We are implementing Tic-Tac-Toe using Python

Coding Part 1: Importing Libraries and Creating Empty Board

```
# Importing all necessary libraries for this project

import numpy as np
import random
from time import sleep

# Creating an empty Tic-tc-toe board

def empty_board():
  board = np.array([
                    [0,0,0],
                    [0,0,0],
                    [0,0,0]
                   ])
  return(board)

print(empty_board())
```

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

Coding Part 2: Check for empty places on board + Randomly place Xs (1) and Os (2).

```
# Check for empty places on the Tic-Tac-Toe Board

def empty_places(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)

board = np.array([

                [1,2,1],
                [0,0,0],
                [1,1,2]

                ])

#print(empty_places(board))

# Select a random place for the player on the Tic-Tac-Toe board.

def random_place(board, player):
  select = empty_places(board)
  current_location = random.choice(select)
  board[current_location] = player
  return(board)

print(random_place(board, 2))
```

```
[[1 2 1]
 [2 0 0]
 [1 1 2]]
```

Coding Part 3: Evaluate for a Winner or a Tie

```
# Checks whether the player has three of their marks in a horizontal row
def row_winner(board, player):
    for x in range(len(board)):
        win = True

        for y in range(len(board)):
            if board[x, y] != player:
                win = False
                continue

        if win == True:
            return(win)
    return(win)

# Checks whether the player has three of their marks in a vertical row
def col_winner(board, player):
    for x in range(len(board)):
        win = True

        for y in range(len(board)):
            if board[y][x] != player:
                win = False
                continue

        if win == True:
            return(win)
    return(win)
```

```
# Check the diagnal rows for a winner

def diag_winner(board, player):
  win = True
  y = 0
  for x in range(len(board)):
    if board[x, x] != player:
      win = False

  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

board = np.array([
                  [0,0,1],
                  [0,1,0],
                  [0,0,0]
])

print(diag_winner(board, 1))
```

```
False
```

```
# Evaluates Whether there is a winner or a Tie

def evaluate_game(board):
  # Winner [0 = indecisive; 1 = Player 1; 2 = Player 2; -1 = Tie]
  winner = 0
  for player in [1, 2]:
        if (row_winner(board, player) or
                col_winner(board, player) or
                diag_winner(board, player)):

            winner = player

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

Coding Part 4: Main function to run/start the game

```
def tic_tac_toe():
    board = empty_board()
    winner = 0
    counter = 1
    print(board)
    sleep(5)

    while winner == 0:
        for player in [1, 2]:
            brd = random_place(board, player)
            print("Board after " + str(counter) + " move")
            print(brd)
            sleep(2)
            counter += 1
            winner = evaluate_game(brd)
            if winner != 0:
                break
    return(winner)


# Driver Code
print("Winner is player: " + str(tic_tac_toe()))
```

```
[[0 0 0]
 [0 0 0]
 [0 0 0]]
Board after 1 move
[[0 0 0]
 [0 0 0]
 [1 0 0]]
Board after 2 move
[[0 0 0]
 [0 0 0]
 [1 0 2]]
Board after 3 move
[[0 0 0]
 [0 0 0]
 [1 1 2]]
Board after 4 move
[[0 0 2]
 [0 0 0]
 [1 1 2]]
Board after 5 move
```

```
[[0 0 2]
 [1 0 0]
 [1 1 2]]
Board after 6 move
[[2 0 2]
 [1 0 0]
 [1 1 2]]
Board after 7 move
[[2 0 2]
 [1 1 0]
 [1 1 2]]
Board after 8 move
[[2 2 2]
 [1 1 0]
 [1 1 2]]
Winner is player: 2
```