In [ ]:
```
#Written by Thomas Fouts, Fran Miguens, and Andrew Shaffer on 10/17
#Compiled by Thomas Fouts on 10/18
#Run the code by running the last cell
```

In [123]:
```
import numpy as np
```

In [176]:
```
def playConnect4(): #Written by Andrew Shaffer and Fran Miguens
#Shaffer wrote outer loop, Fran wrote inner loops

    SENTINEL = -1
    player = 'X' #Set the player as 'X'
    board = makeBoard() #Create the board
    playing = True #Set variable to control how many games we play
    score = [0,0] #Create a list to keep track of score
    while playing: #Continue to play until loop is ended
        print('')
        print(board) #Print the board

        while True: #Inner loop written by Fran
            print('It is '+player+'s turn') #Print out whose turn it is
            spot = int(input('Enter the row you would like to place a pi
ece at, or -1 to quit')) #Take an input for where the player wants to go
            if(spot == SENTINEL): #Allow for the user to enter the game
                break
            if(placePiece(board, spot, player)): #Place the piece, and u
se the boolean result to determine if a piece was placed
                if (player =='X'): #Swap the players
                    player ='O'
                else:
                    player = 'X'
            print(board) #Print out the new board
            winner = getSubMatrix(board) #Call the getSubMatrix method,
 save the result as a list

            if(winner[0]): #If the first element of the winner list is t
rue, end the game
                if(winner[1]=='X'): #If someone one, check to see who wo
n
                    score[0]+=1 #Add to the scores list accordingly
                else:
                    score[1]+=1
                break #Break the inner loop if someone wins

        #Outer loop written by Shaffer
        getScore(score) #Call the get Score method to get the score
        board = makeBoard() #Make a new board for the next game
        playAgain = str(input('Would you like to play again?')).upper()
#Ask the user if they want to play again
        if(playAgain == 'NO'): #If they dont want to play again
            print('Thank you for playing')
            getSeries(score) #Call the get Series method when the series
is over
            break #Break the outer loops
```

In [154]:
```
def getScore(score): #Get score method written by Thomas
    print('X has won '+str(score[0])+' games, and O has won '+str(score[
1])+' games') #Print out how many wins each player has
```

In [158]:
```
def getSeries(score): #Get Series method written by Shaffer
    if(score[0]>score[1]): #Check who has the most wins, and print out t
he winner
        print('X won the series')
    elif(score[1]>score[0]):
        print('O won the series')
    else:
        print('X and O tied the series')
```

In [126]:
```
def makeBoard(): #Written by Fran Miguens
    board = np.full((7,7),' ') #Create a 7x7 array of empty spaced
    for i in range(7): #Iterate through the top row
        board[0][i] = str(i) #Label the rows with a string
    return(board) #Return the board
```

In [162]:
```
def placePiece(board, spot, player): #Written by Fouts and Shaffer
    if(spot<0 or spot>6): #Written by Andrew Shaffer - Check to make sur
e that Ali doesnt enter an invalid number
        print('Please enter a valid spot')  #Checks to make sure that th
e spot entered actually exists
        return False #If it does not exist, return false so that player
 can go again

    if(board[1][spot]!=' '): #Check to see if the row is full, and retur
n false if it is
        print('That row is full')
        return False

    for i in range(2,7): #Written by Thomas Fouts- Iterate through each
 row, checking the column 'spot' to see if it is taken
        if(board[i][spot]!=' '): #As soon as the spot is full, the previ
os spot must be the last free spot
            board[i-1][spot]= player #Place the piece is the previous sp
ot
            return True # Return true to signify that a piece had been p
laced

    board[6][spot] = player #If all of spots are empty, place the piece
 in the bottom spot
    return True #Return true
```

In [128]:
```
def getSubMatrix(board): #Written by Thomas Fouts
    for i in range(0, 4): #Iterates through each of the columns
        for j in range(0,4): #Iterates through each of the rows
            winner = checkWin(board[i:i+4, j:j+4]) #Calls Frans method w
ith a 4x4 matrix, and loops through all possible matrix's
            if(winner[0]): #If Fran's method returns that there was a wi
nner
                print('Congradulation, '+str(winner[1])+'s win!') #Print
the congradulations statement
                return [True,winner[1]] #Return true so the main method
 stops running
    return [False,'null']
```

In [129]:
```
def checkWin(board): #Written by Fran Miguens
    #Default
    winner=((False,''))
    #Checks diagonals and returns winner if true
    if np.all(np.diag(board)=='X') or np.all(np.diag(np.fliplr(board))==
'X'):
        winner=((True,'X')) #Return a tuple of if someone won and who wo
n
        return winner
    elif np.all(np.diag(board)=='O') or np.all(np.diag(np.fliplr(board))
=='O'):
        winner=((True, 'O'))
        return winner
    #Checks rows and columns
    for x in range(0,3):
        #Checks Rows
        if board[x][0] == 'X' and board[x][1] == 'X' and board[x][2] ==
'X' and board[x][3]=='X':
            winner=((True,'X'))
        elif board[x][0] == 'O' and board[x][1] == 'O' and board[x][2] =
= 'O' and board[x][3]=='O':
            winner=((True,'O'))
        #Checks Columns
        elif board[0][x] == 'X' and board[1][x] == 'X' and board[2][x] =
= 'X' and board[3][x]=='X':
            winner=((True, 'X'))
        elif board[0][x] == 'O' and board[1][x] == 'O' and board[2][x] =
= 'O' and board[3][x]=='O':
            winner=((True, 'O'))
    #returns winner
    return winner
```

In [177]:
```
playConnect4()
```

```
[['0' '1' '2' '3' '4' '5' '6']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']]
It is Xs turn
Enter the row you would like to place a piece at, or -1 to quit2
[['0' '1' '2' '3' '4' '5' '6']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' 'X' ' ' ' ' ' ' ' ']]
It is Os turn
Enter the row you would like to place a piece at, or -1 to quit3
[['0' '1' '2' '3' '4' '5' '6']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' 'X' 'O' ' ' ' ' ' ']]
It is Xs turn
Enter the row you would like to place a piece at, or -1 to quit2
[['0' '1' '2' '3' '4' '5' '6']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' 'X' ' ' ' ' ' ' ' ']
 [' ' ' ' 'X' 'O' ' ' ' ' ' ']]
It is Os turn
Enter the row you would like to place a piece at, or -1 to quit3
[['0' '1' '2' '3' '4' '5' '6']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' 'X' 'O' ' ' ' ' ' ']
 [' ' ' ' 'X' 'O' ' ' ' ' ' ']]
It is Xs turn
Enter the row you would like to place a piece at, or -1 to quit2
[['0' '1' '2' '3' '4' '5' '6']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' 'X' ' ' ' ' ' ' ' ']
 [' ' ' ' 'X' 'O' ' ' ' ' ' ']
 [' ' ' ' 'X' 'O' ' ' ' ' ' ']]
It is Os turn
Enter the row you would like to place a piece at, or -1 to quit3
[['0' '1' '2' '3' '4' '5' '6']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' 'X' 'O' ' ' ' ' ' ']
 [' ' ' ' 'X' 'O' ' ' ' ' ' ']
 [' ' ' ' 'X' 'O' ' ' ' ' ' ']]
It is Xs turn
Enter the row you would like to place a piece at, or -1 to quit2
[['0' '1' '2' '3' '4' '5' '6']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' ' ' ' ' ' ' ' ' ' ']
 [' ' ' ' 'X' ' ' ' ' ' ' ' ']
 [' ' ' ' 'X' 'O' ' ' ' ' ' ']
 [' ' ' ' 'X' 'O' ' ' ' ' ' ']
 [' ' ' ' 'X' 'O' ' ' ' ' ' ']]
Congradulation, Xs win!
X has won 1 games, and O has won 0 games
Would you like to play again?no
Thank you for playing
X won the series
```

```
In [ ]:
```