TIC TAC TOE GAME

USING PYTHON

V.PRANAV SHESU RAJU

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

WORKING

- 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, any one of the players won the match or the game is still running).

- Display the message, of who won the match.

USES

The Tic-Tac-Toe game, implemented in Python, serves as an excellent learning tool for beginners and intermediate programmers alike. It teaches foundational concepts such as loops, conditionals, and data structures (like lists or dictionaries). Through implementing game logic—such as checking for wins or preventing opponent wins—it sharpens algorithmic thinking and problem-solving skills. Furthermore, creating a graphical user interface (GUI) version of Tic-Tac-Toe introduces event-driven programming and GUI frameworks like tkinter or Pygame. Overall, Tic-Tac-Toe in Python is not only a fun recreational project but also a valuable educational tool that promotes collaboration, community engagement, and skill development in programming and software development.

ADVANTAGES & DISADVANTAGES

Advantages:

1.Learning Tool
2.Algorithmic Thinking
3.GUI Development
4.AI Experimentation
5.Recreational and Educational

Disadvantages:

1. Simplicity
2. Limited Scope
3. Algorithmic Constraints
4. Over-Simplification
5. User Engagement

FUTURE ENHANCEMENT

Future enhancements for a Tic-Tac-Toe game in Python could significantly elevate its appeal and functionality. These include implementing advanced AI algorithms like reinforcement learning to enhance the computer player's strategy and adaptability. Introducing variations such as larger boards or 3D configurations would add complexity and diversity to gameplay. Networked multiplayer capabilities would enable players to compete online, fostering community interaction. Enhancing the graphical interface with animations, sound effects, and customizable themes would improve user engagement. Features like game statistics tracking, difficulty levels for AI opponents, and tournaments with leaderboards could create a competitive environment. Accessibility features such as text-to-speech and customizable controls would cater to a broader audience. Ensuring cross-platform compatibility would allow players to enjoy the game on different devices seamlessly.

SOURCE CODE FOR TIC TAC TOE GAME

1. The code starts by importing the tkinter package.
2. This is a library that allows us to create graphical user interfaces in Python.
3. Next, we import the messagebox function from this same library.
4. The messagebox function creates a window with an OK button and text input field for the player to enter their move.
5. The next line of code sets up our variables: Player1 = ‘X’, stop_game = False.
6. These are global variables which means they can be accessed anywhere in the program without having to use parentheses or any other special syntax (e. g., if Player1 == “X” and states[r] == 0).
7. Global variables are often used when you want your program’s logic to be able to access all parts of it without needing extra lines of code for each part (e.g., checking if Player1 == “O” and states[r] == 0).
8. Next, we have two functions: clicked(r,c) and check_if_win().
9. The first one checks whether player X has won or not based on what state r is currently in; if so, then it

changes state r back into X; otherwise, it changes state r into O depending on who won last time.

10. The code is a function that checks if the player has won or lost.
11. The code starts by importing tkinter, which is a library for creating graphical user interfaces in Python.
12. The next line imports messagebox, which allows the program to display messages on screen.
13. The first line of the function sets up Player1 as “X”.
14. Then states[r] is set to 0 and stop_game is set to False because this function will be checking if the player has won or lost.
15. If they have won then states[r] will be changed to “X” and Player1 will be changed to “O”.
16. If they have lost then states[r] will be changed to “O” and Player1 will be changed.
17. The code starts by declaring a global variable called stop_game.
18. The code then declares three loops that iterate through the states of the game board.
19. Inside each loop, if any state is true, then stop_game is set to True, and winner is displayed in messagebox with the corresponding state value.

20. The check_if_win() function checks if there are any winning combinations on the board.
21. If so, it displays “Winner” in messagebox with the corresponding state value and sets stop_game to True.
22. The code checks if the player has won.
23. If they have, it displays a messagebox with the states of the game and their corresponding values.

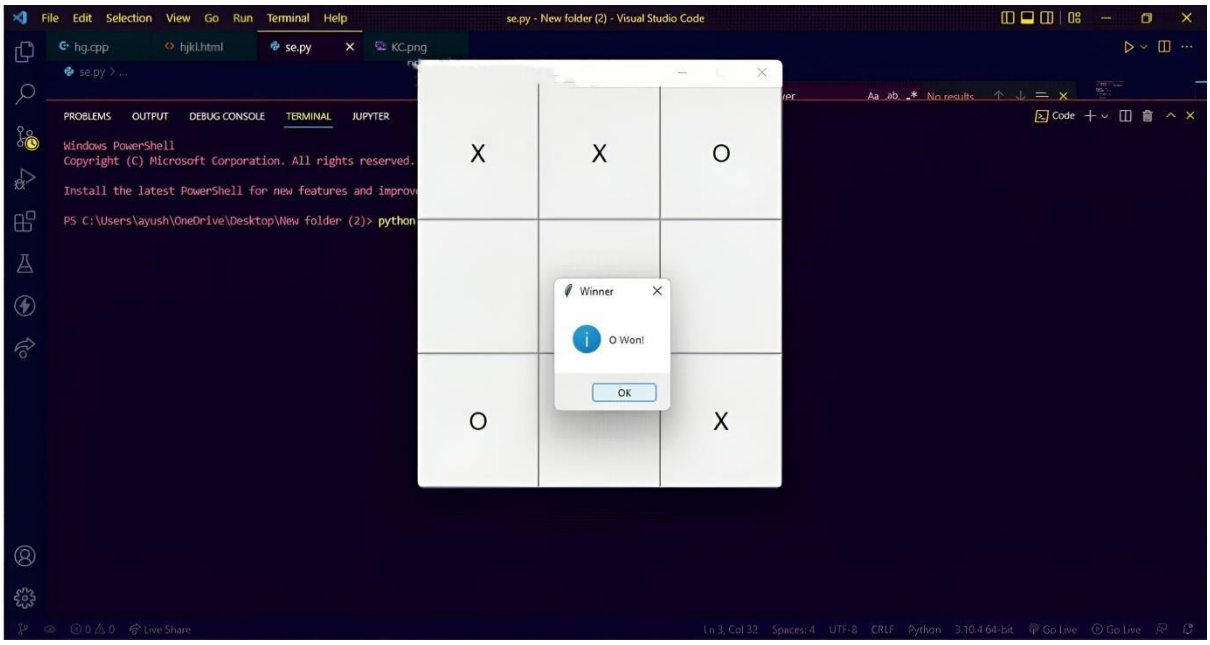
CODE:

```python
#importing Packages from tkinter
from tkinter import *
from tkinter import messagebox

Player1 = 'X'
stop_game = False

def clicked(r,c):

    #player
    global Player1
    # global stop_game

    if Player1 == 'X' and states[r] == 0 and stop_game == False:
        b[r].configure(text = 'X')
        states[r] = 'X'
        Player1='O'


    if Player1 == 'O' and states[r] == 0 and stop_game == False:
        b[r].configure(text = 'O')
        states[r] = "O"
        Player1 = "X"

    check_if_win()
    # check_if_tie()
    # if check_if_win() == False:
    #     tie = messagebox.showinfo("tie","its tie")
    #     return tie
def check_if_win():
    global stop_game
    # count = 0

    for i in range(3):
        if states[i][0] == states[i][1] == states[i][2] !=0:
            stop_game = True

            winner = messagebox.showinfo("Winner", states[i][0] + " Won")
            # disableAllButton()
            break
```

```
    # for j in range(3):
        elif states [0][i] == states[1][i] == states[2][i] != 0:
            stop_game = True

            winner = messagebox.showinfo("Winner", states[0][i]+ " Won!")
            break

        elif states[0][0] == states[1][1] == states[2][2] !=0:
            stop_game = True

            winner = messagebox.showinfo("Winner", states[0][0]+ " Won!")
            break

        elif states[0][2] == states[1][1] == states[2][0] !=0:
            stop_game = True

            winner = messagebox.showinfo("Winner" , states[0][2]+ " Won!")
            break

        elif states[0][0] and states[0][1] and states[0][2] and states[1][0] and sta
            stop_game = True

            winner = messagebox.showinfo("tie", "Tie")

# Design window
#Creating the Canvas
root = Tk()
# Title of the window
root.title("GeeksForGeeks-:Tic Tac Toe")
root.resizable(0,0)

#Button
b = [
     [0,0,0],
     [0,0,0],
     [0,0,0]]

#text for buttons
states = [
     [0,0,0],
     [0,0,0],
     [0,0,0]]

for i in range(3):
    for j in range(3):

        b[i][j] = Button(
                        height = 4, width = 8,
                        font = ("Helvetica","20"),
                        command = lambda r = i, c = j : clicked(r,c))
        b[i][j].grid(row = i, column = j)
```

```
# Design window
#Creating the Canvas
root = Tk()
# Title of the window
root.title('GeeksForGeeks-:Tic Tac Toe")
root.resizable(0,0)

#Button
b = [
     [0,0,0],
     [0,0,0],
     [0,0,0]]

#text for buttons
states = [
     [0,0,0],
     [0,0,0],
     [0,0,0]]

for i in range(3):
    for j in range(3):

        b[i][j] = Button(
                        height = 4, width = 8,
                        font = ('Helvetica","20"),
                        command = lambda r = i, c = j : clicked(r,c))
        b[i][j].grid(row = i, column = j)
```

OUTPUT:



SUBMITTED BY-

V.PRANAV SHESU RAJU