import math

# Constants

EMPTY = 0

HUMAN = 1

AI = 2

# Utility function to check if the game has ended

def is\_terminal(board):

# Check for rows

for i in range(3):

if board[i\*3] == board[i\*3+1] == board[i\*3+2] != EMPTY:

return True

# Check for columns

for i in range(3):

if board[i] == board[i+3] == board[i+6] != EMPTY:

return True

# Check for diagonals

if board[0] == board[4] == board[8] != EMPTY:

return True

if board[2] == board[4] == board[6] != EMPTY:

return True

# Check for tie

if EMPTY not in board:

return True

return False

# Utility function to evaluate the board state

def evaluate(board):

if board[0] == board[1] == board[2] == AI:

return 10

elif board[3] == board[4] == board[5] == AI:

return 10

elif board[6] == board[7] == board[8] == AI:

return 10

elif board[0] == board[3] == board[6] == AI:

return 10

elif board[1] == board[4] == board[7] == AI:

return 10

elif board[2] == board[5] == board[8] == AI:

return 10

elif board[0] == board[4] == board[8] == AI:

return 10

elif board[2] == board[4] == board[6] == AI:

return 10

elif board[0] == board[1] == board[2] == HUMAN:

return -10

elif board[3] == board[4] == board[5] == HUMAN:

return -10

elif board[6] == board[7] == board[8] == HUMAN:

return -10

elif board[0] == board[3] == board[6] == HUMAN:

return -10

elif board[1] == board[4] == board[7] == HUMAN:

return -10

elif board[2] == board[5] == board[8] == HUMAN:

return -10

elif board[0] == board[4] == board[8] == HUMAN:

return -10

elif board[2] == board[4] == board[6] == HUMAN:

return -10

else:

return 0

# MiniMax function

def minimax(board, depth, is\_maximizing):

if is\_terminal(board):

return evaluate(board)

if is\_maximizing:

best\_score = -math.inf

for i in range(9):

if board[i] == EMPTY:

board[i] = AI

score = minimax(board, depth + 1, False)

board[i] = EMPTY

best\_score = max(score, best\_score)

return best\_score

else:

best\_score = math.inf

for i in range(9):

if board[i] == EMPTY:

board[i] = HUMAN

score = minimax(board, depth + 1, True)

board[i] = EMPTY

best\_score = min(score, best\_score)

return best\_score

# Function to find the best move for the AI

def find\_best\_move(board):

best\_score = -math.inf

best\_move = None

for i in range(9):

if board[i] == EMPTY:

board[i] = AI

score = minimax(board, 0, False)

board[i] = EMPTY

if score > best\_score:

best\_score = score

best\_move = i

return best\_move

# Function to display the board

def display\_board(board):

for i in range(3):

row = ""

for j in range(3):

if board[i\*3+j] == EMPTY:

row += "- "

elif board[i\*3+j] == HUMAN:

row += "X "

else:

row += "O "

print(row)

# Main game loop

def play\_game():

board = [EMPTY] \* 9

current\_player = HUMAN

while not is\_terminal(board):

display\_board(board)

if current\_player == HUMAN:

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

while board[move] != EMPTY:

move = int(input("Invalid move. Try again: "))

board[move] = HUMAN

else:

move = find\_best\_move(board)

board[move] = AI

current\_player = HUMAN if current\_player == AI else AI

display\_board(board)

if evaluate(board) == 10:

print("AI wins!")

elif evaluate(board) == -10:

print("You win!")

else:

print("It's a tie!")

# Start the game

play\_game()