CSCI 3003: Introduction to Computing in Biology

Lab Assignment #1

5 points

Assigned: Thursday, September 5th, 2019

Due: Thursday, September 12th, 2019, 11:55pm

Goals of this lab:

1. Practice using Linux for writing and running Python scripts.
2. See and run your first Python script and explore Python syntax.
3. Use the Moodle Assignment Submit Tool for submitting your lab assignment.
4. Extra: Install python3 on your personal machine.

Learning Linux/Unix

What's Linux?

Linux is the operating system that the CSE labs machines run. It was written and developed as a free operating system modeled after a popular system called UNIX created by Bell labs in the late 1970s. It is popular amongst programmers and computer scientists.

This is weird! Why do we have to use Linux?

Although modeled after operating systems used in the 70s, Linux is a cutting edge programming environment that is constantly being updated. Contrary to popular belief, Linux is installed on more hardware than any other OS in the world, including most supercomputing systems. Its free and open nature makes it an ideal choice for developing software in academic and industrial settings. While it may seem weird at first, Linux was designed around simple design principles which, as you will see, allow you to interact with the computer using many different interfaces that are referred to in the computing world as shells.

Breaking out of your SHELL

Computer scientists get a kick out of clever names. Since modern computer hardware has been widely available there has been a constant effort to abstract away all the ugly, esoteric aspects of the machine and replace it with interfaces that humans can interact with. The computer's hardware is rarely ever dealt with directly. Instead, the operating system was developed and most if not all direct interaction with the computer is done through simple abstractions such as files, directories, URLs, and programs. All of the specifics of these are handled behind the scene by the operating system and silently converted to their bitwise equivalents. In order to interact with the OSs efficiently, special programs called shells were designed to provide arguments to programs written to interact with the operating system. The word shell is used to describe the outermost program meant to hide away all the complicated aspects of the machine; it's the program that has the most interaction with humans! Shells are like any other program under the hood and come in a variety of flavors; in fact you might be surprised that you are interacting with a shell right now!

Clicking and dragging graphical icons and buttons provides an interface between you and the computer that allows you to easily access files, open URLs, and change directories! These types of shells are called graphical user interfaces (GUIs) and are the most familiar way we interact with machines. Another type of interface is called the command line interface (CLI), which is the default for most popular shells out there. Most things you can do with a GUI shell, you can also do with the CLI shell. So why bother learning how to navigate Linux using the command line? Although it might be sore on the eyes, text-based shells provide a much more efficient, reproducible work environment than graphical interfaces. Have you ever had to explain to someone how to access their email or edit an Excel spreadsheet over the phone? The shell provides a step-by-step protocol (strangely similar to a protocol for isolating DNA or running a gel, for example) for running programs in order to perform a task, and by nature, it is extremely specific. So don't be afraid to break out of your proverbial graphical shell interface. Like most things that look complicated, once it has been broken down into sizeable chunks, it becomes apparent that is wasn't that hard all along.

A short Linux Tutorial

There are a few key commands that will be helpful in getting around the shell on the lab machines. Throughout the rest of this document, text following a "%" and formatted in `fixed width font` is meant to be executed from the shell. Filenames will be *italicized* and command names will be enclosed in `back ticks`. Below is a short summary of the key Linux commands or you can go to a full tutorial by following this link:

http://www.ee.surrey.ac.uk/Teaching/Unix/unix1.html

Using the Linux commands you've learned in class, perform the following tasks:

- Open a shell. Click the "Terminal" icon in the toolbar on the left of the desktop.
- Create a directory in your Documents directory called CSCI3003 using the `mkdir` command:

% mkdir Documents/CSCI3003

- Change to the newly created directory using the `cd` command:

% cd Documents/CSCI3003

- Create a subdirectory in the *CSCI3003* directory called *Lab1* and `cd` into it:

% mkdir Lab1

- Check that the newly created directory has been made using the `list` command:

% ls

% cd Lab1

- Download and move *Lab1.py* into your lab directory. You need to substitute which folder you downloaded the python file into.

% mv <sourceFile> <destinationFile>

- You can remove files using the `rm` command. Remember, when you remove a file from the shell, it is **GONE FOREVER**! Always back up your code with the copy (`cp`) command:

% cp <sourceFile> <destinationFile>

% rm <destinationFile>

Your first Python script:

Using the commands outlined above, you should have set up a working space to run your first script. Make sure you are in the directory containing the script and open it with a text editor:

% gedit Lab1.py &

- Look over the script, notice how the commands are structured and start guessing what the program might do.
- There are two different flavors of python installed in the lab machines: python 2 and python 3 (with multiple different subversions of each of them). The commands used to invoke python 2 and 3 are 'python' and 'python3' respectively. For this class, we are going to use python3. The command 'python3' will run the default python3 version installed in the system. To use a specific version of python 3 from a list of available options, load the module corresponding to the version. An example is shown below:

% module load soft/python/3.7

- Run the script from the terminal using the `python3` command. Follow the instructions on the screen for some brief information about python. When the script has finished running, there should be an output file called '*lab1_output.txt*'.

```
% python3 Lab1.py
```

- Change the script any way you would like and see what the changes do. Explore.

Running your script using Spyder:

The goal of this section is to show you how to program using an integrated development environment (IDE). You will see how an IDE works compared to the standard "text editor + terminal" configuration (you may notice that they are basically the same – one is just fancier!).

Spyder (installation instructions below) is an IDE for python. An IDE is a piece of software that combines a text editor, a terminal, and other useful tools such as autocompletion, a debugger, and a variable explorer in one package. IDEs can be very convenient, and especially so for learning programming as a beginner. In all future python exercises, we would like you to use Spyder to write and run your code.

Spyder is part of the Anaconda python distribution. This is already installed on CSELabs computers. To load this software, first load the anaconda module.

```
% module load soft/python/anaconda
```

Then, open the spyder software (this will take a little while to load)

```
% spyder &
```

Click on the large folder icon in the upper right of the Spyder window and set the working directory to your lab 1 folder. Open *Lab1.py* so that it appears in the text editor.

Since you will be running your script again, it will be important to change the names of files exported by the script so you do not overwrite older files! Search for "lab1_output.txt" in *Lab1.py* and create a different filename for this second run (e.g., "lab1_output_2.txt"), using Spyder.

Use "Run" in the Run menu to run *Lab1.py* from within Spyder!

Extra: Installing python on your personal machine:

One of the major attractive features of python is that it is cross platform. We will be using python3 in this course. If you prefer to complete labs on your personal machine, make sure the code will run in python3.

You can download the exact same version of python and the Spyder IDE (integrated development environment) software that you will use on the lab computers. This will give you the same experience

whether you work on your assignments in the computer lab or on your personal computer. (note: this will require ~2.8GB of hard drive space)

We need to install the Anaconda python distribution (the latest is **2019.07**), which includes python and the Spyder IDE. To install, go to **https://www.anaconda.com/download/** and download the appropriate installer for your operating system; **make sure you install the python3 version** (Click on **'Python 3.7 version'** as for now). Download the installer appropriate for your operating system, and choose the graphical installer where available (Windows and Mac).

Follow the installation instructions, which are quite straightforward. Feel free to contact the TA if you are having issues.

With this python installation, you will also be to run python from the command line or from Spyder. If you want to run python from the command line, make sure that during the installation, you say "yes" if it asks you to make Anaconda your default python distribution.

Submit to Canvas

You should submit two files on Canvas: the *Lab1.py* script and the output file *lab1_output.txt* generated from running this script.

You're finished with Lab 1!