Response to Peer Feedback

1) The mutability point raised is valid, so we have altered the `board()` method to return a deep copy of the board. As per the API design principles, this will help to minimize not only mutability but also accessibility, since the user will no longer have access to or be able to mutate the original board used by the game.

2) This is a typo and has been changed so that both `TURN_PLAYER_1` and `TURN_PLAYER_2` are accepted. This typo in both the documentation and implementation would lead the user to believe that player 2 can't play first, which is not the case.

3) Even though the point being raised here is reasonable, we have decided that it is beyond the scope of the API. The API should be "as simple as possible but no simpler." In this case, we want the API to focus on game logic and for it to be the user's job to display the board. This allows for different users to display the board however they wish without feeling constrained by the API. Furthermore, many users will likely create a GUI and therefore not need the printing functionality. Besides, the `board()` method returns the board, which should be enough for the user to understand the board's state at any moment in the game.

4) We decided to make `drop_checker(col)` return the `game_state` because it is very convenient for the user. Otherwise, the user would usually have to call `drop_checker(col)` and `current_game_state()` sequentially to check if the move they just made ended the game or not. This implementation also doesn't hurt the user, as they can choose to use the `game_state` returned by `drop_checker(col)` or not and whatever decision they make shouldn't affect their implementation. While yes, the design principles do suggest that 'components should be small and composable,' in this case, `drop_checker(col)` returning the game state would not negatively affect users who do not need it returned.

5) Similarly to feedback 3 above, the API's focus is on game logic. Printing or other kinds of displaying can be handled by the user depending on their user interface requirements. This allows flexibility on the user's side to come up with robust CLIs or GUIs.

6) We believe that having a method that returns if the game is over is a very useful convenience method, as the API should not force the user to do anything that it could make itself. We therefore made the `is_game_in_progress()` method public.

7) The word "inputted" was changed to "the provided column" to add more clarity.

API Design Rationale

In addition to the design decisions regarding the feedback above, we also made the following decisions when designing our API:

- `drop_checker(col)` method makes the move into the specified column for the current player.
    - Naming: drop_checker represents the actual action the player does when they are playing connect 4, which is to drop a checker into the grid.
    - The method only takes the column to drop a checker into. There is no need to say which player is moving because the most common use case is to play following the standard game rules, that is, alternating plays between players.
    - “Fail fast or better yet don’t fail at all”: Since the method automatically plays for the current player, it doesn’t rely on the user to specify who is making the move. This makes it impossible to make a wrong move in this regard.
    - “Don’t make users do anything the library could do for them”: The method validates the input, drops the checker to the bottom of the column, and checks for any game wins. All of the game logic is handled by the API and is contained within one method.

- We used enums instead of strings because they are more readable and safe compared to strings, enums makes it impossible to do what's wrong, and more importantly they fail fast i.e most of the IDEs and editors will warn the user if they didn’t use enums correctly.

- “APIs should be approachable” principle guided the design which allows users to play within only two steps. The basics of playing a game only requires the user to initialize a new game and then drop checkers with the `drop_checker(col)` method.

- “An API must be appropriate to its audience” principle is what we consulted to make columns range from 1 - 7 rather than 0 - 6. The reason 1 - 7 is more appropriate to the user is because it makes more sense to players.