2016-09-26 Lab

There are two parts to this lab: writing test cases (for which you will have about an hour), and writing code (the rest of the lab). To start, download `testcases_starter.py` and `connect_four_starter.py` from Moodle.

Introduction

In this lab, you will be finishing an implementation of a text-based Connect Four game. Because writing the complete game is non-trivial, I am providing you with a lot of the infrastructure, including printing the board. However, three key functions have been left out:

- `lowest_empty_row`, which returns the lowest row in a column that is empty
- `get_input`, which gets from the user which column to play next
- `has_connect_four`, which checks if someone has won

Part 1: Writing Test Cases

Before you start writing code, however, you need to come up with test cases to make sure your code is correct. The practice of writing test cases first, before writing code, is called test-driven development. For this lab specifically, you will be writing test cases for the `has_connect_four` function.

Take a look at `test.py` from Moodle. The file already has two test cases, which you can copy and modify for your own purpose. You can also look through `connect_four.py` for descriptions of what the functions do, but the big idea is that `has_connect_four` should return `True` if the board has a winning connect four, and `False` otherwise. Let's take a look at the first test case:

```
board = str_to_board("""
0  | _  _  _  _  _  _  _ |
1  | _  _  _  _  _  _  _ |
2  | _  _  _  _  _  _  _ |
3  | _  _  _  _  _  _  _ |
4  | _  _  _  _  _  _  _ |
5  | _  _  _  _  _  _  _ |
   +---------------------+
     0  1  2  3  4  5  6
""")
if has_connect_four(board):
    fail(board)
```

The `str_to_board()` function creates a board (a nested list of strings) from a string. Since this board is empty, the correct return value from `has_connect_four` is `False`. The second test case is similar, except this time `has_connect_four` should return `True`. **You must call the `fail` function if a test case fails.** Note that the boards you use as your test cases for `has_connect_four` DO NOT have to be valid boards; look at the third test case as an example. However, your boards to test `lowest_empty_row` *should* obey gravity (ie. you should not have floating tokens). You do not have to write test cases for `get_input`.

Your job, for the first 30-60 minutes, is to write more test case. This is so that when you actually write `has_connect_four`, you will know whether your function is correct or not. **Your test cases will be graded based on how many people get them wrong**. Make sure you think of tricky cases. An incorrect test case - saying that `has_connect_four` should return `True` on an empty board, for example - will get you an automatic 0 for this part of the lab. Keep in mind that test cases are for you to make sure your function works, and do not have to be exhaustive.

Submit your test cases to the "Connect Four Test Cases" project on the autograder when you are ready.

Part 2: Writing Code

DO NOT START THIS PART OF THE LAB UNTIL YOU HAVE SUBMITTED YOUR TESTS CASES

The coding part of this lab is completing the two functions at the top of `connect_four.py`. Before you start, however, you will need to know a little more about how the board is represented. As I mentioned earlier, the board is really a list of lists of strings. The board from the second test case is represented as:

```
board = [ ["_", "_", "_", "_", "_", "_", "_"],
          ["_", "_", "_", "_", "_", "_", "_"],
          ["_", "_", "_", "O", "X", "_", "_"],
          ["_", "_", "O", "X", "X", "O", "X"],
          ["_", "_", "X", "O", "O", "O", "X"],
          ["_", "X", "X", "O", "O", "X", "O"] ]
```

For your convenience, three constants have been defined to make your code easier to understand:

```
EMPTY = "_"
EX = "X"
OH = "O"
```

So, for example, to check that the piece at row 5, column 1 is an `X`, you would do:

```
board[5][1] == EX
```

You will need to write three functions for this lab:

- `lowest_empty_row`, which returns the lowest row in a column that is empty
- `get_input`, which gets from the user which column to play next
- `has_connect_four`, which checks if someone has won

There are comments in the file for what the functions should do. **Do not change any other function in `connect_four.py`.**

When you are done, submit your code to the `Connect 4` project on the autograder, and fill out the peer evaluation.