IMPORTANT: Python assessment (due Saturday 11:59PM)

Important note: You have until Saturday at 11:59PM to complete the Python assessment. Please do not let this assessment interfere with the I-School social event tonight. Also, please do not let this assessment interfere with your attending classes today or Friday.

All of the place-out assessments have been graded and we've sent e-mail to everyone participating in the assessments. (if you did place out, but have changed your mind and would like to take the course, you are very welcome to do so.)

Below is the Python assessment. Depending on your ability in Python, it may take as little as 10 minutes or as long as 4 hours to complete. To make sure that time is not an issue, we are allowing folks to submit the assessment anytime between now and Saturday at 11:59.

I heard a rumor that there is an I-School student social event (called "Thirsty Thursday," I believe) tonight. If that sounds like something you'd like to attend, please do so -- I don't believe that it should interfere with your completing the Python assessment.

During the next few days, the Info 206 team will be monitoring Piazza and our e-mail to answer any questions that may come up. Good luck on the Python assessment, and we'll see you in class on Tuesday morning!

All students need to show basic proficiency in Python to enroll in Info 206. To do this, we are giving the following easy assessment, which is due by Saturday, 8/30, by 9PM.

Step 1: Sign up for Code Academy http://www.codecademy.com/ - it is free. Go to the Python course and go to the Battleship http://bit.ly/206-python. Work through the entire example up to step 19/19. (If anything is unfamiliar to you, feel free to work through earlier parts of the codeacademy course. You do NOT need to turn this step in.)

Step 2: Now we are going to modify the program, basically along the lines of Extra Credit 02 in lesson 19/19. Here are the modifications:

(a) The first battleship will be size 3x1, which will have a random vertical orientation half the time and a horizontal orientation half of the time. For example, the battleship could occupy three consecutive spots on a single row or a single column. The location of the battleship should be random, but the entire battleship must fit within the 5x5 board.

(b) The second battleship will be size 2x1, and it will also have a random vertical and horizontal orientation. It will also have a random location (on the 5x5 board), but it may not touch the first battleship.

(c) A player "wins" if the player guesses *any* position on which *any* battleship is located. The player does not need to find the location of both battleships or the full spread of a battleship; rather, the player just needs to find one location on the board which also is occupied by (part of) a battleship.

(d) If the player guesses row 17 and column 17, then the following should happen:

(i) no error message ("Oops, that's not even in the ocean") is printed
(ii) instead a debug printout showing the location of the battleships (along the lines of lesson 9/18) is printed
(iii) the "guess" does not count against the four guesses that you give the user.

Put the final program in "scrypt.py" and e-mail it to i206-instructors@ischool.berkeley.edu by 11:59PM on Saturday.