TIC TAC TOE GAME IN PYTHON

TABLE OF CONTENTS

GROUP MEMBERS

MUKARRAM KHAN: (BCB-23S-043)

BABAR ALI: (BCB-23S-053)

AZAN SHAHZAD: (BCB-23S-021)

INTRODUCTION

Tic Tac Toe is a classic 3x3 grid game for two players. The objective is to align three symbols in a row, vertically, horizontally, or diagonally. In this Python implementation, players take turns placing their symbols on the grid. The game ends when one player wins or the grid is full, resulting in a draw.

WORKING

Tic Tac Toe is a classic 3x3 grid game where two players, typically represented by "X" and "O", take turns playing. The game starts with an empty grid, and Player X makes the initial move. Players then alternate placing their symbol on the grid until one player achieves a winning condition or the grid is full. Winning conditions involve aligning three symbols in a row, horizontally, vertically, or diagonally. If a player successfully aligns three symbols, they win the game. If all cells on the grid are filled and no player has achieved a winning pattern, the game ends in a draw.

Players input their moves by specifying the row and column where they want to place their symbol, with the program ensuring input validation to prevent moves outside the grid or on already occupied cells. Overall, Tic Tac Toe is a straightforward yet engaging game that encourages strategic thinking and pattern recognition.

CONTRIBUTION

1) MUKARRAM KHAN PRIMARILY FOCUSES ON SETTING UP THE GRAPHICAL USER INTERFACE (GUI) USING TKINTER AND MANAGING USER EVENTS SUCH AS BUTTON CLICKS. HERE'S THE PART OF THE CODE HANDLED BY THIS MUKARRAM:

- This part sets up the main window, labels, buttons, and the game board grid.
- It creates a window using Tk().
- Sets the title of the window to "Tic-Tac-Toe".
- Creates a label to display whose turn it is.
- Adds a restart button to start a new game.
- Creates a frame to contain the game board buttons.
- Uses nested loops to create a 3x3 grid of buttons representing the game board.
- Associates each button with the next_turn function using the command parameter.

2) **Azan Shahzad** handles the game logic, including checking for winners and managing player turns.

•next_turn function handles the player's move and updates the label accordingly.

•check_winner function checks if any player has won or if the game is a tie.

•empty_spaces function checks if there are any empty spaces left on the board.

•new_game function resets the game board and starts a new game with a randomly selected player.

3) Babar Ali focuses on visually indicating the winning combination on the game board.

- This member enhances the check_winner function to change the background color of the winning combination to green.
- If there's a tie, the background color of all buttons is changed to yellow.

CODE:

```
FROM TKINTER IMPORT *
IMPORT RANDOM

DEF NEXT_TURN(ROW, COLUMN):

    GLOBAL PLAYER

    IF BUTTONS[ROW][COLUMN]['TEXT'] == "" AND CHECK_WINNER() IS FALSE:

        IF PLAYER == PLAYERS[0]:

            BUTTONS[ROW][COLUMN]['TEXT'] = PLAYER

            IF CHECK_WINNER() IS FALSE:
                PLAYER = PLAYERS[1]
                LABEL.CONFIG(TEXT=(PLAYERS[1]+" TURN"))

            ELIF CHECK_WINNER() IS TRUE:
                LABEL.CONFIG(TEXT=(PLAYERS[0]+" WINS"))

            ELIF CHECK_WINNER() == "TIE":
                LABEL.CONFIG(TEXT="TIE!")

        ELSE:

            BUTTONS[ROW][COLUMN]['TEXT'] = PLAYER
```

```
        check_winner() is False:
            player = players[0]
            label.config(text=(players[0]+" turn"))

        elif check_winner() is True:
            label.config(text=(players[1]+" wins"))

        elif check_winner() == "Tie":
            label.config(text="Tie!")

    def check_winner():

        for row in range(3):
            if buttons[row][0]['text'] == buttons[row][1]['text'] ==
                buttons[row][2]['text'] != "":
                buttons[row][0].config(bg="green")
                buttons[row][1].config(bg="green")
                buttons[row][2].config(bg="green")
                return True

        for column in range(3):
            if buttons[0][column]['text'] == buttons[1][column]['text'] ==
                buttons[2][column]['text'] != "":
                buttons[0][column].config(bg="green")
                buttons[1][column].config(bg="green")
                buttons[2][column].config(bg="green")
                T
```

```
    if buttons[0][0]['text'] == buttons[1][1]['text'] == buttons[2][2]['text'] !=
                                                                        "":
                buttons[0][0].config(bg="green")
                buttons[1][1].config(bg="green")
                buttons[2][2].config(bg="green")
                return True

    elif buttons[0][2]['text'] == buttons[1][1]['text'] == buttons[2][0]['text']
                                                                    != "":
                buttons[0][2].config(bg="green")
                buttons[1][1].config(bg="green")
                buttons[2][0].config(bg="green")
                return True

            elif empty_spaces() is False:

                for row in range(3):
                for column in range(3):
            buttons[row][column].config(bg="yellow")
                return "Tie"

                else:
                return False


        def empty_spaces():
```

```
            FOR COLUMN IN RANGE(3):
                IF BUTTONS[ROW][COLUMN]['TEXT'] != "":
                    SPACES -= 1

            IF SPACES == 0:
                RETURN FALSE
            ELSE:
                RETURN TRUE

        DEF NEW_GAME():

            GLOBAL PLAYER

            PLAYER = RANDOM.CHOICE(PLAYERS)

            LABEL.CONFIG(TEXT=PLAYER+" TURN")

            FOR ROW IN RANGE(3):
                FOR COLUMN IN RANGE(3):
                    BUTTONS[ROW][COLUMN].CONFIG(TEXT="",BG="#F0F0F0")

        WINDOW = TK()
        WINDOW.TITLE("TIC-TAC-TOE")
        PLAYERS = ["X","O"]
        PLAYER = RANDOM.CHOICE(PLAYERS)
        BUTTONS = [[0,0,0],
```

```
                                [0,0,0]]

    label = Label(text=player + " turn", font=('consolas',40))

                        label.pack(side="top")

reset_button = Button(text="restart", font=('consolas',20),
                        command=new_game)

                    reset_button.pack(side="top")

                        frame = Frame(window)

                            frame.pack()

                        for row in range(3):

                        for column in range(3):

                    buttons[row][column] = Button(frame,
    text="",font=('consolas',40), width=5, height=2,

                        command= lambda row=row, column=column:
                    next_turn(row,column))

    buttons[row][column].grid(row=row,column=column)

                            window.mainloop()
```

THANK YOU!