Computer Science — Python — HW #10

Assigned on Monday, 2017-03-06. Due on Tue, 2017-03-14. Not HW will be assigned this Wednesday.

1. Read chapters 10 and 11 of Think Python this week. (Continuing with two chapters per week.)
2. **[Turn in]** "Craps". [Please think about this before Wednesday, 2016-03-08. We'll work on this in teams in class.]

Here are the (slightly simplified) rules of each "round" of a dice game called craps.

* A player starts out by rolling a pair of six-sided dice.
* If the first roll (i.e., the sum of the two dice) is a 2, 3, or 12, then the player loses.
* If the first roll is a 7 or 11, then the player wins.
* If the first roll isn't any of those listed above, then the first roll becomes the player's "point", and play continues, with the player repeatedly rolling the same pair of six-sided dice. For this continued play:
  + If the player rolls a 7 before rolling the "point", then the player loses the round.
  + If the player rolls their "point" before rolling a 7, then the player wins the round.
  1. Download the script **craps.py.zip** from <http://www.nyhs.org/Page/661>
  2. Fill in the code for **get\_round\_outcome**, so that it plays a round of Craps, and returns either **LOSE** or **WIN**, as appropriate.
     + **Hint:** Make a list of the information you need to track (e.g., the player's "point"), and then name variables that you will introduce.
     + Recall how the **'in'** operator works in Python:

3 in [1,2,3] # Value is True

5 in [1,2,3] # Value is False

* 1. Submit your modified version of the script **craps.py** (as an **attachment**, of course), together with the output from running it (i.e., the line printed out at the end of the script, after the call to **play\_set**).

1. **[Turn in]** Find-the-pebble.

I wrote the first version of a game I'll call Find the Pebble. But there's a bug in the initial text version. Specifically, it seems to take way too long to find the pebble. Boo!

The idea of the game is that the user first chooses the dimensions of a game board. Then the computer chooses the location of a pebble that will stay at a fixed location on the board while the player chooses locations by specifying x and y coordinates. For each guess the user makes, the computer tells the user how far off their guess is, in terms of Cartesian distance (i.e., the square root of the sum of the squares of the x- and y- distances).

It seems impossible to win the game for a large board. If I choose a smaller dimension, I eventually win, but it still seems to take forever! There might be a problem with the distance function, but maybe not.

* 1. Download the script **find\_the\_pebble.py.zip** from <http://www.nyhs.org/Page/661>
  2. Take a look at the code and run the script. See if you agree that it doesn't run the way you'd hope it would.
  3. Fix whatever the problem/bug is, and submit the corrected version of **find\_the\_pebble.py** as an **attachment**.
  4. You're welcome to collaborate on this, but please don't reveal the bug(s) to anyone you're not collaborating with. The learning benefits come from working through the problem—not from being handed the answer.

A few debugging tips:

* Choose small values for the dimension of the game board. It might be easier to see what's going on.
* Imagine you're the computer executing the code, step by step. Seek to fully understand it.
* As needed, add print statements to show where the "flow of execution" goes. For example, you can add a new first statement to a function (suppose it's called 'my\_func') of the form:

print('Now entering function my\_func')

* As needed, add statements to print out values that might help shed light on what's going on.

1. **[Turn in]** Submit three more ideas for your course project.