def print\_board(board):

    for row in range(3):

        print(' | '.join(board[row]))

        if row < 2:

            print('---------')

def check\_winner(board):

    # Check rows, columns, and diagonals

    for i in range(3):

        if board[i][0] == board[i][1] == board[i][2] != ' ':

            return board[i][0]

        if board[0][i] == board[1][i] == board[2][i] != ' ':

            return board[0][i]

    if board[0][0] == board[1][1] == board[2][2] != ' ':

        return board[0][0]

    if board[0][2] == board[1][1] == board[2][0] != ' ':

        return board[0][2]

    return None

def is\_draw(board):

    return all(cell != ' ' for row in board for cell in row)

def tic\_tac\_toe():

    board = [[' ' for \_ in range(3)] for \_ in range(3)]

    current\_player = 'X'

    while True:

        print\_board(board)

        move = int(input(f"Player {current\_player}, enter your move (1-9): ")) - 1

        row, col = divmod(move, 3)

        if board[row][col] != ' ':

            print("Invalid move! Try again.")

            continue

        board[row][col] = current\_player

        winner = check\_winner(board)

        if winner:

            print\_board(board)

            print(f"Player {winner} wins!")

            break

        if is\_draw(board):

            print\_board(board)

            print("It's a draw!")

            break

        current\_player = 'O' if current\_player == 'X' else 'X'

if \_name\_ == "\_main\_":

    tic\_tac\_toe()

**‘’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’’**

**Explanation for Tic tac toe game using python**

**Objective:**

The objective of the Tic-Tac-Toe game is to create an engaging and interactive experience where players can choose to compete as "X" or "O" against either another player or an AI opponent powered by the Minimax algorithm. The game implements the classic rules of Tic-Tac-Toe, allowing players to take turns placing their symbols on a 3x3 grid with the goal of connecting three in a row. It ensures smooth gameplay flow, from player selection to move execution and result display, while utilizing Pygame to deliver an appealing visual interface that makes the game enjoyable and accessible for players of all ages, ultimately enhancing logical thinking and decision-making skills.

**Explanation:**

First as we run the runner.py file it pops a window using pygame extension.



**Initial Screen:**

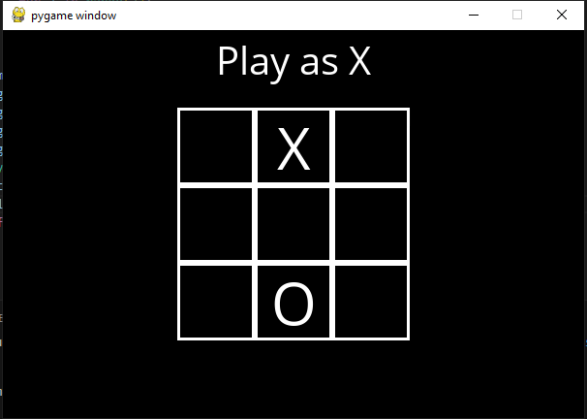
When we first run the game, we see a black screen with the title "Play Tic-Tac-Toe" at the top.

Below the title, there are two buttons: "Play as X" and "Play as O." These buttons are rendered using the mediumFont font, and they are displayed in white rectangles on the black background.

**User Interaction:**

we can click either button to select the sign. The game checks for mouse clicks, and if we click the button for "X," the user variable is set to ttt.X. If we click "O," it’s set to ttt.O.

After making a selection, the game pauses briefly (0.2 seconds) to enhance the user experience before moving on to the game board.



**Game Board Setup:**

Once a player is selected, the game draws a 3x3 grid representing the Tic-Tac-Toe board.

Each square is represented by a white rectangle, and if a player has already made a move in that square, their symbol ("X" or "O") is displayed in the center of the square using the moveFont.

**Game Logic:**

The game continuously checks whether it’s the player's turn or the AI's turn using the player function, which determines the current player based on the board state.

The terminal function checks if the game has reached a conclusion (either a win for one player or a tie). If the game is over, a message is displayed indicating the result.

**User Moves:**

If it's the user's turn and the game isn't over, the game checks for mouse clicks on the squares of the board. If the user clicks on an empty square, the game updates the board using the result function, which modifies the board state according to the move made.

**AI Moves:**

If it’s the AI's turn, the game calls the minimax function to determine the optimal move for the AI based on the current board state.

The AI takes a brief pause (0.5 seconds) before making its move, enhancing the interactive experience. The board updates accordingly based on the AI’s decision.

**Game Over:**

When the game ends (either through a win or a tie), the message is updated to reflect the result. A "Play Again" button appears, allowing players to reset the game.

If the player clicks the "Play Again" button, the game resets the user, board, and ai\_turn variables to their initial states, allowing for a new game.