Chess Game Report

Objective:

The objective of this project is to implement a simplified version of a chess game in Python, incorporating basic move generation, game state evaluation, and minimax for automated gameplay.

1. Code Structure:

The code begins by importing necessary modules (copy) and defining constants such as BOARD\_SIZE, PIECE\_VALUES, and PIECE\_SYMBOLS for board representation and evaluation.

Key functions include:

evaluate\_board: Evaluates the current board state based on piece values.

get\_valid\_moves and get\_piece\_moves: Generate valid moves for players and individual pieces, respectively.

Check and game over functions (is\_check, is\_in\_check, is\_checkmate, is\_stalemate, check\_game\_over) handle game state evaluation.

The minimax algorithm (minimax) with alpha beta pruning is implemented for AI move selection.

make\_move applies a move to the board, and print\_board displays the board state.

2. Move Generation and Evaluation:

The get\_valid\_moves function generates all valid moves for a player by iterating through the board and calling get\_piece\_moves for each piece.

get\_piece\_moves implements move generation logic for different pieces, including pawn, knight, bishop, rook, queen, and king moves.

Evaluation of the board state is based on a simple piece count evaluation function using predefined piece values (PIECE\_VALUES).

3. Game Logic and AI:

The game loop (play\_game) manages player turns, allowing human input for moves and using the minimax algorithm for AI move selection.

The AI player evaluates possible moves using the minimax algorithm with alpha beta pruning, aiming to maximize its evaluation score.

4. Testing and Execution:

The code can be executed to play a game of chess, with the AI making moves based on the minimax algorithm.

The game checks for checkmate and stalemate conditions to determine the end of the game and announce the result accordingly.

5. Future Improvements:

Additional features such as pawn promotion, en passant capture, and castling can be implemented to enhance gameplay realism.

Advanced evaluation functions can be developed for more sophisticated AI gameplay strategies.

Conclusion:

In conclusion, this project successfully implements a basic chess game with AI capabilities using Python. Further enhancements and feature additions can be explored to create a more comprehensive and engaging chess playing experience.