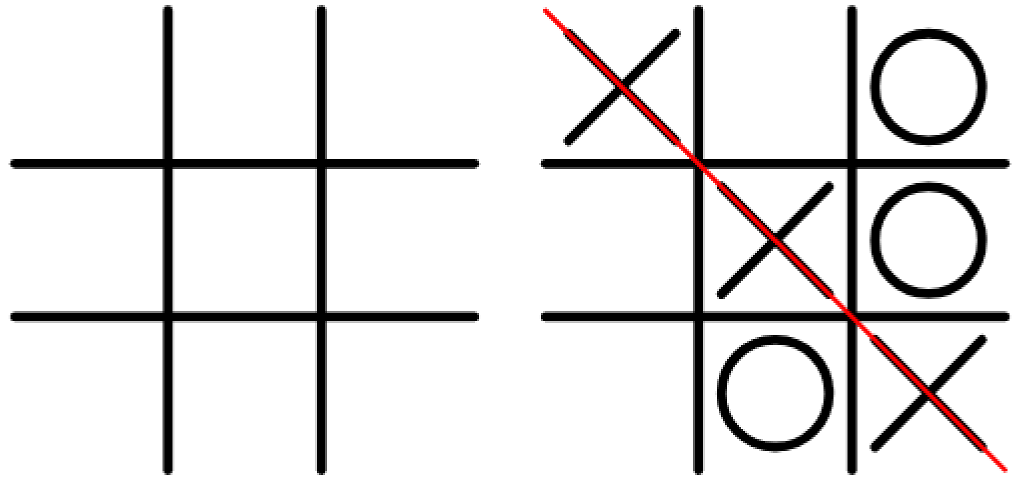
Lab 7. Tic Tac Toe

Yongjae Yoo, Ph. D.
Assistant Professor
Department of Artificial Intelligence
Hanyang University ERICA

Tic Tac Toe

- A board game on 3x3 grid
  - Two players plays with X or O marks.
  - Either side can win if makes a single line of symbols.



Today,

- We will start implementing a text-based Tic-Tac-Toe first,
- Then we will test a random pick AI
- And build your best strategy on the code.

TicTacToe.ipynb

- Codes

```python
from IPython.display import clear_output

def display_board(board):
    clear_output()
    print('   |   |')
    print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
    print('   |   |')
    print('---|---|---')
    print('   |   |')
    print(' ' + board[4] + ' | ' + board[5] + ' | ' + board[6])
    print('   |   |')
    print('---|---|---')
    print('   |   |')
    print(' ' + board[1] + ' | ' + board[2] + ' | ' + board[3])
    print('   |   |')
```

TicTacToe.ipynb

- Codes
  - (Note that we will use “input”)

```
def player_input():
    marker = ''
    while not (marker == 'O' or marker == 'X'):
        marker = input('Player: Do you want to be X or O? ').upper()

    if marker == 'X':
        return ('X', 'O')
    else:
        return ('O', 'X')

def place_marker(board, marker, position):
    board[position] = marker
```

TicTacToe.ipynb

- Win check part

```python
def win_check(board, mark):
    return ((board[7] == mark and board[8] == mark and board[9] == mark) or
            (board[4] == mark and board[5] == mark and board[6] == mark) or
            (board[1] == mark and board[2] == mark and board[3] == mark) or
            (board[7] == mark and board[4] == mark and board[1] == mark) or
            (board[8] == mark and board[5] == mark and board[2] == mark) or
            (board[9] == mark and board[6] == mark and board[3] == mark) or
            (board[7] == mark and board[5] == mark and board[3] == mark) or
            (board[9] == mark and board[5] == mark and board[1] == mark))
```

TicTacToe.ipynb

- Game functions – deciding who is first

```
import random

def choose_first():
    if random.randint(0,1) == 0:
        return 'Player'
    else:
        return 'AI'

def space_check(board, position):
    return board[position] == ' '


def replay():
    return input('Do you want to play again? Enter Yes or No - ').lower().startswith('y')
```

TicTacToe.ipynb

- Check whether the board is full or not and gets player’s input.

```
def full_board_check(board):
    for i in range(1,10):
        if space_check(board,i):
            return False
    return True

def player_choice(board):
    while True:
      position = input('Choose your next position (1-9)  ')
      if (space_check(board, int(position))):
        return int(position)
```

TicTacToe.ipynb

- Starting the game

```python
print('Welcome to Tic Tac Toe!')

while True:
    theBoard = [' ']*10
    player1_marker, player2_marker = player_input()
    turn = choose_first()
    print(turn + ' will go First')

    game_on = True
```

Gets input and making Decisions

```
while game_on:
        if turn == 'Player':
            display_board(theBoard)
            position = player_choice(theBoard)
            place_marker(theBoard, player1_marker, position)

            if win_check(theBoard, player1_marker):
                display_board(theBoard)
                print('Congratulations!!! \nPlayer WON the game.')
                game_on = False
            else:
                if full_board_check(theBoard):
                    display_board(theBoard)
                    print('The game is a DRAW!')
                    break
                else:
                    turn = 'AI'
```

AI’s part – Currently Player 2’s role

```
        else:
            display_board(theBoard)
            position = player_choice(theBoard)
            #position = AI_run(theBoard)
            place_marker(theBoard, player2_marker, position)

            if win_check(theBoard, player2_marker):
                display_board(theBoard)
                print('AI WON the game.')
                game_on = False
            else:
                if full_board_check(theBoard):
                    display_board(theBoard)
                    print('The game is a DRAW!')
                    break
                else:
                    turn = 'Player'
    if not replay():
        break
```

Current AI

- If we remove # on `#position = AI_run(theBoard)` and comment out `position = player_choice(theBoard),`
- AI will work – current AI is quite naïve!

```
def AI_run(board):
    #check the board and just random pick
    while True:
      x = random.randint(1,9);
      if(space_check(board, x)):
        break
      else:
        continue

    #find the best one by your logic
    return x
```

Lab Work – Can you make it better?

- Instead of random, you can utilize how to pick the best strate gy in the code AI_run() function.

- You can first identify which “decision(s)” should be made.