NUMBER GUESSING GAME

HOMEWORK 2, DUE THURSDAY, MARCH 6, 2014

Files

- biof309_hw2_solution.pyc - A byte compiled, fully playable solution to this assignment that is not human readable.
- biof309_hw2.py - A partially completed program where you fill in the blanks.
- biof309_test_hw2.py - Autograding program that will test your work.
- biof309_hw2_directions.pdf - This file.

Description of the Program

The object of the game is for the user to guess an integer chosen randomly by the computer. At the start of the game the user chooses the range of values in which the random number should fall. The game can have up to ten players who will take turns guessing. The program keeps track of the time it takes for each player to guess their number and how many turns s/he needed. At the end of the game it prints out a report on how the contestants did, the winner being the one who took the least amount of time to guess their number. The program then asks the user(s) if they want to play again.

Directions

(1) Download the files. Run the solution code. Run the test code on the solution to see that it works.

(2) Rename the template file to follow the naming convention: yourlastname_firstinitial_hw2.py

(3) Fill in the blanks. You are not allowed to change the function signatures, meaning you can't change the name of the function you're filling in, nor can you change the arguments to those functions. It is also not allowed to change any other part of the program, including the other functions and their code. Play your game, and test that it emulates the behavior and functionality of the solution.

(4) Run the autograder code on your file on the command line with the command python biof309_test_hw2.py yourlastname_firstinitial_hw2.py to see how you did.

(5) Send me an email with your program as an attachment with the subject "<your last name>hw2 submission".

Concepts used

The concepts used in this program may include, but is not limited to, information capture (i.e., `input()` etc.), conditional statements (i.e., `if ... else` etc), dummyproofing (i.e., checking to see whether user's input is valid), loops, using an infinite loop (i.e., `while( True ):`, using `break` to get out), etc...

Python Packages to be Used

`time`, `random`

Tips

- Each function has a docstring that says exactly what the function is supposed to do. Read the docstrings very carefully.
- You should read and understand the other parts of the program to see how the other parts will use your function, as well as for tips on how to structure your functions.
- This program should PEDANTICALLY check user input for validity. While you're writing your code, try to think about all the ways the user can enter the wrong thing, and write the code with guardrails to make sure it doesn't crash if the user enters garbage. At no point in playing the game should any SyntaxError or uncaught Exception be raised.
- While you're writing the code for your two functions, you might find it easier to test just the function you're working on independently, rather than run the whole program. You can do that by calling the functions individually from the "main" part of the program at the bottom of the template file, e.g., comment out the call to RunGame() and make the call to your function instead.
- When you run your code, you may observe that the program has a logic error (a bug) because it does not behave as intended. In order to track down bugs, you can insert print statements printing out the values of the variables you're using, or you can use the Python debugger to step through your code line by line to see what it's doing in real time. To do the latter, insert the line "import pdb; pdb.set_trace()" somewhere above the code where you suspect the bug is and run the program. The program will stop at the line after the import. Google around to learn about the commands used to drive the Python debugger.