Grader Description:

This lab is for making a single specified move on a specified Othello board.

There are up to three command line inputs to the script for this lab, in the order of board, token, move. All input are optional. board is a length 64 string from characters in {*"xo.XO"}. If not provided, board is a starting Othello board: '.'*27+'ox......xo'+'.'*27. The token, 'x' or 'o', is the side that is to play next. It defaults to the side that would play next in the absence of any pass moves. If only one side can play in the current move, however, then it defaults to that side. The move (defaulting to no move) is an integer in [0,63], but it may also be given in A1 notation, where the letter indicates the column label (A-H) and the 2nd character corresponds to the 1-based row (1-8). Case is to be ignored for all inputs.

The output should be two snapshots, a before and after snapshot. However, if no move is given, then only a before snapshot should be displayed.

A snapshot is a move, a 2D representation of a board, a 1D representation of the board (ie. a string), a score, and a listing of the possible moves at that point, in the given order. There is no move for the first snapshot, but for the second snapshot (if there is one), a move could be shown in the format x plays to 25 or o moves to 36. The move should be given in standard form (ie. an integer in range(64)). The 2D representation should indicate the possible move locations with an asterisk in place of a period - you may find this helpful for your own debugging purposes. A score should look like: 28/31 where the first number is the number of x tokens and the second number is the number of o tokens on the board.
The listing of possible moves should be in the form of:
Possible moves for o: 24, 15, 7, 63

There are two special cases that should be addressed. If the game is over (hence, no moves), then there is no final listing of possible moves. The other special case is where the resultant board is a pass for the opposite player. In this case, the first player is to go again, and the list of moves should reflect it. In the testing for this lab, the script will never receive inconsistent input (such as an illegal move).

Eric's Notes:

Again, the first couple labs are more annoying than anything.

Let's start with the inputs. All of them are optional, so we have to identify the type of inputs. Luckily, this is a simple task. Think about the differences between inputs. If you have too much free time, you can also use set subtraction (a fun python feature).

Actually making the move should take minimal extra coding from the last lab, since you are still only processing a singular board. In other words, finding possible moves and making those moves should go hand in hand.

Snapshots are easy and aren't required for the later labs.

Final note, make sure your code is modular! Use functions and write comments. Making simple and fast functions now will make later labs a breeze.