

```

from math import inf as infinity
from random import choice
import platform
import time
import numpy as np

```

Declaring the board below. Please uncomment 4x4 board to test for that size.

The functions written below work for nxn matrices but a lot of computation is involved.

```

COMP = 1
HUMAN = -1
'''BOARD = [
    [0, 0, 0, 0],
    [0, 0, 0, 0],
    [0, 0, 0, 0],
    [0, 0, 0, 0],
]'''
BOARD = [
    [0, 0, 0],
    [0, 0, 0],
    [0, 0, 0],
]

def isWin(state, player):
    '''
    Checks if either of the players have won the game
    to test for 4x4 uncomment the below list and comment the 3x3 list out
    '''
    '''win_states = [
        [state[0][0], state[0][1], state[0][2], state[0][3]],
        [state[1][0], state[1][1], state[1][2], state[1][3]],
        [state[2][0], state[2][1], state[2][2], state[2][3]],
        [state[3][0], state[3][1], state[3][2], state[3][3]],
        [state[0][0], state[1][0], state[2][0], state[3][0]],
        [state[0][1], state[1][1], state[2][1], state[3][1]],
        [state[0][2], state[1][2], state[2][2], state[3][2]],
        [state[0][3], state[1][3], state[2][3], state[3][3]],
        [state[0][0], state[1][1], state[2][2], state[3][3]],
        [state[0][3], state[1][2], state[2][1], state[3][0]],
    ]'''
    win_states = [
        [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[0][2], state[1][0], state[2][1]]
    ]

```

```

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

    win_array = []
    for i in range(len(BOARD)):
        win_array.append(player)

    if win_array in win_states:
        return True
    else:
        return False

def evaluate(state):
    if isWin(state, COMP):
        score = 1
    elif isWin(state, HUMAN):
        score = -1
    else:
        score = 0
    return score

def isGameOver(state):
    return isWin(state, COMP) or isWin(state, HUMAN)

def getEmptyCells(state):
    cells = []

    for i in range(len(state)):
        for j in range(len(state[i])):
            if state[i][j] == 0:
                cells.append([i, j])

    return cells

def isValidMove(x, y):
    if [x, y] in getEmptyCells(BOARD):
        return True
    else:
        return False

def setMove(x, y, player):
    if isValidMove(x, y):
        BOARD[x][y] = player
        return True
    else:
        return False

```

```

def minimax(state, depth, player):
    """
        Minimax is a Game Theory based algorithm that is frequently used in Reinforcement Learning.
        The computer considers all possible moves before a move is played. This ensures that
        optimal move is made always
    """
    if player == COMP:
        best = [-1, -1, -infinity]
    else:
        best = [-1, -1, +infinity]

    if depth == 0 or isGameOver(state):
        score = evaluate(state)
        return [-1, -1, score]

    for cell in getEmptyCells(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
        else:
            if score[2] < best[2]:
                best = score

    return best

def printBoard(c, h):
    piece = {
        1: c,
        -1: h,
        0: ' ',
    }

    print("-----"*len(BOARD))
    for i in range(len(BOARD)):
        for j in range(len(BOARD[i])):
            print("{} {}".format(piece[BOARD[i][j]]), end=' ')
        print()
    print("-----"*len(BOARD))

def ComputerTurn(c, h):
    print("Computer's turn : ")
    depth = len(getEmptyCells(BOARD))
    if depth == 0 or isGameOver(BOARD):
        return

```

```

if depth == len(BOARD)*len(BOARD):
    x = np.random.randint(len(BOARD))
    y = np.random.randint(len(BOARD))
else:
    move = minimax(BOARD, depth, COMP)
    x, y = move[0], move[1]

setMove(x, y, COMP)
printBoard(c, h)

def HumanTurn(c, h):
    print("Human's turn : ")
    depth = len(getEmptyCells(BOARD))
    if depth == 0 or isGameOver(BOARD):
        return

    emptyCells = getEmptyCells(BOARD)
    x = 0
    y = 0
    ''' print board here '''
    while True:
        x = int(input("Enter x coordinate : "))
        y = int(input("Enter y coordinate : "))

        if [x, y] in emptyCells:
            break

    setMove(x, y, HUMAN)
    printBoard(c, h)

def playGame(h_choice, c_choice, first):
    while len(getEmptyCells(BOARD)) > 0 and not isGameOver(BOARD):
        if first == 'N':
            ComputerTurn(c_choice, h_choice)
            first = ''

            HumanTurn(c_choice, h_choice)
            ComputerTurn(c_choice, h_choice)

    if isWin(BOARD, COMP):
        print("Computer Wins!")
    elif isWin(BOARD, HUMAN):
        print("Human Wins!")
    else:
        print("Draw!")

h_choice = input("HUMAN: What would you like to be? {X or O} : ").upper()

if h choice == 'X':

```

```

    _
    c_choice = 'O'
else:
    c_choice = 'X'

first = input("HUMAN: Would you like to go first? [Y/N2]: ").upper()
playGame(h_choice, c_choice, first)

```

```

↳ HUMAN: What would you like to be? {X or O} : x
HUMAN: Would you like to go first? [Y/N2]: n

```

Computer's turn :

```

-----
| O ||  ||  |
-----

```

```

|  ||  ||  |
-----

```

```

|  ||  ||  |
-----

```

Human's turn :

Enter x coordinate : 2

Enter y coordinate : 2

```

-----
| O ||  ||  |
-----

```

```

|  ||  ||  |
-----

```

```

|  ||  || X |
-----

```

Computer's turn :

```

-----
| O ||  || O |
-----

```

```

|  ||  ||  |
-----

```

```

|  ||  || X |
-----

```

Human's turn :

Enter x coordinate : 1

Enter y coordinate : 1

```

-----
| O ||  || O |
-----

```

```

|  || X ||  |
-----

```

```

|  ||  || X |
-----

```

Computer's turn :

```

-----
| O || O || O |
-----

```

```

|  || X ||  |
-----

```

```

|  ||  || X |
-----

```

Computer Wins!

