UIT2502---Data Analytics and Visualization Lab

Ex 1c: Game Application using NumPy

Name: Vasundhara.B

Roll No: 3122 21 5002 119

TIT-TAC-TOE

Objective:

The objective of Tic-Tac-Toe is to be the first player to form a line of three of their own symbols (either "X" or "O") horizontally, vertically, or diagonally on the game board.

Game Setup:

The game is played on a 3x3 grid, initially empty.

There are two players, traditionally denoted as "X" and "O."

Gameplay:

Players take turns placing their symbol ("X" or "O") on an empty cell of the grid.

Player "X" typically goes first.

Winning:

A player wins the game if they have three of their symbols in a row, column, or diagonal. There are eight possible winning combinations:

Top row: [0, 0], [0, 1], [0, 2]

Middle row: [1, 0], [1, 1], [1, 2]

Bottom row: [2, 0], [2, 1], [2, 2]

Left column: [0, 0], [1, 0], [2, 0]

Middle column: [0, 1], [1, 1], [2, 1]

Right column: [0, 2], [1, 2], [2, 2]

Diagonal from top left to bottom right: [0, 0], [1, 1], [2, 2]

Diagonal from top right to bottom left: [0, 2], [1, 1], [2, 0]

Draw (Tie):

If all cells on the grid are filled and no player has formed a winning line, the game is a draw (tie).

Game Over:

The game ends when one player wins by forming a line of three symbols or when the game is a tie.

Playing the Game:

Players take turns entering the row and column numbers where they want to place their symbol.

The game enforces valid moves and prevents players from placing their symbols on occupied cells.

Usage of NumPy in the application:

Importing NumPy: The first line of the code **import numpy as np** imports the NumPy library and assigns it the alias "np." This alias is commonly used to refer to NumPy functions and classes throughout the code.

Creating the Game Board: The game board is created using the **np.full()** function. This function creates a new NumPy array with a specified shape and fills it with a given value. In this case, a 3x3 game board is created with all cells initially filled with a space character (" ").

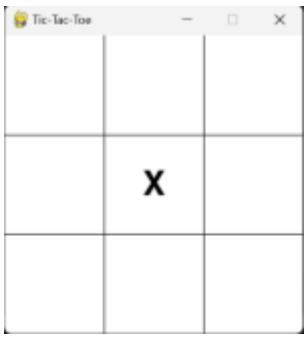
CODE:

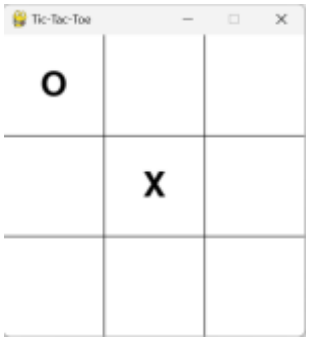
```
import pygame
import numpy as np

# Initialize Pygame
pygame.init()

# Constants for the game
WINDOW_SIZE = 300
GRID_SIZE = 3
CELL_SIZE = WINDOW_SIZE // GRID_SIZE
WHITE = (255, 255, 255)
LINE_COLOR = (0, 0, 0)
LINE_WIDTH = 10
FONT_SIZE = 50
FONT = pygame.font.Font(None, FONT_SIZE)

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

# Create the game board using NumPy
board = np.full((GRID_SIZE, GRID_SIZE), "")

# Function to draw the grid lines
def draw_grid():
    for i in range(1, GRID_SIZE):
        pygame.draw.line(window, LINE_COLOR, (0, i * CELL_SIZE), (WINDOW_SIZE, i *
CELL_SIZE))
        pygame.draw.line(window, LINE_COLOR, (i * CELL_SIZE, 0), (i * CELL_SIZE,
WINDOW_SIZE))

# Function to check for a win
def check_win(player):
    return (
        any(np.all(board == player, axis=0)) or
        any(np.all(board == player, axis=1)) or
        np.all(np.diag(board) == player) or
        np.all(np.diag(np.fliplr(board)) == player)
    )

# Main game loop
current_player = "X"
game_over = False
winner = None

while not game_over:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
```

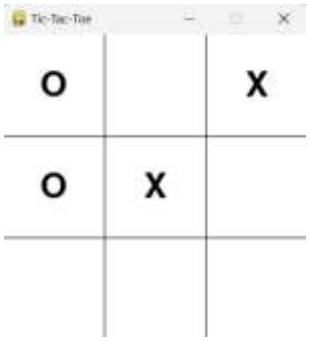
```
            game_over = True
        elif event.type == pygame.MOUSEBUTTONDOWN and not check_win("X") and not
check_win("O"):
            row = event.pos[1] // CELL_SIZE
            col = event.pos[0] // CELL_SIZE
            if board[row, col] == "":
                board[row, col] = current_player
                current_player = "O" if current_player == "X" else "X"

    window.fill(WHITE)
    draw_grid()

    for row in range(GRID_SIZE):
        for col in range(GRID_SIZE):
            if board[row, col] != "":
                text = FONT.render(board[row, col], True, LINE_COLOR)
                text_rect = text.get_rect(center=(col * CELL_SIZE + CELL_SIZE //
2, row * CELL_SIZE + CELL_SIZE // 2))
                window.blit(text, text_rect)

    if check_win("X"):
        game_over = True
        winner = "X"
    elif check_win("O"):
        game_over = True
        winner = "O"
    elif np.all(board != ""):
        game_over = True
        winner = "Tie"

    pygame.display.flip()

if winner == "Tie":
    print("It's a tie!")
else:
    print("Player {} wins!".format(winner))

pygame.display.flip()

# Wait for 5 seconds
pygame.time.wait(5000)

pygame.quit()
```

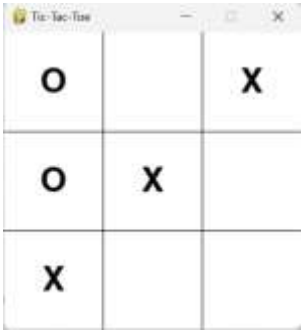
OUTPUT:

(i) PLAYER "X" WINS



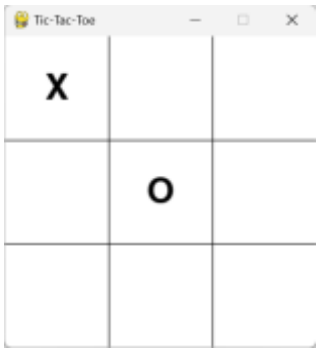


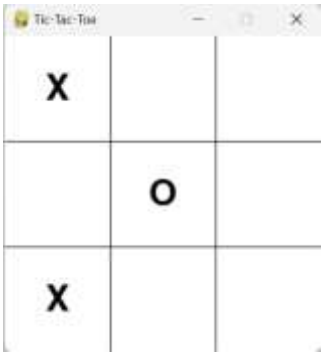


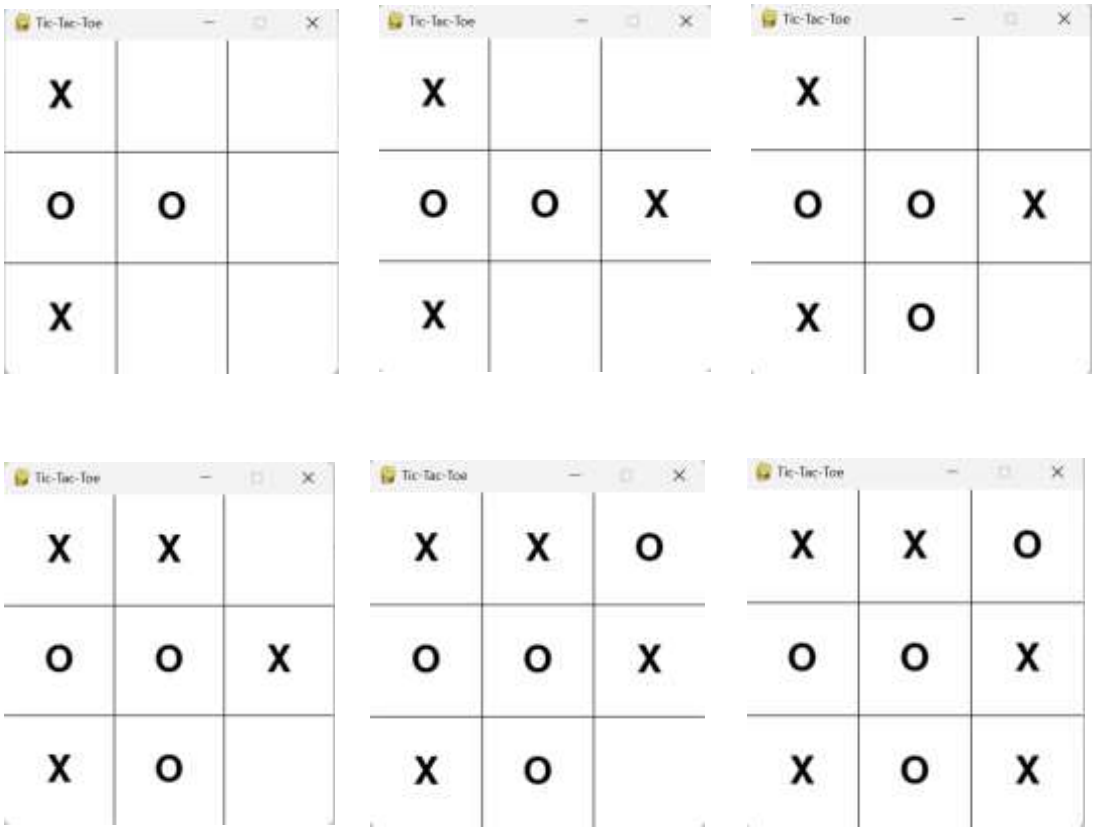



```
pygame 2.5.1 (SDL 2.28.2, Python 3.10.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
Player X wins!
PS C:\Users\B Vasundhara\Documents\Data Analaytics>
```

(ii) IT'S A DRAW

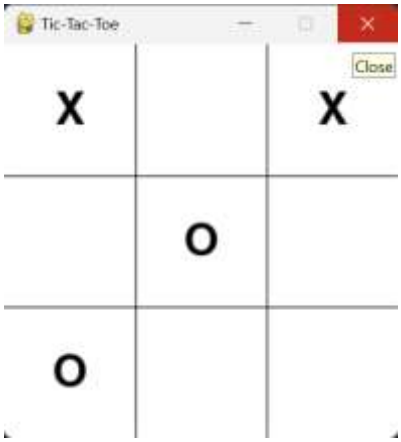







```
pygame 2.5.1 (SDL 2.28.2, Python 3.10.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
It's a tie!
PS C:\Users\B Vasundhara\Documents\Data Analaytics>
```

(iii) WHEN THE GAME IS STOPPED MIDWAY




```
pygame 2.5.1 (SDL 2.28.2, Python 3.10.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
Player None wins!
PS C:\Users\B Vasundhara\Documents\Data Analaytics>
```