Lab 1, CPE 101

Welcome to CSC/CPE 101. This first lab will help you to familiarize yourself with the programming environment provided in the department labs. This is the environment in which you will be working all quarter long, so take this time to ask questions and to get comfortable with the lab setup.

Environment

The lab machines run a distribution of the Linux operating system. For simplicity, and to gain experience in a, potentially, new environment, we will do our coursework in this environment.

Unix/Linux

Open a terminal window. To do so, from the system menu on the desktop toolbar, select Applications → System Tools → Terminal. The Terminal program will present a window with a command-line prompt. At this prompt you can type Linux commands to list files, move files, create directories, etc. For this lab you will use only a few commands. Additional commands can be found from a link on the course website.
In the terminal, type ls at the prompt and hit <Enter>. This command will list the files in the current directory. (In some environments, e.g., Windows, a directory is commonly referred to as a folder.) If you type pwd, the current directory will be printed (it is often helpful to type pwd while you are navigating directories). If you type tree, then you will see a tree-like listing of the directory structure rooted at the current directory.
Create a new directory for your coursework by typing mkdir cpe101. Use ls again to see that the new directory has been created.
Change into this new directory with cd by typing **cd cpe101**.
Create another directory inside the cpe101. Type: **mkdir lab1**
Type **ls** to see if the directory has been created.
To move back "up" one directory, type **cd ..**

To summarize

- **ls** list files in the current directory cd change to another directory.
- **mkdir** creates a new directory.
- **pwd** print (the path of) the current directory.

Though these basic commands are enough to continue with this lab assignment, you should consider working through a Unix tutorial (many can be found on the web) at a later time.

Executing a Program

Type:

```
echo 'print ("Hello World!")' >> lab1.py
```

You can see the contents of lab1.py by typing:

```
more lab1.py
```

To execute the program, type:

```
python lab1.py
```

To summarize

- **mv** move files
- **unzip** extract contents of a .zip file
- **more** display contents of a file
- **python** python interpreter used to execute a program by specifying name of program file

Editing

There are many options for editing a Python program. On the department machines, you will find vi, emacs/xemacs, nano, gedit, and some others. The editor that one uses is often a matter of taste. You are not required to use a specific editor, but we will offer some advice (and we will try to help with whichever one you choose).

If you decide to connect to the department machines from home, then you'll want an editor that can work via a remote connection without much effort (i.e., without installing and running additional software). This makes gedit a less than desirable candidate (unless you want to learn two editors). The others will work fine, though you'll see some differences between xemacs and emacs (which is what you'd use remotely).

vi and emacs are powerful editors once you have learned how to use them. But they require some initial effort to learn their command sequences. Knowledge of one of these will, however, serve you well for quite some time.

nano (or pico, when nano isn't available) is very simple. It will feel, in some respects, like using Notepad. It is quick to learn and easy to use. Some people will mock others for using nano (some people are elitist dorks). You might choose to start with nano and then switch to one of the other editors as you "outgrow" nano (the others provide features that can aid you when programming).

A note concerning tabs: Whichever editor you choose, you should lookup how to convert tabs to spaces so that your Python source files do not contain a mix of tabs and spaces (or avoid tabs altogether).

In general, whichever editor you choose, you should invest some time early to learn how to navigate quickly within a file. More specifically, you will want to learn how to jump to a specific line, to search within the file, and copy/paste lines.

If you cannot decide, then use vim (I strongly advise against emacs). You always have the option of switching at a later time.

You will start the editor by typing the name at the prompt followed by the name of the file that you wish to edit. For instance, vim pset1.py.
To Do:
Open a new file pset.py ($ marks the start of a prompt.)

```
$ vim pset1.py
```

At the top of the file paste the following block of text:

```
###########################
# CPE 101
# Section: <section number>
# Lab 1, Problem Set 1
# Name: <first> <last>
# Cal Poly id: <id>
###########################
```

Replace <section_number>, <first>, <last> and <id> with yours.
Download the instruction for the problem set I from the poly learn and follow the instruction.

Interactive Interpreter

The Python interpreter can be used in an interactive mode. In this mode, you will be able to type a statement and immediately see the result of its execution. Interactive mode is very useful for experimenting with the language and for testing small pieces of code, but your general development process with be editing and executing a file as discussed previously.
Start the interpreter in interactive mode by typing python at the command prompt. You should now see something like the following.

```
Python 2.7.5 (default, Oct 30 2018, 23:45:53)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-36)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The >>> is the interpreter's prompt. You can type an expression at the prompt to see what it evaluates to. Type each of the following (hit enter after each one) to see the result. When you are finished, you can exit the interpreter by either pressing ctrl and d keys simultaneously or typing “exit()” and hitting <enter> key.

Try typing the following line in python interpreter and see what happens:

```
124 % 10
```

Tutorial

All done? Great. Consider using any remaining time by working through a Unix tutorial. This is a good time to get more familiar with the environment and with your editor.