CS150 - Lab Prep 4
Due: Friday Oct. 5, at the beginning of class

For our lab, we're going to be writing a program that randomly generates math problems (i.e. math expressions like `3*4+10`) and counts how many of these questions a person can answer correctly in a fixed amount of time (say 30 seconds).

Half of this lab prep is playing with a few things and there is nothing to hand in for these parts ("Guessing game", "A demo" and "Infinite loops"). For the "Timing" section, answer the two questions (labeled 1 and 2) and bring your answers *on a piece of paper* to class on Friday.

Guessing game

Look at the class notes for Wednesday on the course web page and look at the `number_guessing_game` function. Play with it a few times and then look at the code and make sure you understand what it is doing. If you don't send me an e-mail or come see me in my office hours.

I've also provided a function `number_guessing_game2`, which behaves the same way, but is structured a bit differently. Look at this variant and make sure you understand it as well. Either way of implementing this function is fine.

A demo

(Optional, but strongly suggested)

To get you familiar with what you'll be implementing for your assignment this week, I've provided you with an example program to run. Play with it a little bit before Friday.

Note this will only work in the Mac lab, so you'll need to poke your head in MBH 505. To run it:

- Run `Terminal`. In your dock at the bottom of the screen there should be an icon that looks like:

Click on it. If you can't find it in your dock, type `command + space_bar` to open the search in the upper right, then search for "Terminal" and select it that way.

- In the `Terminal` window type:

  `python /home/dkauchak/PUBLIC/cs150/assignment4/mathwhiz.pyc`

  This should execute the example program in the terminal window. If you ever want to exit or quit early, you can type `control + c`.

Timing

Python has a module call `time` that has a variety of functionality for asking questions about time, dates, etc. If you're curious, you can check out the documentation at:

http://docs.python.org/library/time.html

The `time` function

In this lab, we'll be playing with the `time` function. The `time` function gives the number of seconds that have elapsed since a particular date and time. For example, you can type:

```
import time
print time.time()
```

and you will get the number of seconds elapsed since the starting date.

1. Figure out what date the clock started counting on. To do this, convert the number of seconds that have elapsed into the number of days that have elapsed. Then, go to:

   http://www.timeanddate.com/date/dateadd.html

   and use this online tool to subtract that number of days from the current date.

We won't be using it right now, but there are other functions that allow you to get the current time in a more human friendly way. You can look at the documentation for the `time` module for more information.

Timing user feedback

Elapsed time from an arbitrary time/data may seem like a weird measurement, however, one useful thing we can do easily is figure out how long a particular activity takes. We record how many seconds have elapsed in a variable using `time()`, then, at some later point after some time has elapsed, record the time again using `time()` in another variable. If we take the difference between these two time readings, we get the time elapsed.

2. Write a set of statements that uses `raw_input` to ask the user for their name, times how long it takes them to enter their name in seconds, then prints out how many seconds elapsed. You'll have to record the time before and after the user enters the text.

Infinite loops

`for` loops run for a fixed number of iterations and we don't have to worry about them not finishing. With a `while` loop though, if the bool expression we're looping on never becomes `False` then the loop will never end. For example, the following is an infinite loop and will NEVER end:

```
while True:
    print "hello"
```

In WingIDE, create a new file and enter this text, then run the program. What happens? What should happen? (rhetorical questions)

Unfortunately, one of the misleading behaviors of Wing (not Python) is that if it enters an infinite loop, the program appears to have frozen. If this happens, the easiest way to "stop" it is to go under "Options" in the Python shell and select "Restart Shell".

It would be much more useful for debugging to actually print out the infinite "hello"s so that you can figure out where the problem is. If instead of running it with the green arrow you use the debug button:

You'll see the output continuously displayed in the Debug I/O window in the bottom left. To stop the debugging run, click on the "stop" button next to the debug button.

If you're program ever hangs during this lab, it's likely an infinite loop. Use this debugging technique to help you figure out where it's coming from.