CS 150: Intro to OOP, Spring 2012
Assignment 9

Due *via email* by 11:59pm on Friday, May 4, 2012

For this assignment, you must work individually. Please note the distinction made in our academic integrity policy between general course material and work which is submitted for this course. We consider the use of the Python language syntax and the cs1graphics package (not needed for this assignment) in the category of general course material, which you may discuss freely. However, you must avoid any discussion of code which is specific to this assignment. You should not receive direct help from others, nor should you share your own source code with others.

The Assignment

Your objective in this assignment is to write a client and server that can play the game of Bulls and Cows against each other. This is a two player game, where (for us) the client will try to guess the secret 4-digit number that the server has. The server's secret number is composed of four unique digits. So for example, 1234 is a valid secret number, but 1123 is not.

Gameplay

The game proceeds in rounds. In each round the client sends a guess for the server's secret, and the server responds back with how close the guess was. More specically, when the client sends a guess, the server will tell the player how many digits of the guessed number matched the secret and were in the correct position (called bulls), and how many digits matched the secret but were in an incorrect position (called cows). Examples:

- For secret 1234, guess 1873 has: one bull (the digit 1) and one cow (the digit 3)
- For secret 1234, guess 1384 has: two bulls (the digits 1 and 4) and one cow (the digit 3)
- For secret 1234, guess 6789 has: zero bulls and zero cows

The client wins when the user guesses the secret number of the server (which hopefully will happen for each of you).

You can read more about bulls and cows here: `http://en.wikipedia.org/wiki/Bulls_and_cows`

Networking protocol

Your protocol will work with the TCPServer and BaseRequestHandler classes in Python, similar to our webserver and mathserver examples in class. In particular, you will need to write a Handler class that inherits from BaseRequestHandler for the server side.

In addition, you will need to come up with a protocol that allows the client to send a guess and the server to respond with information. For example, you might send `GUESS: wxyz` as the client's message for a guess. The server will need a way to send back either the number of bulls and cows, or a WIN message to the client. You may design your own protocol, but please document it in comments and docstrings for the class.

The client will must also give the user the option of requesting a new secret number from the server, so that it can play the game multiple times with different numbers

Input

Your code (for either the server or client) will need to prompt for input before it begins running.

The server needs to prompt for the secret number, as well as the port it will run on. (Please make this an actual raw_input command, so that I can change the port easily.)

The client will also prompt for the port number, so that it can connect to the same one that your server is currently running on. It should also provide a menu of appropriate options, or at least a yes/no question of if they want to play the game. It should then send a request to the server for a secret number to be picked, and allow you to enter guesses. Each guess along with the server's response should be printed.

Implementation

In essence, there are three parts you need to complete:

- **Getting a secret number and checking a guess against the secret number and calculating the proper response**. This is not the focus of this assignment, and you can indeed nd an implementation for this here: `http://rosettacode.org/wiki/Bulls_and_Cows#Python`

  Feel free to use their code (suitably modified) as needed, although you will probably need to make a few changes to have it run for the server.

- **Sending and receiving messages on the network**. This is really two parts, the server which handles incoming guesses from the opponent, and the client which issues guesses to the opponent. See our webserver example or mathserver example for relevant information and examples.

Extra Credit

For the homework submission, you only need to set up the game so that you can play it against the server (which will generate a random secret number).

For extra credit, code a better client which will make guesses for you until it discovers the correct number. Your program can blindly make guesses until it stumbles upon the correct secret number, or it can try to learn something from what the opponent responds with. This part is completely optional, and Ill leave it up to you. It would be cool to implement some simple calculations to reduce the number of guesses (and would be worth more extra credit).

Note that you are welcome to research better strategies - there are a ton of references available on this online. Please document any sources you do use in a comment, however.