Simulate a tic-tac-toe game and look at strategies for winning

Picture with rows and columns. 3x3 board - put crosses and noughts - when do you win?

You win when you have row of column or diagnol with same symbol. ¶

Q1) build our board - createBoard()

In [1]:
```
import numpy as np
def createBoard():
    board = np.zeros((3,3))
    return board
print(createBoard())
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

Q2) assume there are two players- player 1 and player 2 - will take their turns to change the value of the array from 0 to a 1 or 2. This indicates the player that makes that move

In [2]:
```
def placeMarker(board, player, position):
    ####################################
    #board - 3x3 board we just created #
    #player - either 1 or 2            #
    #position - touple of 2            #
    ####################################
    #print(isTaken(board,position))
    #while isTaken(board, position)==True:
    #    pos_y = input('Please input a new row, the one you entered is t
aken')
    #    pos_x = input('Now please enter the column')
    #    position=(pos_y, pos_x)
    check=board[int(position[0])][int(position[1])]
    if check ==0:
        board[int(position[0])][int(position[1])]=player
    else:
        print('This spot is taken')
    return board
```

In [3]:
```
def playGame():
    board=createBoard()
    while checkWin(board):
        print(possiblePositions(board))
        place = tuple(input('Where do you want to go, Player 1: '))
        placeMarker(board, 1, place)
        print(possiblePositions(board))
        place = tuple(input('Where do you want to go, Player 2: '))
        placeMarker(board, 2, place)
```

Homework- create a function called possiblePositions - this methods returns a list of all available positions(tuples) on the board that are available; not occupied

and

make one modification to the placeMarker and called v1

In [4]:
```
def possiblePositions(board):
    listOfCoor=[]
    row=0
    column=0
    for x in board:
        for y in x:
            if y==0:
                listOfCoor.append((row,column))
            column+=1
        row+=1
        column=0
    return listOfCoor
boards=createBoard()
print(possiblePositions(boards))
```

```
[(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2,
2)]
```

In [5]:
```
def isTaken(board, position):
    for x in possiblePositions(board):
        if position == x:
            return True
    return False
```

In [6]:
```
def diag_win(board, player):
    if np.all(np.diag(board)==player) or np.all(np.diag(np.fliplr(board
))==player):
        return True
    else:
        return False
```

In [7]:
```
def across_win(board, player):
    for x in range(0,2):
        if board[x][0] == player and board[x][1] == player and board[x][
2] == player:
                return True
    return False
```

In [8]:
```
def down_win(board, player):
    for y in range(0,2):
        if board[0][y] == player and board[1][y] == player and board[2][
y] == player:
            return True
    return False
```

In [9]:
```
def board_full(board):
    zeros=0
    for row in board:
        for cell in row:
            if cell == 0:
                zeros+=1
    if zeros==0:
        return True
    else:
        return False
```

In [ ]:
```
def checkWin(board):
    for i in range(1,2):
        if diag_win(board,i) or across_win(board,i) or down_win(board,i
):
            print('Player '+str(i)+' has won!!!')
            return False
    if board_full(board):
        print('It is a tie!!!')
        return False
    return True
```

In [ ]:
```
playGame()
```

```
[(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2,
2)]
[(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2,
2)]
Where do you want to go, Player 1: 00
[(0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)]
Where do you want to go, Player 2: 10
[(0, 1), (0, 2), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)]
Where do you want to go, Player 1: 01
[(0, 2), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)]
Where do you want to go, Player 2: 11
[(0, 2), (1, 2), (2, 0), (2, 1), (2, 2)]
Where do you want to go, Player 1: 02
[(1, 2), (2, 0), (2, 1), (2, 2)]
Where do you want to go, Player 2: 12
Player 1 has won!!!
[(2, 0), (2, 1), (2, 2)]
Where do you want to go, Player 1: 22
[(2, 0), (2, 1)]
```

In [ ]:
```
board=createBoard()
board[0][1]
position=(0,0)
position[0]
board[position[0]][position[1]]
```

In [ ]:
```
position = tuple(input(''))
```

In [ ]:
```
position[0]
```

In [ ]:
```

```