Lab-1 Tic-Tac-Toe bot

→ Pseudocode:

- Function to determine winner of a given board

```
def winner (board):
    d1 = {set([board(i)[i] for i in range(3)])
    d2 = set([board[2-i][i] for i in range(3)])
    rows = [set([board[i][j] for j in range(3)])
            for i in range(3)]
    Col = [set([board[j](i)] for j in range(3)])
            for i in range(3)]
    if d1 == {'x'} or d2 == {'x'} or
        {'x'} in rows or {'x'} in cols:
        winner = 'x'
    elif d1 == {'0'} or d2 = {'0'} or {'0'} in
        rows or {'0'} in cols:
        winner = '0'
    else:
        return None
```

~~Determine a score for~~

- To determine next move of bot:

```
~~def function~~ def CheckMove(board, ~~[illegible]~~, bot):
    for move in possibleMoves(board, player)
    boardcopy = ~~if~~ applyMove (board)
        if winner(boardcopy) == bot:
            return move
        elif winner(boardcopy) == player:
            ~~ret~~ ~~Continue~~ return
        else:
            return CheckMove(boardcopy, ~~move~~, bot)
    if terminal(board):
        return None
```

- Initial Condition: Empty 3x3 array with all cells filled with None
- Final Condition: A final move (integer b/w 0-8) that the bot can play on the existing board

- Utility Function

```
def applyMove (board, player~~X~~, move):
    boardcopy = board.deepcopy()
    boardcopy [move//3] [move % 3] = player
    return boardCopy

def Terminal (board):
    if ~~(board~~ winner (board) is not None ~~None~~ : ~~or~~ ~~and~~
    ~~(winner (board)~~) return True
    ~~else~~ elif winner (board) is None:
        for i in board
            if None in i:
                return False
    return True
```

- ~~def~~ Initialization Code:

```
board = [[None ~~for~~ _ in range(3)] for _ in range(3)]

~~player = 'O'~~
~~bot = 'X'~~

~~# Apply move inputted~~
```

Date ________
Page ________

```
def possible Moves (board): ~~[ ans=[]~~
                                  ans = []
    ~~for i in range(3):~~
        ~~for j in range(3):~~
            ~~if board[i][j] == None:~~
                ~~ans.append (~~
    for i in range (9):
        if board [i//3] [i%3] == None:
            ans.append (i)
    return ans
```

- Initialization Code:

```
    board = [[None for _ in range(3)] for _ in
                                      range (3)]

    Player = 'x'
    bot = 'o'
    Current = 0
    ~~if Current %2 ==0:~~ while not terminal(board):
        ~~apply Move (~~
        if Current % 2 == 0:    # Player
            ~~move apply Mo~~
            board = apply Move (int(input("Move: "))
            Current ~~++~~ += 1
        else:
            ~~board = apply Move (~~
            board = apply Move ( Check Move (board, bot))
        ~~if win if winner (board~~ Current += 1
        ~~print (f" {} win~~
        if winner (board) != None:
            Print ( f" {} wins!", winner (board))
        else:
            Continue
```

[illegible]

Date
Page

O/p:

```
None    None    None
None    None    None
None    None    None

X       N       N
N       N       N
N       N       N

Enter move (0-8) : 2.
X       N       O
N       N       N
N       N       N

X       N       O
X       N       N
N       N       N

Enter move (0-8) :    4
X       N       O
X       O       N
N       N       N

X wins !
```

[illegible]