TITLE PAGE

Tic-Tac-Toe Solver

Personal Details:

- Name: Tejasva Pratap Singh Tomar
- BRANCH : CSE AI
- SEC : D
- Roll No.: 202401100300264
- Subject: AI MSE

Date: [11/03/2025]

INTODUCTION

Tic Tac Toe, also known as Noughts and Crosses or Xs and Os, is a classic two-player game played on a 3x3 grid. Each player chooses a mark—traditionally “X” or “O”—and takes turns placing their mark in an empty cell. The goal is to be the first player to align three of your marks horizontally, vertically, or diagonally. If all nine cells are filled without any player achieving three-in-a-row, the game ends in a tie.

Despite its simplicity, Tic Tac Toe provides a foundational example of strategy and combinatorial game theory. It is commonly used as a teaching tool in introductory programming courses to illustrate concepts such as arrays or lists, control flow, and basic artificial intelligence (AI) techniques (e.g., checking for a win, making strategic moves, or even implementing a minimax algorithm). By developing a Tic Tac Toe game, students can practice designing a logical sequence of steps, maintaining a game state, and implementing interactive or automated decision-making

Methodology of Tic Tac Toe Game

The development of a Tic Tac Toe game follows a structured approach to ensure smooth gameplay and logical decision-making. Below are the key steps involved in the methodology:

1. Problem Understanding & Game Rules

- The game is played on a 3x3 grid.
- Two players take turns marking the cells with either ‘X’ or ‘O’.
- A player wins if they align three of their marks horizontally, vertically, or diagonally.
- If all cells are filled without a winner, the game ends in a draw.

2. Designing the Game Board

- The board is represented using a **2D list (array)** in Python.
- Each cell in the board is initialized as empty (" ") and updated as players make moves.
- A function is created to print the board after each move.

3. Handling Player Moves

- The game accepts user input to place a mark on an available position.
- A function checks if the selected move is valid (i.e., the cell is empty).

4. Checking for a Winner

- After every move, the program checks whether any player has won.
- This is done by evaluating all rows, columns, and diagonals for three matching marks.

5. Game Loop & User Interaction

- The game runs in a loop until there is a winner or the board is full.
- Players are alternated in each round.
- The final result is displayed, and the user can restart the game if desired.

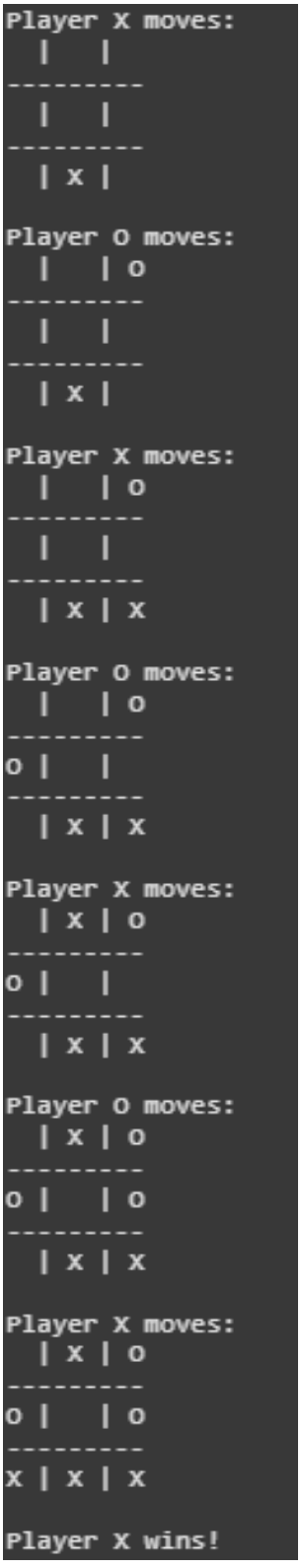
CODE

```
import random


def create_board():
    # Create a 3x3 board with empty spaces
    return [[" " for _ in range(3)] for _ in range(3)]


def print_board(board):
    # Print the board in a clear format
```

```
    for i in range(3):
        print(" | ".join(board[i]))
        if i < 2:
            print("-" * 9)
    print()

def available_moves(board):
    # Return a list of empty positions as (row, col) tuples
    moves = []
    for i in range(3):
        for j in range(3):
            if board[i][j] == " ":
                moves.append((i, j))
    return moves

def check_win(board, player):
    # Check rows and columns
    for i in range(3):
        if all(cell == player for cell in board[i]):
            return True
        if all(board[r][i] == player for r in range(3)):
            return True
    # Check both diagonals
    if board[0][0] == player and board[1][1] == player and board[2][2] == player:
        return True
    if board[0][2] == player and board[1][1] == player and board[2][0] == player:
        return True
    return False
```

```
def play_game():
    board = create_board()
    current_player = "X"  # Player X starts
    while True:
        moves = available_moves(board)
        if not moves:
            print("It's a tie!")
            break

        # Choose a random move for the current player
        move = random.choice(moves)
        board[move[0]][move[1]] = current_player
        print(f"Player {current_player} moves:")
        print_board(board)

        # Check for a win
        if check_win(board, current_player):
            print(f"Player {current_player} wins!")
            break

        # Switch player
        current_player = "O" if current_player == "X" else "X"

play_game()
```

OUTPUT



References / Credits

For concept and idea : Geeksforgeeks(https://www.geeksforgeeks.org/python-implementation-automatic-tic-tac-toe-game-using-random-number/)

For code refining : ChatGPT