In [2]:

```
# Tic-Tac-Toe Program using
# random number in Python

# importing all necessary libraries
import numpy as np
import random
from time import sleep

# Creates an empty board


def create_board():
    return(np.array([[0, 0, 0],
                     [0, 0, 0],
                     [0, 0, 0]]))

# Check for empty places on board


def possibilities(board):
    l = []

    for i in range(len(board)):
        for j in range(len(board)):

            if board[i][j] == 0:
                l.append((i, j))
    return(l)

# Select a random place for the player


def random_place(board, player):
    selection = possibilities(board)
    current_loc = random.choice(selection)
    board[current_loc] = player
    return(board)

# Checks whether the player has three
# of their marks in a horizontal row

```

```
42
43 def row_win(board, player):
44     for x in range(len(board)):
45         win = True
46
47         for y in range(len(board)):
48             if board[x, y] != player:
49                 win = False
50                 continue
51
52         if win == True:
53             return(win)
54     return(win)
55
56 # Checks whether the player has three
57 # of their marks in a vertical row
58
59
60 def col_win(board, player):
61     for x in range(len(board)):
62         win = True
63
64         for y in range(len(board)):
65             if board[y][x] != player:
66                 win = False
67                 continue
68
69         if win == True:
70             return(win)
71     return(win)
72
73 # Checks whether the player has three
74 # of their marks in a diagonal row
75
76
77 def diag_win(board, player):
78     win = True
79     y = 0
80     for x in range(len(board)):
81         if board[x, x] != player:
82             win = False
83     if win:
```

```
            return win
        win = True
        if win:
            for x in range(len(board)):
                y = len(board) - 1 - x
                if board[x, y] != player:
                    win = False
        return win

# Evaluates whether there is
# a winner or a tie


def evaluate(board):
    winner = 0

    for player in [1, 2]:
        if (row_win(board, player) or
                col_win(board, player) or
                diag_win(board, player)):

            winner = player

    if np.all(board != 0) and winner == 0:
        winner = -1
    return winner

# Main function to start the game


def play_game():
    board, winner, counter = create_board(), 0, 1
    print(board)
    sleep(2)

    while winner == 0:
        for player in [1, 2]:
            board = random_place(board, player)
            print("Board after " + str(counter) + " move")
            print(board)
            sleep(2)
            counter += 1
```

```
            winner = evaluate(board)
            if winner != 0:
                break
    return(winner)


# Driver Code
print("Winner is: " + str(play_game()))

```

```
[[0 0 0]
 [0 0 0]
 [0 0 0]]
Board after 1 move
[[0 0 0]
 [1 0 0]
 [0 0 0]]
Board after 2 move
[[0 0 0]
 [1 0 0]
 [0 0 2]]
Board after 3 move
[[0 0 1]
 [1 0 0]
 [0 0 2]]
Board after 4 move
[[0 2 1]
 [1 0 0]
 [0 0 2]]
Board after 5 move
[[0 2 1]
 [1 1 0]
 [0 0 2]]
Board after 6 move
[[2 2 1]
 [1 1 0]
 [0 0 2]]
Board after 7 move
[[2 2 1]
 [1 1 0]
 [0 1 2]]
Board after 8 move
[[2 2 1]
 [1 1 2]
 [0 1 2]]
Board after 9 move
[[2 2 1]
 [1 1 2]
 [1 1 2]]
Winner is: 1
```

In [ ]:
```
1
```