Assignment 7: Battleship

This assignment is to be completed individually. It is not a team project.

You must document (using comments in your code) all resources (beyond our official textbook, Sakai, the class discussion forum, or the class website) that you used to help you complete this assignment. For example, if you referenced a code example from an online website such as Stackoverflow or github you could acknowledge it as follows:

```
// Next we print 'Hello, World' to the console.
// Based on help from: https://stackoverflow.com/questions/826948/syntax-error-on-print-with-python-3
print("Hello, world!")
```

You must provide an attribution to any resource you consulted to complete this assignment except for our official textbook, Sakai, class discussion forum, or the class website. Resources that require attribution include — but are not limited to — websites, books, notes from other students, tutors, or help from other people (friends, classmates, etc.). Under no circumstances are you to look at or copy any material related to another student's solution for this assignment.

To repeat, you must attribute **any resource** you consulted to complete this assignment other than our official textbook, Sakai, class discussion forum, or the class website. Failure to provide attribution is a violation of the honor code.

Overview

In the last and final assignment for this semester, you'll be developing an electronic version of the classic board game Battleship. If you are not familiar with this 2-player game, I suggest that you take a look at the following resources:

- Read Description of the Battleship Game on Wikipedia
- Watch How to Play: Battleship on YouTube
- Try playing a free online version of the game

For this assignment, we'll use a 10x10 board with following the 5 ships:

- Aircraft Carrier (size 5)
- Battleship (size 4)
- Submarine (size 3)
- Destroyer (size 3)
- Patrol boat (size 2)

At a high level, the game will proceed as follows:

- Players first position their own boats on the grid such that no boats overlap. Boats can be positioned vertically or horizontally in any position as long as the entire boat fits on the grid without overlapping another boat.
- Once the boats are positioned, the game play begins with each player alternately choosing a spot on their opponent's board to attack. Players are able to see both their own board and their opponent's board, both of which show the history of attacks by the two players. However, players can only see the positions of boats on the own board. When they see their opponent's a player can only see the history of where they have attacked, and if those shots resulted in a hit or miss.
- Players choose spots to attack with the goal of "sinking" a ship. A ship is considered sunk when all of its spots on the grid have been attacked. The first player to sink all of his/her opponents ships wins the game.

Now that you are familiar with the basics of the game, you should be ready to begin developing your own version. In class, we'll discuss what an object-oriented design for the program should look like. I am also providing you with skeleton code that has a fully functional 2-player version of Battleship.

You job in this assignment will be to extend that skeleton code, based on the design we discuss in class, to support a 1-player version. To do this, you'll need to create a computerized player that can play the game without any user input. You'll also need to add code to keep track of some game statistics.

As usual, the assignment is split into basic- and advanced-level requirements as outlined below.

Skeleton Code

Like A6, you won't be starting from scratch for this assignment. Instead, you should begin with the following two Python files. If you download and run the code, you will find that it includes a fully-functional Battleship game. It even asks if you want to play with 1 player or 2. However, the skeleton code will not respect your choice. It will ONLY work as a 2 player game, no mater what you ask for. Making it work in the 1 player mode is the basic requirement for this assignment.

- Class definitions: battleship.py
- Main code to launch the game: assignment7_changetoyouronyen.py

Your very first order of business, after reading through the assignment, should be to download, rename, and run this skeleton program. First, make sure it runs successfully. Then, make sure you understand how it works. In particular, pay attention to the class definitions and how they interact to support the game. Play a few times with the debugger to see how it all works.

Basic Requirements

Satisfying all basic requirements perfectly, with no points deducted for any reason, would earn a maximum score of 8 out of 10 for this assignment.

Your program should allow 1 player to play a game of Battleship against a computer opponent. That includes:

- A computer player that functions like a human player, but without any user input.
- Your program must allow users to choose between a 1 player game (a human player against a computer player) or a 2 player game (two human players; provided in the skeleton code). Both modes should be fully functional when you submit your final program.
- Your computer player must generate different boat positions and attacks each time a game is played. You **cannot** "hard code" your computer player to make the exact same moves (boat positions and attacks) each time.

Your algorithm for the computer player's attacking strategy can be as simple or complicated as you wish. The key requirement is that your algorithm attacks the human player's board and would, at least eventually, result in the sinking of all ships if a game were to last long enough. You can find an interesting discussion of different algorithmic strategies on this website.

The console output produced by your program should be nearly identical to the sample output provided below. While the specific boat and attack positions will vary game-to-game, the prompts and messages printed to the console when using your program should match those in the sample output to receive full credit.

Advanced Requirements

Satisfying both the basic and advanced requirements perfectly, with no points deducted for any reason, will result in a full 10 out of 10 score for this assignment.

Expand on the basic requirements by extending your program as follows.

- Add code to keep track of how many times each player attacks the opponent, how many times those attacks miss, and how many times they hit.
- Display the current statistics for each player (attacks, hits, misses) at the start of each turn.

Sample Output

An example of the output produced by my solution to this assignment can be found here.

Grading Criteria

This assignment will be graded on a 10 point scale. Your grade for this assignment will be based on a combination of factors including:

- Correct functionality (e.g., Does your Python code do what it is supposed to do as outlined in the basic and/or advanced requirements?)
- Clarity of your solution (e.g., Did you solve the problem directly and efficiently? Or is your answer excessively complex and/or inefficient?)
- Coding style (e.g., Is your code readable with comments, meaningful variable names, and good whitespace/indentation?)
- Meets submission requirements (see below)

Seeking Help

For general questions about Python, please use the class forum on Piazza to seek assistance. For questions that are personal in nature or that would reveal a solution to the assignment, you ask for help by email or during office hours. However, please note that emailed questions will not receive an immediate response. It is likely that it will take 24-48 hours for me to respond.

Submitting Your Solution

Please Note: You must name the main python file for your assignment "`assignment<number>_<onyen>.py`". For example, for assignment 3 I would name my file `assignment3_gotz.py` because my onyen is gotz.

Using the wrong name for your file will be cost you points on your assignment grade. Please follow this requirement carefully!

Please submit your assignment via Sakai. You should submit a zip file containing your entire project folder. To create the zip file, follow the submission instructions that have been posted at the bottom of the "Other Information" page on our course website.

The due date for this assignment can be found on the course schedule.