**Assignment: Tic-Tac-Toe Game Development**

Objective:

Develop a simple, console-based Tic-Tac-Toe game in Python that allows two players to play against each other on the same computer. The game should alternate turns between the players, accept player input to mark the board, and check for win conditions or a draw.

Requirements:

1. **Game Board Representation**: Implement a 3x3 grid where players can place their marks. The grid can be displayed in the console using numbers or letters to represent each cell.
2. **Player Input**: Allow the players to input their moves. Ensure the program checks for valid input (e.g., the cell is within the grid and not already taken).
3. **Win Condition**: Implement the logic to check if a player has won the game (three of their marks in a row, column, or diagonal).
4. **Draw Condition**: The game should check for a draw (no more spaces left to play and no winner).
5. **Game Loop**: Implement the game loop so that the game alternates turns between the two players, displays the updated board after each move, and checks for game end conditions after each move.
6. **Restart Option**: After a game concludes (win or draw), allow the players to start a new game without restarting the program.

Considerations:

* **Code Structure**: Aim for clean, readable code. Use functions to organize your code, separating the logic for displaying the board, taking player input, checking win conditions, etc.
* **Input Validation**: Ensure your program gracefully handles invalid input, such as out-of-bounds or already occupied cells, without crashing.
* **User Interface**: While this is a console-based game, strive for a user-friendly interface. Clearly indicate how players can make a move and display messages for invalid moves, turn changes, and game outcomes.

Tips and Tricks:

* **Modularize Your Code**: Break down your program into smaller functions, such as **print\_board()**, **take\_turn()**, **check\_winner()**, and **check\_draw()**. This approach makes your code more organized, readable, and easier to debug.
* **Use Data Structures Wisely**: Represent the game board as a list of lists. This structure simplifies accessing and updating the board's cells based on player input.
* **Loop Control**: Use loops wisely to manage the game's flow, such as a main game loop that continues until a win or draw condition is met. Within this loop, alternate between players' turns.
* **Keep It Simple**: Start with the basic functionality. Ensure the core game works flawlessly before adding enhancements like the restart option or improved input validation.

Extension (Optional):

Once you have the basic game working, consider adding one or more of the following features to enhance your project:

* **AI Opponent**: Implement a simple AI opponent that makes moves based on a set of rules (e.g., block the player from winning, choose a random free spot).
* **Score Tracking**: Keep track of and display the score across multiple games.
* **GUI Implementation**: As an advanced extension, explore creating a graphical user interface for the game using a library like Tkinter.

Submission Guidelines:

* Submit your Python script file(s) containing the complete code for the game.
* Include comments in your code to explain the purpose of functions and key sections.
* Ensure your code is thoroughly tested and free of bugs.