CS3010-1 Python, Fall 2014: Problem Set 1

Daniel Bauer

Total points: 20
Due date: Sep 16, 11:59pm EST

Submission instructions:
Place the files for all problems in a directory named `[your_uni]_week[`X`]`, where X is the number of the problem set. For instance if your uni is xy1234 and you are submitting the problem set for the first week, the directory should be called `xy1234_week1`. Either zip or tar and gzip the directory (using `tar -c xy1234_week1 | gzip > xy1234_week1.tgz`) and upload it to your directory in the drop box for this class on Courseworks.

For parts 2 to 4, download the answer template (`homework1.py`) and fill in your solutions.

It is often easier to experiment with the shorter problems in the interpreter's interactive mode, before filling the final solution into the answer template. Please pay attention to the general guidelines/homework policy on the course website.

Part 0 (0 points)

1. Make sure you have a version of Python 3 installed. Open a terminal/command window and try to run the Python interpreter:

   ```
   $ python
   Python 3.4 (default, May 21 2014, 01:39:38)
   [GCC 4.2.1 Compatible Apple LLVM 5.1 (clang-503.0.40)] on darwin
   Type "help", "copyright", "credits" or "license" for more information.
   >>>
   ```

   On some systems you need to start Python with the command `python3` or `python3.4` to run the correct version.

   If Python is not installed or you have the wrong version, you will need to get a new version. For Windows: download the installer for version $3.4.1$ and run it (packages are available at `https://www.python.org/downloads/`). For Mac: either download and run the Mac OS X installer, or use a package manager such as mac ports ($>$ `$ sudo port install python34`). For Linux, it's best to use your distribution's package manager (Ubuntu/Debian: $>$ `$ sudo apt-get install python3`).

2. Familiarize yourself with the Style Guide for Python Code (PEP 8) at `http://www.python.org/dev/peps/pep-0008/`.

   PEPs (Python Enhacement Proposals) are design documents written by the Python developers. They document proposed and implemented features, as well as general design decisions and guidelines.

Part 1 (5 points) - Control Flow

Implement a simple guessing game. The program chooses a secret number x between 1 and 10. The player can guess numbers repeatedly. After each guess, the program prints a hint:

- if the difference between x and the guess is greater than 5, the program prints 'not even close'.
- if the difference between x and the guess is between 3 and 5, the program prints 'close'.
- if the difference between x and the guess is less than 3, the program prints 'almost there'.

If the player guesses x correctly, the program prints a message and terminates.

The program keeps track of the number of guesses. If the user cannot guess x in his fifth guess, the program informs him that the game is lost and terminates.

Hints:

- Your program should be in a single file 'guessing.py'. It should run from the command line with 'python guessing.py'.
- At the beginning of your source file include.

```
import random
```

  To guess a random number use

```
# chose a random number between 1 and 10
secret_number = random.choice(range(1,11))
```

- To read in a new guess use

```
guess = int(input('Guess a number between 1 and 10:'))
```

- Use a loop to ask for guesses and print hints repeatedly.

Part 2 (5 points)

For each line in the following interactive Python interpreter session, write a short justification for the output, clearly indicating which objects are in memory and which variables refer to them [1].

```
>>> y,x = [1], [1]
>>> x is y
False
>>> x = y
x is y
True
>>> y = y + x
```

[1] Ignore garbage collection for this exercise

```
>>> x.append(y)
>>> x == y
False
```

Add your explanations as comments to the `homework1.py` template.

Part 3 (5 points) - Lists and for-Loops

Use two nested `for` loops to compute the value of expressions of the form

$$(a_1 + ... + a_m) \cdot (b_1 + ... + b_m).$$

Assume that a and b are arbitrary sequences of numbers.

For instance, for `a = [1, 2, 4]` and `b = [2, 3]` the result would be 35.

Part 4 (5 points) - Loops, Strings, and Indices

Given a string `s` create a second string `s2` that is a copy of `s` but without characters that have an identical element next to them.

For instance the, string

```
s = 'abbcaabcaa'
```

should produce the answer

```
'acbc'.
```

Pay attention to the first and last element.

Hint: Instead of iterating over elements of `s`, you should iterate over indices of `s` (use `range`). This should work for arbitrary sequences of arbitary length. You can refer to a specific character in a string using the index operation `s[index]`. The index of the first character is 0. In the above example `s[0]` equals `'a'` and `s[3]` equals `'c'`.