CS 1110, LAB 1: PYTHON EXPRESSIONS; ASSIGNMENT TO VARIABLES

http://www.cs.cornell.edu/courses/cs1110/2013sp/labs/lab01/lab01.pdf

D. GRIES, L. LEE, S. MARSCHNER, AND W. WHITE

First Name: _____ **Last Name**: _____ **Net-ID**: _____

This lab serves to (i) get you started immediately with one way to interact with Python, namely, via the command shell and prompt; and (ii) give you hands-on experience with and develop "muscle memory" for writing Python expressions and assignment statements. Essentially, we're starting with a "grammar drill", just like when you begin learning a new human language.

1. Getting Started with Python

If your primary computer is a laptop, bring it to the lab to work on, as lab is an *excellent* opportunity to get started with Python on your machine. Installation instructions: http://www.cs.cornell.edu/courses/cs1110/2013sp/materials/python.php. Ask a consultant for help if you run into problems; that is why they are there.

Read *just* the one or two relevant paragraphs in this tutorial on how to start up the appropriate command shell for the computer you're working on, viz., Command Prompt on Windows, Terminal on Mac: http://www.cs.cornell.edu/courses/cs1110/2013sp/materials/command.php. Start the command shell up, and type `python` at the prompt. On an ACCEL-lab computer, you should see output that looks something like this:

```
ActivePython 2.7.2.5 (ActiveState Software Inc.) based on
Python 2.7.2 (default, Jun 24 2011, 12:22:14)
[MSC v. 1500 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

To make Python evaluate an expression, type it in after the `>>>`; when you hit "return", Python will give you the result of the evaluation, and then provide you with another `>>>` prompt.

Here are some useful shortcuts for working at the prompt. 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.

2. Lab Instructions

Below is a list 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 "?". Then, use Python to compute the same expression. (It may be easiest to cut-and-paste from the online version of these instructions; a clickable link is just under the title of this document.) Write down Python's result in the third column.

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. You are not really being graded

on complete correctness in these labs (see paragraphs below) so make your best guess at what is happening; your answer will help the staff understand how to better aid you.

On this lab, don't waste too much time trying to figure things out yourself! If you don't understand something, ask a staff member immediately. It is very important that you understand *how* each expression is evaluated, so if an answer doesn't make sense, ask someone. The lab instructors and consultants are in the lab to help. They will look over your shoulder from time to time and give you advice.

Getting credit for lab completion. Labs are graded on effort, not correctness. When you are finished, show your written answers to this lab to your lab instructor, who will ask you a few questions and, if satisfied that you've made a good-faith effort to understand what's going on, record that you completed the lab. The physical piece of paper is yours to keep. If you do not finish during the lab, finish it within the next few days and show it to *your* lab instructor next week.

3. Lab Problems

3.1. int and float Expressions.

Expression	Expected Value	Calculated Value	Reason for Calculated Value
2 * 3			
2 ** 3			
5 + 2 * 5			
(5 + 2) * 5			
-4 - -4 - -4			
2 ** 2 ** 0			
(2 ** 2) ** 0			
6 / 2			
6 / 4			
6.0 / 4			
6 / 4.0			

Expression	Expected Value	Calculated Value	Reason for Calculated Value
5.0 + 2.0			
9.0 * 0.5			
9.0 ** 0.5			
(5 + 2.1) * 5			
4.0 - 3 - 3			
6 % 2			
7 % 2			
6.2 % 4			

3.2. **Types and Casting.**

Expression	Expected Value	Calculated Value	Reason for Calculated Value
float(4)			
int(4)			
int(5.3)			
float(int(5.3))			
int(-5.3)			
float(7) / 4			
7 / float(4)			
float(7 / 4)			
type(4)			
type(7 / 4.0)			

3.3. **boolean Expressions, Mostly.**

Expression	**Expected Value**	**Calculated Value**	**Reason for Calculated Value**
$3 < 5$			
$3 < 5$ and $5 < 3$			
True			
true			
True and False			
True and True			
False			
True or False			
False or False			
not True			
not False			
not not False			
not False and True			
not (False or True)			
True and False and True			
True or (False and True)			
False and (5 / 0 == 1)			
(5 / 0 == 1) and False			

Why does the last expression in the table above not "work" but the one above it does?

3.4. **String Expressions.**
Pay close attention to spaces and to the three types of quotation marks being used: ' (single quote), " (double quote), and ` (backquote: next to the "1" key on the keyboard). It might be best to cut and paste here.

Expression	Expected Value	Calculated Value	Reason for Calculated Value
"Truth " + "is " + "best"			
"Truth" + "is" + "best"			
"Truth " + ("is " + "best")			
"Truth " + 'is ' + "best"			
' "Truth" '			This is a string containing double quotes.
" 'Truth' "			This is a string containing single quotes.
" "Truth" "			
4 / 2 + "a"			
'4 / 2' + "a"			
`4 / 2` + "a" (update from printout)			Backquotes evaluate an expression and convert result to string
'Truth'			
"Truth"			
`Truth`			

One more page to go!

3.5. **Variables and Assignment Statements.**
We now turn to 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; it evaluates the expression on the right-hand side of the = (in this case, $3 < 5$) and 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; write "None" in the table below to indicate this (we've done one for you).

Statement or Expression	**Expected Value**	**Calculated Value**	**Reason for Calculated Value**
i = 2	None		
j = 1			
j			
j = j + i			
j			
i			
w = "Hello"			
i + w			