Assignment 1: Manipulating the Game Board

Objective

In this assignment, you will implement and test functions that manipulate the Connect 4 game board, allowing players to place tokens in specific columns and check for available spaces.

Instructions

1. Complete each function according to the specifications described below.
2. After implementing a function, run `tests.py` to confirm that it works as specified. Many functions rely on previously written functions, so complete them in order. Do not move on to the next function until the current one works correctly.
3. If a test fails, fix the related function and re-run your tests.
4. The `tests.py` file includes test cases for several functions. However, some functions do not yet have tests. You will need to write your own tests for those functions.

Tips for Testing:

- Make multiple test calls to each function. Consider edge cases and potential issues that could break the function.
- Test **edge cases**, such as:
  - Placing tokens in the first and last rows/columns.
  - Dropping tokens into full columns.
- For functions that return Boolean values, test both valid and invalid cases.

Function Descriptions

`make_board(num_rows=6, num_cols=7)`

Your first task is to create the game board. A Connect 4 board is a grid with multiple rows and columns. This function should:

- Create a **2D list** (a list of lists) with the specified number of rows and columns.
- Default to a **6-row by 7-column** grid unless different values are provided.
- Initialize all slots with `0`, where each `0` represents an empty space.
- Return the **2D list**.

Note: This function is purely for **data management** and does not control the game's visual appearance.

`row_is_valid(board, row)`

This function checks if the given row index is valid. A row index is **valid** if:

- It is greater than or equal to `0`.
- It is **less than** the total number of rows in the board.

This ensures that players do not attempt to interact with a non-existent row.

`column_is_valid(board, column)`

Similar to `row_is_valid`, this function checks whether the provided **column index** is within bounds. A column index is **valid** if:

- It is greater than or equal to `0`.
- It is **less than** the total number of columns in the board.

This ensures that players only drop tokens into valid columns.

`location_is_valid(board, row, column)`

This function combines the checks from `row_is_valid` and `column_is_valid`. It should:

- Return `True` if **both** the row and column indices are within the board's bounds.
- Return `False` otherwise.

Hint: Instead of rewriting the logic from `row_is_valid` and `column_is_valid`, call those functions inside this one.

`location_empty(board, row, column)`

This function checks whether a given location on the board is **empty**. It should:

- Return `True` if the board slot at the given row and column contains `0`.
- Return `False` if the slot is occupied by a token.

`place_token(board, token, row, column)`

This function places a player's token on the board at the specified row and column. The token should replace the `0` at that position.

- The **token** can be any value, but use `1` and `2` to represent Player 1 and Player 2, respectively.
- It **does not** need to check if the space is empty before placing a token. Overwriting an existing token is allowed. This design makes the function reusable for different types of games.

`column_available(board, column)`

This function checks if there is space available in a specific column to drop a token.

- A column **has space** if the topmost row in that column is empty (`0`).
- If the first row is already occupied, the column is full, and no more tokens can be dropped.
- The function should return:
    - `True` if there is space available.
    - `False` if the column is full.

`drop_token(board, column, token)`

This function **drops a player's token** into a specified column. It should:

- Find the **lowest available space** in the column (i.e., the first empty row from the bottom).
- Place the token in that row.
- Return the **row index** where the token was placed.
- If the column is **full**, return `-1`.

Hint: In a physical Connect 4 game, a token falls until it reaches either the bottom of the board or another token. However, instead of checking from the top down, a **more efficient approach** is to:

1. Start checking from the **bottom row** of the grid.
2. Stop at the **first empty slot** found.
3. Place the token and return the row index.