Week 01

TIC TAC TOE

import math

import copy

X = "X"

O = "O"

EMPTY = None

def initial\_state():

    return [[EMPTY, EMPTY, EMPTY],

            [EMPTY, EMPTY, EMPTY],

            [EMPTY, EMPTY, EMPTY]]

def player(board):

    countO = 0

    countX = 0

    for y in [0, 1, 2]:

        for x in board[y]:

            if x == "O":

                countO = countO + 1

            elif x == "X":

                countX = countX + 1

    if countO >= countX:

        return X

    elif countX > countO:

        return O

def actions(board):

    freeboxes = set()

    for i in [0, 1, 2]:

        for j in [0, 1, 2]:

            if board[i][j] == EMPTY:

                freeboxes.add((i, j))

    return freeboxes

def result(board, action):

    i = action[0]

    j = action[1]

    if type(action) == list:

        action = (i, j)

    if action in actions(board):

        if player(board) == X:

            board[i][j] = X

        elif player(board) == O:

            board[i][j] = O

    return board

def winner(board):

    if (board[0][0] == board[0][1] == board[0][2] == X or board[1][0] == board[1][1] == board[1][2] == X or board[2][0] == board[2][1] == board[2][2] == X):

        return X

    if (board[0][0] == board[0][1] == board[0][2] == O or board[1][0] == board[1][1] == board[1][2] == O or board[2][0] == board[2][1] == board[2][2] == O):

        return O

    for i in [0, 1, 2]:

        s2 = []

        for j in [0, 1, 2]:

            s2.append(board[j][i])

        if (s2[0] == s2[1] == s2[2]):

            return s2[0]

    strikeD = []

    for i in [0, 1, 2]:

        strikeD.append(board[i][i])

    if (strikeD[0] == strikeD[1] == strikeD[2]):

        return strikeD[0]

    if (board[0][2] == board[1][1] == board[2][0]):

        return board[0][2]

    return None

def terminal(board):

    Full = True

    for i in [0, 1, 2]:

        for j in board[i]:

            if j is None:

                Full = False

    if Full:

        return True

    if (winner(board) is not None):

        return True

    return False

def utility(board):

    if (winner(board) == X):

        return 1

    elif winner(board) == O:

        return -1

    else:

        return 0

def minimax\_helper(board):

    isMaxTurn = True if player(board) == X else False

    if terminal(board):

        return utility(board)

    scores = []

    for move in actions(board):

        result(board, move)

        scores.append(minimax\_helper(board))

        board[move[0]][move[1]] = EMPTY

    return max(scores) if isMaxTurn else min(scores)

def minimax(board):

    isMaxTurn = True if player(board) == X else False

    bestMove = None

    if isMaxTurn:

        bestScore = -math.inf

        for move in actions(board):

            result(board, move)

            score = minimax\_helper(board)

            board[move[0]][move[1]] = EMPTY

            if (score > bestScore):

                bestScore = score

                bestMove = move

        return bestMove

    else:

        bestScore = +math.inf

        for move in actions(board):

            result(board, move)

            score = minimax\_helper(board)

            board[move[0]][move[1]] = EMPTY

            if (score < bestScore):

                bestScore = score

                bestMove = move

        return bestMove

def print\_board(board):

    for row in board:

        print(row)

# Example usage:

game\_board = initial\_state()

print("Initial Board:")

print\_board(game\_board)

while not terminal(game\_board):

    if player(game\_board) == X:

        user\_input = input("\nEnter your move (row, column): ")

        row, col = map(int, user\_input.split(','))

        result(game\_board, (row, col))

    else:

        print("\nAI is making a move...")

        move = minimax(copy.deepcopy(game\_board))

        result(game\_board, move)

    print("\nCurrent Board:")

    print\_board(game\_board)

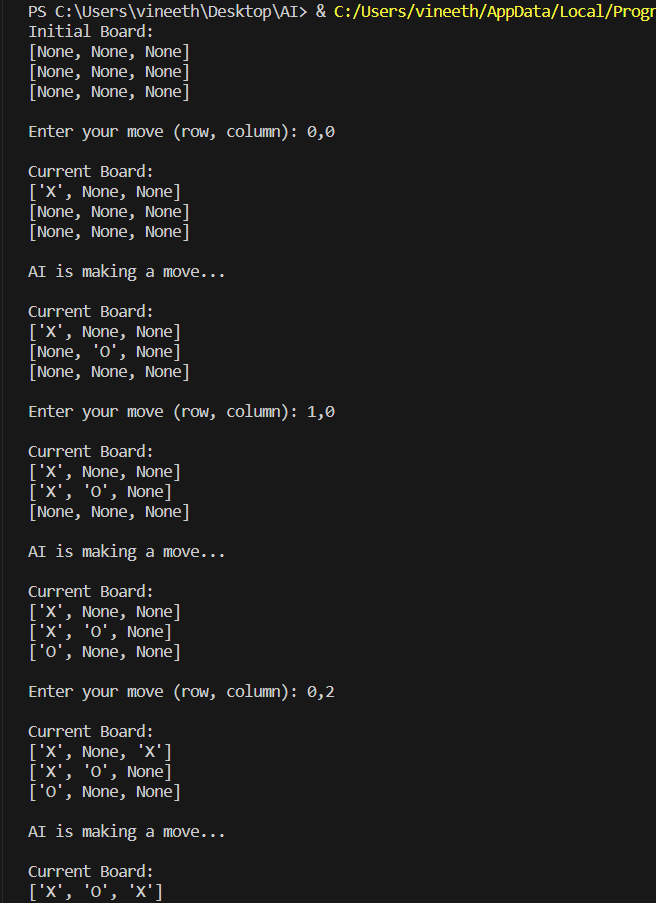
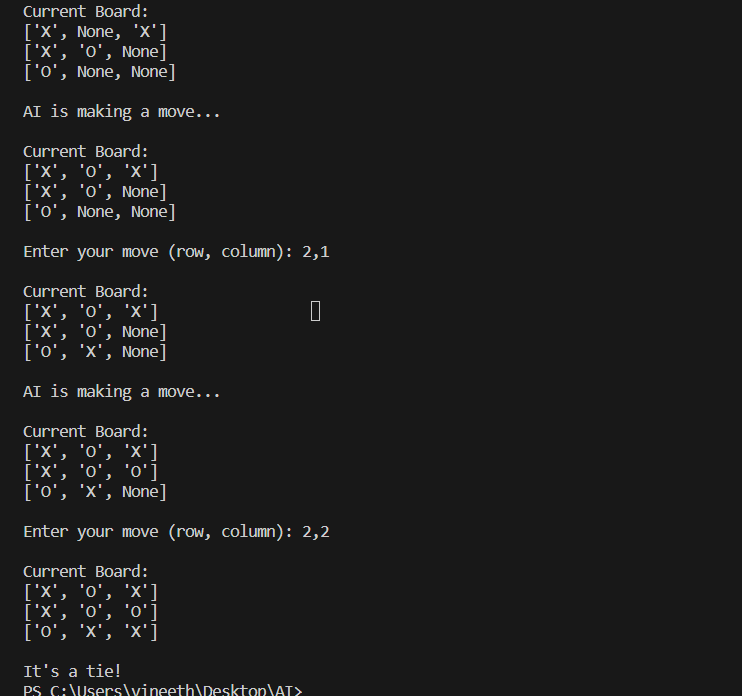
# Determine the winner

if winner(game\_board) is not None:

    print(f"\nThe winner is: {winner(game\_board)}")

else:

    print("\nIt's a tie!")

OUTPUT: