Putting your house in order

Now that you know a little programming syntax the next steps will be A.) learning how to write useful code, and B.) setting up your computer environment so that you can execute it efficiently, minimizing clutter, confusion, and errors. The first is rather elusive, and will only come from lots of practice, but we can easily give you some help on the latter. I want to emphasize three core concepts:

1.) Have your code in only one place
2.) Write lots of comments and notes to yourself
3.) Use version control as much as possible

These will help you be organized and effective, but it takes some practice and habit building. The first is in some ways the most important, because without it, the other two aren't very helpful, so that's what we'll work on now. Writing comments and docstrings is something you already know how to do, and, just like keeping a good lab notebook, will come with practice. The last, version control, is the subject of our second invited speaker, Cheng Lee.

You couldn't have your code all in one place if it always needs to be in the directory you are currently in to execute. So what you need to do is tell your bash shell where to look. The list of directories where bash looks by default is held in the variable `$PATH`. To see what directories are in your `$PATH`, type at the command line:

```
echo $PATH # echo is the Unix version of print in Python
```

When you type a command and hit enter, Unix looks for programs by this name in all these directories. Python also has a list of directories it looks in for modules when you call `import`. This is, unsurprisingly, held in a variable called `$PYTHONPATH`. At the command line, type:

```
echo $PYTHONPATH

# Or at the python interpreter
>>> import sys
>>> print sys.path
```

The next step is to make yourself a directory (with `mkdir`) that all your future scripts will go. This should be somewhere convenient. Then get the full path to this location by moving into it (with `cd`) and typing `pwd` ('path to working directory').

You will then need to append this full path to your `$PATH`. Let's say your scripts directory is in your home directory. Home is a special directory, like root, and is specified by two special variables: the tilde (~), and `$HOME` – these are interchangeable. You can edit your `$PATH` by editing a *hidden* text file called `.bashrc` in Ubuntu, or `.bash_profile` or `.profile` on Macs. These files should be in `$HOME`, but to see them you need to type `ls -a` (the –a argument reveals hidden files). Some Macs don't have a `.profile`, in which case you just create one – the period in front of the file name will keep it hidden. Add these lines to the bottom of your profile document. Change nothing else!

```
export PATH=$PATH:$HOME/scripts
export PYTHONPATH=$PYTHONPATH:$HOME/scripts
```

If your scripts directory is not in `$HOME`, put the full path to it in place of `$HOME/scripts`. Now close your terminal, open it back up, and you should be able to execute scripts in scripts/ from anywhere on your computer