Guess the Number - Code Clinic tips

This page include helpful tips based on our experience in helping students in the "Code Clinic" (interactivepython@online.rice.edu). Be sure and take a look at these tips if you get stuck.

1. Common issues from previous versions of IIPP
   - Remember that the input handler `input_guess(guess)` in the program template takes a parameter guess that is a **string**. Your first statement in `input_guess` should convert guess into an integer using the function `int(guess)`.
   - You should **not use** a While loop in this mini-project. If you are tempted to use one, stop and think about how input fields work. Each call to `input_guess` advances the game by one guess, not lots of guesses.
2. Testing `input_guess()`

   The key function in "Guess the number" is `input_guess()`, the event handler for your input field. It takes a number (as as string, compares it to the global variable containing the secret number, and prints out one step of the game. To help you determine whether you implementation of `input_guess()` is working, we have prepared the following testing template with the same structure as the practice exercises.

   Paste your implementation of "Guess the number" into the template below and run the template. The bottom of the template contains a sequence of calls to `input_guess()` for three games of "Guess the number". Uncomment each sequence of calls and check whether the output in the console matches that provided in the comments below. Note that your output doesn't have to be identical, just of a similar form. If the global variable containing your secret number is **not** called `secret_number`, you should change the assignment in the testing code to use your variable name for the secret number.

   http://www.codeskulptor.org/#examples-gtn_testing_template.py
3. Maintaining the current guess range and starting/restarting your game

   For the last two steps, you need to vary the range of your game. I suggest that you create a global variable (call it `num_range`) that contains the range for the current game. Now, your button handlers can simply set `num_range` appropriately and call the function `new_game()` that restarts your game.

   This function `new_game()` should initialize all of the appropriate global variables as well as print out any necessary messages when you start/restart a game. You should call `new_game()` in four places: in each of the two buttons handlers, after a completed game in `input_guess()` and before `frame.start()` to initialize the first game.
4. Limiting the number of global variables

Many of the students that sought help in the Code Clinic in previous session of IIPP made two copies of each of their global variables, one for each range. For example, they might have had `secret_number100` and `secret_number1000`. This design choice ignored the suggested mini-project development process and makes your code very complex. Just keep a single copy of each global variable and initialize it correctly based on the current range of guesses (also stored as a global variable). This choice will make your code much simpler.

5. Terminating the game correctly with limited guesses

   When you modify your code to restrict a game to 7 or 10 guess, there is a temptation to just add a final `if` clause to your existing code yielding something like this

```
if correct:
        print "Correct"
elif higher:
        print "Higher"
elif lower:
        print "Lower"
elif out of guesses:
        print "Out of guesses"
```

   Note that structuring you code in this way won't work since one of the first three cases will evaluate to `True` before the last clause is ever checked. Try checking for whether the player is out of guesses first instead.

Created Wed 28 Jan 2015 3:14 AM CST

Last Modified Wed 28 Jan 2015 3:30 AM CST