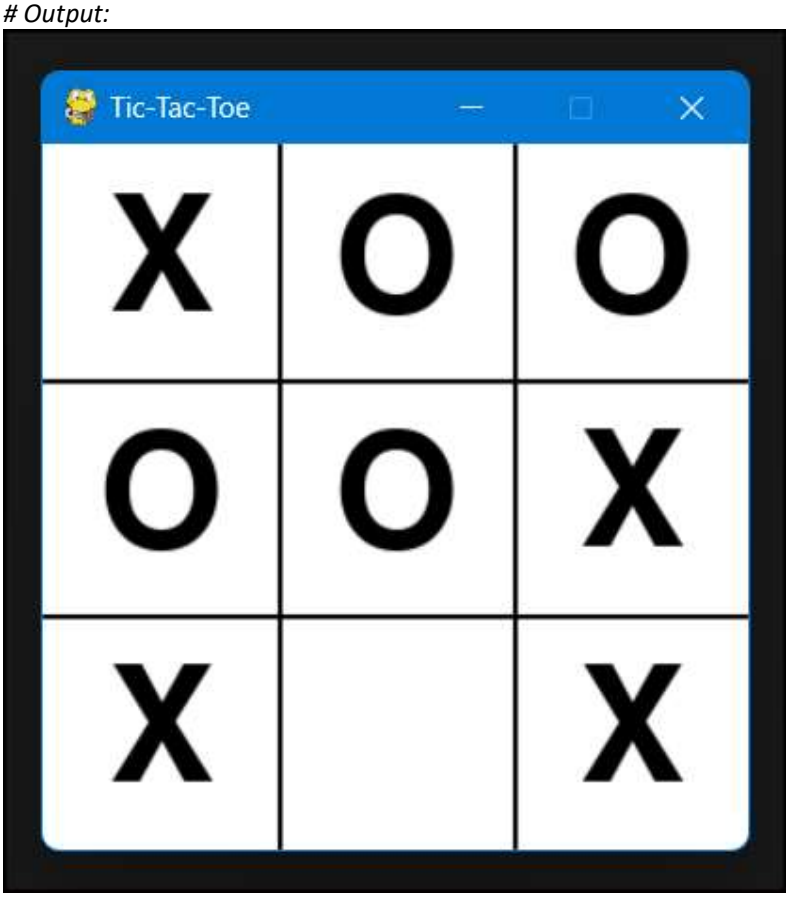
```
# Name: Aniket Sarjerao Sable
# Roll No: 54

# ============================================
# AI Practical 05
# Problem Statement:
# Write a program to develop a Tic-Tac-Toe game
# using the appropriate concepts of Game Theory.
# ============================================

# ------------------------------------------------------------
# Importing Libraries
# ------------------------------------------------------------
import pygame
import sys

# ------------------------------------------------------------
# Pygame Initialization
# ------------------------------------------------------------
pygame.init()

# ------------------------------------------------------------
# Constants and Global Variables
# ------------------------------------------------------------
WIDTH, HEIGHT = 300, 300
GRID_SIZE = 3
CELL_SIZE = WIDTH // GRID_SIZE

WHITE = (255, 255, 255)
BLACK = (0, 0, 0)

# Create the game window
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Tic-Tac-Toe")

# Initialize game board
board = [[' ' for _ in range(GRID_SIZE)] for _ in range(GRID_SIZE)]
turn = 'X'  # Player X starts

# ------------------------------------------------------------
# Functions
# ------------------------------------------------------------

# Draw grid lines
def draw_grid():
    for i in range(1, GRID_SIZE):
        pygame.draw.line(screen, BLACK, (i * CELL_SIZE, 0), (i * CELL_SIZE, HEIGHT), 2)
        pygame.draw.line(screen, BLACK, (0, i * CELL_SIZE), (WIDTH, i * CELL_SIZE), 2)

# Draw X or O on the board
def draw_symbol(row, col, symbol):
    font = pygame.font.Font(None, 100)
    text = font.render(symbol, True, BLACK)
    text_rect = text.get_rect(center=((col * CELL_SIZE) + CELL_SIZE // 2,
                                      (row * CELL_SIZE) + CELL_SIZE // 2))
    screen.blit(text, text_rect)

# Check if a player has won
def check_winner(symbol):
    for i in range(GRID_SIZE):
```

```
        if all(board[i][j] == symbol for j in range(GRID_SIZE)) or \
           all(board[j][i] == symbol for j in range(GRID_SIZE)):
            return True
    if all(board[i][i] == symbol for i in range(GRID_SIZE)) or \
       all(board[i][GRID_SIZE - 1 - i] == symbol for i in range(GRID_SIZE)):
        return True
    return False

# Check for draw condition
def is_board_full():
    return all(board[i][j] != ' ' for i in range(GRID_SIZE) for j in range(GRID_SIZE))

# Reset the board
def reset_game():
    global board
    board = [[' ' for _ in range(GRID_SIZE)] for _ in range(GRID_SIZE)]

# ------------------------------------------------------------
# Game Loop
# ------------------------------------------------------------
running = True

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

        elif event.type == pygame.MOUSEBUTTONDOWN and event.button == 1:
            mouseX, mouseY = event.pos
            clicked_row = mouseY // CELL_SIZE
            clicked_col = mouseX // CELL_SIZE

            if board[clicked_row][clicked_col] == ' ':
                board[clicked_row][clicked_col] = turn

                if check_winner(turn):
                    print(f'{turn} wins!')
                    pygame.time.wait(1000)
                    reset_game()
                elif is_board_full():
                    print("It's a draw!")
                    pygame.time.wait(1000)
                    reset_game()
                else:
                    turn = 'O' if turn == 'X' else 'X'

    # Drawing board state
    screen.fill(WHITE)
    draw_grid()

    for row in range(GRID_SIZE):
        for col in range(GRID_SIZE):
            if board[row][col] != ' ':
                draw_symbol(row, col, board[row][col])

    pygame.display.flip()

# Exit
pygame.quit()
sys.exit()
```

 # ----------

 # Output:



 # ----------