CS 1110, LAB 1: GETTING STARTED

http://www.cs.cornell.edu/courses/cs1110/2014fa/labs/lab01.pdf

First Name: ____________________ **Last Name**: ____________________ **NetID**: ______________

This lab serves two purposes. First, it is designed to get you started with Python immediately, particularly with the command shell. Second it gives you hands on experience with Python expressions, which we talked on the first day of class. Learning a computer language is a lot like learning a new human language, and this lab essentially works as a *grammar drill.*

Lab Materials. This handout is not the only part of this lab. In addition, we have two files for you to play with. Lab files are always located on the Labs section of the course web page:

http://www.cs.cornell.edu/courses/cs1110/2014fa/labs

For today's lab you will notice two files.

- `hello1.py` (a text-based Python script)
- `hello2.py` (a graphical Python script)

Create a *new* directory on your hard drive and call is `lab01`. **Put this directory on your Desktop**. In future labs, you can put the folder wherever you want, but this lab is a lot easier if this folder is in your Desktop folder. Download all of the files in to that directory.

Getting Credit for the Lab. This lab handout has several empty boxes that prompt you to answer a question. As part of the lab, you are to write the answers to these questions inside the boxes. When you are finished, you should show your written answers to your lab instructor.

Labs are graded on effort, not correctness. The instructor will ask you a few questions to make sure you understand the material and then swipe your ID card to record your success. This physical piece of paper is yours to keep.

If you do not finish, you have **until the beginning of next lab next week** to work on this lab. Over the next week, you may either show your lab to a consultant during consulting hours, or to your lab instructor at the *beginning* of the next lab session.

Course authors: D. Gries, L. Lee, S. Marschner, W. White

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. You should follow the instructions on the course website for installing Python:

http://www.cs.cornell.edu/courses/cs1110/2014fa//materials/python.php

Ask a consultant for help if you have problems with your installation; that is why they are there. However, you do not need to spend lab time installing Python on your machine. If you want get started with Python now and install it later, you are welcome to work on a machine in the lab.

This first part of the lab is to help you understand the tools that we will be working with this semester. Make sure that you have created the folder `lab01` with the two files for this lab: `hello1.py` and `hello2.py`. You should not have any other files in this folder.

1.1. Working with the Command Shell.

Start the command shell for your computer. This is the Command Prompt on Windows (Located in `Accessories` in the Start Menu on Windows 7, and in `Apps` > `Windows System` in Windows 8), or the Terminal on OS X (Located in `Utilities` in the `Applications` folder). This program works like an Window in your operating system, allowing you to manipulate files and folders. However, it uses typed text commands instead of a mouse.

Like a Window in your operating system, the command shell looks at a specific folder. We call this folder the *active directory*. You can use the command shell to list, copy, and move files in the active directory. You can also change the active directory of the command shell, just like you can change the current folder of a Window.

When the command shell starts out, the active directory is your *home directory*. In both Windows and OS X, this is the folder with your name on it. Open up a Window for your home directory, and put it next to the command shell. Go back to the command shell. If you are using Windows, type

```
dir
```

and hit the **Return** key. If you are using OS X, type

```
ls -l
```

(that is a "dash ell") and hit the **Return** key. In a few words, describe what you see and how it compares to the Window next to the command shell.

We want you to change the active directory to the folder `lab01`. If you followed the instructions above, this folder should be on your Desktop. Your Desktop is actually a folder stored inside your home directory. So the first thing to do is to move to the Desktop. To get there from the home directory, type the command (on either OS)

```
cd Desktop
```

and hit **Return**.

Once again enter either `dir` or `ls -l` (depending on your operating system) into the command shell. In a few words, explain what is different this time and why.

Since `lab01` is a folder inside the Desktop, you can now type

```
cd lab01
```

and hit **Return**. Your active directory should now be `lab01`. One last time, either `dir` or `ls -l` (depending on your operating system) into the command shell and tell us what you see.

To go back to the previous directory, you can use the command "`cd ..`" This should be enough to familiarize you with the command shell for now. If you want to learn more, you should read the more detailed introduction on the course web page:

http://www.cs.cornell.edu/courses/cs1110/2014fa/materials/command.php

1.2. **Working with Python.** There are two ways to run Python. One is to execute "scripts", or files that contain Python commands. The other is to use Python *interactively*, typing in just one command at a time.

The files `hello1.py` and `hello2.py` are both scripts. To run a script, you type `python`, followed by the name of the script, into the command shell. Type

```
python hello1.py
```

and hit **Return**. Does anything happen?

Now open up the file `hello1.py` with Komodo Edit. It is a file with just one line in it:

```
#print 'Hello World!'
```

Delete the `#` character and save the file. Once again, run the script `hello1.py` by typing "`python hello1.py`". What you do see this time?

Finally, run the script `hello2.py`. What happens this time?

To run Python interactively, type the word `python` by itself and hit **Return**. Do that now. You should see several lines of text followed by the symbol `>>>`. This is the *Python prompt*. Like the command shell, it responds to commands that you type. The purpose of the `>>>` is to let you know that you are currently running Python, and that you are no longer working with files and folders.

At the Python prompt, type

```
>>> 1+1
```

and hit **Return** (do not type the `>>>`). What happens?

2. Python Expressions

For the remainder of this lab, you will use Python in interactive mode. The following pages have 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. If you have no idea, write "?".

Next, use Python to compute the same expression. You may find it easier 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. **You should always 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, you should 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 graded on being correct so make your best guess at what is happening; your answer will help the staff understand how to better aid you.

This lab is just supposed to be practice; do not waste too much time trying to figure things out yourself. If you do not understand something, ask a staff member immediately.

2.1. Useful Shortcuts.

If you press the *up-arrow key*, you get the previous expression that you typed in. Pressing up-arrow repeatedly scrolls through all the expressions that you have typed this 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.

2.2. **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			
9.0 * 0.5			
9.0 ** 0.5			
6 % 2			
7 % 2			
6.2 % 4			

2.3. **Comparisons and bool Expressions.**

Expression	**Expected Value**	**Calculated Value**	**Reason for Calculated Value**
3 < 5			
3 < 5 and 5 < 3			
`True`			
`true`			

Expression	**Expected Value**	**Calculated Value**	**Reason for Calculated Value**
`True and False`			
`True and True`			
`True or False`			
`not True`			
`not not False`			
`not (False or 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?

2.4. **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)			
int(-5.7)			
float(7) / 4			
float(7 / 4)			