Console-Based Tic Tac Toe with N by N Grid in Python

Project Overview

Tic Tac Toe is a classic two-player game where the goal is to place three (or more, depending on the grid size) of one's own marks in a row, column, or diagonal. In this project, we will build a flexible version of Tic Tac Toe that allows for an N by N grid, where N is specified by the players.

Step 1: Setup the Game Grid

1. **Initialize the Grid:**
   - Create an N by N grid initialized with empty spaces.
2. **Function:** `initialize_grid` **Purpose:**
   - To create and set up the initial N by N game grid.
3. **Description:**
   - Initializes a grid with N rows and N columns.
   - Each cell in the grid should be initialized to an empty state.

Step 2: Display the Grid

3. **Function:** `print_grid` **Purpose:**
   - To display the current state of the game grid to the console.
4. **Description:**
   - Iterates through each row of the grid.
   - Prints each cell in the grid in a formatted manner to ensure readability.

Step 3: Handle User Input

4. **Function:** `get_user_input` **Purpose:**
   - To obtain the coordinates of the cell where the player wants to place their mark.
5. **Description:**
   - Prompts the player to enter the row and column where they want to place their mark.
   - Validates the input to ensure it is within the valid range of the grid and the cell is not already occupied.

Step 4: Place the Mark

5. **Function:** `place_mark` **Purpose:**
   - To place the player's mark on the selected cell.
6. **Description:**
   - Takes the player's mark and the selected coordinates.
   - Updates the grid with the player's mark at the specified coordinates.

Step 5: Check for Win or Draw

6. **Function:** **`check_win`** **Purpose:**
    - To check if the current move results in a win for the player.
7. **Description:**
    - Checks all possible win conditions (rows, columns, and diagonals) from the position of the last move.
    - Returns `True` if a win condition is met.
8. **Function:** **`check_draw`** **Purpose:**
    - To check if the game has resulted in a draw.
9. **Description:**
    - Checks if the grid is completely filled without any player winning.
    - Returns `True` if no empty cells are left and no win condition is met.

Step 6: Switch Player

8. **Function:** **`switch_player`** **Purpose:**
    - To switch the turn to the other player.
9. **Description:**
    - Alternates between Player 1 and Player 2 after each move.
    - Updates the current player’s mark accordingly.

Step 7: Main Game Loop

9. **Function:** **`play_game`** **Purpose:**
    - To manage the overall game loop, handling the game flow and user interactions.
10. **Description:**
    - Initializes the game grid and sets the starting player.
    - Continuously displays the grid, handles user input, places marks, and checks for win or draw conditions.
    - Ends the game loop if a player wins or the game is a draw.
    - Switches the turn to the other player after each valid move.

Full Function Descriptions

Function: `initialize_grid`

- Initializes an N by N grid with each cell set to an empty state.

Function: `print_grid`

- Displays the current state of the game grid to the console.
- Uses specific characters or numbers to represent empty cells and players' marks.

Function: `get_user_input`

- Prompts the player to enter the row and column where they want to place their mark.
- Validates and returns the input.

Function: `place_mark`

- Takes the player's mark and the selected coordinates.
- Updates the grid with the player's mark at the specified coordinates.

Function: `check_win`

- Checks for a win condition from the position of the last move.
- Verifies rows, columns, and diagonals to determine if a player has won.

Function: `check_draw`

- Checks if the game is a draw by verifying if the grid is full and no win condition is met.

Function: `switch_player`

- Alternates between Player 1 and Player 2 after each move.
- Updates the current player's turn accordingly.

Function: `play_game`

- Manages the main game loop.
- Initializes the game grid and sets the starting player.
- Continuously updates and displays the grid, handles user input, and checks for win or draw conditions.
- Ends the loop when a player wins or a draw condition is met.

Implementation Tips

1. **Grid Representation:** Use a 2D list (list of lists) to represent the N by N grid.
2. **Player Marks:** Use distinct characters or symbols for Player 1 and Player 2 (e.g., 'X' and 'O').
3. **Win Conditions:** Ensure the win condition checking is robust to handle different grid sizes.
4. **Input Validation:** Handle cases where the user input is out of bounds or the chosen cell is already occupied.
5. **Game Loop:** Continuously update and display the grid, handle user input, and manage game state changes based on user actions.
6. **User Experience:** Provide clear instructions and feedback to the players, ensuring the game is easy to understand and play.