CS 1110, LAB 1: EXPRESSIONS AND ASSIGNMENTS

http://www.cs.cornell.edu/courses/cs1110/2018sp/labs/lab01/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.

We suggest you open this lab handout on a Web browser so that you can click on website URLs, copy-and-paste text, etc. You can copy the URL given in the title above, or go to the course website's Labs page.

Getting Credit for the Lab

Labs are graded on effort, not correctness. We just want to see that you tried all the exercises, and to clarify any misunderstandings or questions you have.

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 today's lab session, here are some alternate times/places for checking off this lab:

- at ACCEL Green room consulting hours, listed at `http://www.cs.cornell.edu/courses/cs1110/2018sp/about/staff.php` , from today to Tue Feb 6 inclusive;
- at non-professorial TA office hours from now up to Wed Feb 7 3:45pm, although at TA office hours, questions about course material or assignments take precedence over lab check-offs;
- during the first 10 minutes of your next scheduled lab (Tue Feb 6 or Wed Feb 7).

Beyond that time, the staff have been instructed not to give you credit.

1. Getting Registered on the Lab Recording Machine

Sign in with your netID and password at `https://cs1110.cs.cornell.edu/labs` .

2. Getting Started with Python

Next, 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, read *just* the first one or two paragraphs for your operating system here:

`http://www.cs.cornell.edu/courses/cs1110/2018sp/materials/command.php`

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

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

After you have started the command shell, type `python` (and then hit the return key) 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 3.6.1 |Anaconda 4.4.0 (x86_64)| (default, May 11 2017, 13:04:09)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

3. Expressions

The following pages contain lists of expressions. **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. **For each expression**,

(1) *Evaluate the expression to get its value 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 "?".
(2) Next, use Python to evaluate the same expression, by entering it in at the Python prompt, `>>>`, and then hitting the "return" key. (Alternately, copy-and-paste it from the online version of these instructions.) Write down Python's result in the third column.

If the values in column two and three differ, spend a few seconds trying to figure out why Python gave the answer that it did, and put the reason in the final column. Don't waste too much time trying to figure things out yourself. If you do not understand something, ask a staff member immediately.

3.1. Types and Castin.

Useful Shortcuts: The *up-arrow key* gives you the previous expression that you typed in, and you can hit it repeatedly. 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 different, why?
type(4)			
(You did the entire row above before proceeding, right? :)			
type(4.0)			
float(4)			
int(4.3)			
int(5.7)			

3.2. **Expressions with ints and floats.**

Expression	**Expected Value**	**Calculated Value**	**If any difference, why?**
4 + 2 * 5			
(4 + 2) * 5			
-4 - -4 - -4			
2 ** 3			
2 ** 3 ** 0			
(2 ** 3) ** 0			
3.0/2.0			
3 // 2			.
3 / 2			(Note that float division treats its arguments as floats.)
float(3 // 2)			
3.0 // 2.0			(Note: This is, admittedly, a weird case, but for completeness' sake we include it to point out that int division on floats converts its arguments to ints, but then converts its int answer to a float)
6 % 2			
7 % 2			

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). Copy-paste from the online handout might be your best option here.

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

3.5. **Variables and Assignment Statements.**

The last part of this lab involves assignment statements. An assignment statement like

```
b = 3 < 5
```

is a command to Python to:

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

(2) store the expression's value in the variable on the left-hand side of the =, in this case, b.

Because the assignment statement is not an expression, Python will not 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			