Coding Standard

1. Naming Conventions

- **Variables and Functions:**
  - Use snake_case for function and variable names (e.g., `random_player`, `get_best_few_shot_examples_async`).
  - Names should be descriptive, indicating the purpose of the variable or function (e.g., `invoke_player_async`, `complain_if_unknown_player`).
- **Constants:**
  - Use UPPERCASE for constant variables and tuple names (e.g., `known_players`).
- **Classes:**
  - Use PascalCase for class names, though none are currently present in the provided code.

2. Code Structure and Organization

- **Modularity:**
  - Group related functionalities together. For example, all player-related logic (random, human, minimax, LLM) is separated into distinct functions.
- **Single Responsibility:**
  - Each function should handle a single responsibility (e.g., `random_player` for selecting a random move, `minimax_player` for implementing minimax logic).
- **Asynchronous Handling:**
  - Use async functions for handling LLM-based players (e.g., `minimal_gpt4_player_async`, `cot_player_without_few_shot_async`).
  - Avoid blocking operations in asynchronous code.

3. Error Handling

- **Exceptions:**
  - Raise exceptions with meaningful error messages in functions like `complain_if_unknown_player` and `complain_if_unknown_x_or_o`.
  - In critical cases, fall back to default behavior when exceptions are raised (e.g., fallback to `random_player` in case of an exception in `invoke_player_async`).
- **Logging:**
  - Track game progression through logs (e.g., turn details, board state, move costs).

4. Asynchronous Operations

- Asynchronous functions are utilized for any operation that involves LLM API calls (e.g., GPT-4, Gemini). Ensure the code is properly awaited to handle non-blocking I/O.
- Functions returning asynchronous tasks should follow the convention `*_async` to clearly indicate that they are asynchronous.

5. Code Readability

- **Comments and Docstrings:**
  - Include inline comments for explaining non-trivial logic.
  - Use comments to explain assumptions or decisions, e.g., why the fallback to random player is used when exceptions occur.
  - Add docstrings for functions explaining the parameters and return values.
- **Spacing and Indentation:**
  - Follow consistent 4-space indentation.
  - Ensure logical separation of blocks using blank lines (e.g., between function definitions and logical blocks within functions).