CS 1110, LAB 1: EXPRESSIONS AND ASSIGNMENTS

http://www.cs.cornell.edu/courses/cs1110/2017sp/labs/lab01.pdf

First Name: _____ **Last Name**: _____ **NetID**: _____

Learning goals: (1) get hands-on experience using Python in interactive mode via the command shell; (2) get hands-on experience with Python types, expressions, and variables.

Learning a computer language is a lot like learning a new human language. This lab essentially works as a *grammar drill* before we get started with programming itself.

Getting Credit for the Lab

Labs are graded on effort, not correctness.

When finished, show your written answers to a lab instructor. They will ask you some questions to ensure your understanding and record your lab completion. You can keep the physical handout.

If you don't finish during the lab period, you can get your lab work checked off by showing it to a staff member either (a) in consulting hours[1] up to the the day **before** your next scheduled lab section, or (b) **in the first 10 minutes of (your) lab next week (Tu Feb 7 or Wed Feb 8)**. Beyond that, the lab will be late and the staff have been instructed not to give you credit.

1. Using a Lab Computer vs. Your Own Laptop

The labs provide computers with the "right" version of Python pre-installed.

But you are also welcome to bring your own laptop to work on in lab. To get it properly set up, follow the instructions on the course website to get the CS1110 2017sp version of Python:

http://www.cs.cornell.edu/courses/cs1110/2017sp/materials/python.php

Ask a staff member for help if you have problems with installation; giving help is why we're here!

2. Getting Started with Python

Next, you need to start up the command shell on the computer you are working on. This would be the Command Prompt on Windows, or the Terminal on OS X (Macs). If you are unsure of what these mean, look at the tutorial on the course website and read *just* the first one or two paragraphs for your operating system:

http://www.cs.cornell.edu/courses/cs1110/2017sp/materials/command.php

You do not need to read the sections on navigating directories; we will exercise that in a later lab.

Course authors: E. Andersen, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White

[1] For schedule, see the "CS 1110 Consultant Calendar" on the Staff webpage, http://www.cs.cornell.edu/courses/cs1110/2017sp/about/staff.php .

After you have started the command shell, type `python` at the prompt. You should see "banner" output that looks something like this, with the word "Anaconda" somewhere (the following output was generated on a Mac):

```
Python 2.7.12 |Anaconda 4.1.1 (x86_64)| (default, Jul  2 2016, 17:43:17)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2336.11.00)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>>
```

To make Python evaluate an expression, such as `1 + 1`, type the expression in after the `>>>`. When you hit the "return" key, Python displays the value produced by evaluating the expression, and then shows another `>>>` prompt.

3. Expressions

The following pages contain lists of expressions. For each expression, first *compute the expression in your head, without Python.* Write down what you think the value is in the second column of the table. If you have no idea, write "?".

Next, use Python to compute the same expression. You may find it easier to copy-and-paste from the online version of these instructions. Write down Python's result in the third column. **Proceed row-wise (fill in the second and third column of a row before moving on to the next row)**. You want to learn from earlier examples before moving on to the next one.

If the two values are different, try to figure out why Python gave the answer that it did. Come up with a reasonable explanation and put it in the final column.

Do not waste too much time trying to figure things out yourself. If you do not understand something, ask a staff member immediately.

3.1. **int and float Expressions.**

Useful Shortcuts: If you press the *up-arrow key*, you obtain the previous expression that you typed in. Pressing up-arrow repeatedly scrolls through all the expressions that you have typed earlier in the session; if you go too far back, you can press the *down-arrow key* to get to a later expression. The *left and right-arrow keys* move the text cursor on a single line. Using all of these keys together, you can take a previously-used expression, modify it, and try it again.

Expression	**Expected Value**	**Calculated Value**	**If any difference, why?**
4 + 2 * 5			
(You did the entire row above before proceeding, right? :)			
(4 + 2) * 5			
-4 - -4 - -4			
2 ** 3			
2 ** 3 ** 0			
(2 ** 3) ** 0			
6 / 2			
6 / 4			
6.0 / 4			
6 / 4.0			
6 % 2			
7 % 2			

3.2. **Types and Casting.**

Expression	**Expected Value**	**Calculated Value**	**If different, why?**
float(4)			
int(4)			
int(5.3)			
float(int(5.3))			
int(5.7)			
float(7) / 4			
7 / float(4)			
float(7 / 4)			
type(4)			
type(7 / 4.0)			

3.3. **Comparisons and Boolean Expressions.**

Expression	Expected Value	Calculated Value	If different, why?
$3 < 5$			
$3 < 5$ and $5 < 3$			
`True`			
`true`			
`True and False`			
`True and True`			
`True or False`			
`False or False`			
`not True`			
`not not False`			
`not False and True`			
`not (False or True)`			
`True and False and True`			
`True or (False and True)`			
$(5 / 0 == 1)$ `and False`			
`False and` $(5 / 0 == 1)$			

Why does the last expression in the table above "work" but the one above it doesn't?

3.4. **String Expressions.**

Pay close attention to spaces and to the different types of quotation marks being used; we use both ' (single quote) and " (double quote). Also, note the difference between " (a double quote) and ''(two consecutive single quotes); we've added color-coding to the paired single quotes to help you see the difference.

Expression	**Expected Value**	**Calculated Value**	**If different, why?**
'now ' + 'here now' + 'here'			
"now " + "here now" + "here"			
'A double quote: "'			
'A double quote: "			
''+ 'ok'			
''+ '4 / 2'			
''+ 4 / 2			
'' + `str`(4 / 2)			

3.5. **Variables and Assignment Statements.**

The last part of this lab involves assignment statements. You need to know the difference between expressions, which you've been working with so far, and assignment statements. An assignment statement like

```
b = 3 < 5
```

is a command to do something. In particular, this command

(1) evaluates the expression on the right-hand side of the = (in this case, $3 < 5$), and

(2) stores its value in the variable on the left-hand side of the =, in this case, b.

Because it is not an expression, Python will not actually output a result when you type it in; it will just perform the command silently.

In the table below, the first column contains *either* an expression or a command. If it is an expression, write the value. If it is a command, you should just write "None" (we have done the first one for you). Because some of the entries are commands, it is important that you *enter the expressions or commands in exactly the order they are given.*

Statement or Expression	**Expected Value**	**Calculated Value**	**If different, why?**
i = 2	None		
i			
j			
j = 1			
j			
j = j + i			
j			
i			
w = 'Hello'			
i + w			