**Logic implemented for the game and what does each function do**

**1. set\_tile(row, column) Function**

* **Purpose:** This function handles the action of a player clicking on a tile (button) in the Tic-Tac-Toe grid.
* **Logic:**
  + **Check if the game is over:** If the game is over (game\_over is True), the function returns immediately, preventing any further moves.
  + **Check if the tile is already taken:** If the tile at the specified row and column already has a symbol ("X" or "O"), the function returns to prevent overriding the existing symbol.
  + **Update the board:** If the tile is empty, it is filled with the current player's symbol (curr\_player), which is either "X" or "O".
  + **Check for a winner:** The function check\_winner() is called to determine if the current move resulted in a win. If so, the game ends, the score is updated, and the game status is displayed.
  + **Check for a draw:** If all tiles are filled (turns == 9) and there's no winner, it's a draw, the score is updated, and the game ends.
  + **Switch player:** If the game is not over, the function switches the current player from "X" to "O" or vice versa, and updates the label to indicate whose turn it is.
* **Implementation:** The function uses basic control structures like if statements to manage the game's flow, updating the game board and checking the game state.

**2. check\_winner() Function**

* **Purpose:** This function checks whether there is a winning combination on the board.
* **Logic:**
  + **Check rows and columns:** The function loops through each row and each column to check if all three tiles in any row or column contain the same symbol ("X" or "O").
  + **Check diagonals:** It also checks the two diagonals for the same condition.
  + **Return True if there's a winner:** If any of these checks find a winning combination, the function returns True.
  + **Return False if no winner:** If no winning combination is found, the function returns False.
* **Implementation:** This function uses nested loops and if statements to compare the text values of buttons, determining if they are identical and non-empty, which indicates a win.

**3. update\_tile\_colors() Function**

* **Purpose:** This function highlights the tiles that form the winning combination by changing their background and foreground colors.
* **Logic:**
  + **Check rows and columns:** Similar to check\_winner(), the function checks each row and column for a winning combination.
  + **Check diagonals:** It also checks the diagonals.
  + **Update colors:** If a winning combination is found, it changes the background color to "lightgrey" and the text color to "yellow" for the winning tiles.
* **Implementation:** This function directly modifies the button properties (bg and fg) to visually indicate the winning tiles.

**4. new\_game() Function**

* **Purpose:** This function resets the game to start a new round.
* **Logic:**
  + **Reset variables:** It resets the turns counter to 0, sets game\_over to False, and sets the current player (curr\_player) back to "X".
  + **Reset the board:** The function loops through all the tiles on the board, clearing their text, and resetting their colors to the default.
  + **Update the label:** It updates the label to indicate that it's "X's turn".
* **Implementation:** The function uses loops to iterate over the board and reset each button's state, preparing the game for a fresh start.

**5. exit\_game() Function**

* **Purpose:** This function closes the game window.
* **Logic:**
  + **Destroy the window:** The function calls window.destroy() to close the Tkinter window and terminate the application.
* **Implementation:** A simple function with one line of code to end the game by closing the GUI.

**6. Game Setup**

* **Score Dictionary:** The score dictionary keeps track of the number of wins for "X", "O", and draws.
* **Initial Variables:**
  + curr\_player is set to "X", indicating that "X" starts the game.
  + turns is initialized to 0, to count the number of moves made.
  + game\_over is set to False, indicating that the game is active.
* **Window Setup:** A Tkinter window is created and titled "Tic Tac Toe". A frame is used to hold the grid of buttons and other widgets.
* **Label:** A label is placed at the top to show whose turn it is or display the game result.
* **Board:** A 3x3 grid of buttons is created, each representing a tile on the Tic-Tac-Toe board. The buttons are configured with a font, colors, size, and a command that links to the set\_tile() function.
* **Restart and Exit Buttons:** Buttons for restarting the game and exiting the application are placed below the grid.

**7. Main Loop**

* **Purpose:** The window.mainloop() command starts the Tkinter event loop, which waits for user interactions (like button clicks) and updates the GUI accordingly.