Title: Tower Blaster Game Documentation

Introduction: The provided code is an implementation of a simplified version of the "Tower Blaster" game where a human player competes against a computer opponent to organize a tower of bricks from lowest to highest as quickly as possible. Each round, players can choose a brick from either the main pile or discard pile and swap it with a brick in their tower. The game continues until one player has their tower completely sorted.

1. **Importing Libraries**:
    - The `random` library is imported within the `main` and `shuffle_bricks` functions to allow for shuffling the main pile of bricks.
2. **Function Definitions**:
    - `setup_bricks`: Initializes the main pile with bricks numbered 1 to 60 and an empty discard pile.
    - `shuffle_bricks`: Shuffles the main pile using the `random.shuffle` method.
    - `check_bricks`: Checks if the main pile is empty, and if so, shuffles the discard pile, moves all bricks to the main pile, and clears the discard pile.
    - `get_top_brick`: Prompts the user to choose a pile, removes the top brick from the chosen pile, and returns the brick.
    - `deal_initial_bricks`: Alternately deals the first 20 bricks between the computer and human towers.
    - `add_brick_to_discard`: Adds a specified brick to the discard pile.
    - `find_and_replace`: Replaces a specified brick in a specified tower with a new brick and moves the replaced brick to the discard pile.
    - `computer_play`: Defines the computer's strategy to play its turn, which involves comparing the top bricks of the main and discard piles with the bricks in its tower.
    - `main`: The main function where the game is orchestrated. It initializes the game, shuffles the bricks, deals the initial bricks, and enters a game loop where the human and computer players take turns.
3. **Game Loop (within the `main` function)**:
    - The game loop continues until a player achieves a Tower Blaster, i.e., their tower is sorted in ascending order.
    - In each round, the human player is prompted to choose a brick and specify a brick to replace in their tower.
    - The computer player follows a set strategy to choose and replace a brick in its tower.
    - After each round, the program checks if either player has achieved a Tower Blaster and ends the game if so.
4. **Execution**:
    - The script is executed by calling the `main()` function within the `if __name__ == "__main__":` block at the end of the script.
5. **Error Handling**:
    - Error handling is implemented in the `main` function's game loop using a `try`-`except` block to catch any exceptions that may occur during a round

of the game, allowing the computer to continue playing its turn.

6. **Output**:
    - Various print statements throughout the code provide feedback on the current state of the game, such as the content of the main and discard piles, the towers, and the top bricks.
7. **Comments**:
    - The code contains comments explaining the purpose and functionality of each function and major blocks of code within the functions.
8. **Variables**:
    - Descriptive variable names are used throughout the code to indicate the purpose of each variable, such as `main_pile`, `discard`, `computer_tower`, `human_tower`, etc.
9. **User Input**:
    - User input is gathered using the `input` function to prompt the human player for actions during their turn.

This documentation provides an overview of the code structure, the purpose of each function, and the flow of the game.