Space Technology**

Date  
MARCH 29, 2023

SYED ABDULLAH ASHAR

ARTIFICIAL INTELLEGENCE

MS. RIDA

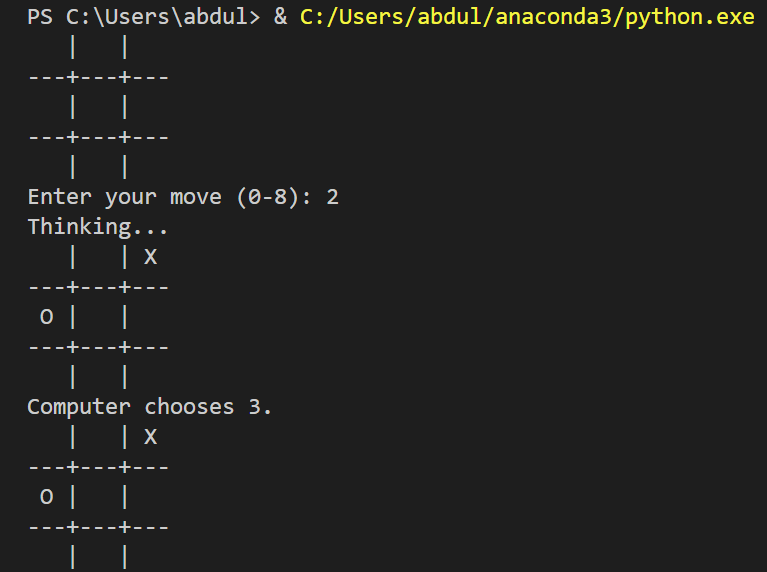
**ASSIGNMENT: 2**

A picture containing graphical user interface

Description automatically generated

MINIMAX ALGORITHM

**OUTPUT:**

****

**Text

Description automatically generated**

**Text

Description automatically generated**

**CODE:**

import random

# SYED ABDULLAH ASHAR    200901074      SEC B

# The board is represented as a list of strings, where each string

# represents a row on the board. A blank space is represented by ' '.

board = [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']

# The player is represented by 'X' and the computer is represented by 'O'.

player = 'X'

computer = 'O'

def get\_possible\_moves(board):

    moves = []

    for i in range(9):

        if board[i] == ' ':

            moves.append(i)

    return moves

def get\_score(board, player):

    # Check rows

    for i in range(0, 9, 3):

        if board[i] == board[i+1] == board[i+2] == player:

            return 1

    # Check columns

    for i in range(3):

        if board[i] == board[i+3] == board[i+6] == player:

            return 1

    # Check diagonals

    if board[0] == board[4] == board[8] == player:

        return 1

    if board[2] == board[4] == board[6] == player:

        return 1

    # No winner yet

    return 0

def make\_move(board, move, player):

    board[move] = player

    return board

# Returns the result of undoing the move on the board.

def undo\_move(board, move):

    board[move] = ' '

    return board

def get\_best\_move(board, player):

    if player == computer:

        # Maximizing player (computer)

        best\_score = -float('inf')

    else:

        # Minimizing player (user)

        best\_score = float('inf')

    possible\_moves = get\_possible\_moves(board)

    best\_move = possible\_moves[0]

    for move in possible\_moves:

        board = make\_move(board, move, player)

        if get\_score(board, player) == 1:

            # If the move results in a win, return it immediately

            undo\_move(board, move)

            return move

        if len(get\_possible\_moves(board)) == 0:

            # If the move results in a tie, score it as 0

            score = 0

        else:

            score = get\_best\_move(board, player=computer if player==player else player)

        undo\_move(board, move)

        if player == computer:

            if score > best\_score:

                best\_score = score

                best\_move = move

        else:

            if score < best\_score:

                best\_score = score

                best\_move = move

    return best\_move

# Prints the current board state.

def print\_board(board):

    print(f' {board[0]} | {board[1]} | {board[2]} ')

    print('---+---+---')

    print(f' {board[3]} | {board[4]} | {board[5]} ')

    print('---+---+---')

    print(f' {board[6]} | {board[7]} | {board[8]} ')

# The main game loop.

while True:

    # The user goes first.

    print\_board(board)

    # Check if the game is over.

    if get\_score(board, player) == 1:

        print(f'{player} wins!')

        break

    if len(get\_possible\_moves(board)) == 0:

        print('Tie game!')

        break

    # Get the user's move.

    while True:

        move = int(input('Enter your move (0-8): '))

        if move in get\_possible\_moves(board):

            board = make\_move(board, move, player)

            break

        else:

            print('Invalid move. Try again.')

    # Check if the game is over.

    if get\_score(board, player) == 1:

        print\_board(board)

        print(f'{player} wins!')

        break

    if len(get\_possible\_moves(board)) == 0:

        print\_board(board)

        print('Tie game!')

        break

    # Get the computer's move.

    print('Thinking...')

    move = get\_best\_move(board, computer)

    board = make\_move(board, move, computer)

    # Print the board and the computer's move.

    print\_board(board)

    print(f'Computer chooses {move}.')