**Practical Number -7**

**TIC-TAC-TOE**

**Algorithm**

* Create a board using a 2-dimensional array and initialize each element as empty.
  + You can represent empty using any symbol you like. Here, we are going to use a hyphen. '-'.
* Write a function to check whether the board is filled or not.
  + Iterate over the board and return false if the board contains an empty sign or else return true.
* Write a function to check whether a player has won or not.
  + We have to check all the possibilities that we discussed in the previous section.
  + Check for all the rows, columns, and two diagonals.
* Write a function to show the board as we will show the board multiple times to the users while they are playing.
* Write a function to start the game.
  + Select the first turn of the player randomly.
  + Write an infinite loop that breaks when the game is over (either win or draw).
    - Show the board to the user to select the spot for the next move.
    - Ask the user to enter the row and column number.
    - Update the spot with the respective player sign.
    - Check whether the current player won the game or not.
    - If the current player won the game, then print a winning message and break the infinite loop.
    - Next, check whether the board is filled or not.
    - If the board is filled, then print the draw message and break the infinite loop.
  + Finally, show the user the final view of the board.

### Code

import random

class TicTacToe:

    def \_\_init\_\_(self):

        self.board = []

    def create\_board(self):

        for i in range(3):

            row = []

            for j in range(3):

                row.append('-')

            self.board.append(row)

    def get\_random\_first\_player(self):

        return random.randint(0, 1)

    def fix\_spot(self, row, col, player):

        self.board[row][col] = player

    def is\_player\_win(self, player):

        win = None

        n = len(self.board)

        # checking rows

        for i in range(n):

            win = True

            for j in range(n):

                if self.board[i][j] != player:

                    win = False

                    break

            if win:

                return win

        # checking columns

        for i in range(n):

            win = True

            for j in range(n):

                if self.board[j][i] != player:

                    win = False

                    break

            if win:

                return win

        # checking diagonals

        win = True

        for i in range(n):

            if self.board[i][i] != player:

                win = False

                break

        if win:

            return win

        win = True

        for i in range(n):

            if self.board[i][n - 1 - i] != player:

                win = False

                break

        if win:

            return win

        return False

        for row in self.board:

            for item in row:

                if item == '-':

                    return False

        return True

    def is\_board\_filled(self):

        for row in self.board:

            for item in row:

                if item == '-':

                    return False

        return True

    def swap\_player\_turn(self, player):

        return 'X' if player == 'O' else 'O'

    def show\_board(self):

        for row in self.board:

            for item in row:

                print(item, end=" ")

            print()

    def start(self):

        self.create\_board()

        player = 'X' if self.get\_random\_first\_player() == 1 else 'O'

        while True:

            print(f"Player {player} turn")

            self.show\_board()

            # taking user input

            row, col = list(

                map(int, input("Enter row and column numbers to fix spot: ").split()))

            print()

            # fixing the spot

            self.fix\_spot(row - 1, col - 1, player)

            # checking whether current player is won or not

            if self.is\_player\_win(player):

                print(f"Player {player} wins the game!")

                break

            # checking whether the game is draw or not

            if self.is\_board\_filled():

                print("Match Draw!")

                break

            # swapping the turn

            player = self.swap\_player\_turn(player)

        # showing the final view of board

        print()

        self.show\_board()

# starting the game

tic\_tac\_toe = TicTacToe()

tic\_tac\_toe.start()

### OUTPUT

Text

Description automatically generated