인공지능

(3주차 과제)

-Tic Tac Toe-

20164091

송희령

1 > 틱택토 코드(Com VS Com)

#게임판

game\_board = [' ',' ',' ',' ',' ',' ',' ',' ',' ']

틱택토의 게임판인 9칸의 배열 game\_board를 만들어 줍니다.

#비어있는 칸을 찾아서 리스트로 변환

def empty\_cells(board):

    cells = []

    for x, cell in enumerate(board):

        if cell == ' ':

            cells.append(x)

    return cells

#비어있는 칸에는 놓을 수 없다.

def valid\_move(x):

    return x in empty\_cells(game\_board)

#위치 x에 놓는다

def move(x,player):

    if valid\_move(x):

        game\_board[x] = player

        return True

    return False

게임의 O,X를 놓을 수 있도록 빈칸을 찾고 놓게끔 하는 함수들 입니다.

#현재 게임보드를 그린다

def draw(board):

    for i, cell in enumerate(board):

        if i%3 == 0:

            print('\n---------------')

        print('|',cell,'|',end='')

    print('\n---------------')

현재의 게임판을 출력하는 draw 함수 입니다.

#보드의 상태를 평가한다.

def evaluate(board):

    if check\_win(board,'X'):

        score = 1

    elif check\_win(board,'O'):

        score = -1

    else:

        score = 0

    return score

#1차원 리스트에서 동일한 문자가 수직선이나 수평선, 대각선으로 나타나면

#승리한 것으로 한다.

def check\_win(board,player):

    win\_conf=[

        [board[0],board[1],board[2]],

        [board[3],board[4],board[5]],

        [board[6],board[7],board[8]],

        [board[0],board[3],board[6]],

        [board[1],board[4],board[7]],

        [board[2],board[5],board[8]],

        [board[0],board[4],board[8]],

        [board[2],board[4],board[6]]

    ]

    return [player, player, player] in win\_conf

#X와 O중 승리자를 결정한다.

def game\_over(board):

    return check\_win(board,'X') or check\_win(board,'O')

보드의 승리 상태를 결정하는 함수와 보드의 승패, 그리고 승자를 가려주는 함수들 입니다.

#미니맥스 알고리즘을 구현한다.

#이 함수는 순환적으로 호출된다.

def minimax(board,depth,maxPlayer):

    pos = -1

    #단말 노드이면 보드를 평가하여 위치와 평가값을 반환한다.

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

        return -1, evaluate(board)

    if maxPlayer:

        value = -10000#음의 무한대

        for p in empty\_cells(board):

            board[p] = 'X'

            #경기자를 교체하여 minimax()를 순환호출환다.

            x, score = minimax(board,depth-1,False)

            board[p] = ' '

            if score > value:

                value = score

                pos = p

    else:

        value = +10000#양의 무한대

        #자식 노드를 하나씩 평가하여 최선의 수를 찾는다.

        for p in empty\_cells(board):

            board[p] = 'O' #보드의 p 위치에 O를 놓는다.

            #경기자를 교체하여 minimax()를 순환호출한다.

            x, score = minimax(board, depth-1, True)

            board[p] = ' '#보드는 원 상태로 돌린다.

            if score < value:

                value = score

                pos = p

    return pos, value

미니맥스 알고리즘을 이용하여 틱택토의 다음 수를 결정하게끔하는 코드입니다.

player = 'X'

#메인프로그램

while True:

    draw(game\_board)

    if len(empty\_cells(game\_board)) == 0 or game\_over(game\_board):

        break

    i, v = minimax(game\_board, 9, player == 'X')

    move(i,player)

    if player == 'X':

        player = 'O'

    else:

        player = 'X'

if check\_win(game\_board, 'X'):

    print("X 승리!")

elif check\_win(game\_board,'O'):

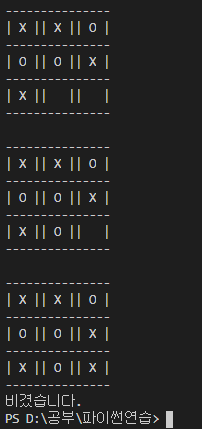
    print("O 승리!")

else:

    print("비겼습니다.")

이제 메인함수로 게임을 진행하도록 하는 코드입니다.

2 > 틱택토 결과(Com VS Com)

실행시 정상 작동하는 것을 볼 수 있습니다.

3 > 틱택토 코드(User VS Com)

위의 틱택토 코드와 유사점이 많아 변경점만 소개해 드리겠습니다.

#현재 게임보드를 그린다

def draw(board):

    for i, cell in enumerate(board):

        if i%3 == 0:

            print('\n---------------')

        print('|',cell,'|',end='')

    print('\n---------------')

#유저 전용 게임보드 그리기

def draw\_user(board):

    tmp = 1

    j = 0

    for i, cell in enumerate(board):

        if i%3 == 0:

            print('\n---------------' + "\t---------------")

        print('|',cell,'|',end='')

        j += 1

        if j == 3:

            for i in range(0,3):

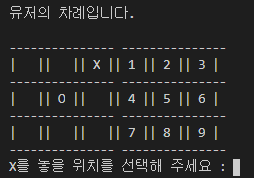
                print('|',tmp,'|',end='')

                tmp += 1

            j = 0

    print('\n---------------' + "\t---------------")

유저가 게임판을 보고 쉽게 입력할 수 있게끔 유저턴에서만 실행되는 draw함수를 새로 만들어 주었습니다.

출력시 게임판 옆으로 입력칸의 숫자칸이 보이게 됩니다.

해당 칸에 맞는 숫자 입력시 해당 칸에 놓게 됩니다.

#메인프로그램

while True:

    draw(game\_board)

    if len(empty\_cells(game\_board)) == 0 or game\_over(game\_board):

        break

    if player == 'X':

        print("\n유저의 차례입니다.")

        draw\_user(game\_board)

        a = input("X를 놓을 위치를 선택해 주세요 : ")

        if game\_board[int(a) - 1] != " " :

            print("다시 선택해 주세요.")

            continue

        game\_board[int(a) - 1] = 'X'

        player = 'O'

    else:

        i, v = minimax(game\_board, 9, player == 'X')

        move(i,player)

        player = 'X'

if check\_win(game\_board, 'X'):

    print("유저 승리!")

elif check\_win(game\_board,'O'):

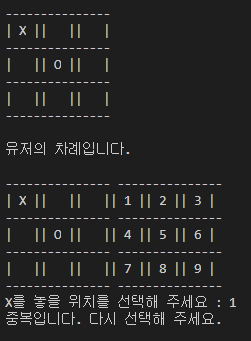
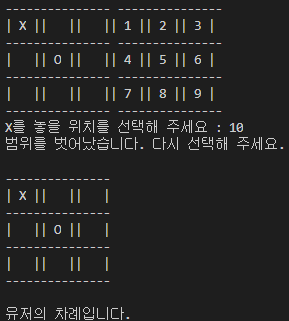
    print("컴퓨터 승리!")

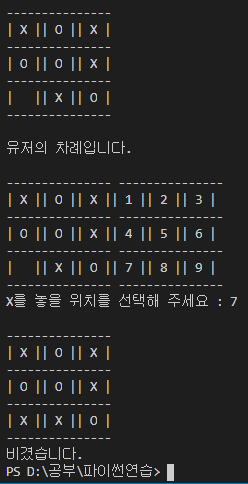
else:

    print("비겼습니다.")

메인 부분에선 유저턴과 컴퓨터 턴에 맞게 실행 방법을 다르게 주었습니다. 컴퓨터 턴에서만 알고리즘이 작동하게 하고, 유저턴에선 말판 입력 부분을 그대로 컴퓨터에게 전달하도록 하였습니다. 또한 중복 입력 방지를 위해 조건문도 설정해 주었습니다.

4 > 틱택토 결과(User VS Com)

출력 결과 정상 작동하는 것을 확인할 수 있습니다.