PPLP MINI PROJECT : 2021-22

Topic:- Tic-Tac-Toe with a Click-to-play UI.

Group Members:-

Name	Roll No.
Yash Bhavsar	PF42
Omkar Bhosale	PF44
Rashi Jain	PF48
Aniket Adhekar	PF49
Mrunal Dande	PF57

Introduction:-

Tic-tac-toe (American English), noughts and crosses (British English), or Xs and Os is a paper-and-pencil game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner.

tkinter Python library is used to create the GUI. Two options are available to play the game, **along with the system or with another player**.

A small winning strategy is used to play with the system. The system will try to find the best place to put its naught or cross by which the system will win or try to stop players to win.

Methodology used:-

Tkinter is the inbuilt python module that is used to create GUI applications. It is one of the most commonly used modules for creating GUI applications in Python as it is simple and easy to work with. You don't need to worry about the installation of the Tkinter module separately as it comes with Python already. It gives an object-oriented interface to the Tk GUI toolkit.

Some other Python Libraries available for creating our own GUI applications are:-

- Kivy
- Python Qt
- wxPython

Among all Tkinter is most widely used.

Widgets in Tkinter are the elements of GUI application which provides various controls (such as Labels, Buttons, ComboBoxes, CheckBoxes, MenuBars, RadioButtons and many more) to users to interact with the application. TKinter is widely used for developing GUI applications. Along with applications, we can also use Tkinter GUI to develop games.

To create a Tkinter :-

- Importing the module – tkinter
- Create the main window (container)
- Add any number of widgets to the main window
- Apply the event Trigger on the widgets.

Approach:-

- Create a landing page containing selection buttons: Single-player or multiplayer.
- Create a game-board containing nine tiles to play the game along with other details (i.e. playing with a system or another player, whose turn etc.).
- Allow the player to press the tile and check the status of the game (i.e. Tie game, anyone of the players won the match or the game is still running).
- Display the message, who won the match.

Description of other functions:-

- gameboard_pc() and gameboard_pl() will create the another geometry to play the game. It will add 9 buttons on 3×3 board of the game (Three rows of buttons containing three buttons each).
- get_text_pc() and get_text() functions will put the text on buttons as it pressed.
- pc() function will decide the next move of the system.
- winner() function will check whether the player won the match or not.
- isfree() function will check whether the player can put it's a coin or not.
- isfull() function will check the board is full or not.

1) **The Button function:-** The above function makes it possible to define a custom button with custom properties.
2) **The Change function:-** The change function switches the 'O' to 'X' and vice-versa for the other player to play his chance.
3) **The Reset function:-** The reset function resets the state of all buttons to Normal and clears the text on them.
4) **The Check function:-** The check function checks the board row-wise,column-wise and diagonal-wise for equality and displays the result of the game.
5) **The Click function:-** The above function handles button clicks on the board. This calls the check() function(mentioned as function 4 above) and the change_a() function(mentioned as function 2 above) and sets the state and text of the button clicked.

InSome major points that help you understand the structure of the code:-

We have used a class to have all the methods in one place. It can easily be a reusable bundle in some other code as well. Next, we have defined different functions for each responsibility, even if it is a small task. It helps to maintain the code with ease. The above two approaches help us update the app effortlessly if we want to update the game.

Result and Conclusion:-

Code of project:-

```
# Tic Tac Toe game with GUI
```

```
# using tkinter
# importing all necessary libraries
import random
import tkinter
from tkinter import *
from functools import partial
from tkinter import messagebox
from copy import deepcopy
# sign variable to decide the turn of which player
sign = 0
# Creates an empty board
global board
board = [[" " for x in range(3)] for y in range(3)]
# Check l(O/X) won the match or not
# according to the rules of the game
def winner(b, l):
    return ((b[0][0] == l and b[0][1] == l and b[0][2] == l) or
            (b[1][0] == l and b[1][1] == l and b[1][2] == l) or
            (b[2][0] == l and b[2][1] == l and b[2][2] == l) or
            (b[0][0] == l and b[1][0] == l and b[2][0] == l) or
            (b[0][1] == l and b[1][1] == l and b[2][1] == l) or
            (b[0][2] == l and b[1][2] == l and b[2][2] == l) or
            (b[0][0] == l and b[1][1] == l and b[2][2] == l) or
            (b[0][2] == l and b[1][1] == l and b[2][0] == l))
# Configure text on button while playing with another player
def get_text(i, j, gb, l1, l2):
    global sign
    if board[i][j] == ' ':
        if sign % 2 == 0:
            l1.config(state=DISABLED)
```

```
            l2.config(state=ACTIVE)
            board[i][j] = "X"
        else:
            l2.config(state=DISABLED)
            l1.config(state=ACTIVE)
            board[i][j] = "O"
        sign += 1
        button[i][j].config(text=board[i][j])
    if winner(board, "X"):
        gb.destroy()
        box = messagebox.showinfo("Winner", "Player 1 won the match")
    elif winner(board, "O"):
        gb.destroy()
        box = messagebox.showinfo("Winner", "Player 2 won the match")
    elif(isfull()):
        gb.destroy()
        box = messagebox.showinfo("Tie Game", "Tie Game")
# Check if the player can push the button or not
def isfree(i, j):
    return board[i][j] == " "
# Check the board is full or not
def isfull():
    flag = True
    for i in board:
        if(i.count(' ') > 0):
            flag = False
    return flag
# Create the GUI of game board for play along with another player
def gameboard_pl(game_board, l1, l2):
    global button
```

```
    button = []
    for i in range(3):
        m = 3+i
        button.append(i)
        button[i] = []
        for j in range(3):
            n = j
            button[i].append(j)
            get_t = partial(get_text, i, j, game_board, l1, l2)
            button[i][j] = Button(
                game_board, bd=5, command=get_t, height=4, width=8)
            button[i][j].grid(row=m, column=n)
    game_board.mainloop()
# Decide the next move of system
def pc():
    possiblemove = []
    for i in range(len(board)):
        for j in range(len(board[i])):
            if board[i][j] == ' ':
                possiblemove.append([i, j])
    move = []
    if possiblemove == []:
        return
    else:
        for let in ['O', 'X']:
            for i in possiblemove:
                boardcopy = deepcopy(board)
                boardcopy[i[0]][i[1]] = let
                if winner(boardcopy, let):
                    return i
```

```
        corner = []
        for i in possiblemove:
            if i in [[0, 0], [0, 2], [2, 0], [2, 2]]:
                corner.append(i)
        if len(corner) > 0:
            move = random.randint(0, len(comer)-1)
            return corner[move]
        edge = []
        for i in possiblemove:
            if i in [[0, 1], [1, 0], [1, 2], [2, 1]]:
                edge.append(i)
        if len(edge) > 0:
            move = random.randint(0, len(edge)-1)
            return edge[move]
# Configure text on button while playing with system
def get_text_pc(i, j, gb, l1, l2):
    global sign
    if board[i][j] == ' ':
        if sign % 2 == 0:
            l1.config(state=DISABLED)
            l2.config(state=ACTIVE)
            board[i][j] = "X"
        else:
            button[i][j].config(state=ACTIVE)
            l2.config(state=DISABLED)
            l1.config(state=ACTIVE)
            board[i][j] = "O"
        sign += 1
        button[i][j].config(text=board[i][j])
    x = True
```

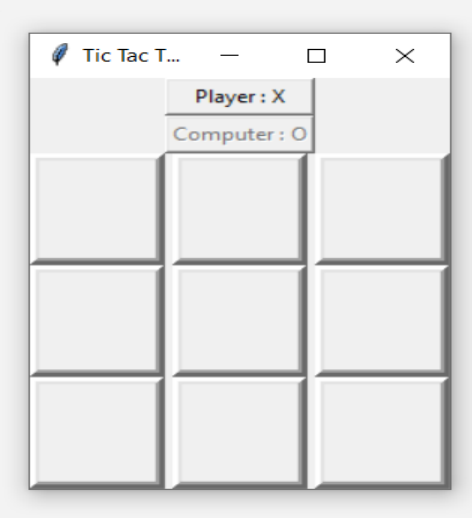
```
    if winner(board, "X"):
        gb.destroy()
        x = False
        box = messagebox.showinfo("Winner", "Player won the match")
    elif winner(board, "O"):
        gb.destroy()
        x = False
        box = messagebox.showinfo("Winner", "Computer won the match")
    elif(isfull()):
        gb.destroy()
        x = False
        box = messagebox.showinfo("Tie Game", "Tie Game")
    if(x):
        if sign % 2 != 0:
            move = pc()
            button[move[0]][move[1]].config(state=DISABLED)
            get_text_pc(move[0], move[1], gb, l1, l2)
# Create the GUI of game board for play along with system
def gameboard_pc(game_board, l1, l2):
    global button
    button = []
    for i in range(3):
        m = 3+i
        button.append(i)
        button[i] = []
        for j in range(3):
            n = j
            button[i].append(j)
            get_t = partial(get_text_pc, i, j, game_board, l1, l2)
            button[i][j] = Button(
```

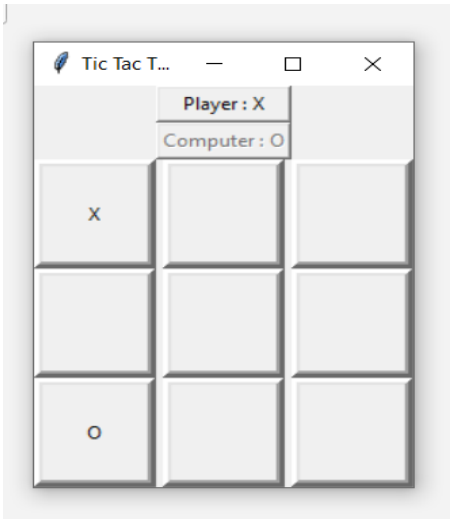
```
            game_board, bd=5, command=get_t, height=4, width=8)
        button[i][j].grid(row=m, column=n)
    game_board.mainloop()
# Initialize the game board to play with system
def withpc(game_board):
    game_board.destroy()
    game_board = Tk()
    game_board.title("Tic Tac Toe")
    l1 = Button(game_board, text="Player : X", width=10)
    l1.grid(row=1, column=1)
    l2 = Button(game_board, text = "Computer : O",
            width = 10, state = DISABLED)
    l2.grid(row = 2, column = 1)
    gameboard_pc(game_board, l1, l2)
# Initialize the game board to play with another player
def withplayer(game_board):
    game_board.destroy()
    game_board = Tk()
    game_board.title("Tic Tac Toe")
    l1 = Button(game_board, text = "Player 1 : X", width = 10)
    l1.grid(row = 1, column = 1)
    l2 = Button(game_board, text = "Player 2 : O",
            width = 10, state = DISABLED)
    l2.grid(row = 2, column = 1)
    gameboard_pl(game_board, l1, l2)
# main function
def play():
    menu = Tk()
    menu.geometry("250x250")
    menu.title("Tic Tac Toe")
```

```
    wpc = partial(withpc, menu)
    wpl = partial(withplayer, menu)
    head = Button(menu, text = "---Welcome to tic-tac-toe---",
                  activeforeground = 'red',
                  activebackground = "yellow", bg = "red",
                  fg = "yellow", width = 500, font = 'summer', bd = 5)
    B1 = Button(menu, text = "Single Player", command = wpc,
                activeforeground = 'red',
                activebackground = "yellow", bg = "red",
                fg = "yellow", width = 500, font = 'summer', bd = 5)
    B2 = Button(menu, text = "Multi Player", command = wpl, activeforeground = 'red',
                activebackground = "yellow", bg = "red", fg = "yellow",
                width = 500, font = 'summer', bd = 5)
    B3 = Button(menu, text = "Exit", command = menu.quit, activeforeground = 'red',
                activebackground = "yellow", bg = "red", fg = "yellow",
                width = 500, font = 'summer', bd = 5)
    head.pack(side = 'top')
    B1.pack(side = 'top')
    B2.pack(side = 'top')
    B3.pack(side = 'top')
    menu.mainloop()
# Call main function
if __name__ == '__main__':
    play()
```

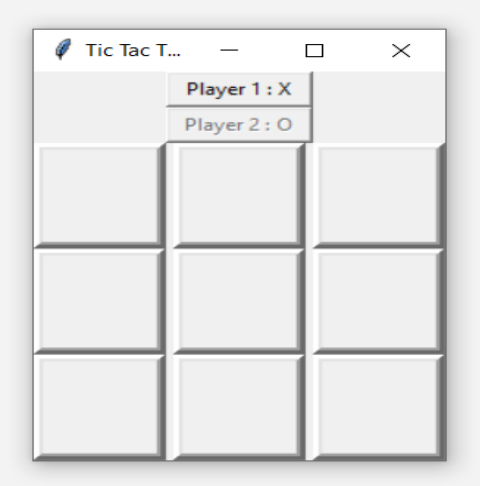
Snapshots of output:-

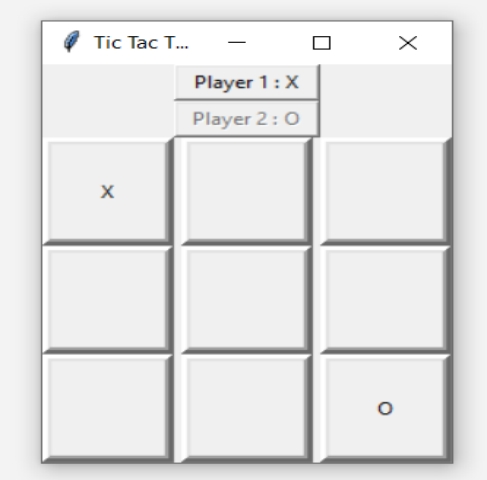
Single player:-



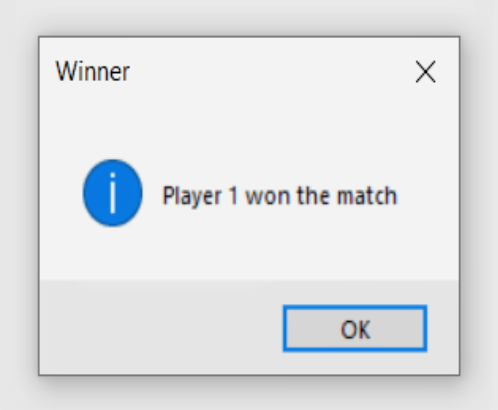


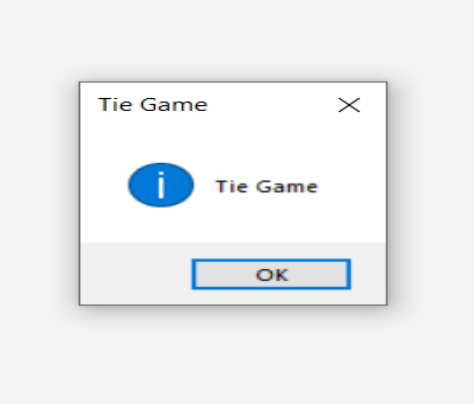
Multiplayer player:-





Winner or Tie:-





Conclusion:-

- The time complexity for implementing tic tac toe in python is O(1).
- The minimum number of turns required to win is 5.
- If both players play optimally, the result will end in a draw.
- You can add much exciting stuff to this project like you can ask to play again, or you can add a scoreboard displaying the number of wins of each player. You can ask the players for their names. There is so much you can do to improve your code and hence yourself. Try to run this tic tac toe in python in your local system, and we are there for you if you face any problems.
