Mini Project file
(Tic Tac Toe Game)

Student Name: Simran **UID: 24MCC20089**

Branch: UIC **Section/Group: 24MCD-1A**

Semester: 1st **Date of Performance: 21-09-24**

Subject Name: Python Programming **Subject Code: 24CAH-606**

AIM: -

To develop a simple graphical Tic-Tac-Toe game using Python's Tkinter library that allows a player to compete against a computer opponent.

OVERVIEW: -

This project implements a two-player game (player vs. computer) of Tic-Tac-Toe. The player uses 'X' and the computer uses 'O'. The game features a graphical user interface (GUI) that displays the game board, tracks scores, and updates the status based on player moves and outcomes.

TASK TO BE DONE:-

- **Set Up the GUI**: Create a window for the game using Tkinter.
- **Implement Game Logic**: Code the rules for player moves, computer moves, win conditions, and checking if the board is full.
- **Handle Game States**: Enable and disable buttons based on the game's current state (ongoing, win, tie).
- **Score Tracking**: Maintain a score count for the player and display it.
- **Reset Functionality**: Allow the player to reset the game and start over.

STEPS:-

1) **Set Up the Environment**
    a. Install Python and Tkinter.
    b. Create a new Python file for the project.

2) **Initialize Game Variables**
    a. Create a board using a list to represent the Tic-Tac-Toe grid.
    b. Initialize a score counter to keep track of the player's score.
3) **Define Game Functions**
    a. **insert_letter(letter, pos)**: Inserts a letter ('X' or 'O') at a specified position if that space is free.
    b. **space_is_free(pos)**: Checks if a specific position on the board is unoccupied.
    c. **print_board()**: Updates the graphical board display based on the current state of the game.
    d. **is_board_full()**: Determines if the board is full.
    e. **is_winner(letter)**: Checks for a winning condition for the specified letter.
    f. **player_move(pos)**: Handles the player's move, updates the board, checks for wins or ties, and calls the computer's move.
    g. **computer_move()**: Implements a simple AI for the computer player to make its move.
    h. **disable_buttons()**: Disables all buttons when the game ends.
    i. **reset_game()**: Resets the game state and board for a new game.
4) **Create the GUI**
    a. Set up the main window using Tkinter.
    b. Create buttons for each cell of the Tic-Tac-Toe grid.
    c. Create a status label to display game messages.
    d. Add a reset button to restart the game.
5) **Event Loop**
    a. Start the Tkinter event loop to run the application.

CODE FOR IMPLEMENTATION: -

```
import tkinter as tk
import random

# Initialize the board and score
board = [' ' for _ in range(10)]
scorecount = 0
```

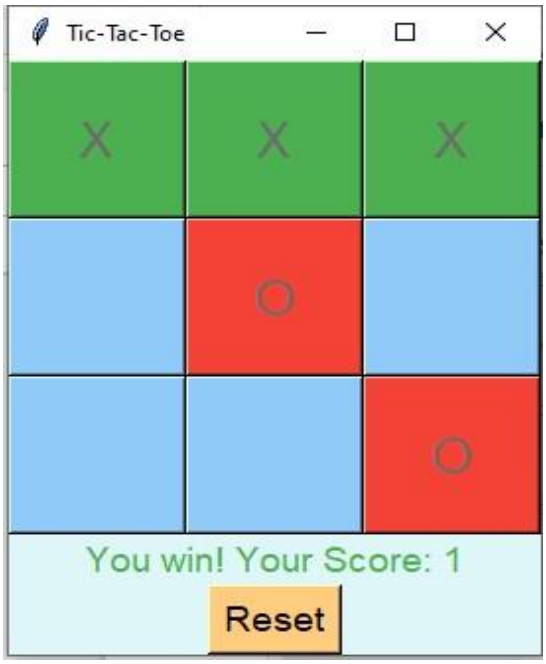
```
def insert_letter(letter, pos):
    if space_is_free(pos):
        board[pos] = letter

def space_is_free(pos):
    return board[pos] == ' '

def print_board():
    for i in range(1, 10):
        buttons[i].config(text=board[i], bg="#90caf9" if board[i] == ' ' else ("#4caf50" if
board[i] == 'X' else "#f44336"))

def is_board_full():
    return board.count(' ') == 1

def is_winner(letter):
    return (
        (board[1] == letter and board[2] == letter and board[3] == letter) or
        (board[4] == letter and board[5] == letter and board[6] == letter) or
        (board[7] == letter and board[8] == letter and board[9] == letter) or
        (board[1] == letter and board[4] == letter and board[7] == letter) or
        (board[2] == letter and board[5] == letter and board[8] == letter) or
        (board[3] == letter and board[6] == letter and board[9] == letter) or
        (board[1] == letter and board[5] == letter and board[9] == letter) or
        (board[3] == letter and board[5] == letter and board[7] == letter)
    )

def player_move(pos):
    global scorecount
    insert_letter('X', pos)
    print_board()

    if is_winner('X'):
        scorecount += 1
```

```
        status_label.config(text=f"You win! Your Score: {scorecount}", fg="#4caf50")
        disable_buttons()
        return

    if is_board_full():
        status_label.config(text="It's a tie!", fg="orange")
        return

    computer_move()
    print_board()
    if is_winner('O'):
        scorecount -= 1
        status_label.config(text=f"Sorry, you lose! Your Score: {scorecount}", fg="#f44336")
        disable_buttons()

def computer_move():
    possible_moves = [i for i in range(1, 10) if space_is_free(i)]
    move = 0

    for letter in ['O', 'X']:
        for i in possible_moves:
            board_copy = board[:]
            board_copy[i] = letter
            if is_winner(letter):
                move = i
                insert_letter('O', move)
                return

    if 5 in possible_moves:
        move = 5
        insert_letter('O', move)
        return

    corners_open = [i for i in possible_moves if i in [1, 3, 7, 9]]
```

```
    if corners_open:
        move = random.choice(corners_open)
        insert_letter('O', move)
        return

    edges_open = [i for i in possible_moves if i in [2, 4, 6, 8]]
    if edges_open:
        move = random.choice(edges_open)
        insert_letter('O', move)

def disable_buttons():
    for button in buttons[1:]:
        button.config(state=tk.DISABLED)

def reset_game():
    global board
    board = [' ' for _ in range(10)]
    print_board()
    for button in buttons[1:]:
        button.config(state=tk.NORMAL)
    status_label.config(text="", fg="black")

# Create the main window
root = tk.Tk()
root.title("Tic-Tac-Toe")
root.configure(bg="#e0f7fa")

# Create buttons for the board
buttons = [None] + [tk.Button(root, text=' ', font=('Arial', 20), width=5, height=2,
                              command=lambda i=i: player_move(i), bg="#81d4fa",
activebackground="#4fc3f7") for i in range(1, 10)]

# Place buttons in a grid
for i in range(3):
```

```
    for j in range(3):
        buttons[i * 3 + j + 1].grid(row=i, column=j)

# Create a status label
status_label = tk.Label(root, text="", font=('Arial', 14), bg="#e0f7fa")
status_label.grid(row=3, column=0, columnspan=3)

# Create a reset button
reset_button = tk.Button(root, text='Reset', font=('Arial', 14), command=reset_game,
bg="#ffcc80", activebackground="#ffb74d")
reset_button.grid(row=4, column=0, columnspan=3)

# Start the game
print_board()

# Run the Tkinter event loop
root.mainloop()
```

OUTPUT:



LEARNING OUTCOMES:-

- **Understanding of Tkinter**: Gain experience in creating GUI applications with window management and widget placement.
- **Game Logic Implementation**: Learn to implement basic game logic, including turns and win conditions.
- **Control Structures**: Utilize control structures like if statements and loops for game management.
- **Functions and Modular Programming**: Understand modular programming by breaking the game into functions for readability.
- **Basic AI Development**: Create simple decision-making logic for the computer player.
- **Event-Driven Programming**: Handle events, such as button clicks, in a GUI environment.
- **Debugging and Testing**: Develop skills in debugging and testing various game scenarios.