Week 15 - Lecture Notes

Welcome to Week 15, our second to last week of the course! Last week, it was mentioned that we were going to spend the last few weeks of the course learning some more practical skills. Last week we went over an important programming concept called recursion. This week, we will go over some handy tools in the Python programming environment, including IPython and PythonAnywhere. There will be no assignment this week; however, please make sure to go over this week's material: it will most definitely make your future programming easier.

Python Shell

During this course, you have been using the python interpreter to run your scripts. For example, *python homework5.py* would execute the code in the homework5.py file. However, there is another way of using the python interpreter, and that is in *interactive mode*. This mode allows you to work with Python in a more interactive way. Sometimes, this mode is referred to as the *python shell*.

What exactly do we mean by more interactive? Well, the Python shell provides what is known as a *REPL* environment, where REPL stats for *Read-Eval-Print Loop*. This means that the interpreter will read in a line of python code, evaluate it, print any results, and keep doing this in a loop until you exit the shell. To see this in action, simply type in *python* at the command line, and you should see something like this:

```
$ python
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The `>>>` is what is know as a prompt, which is where you will type in some python expressions:

```
>>> import os
>>> os.name
'posix'
```

You can type in `exit()` when you are done. This mode is very useful for when you want to debug small fragments of code, or you want to experiment with how some Python syntax works. However, there are some downsides which make this basic shell cumbersome to work with (e.g., lack of tab completion, and poor line editing to name a few).

Introducing IPython

To fix some of these shortcomings, the *IPython* shell was invented. This shell provides a much richer and useful interface than the default one provided in the base Python installation. To install IPython, simply run this from the command line:

```
pip install "ipython[notebook]"
```

Now you can start the interactive shell by running the `ipython` command. To learn more about how to best utilize IPython and all its features, please watch this video series, made up of 5 videos detailing how to best utilize the IPython shell.

IPython Notebooks

IPython provides another way of interacting with python, *IPython Notebooks*. The IPython Notebook is "a web-based interactive computational environment where you can combine code execution, text, mathematics, plots and rich media into a single document". In future courses, you may need to use IPython Notebooks, so you might as well become comfortable starting and running a notebook now.

To run the IPython notebook server, run `ipython notebook` at the command line.

You should see some output like this:

```
[NotebookApp] Using existing profile dir: u'/home/<some_directory>'
[NotebookApp] Serving notebooks from'/home/<some_directory>
[NotebookApp] The IPython Notebook is running at: http://127.0.0.1:8888/
[NotebookApp] Use Control-C to stop this server and shut down all kernels.
```

The program will wait there until you hit Ctrl-C to kill the process. At this point, going to your web browser of choice and entering *http://127.0.0.1:8888/* will send you to the front page of the IPython web interface. To learn more about how to use IPython Notebooks, please watch this video presentation from PyOhio 2013 on how IPython Notebooks present a new way to approach programming in Python.

PythonAnywhere

The last tool that we will go over in this class is a cloud platform called PythonAnywhere. This online environment gives you access to not only a Python programming environment, but a bash console as well as a database setup. You can edit code directly from your browser with their online Python editor, or you can develop code locally and use Git to sync your code. PythonAnywhere also supports a lot of web development frameworks, including Flask, out of the box. Since all of this is available from their website, you can code from anywhere you want, as long as you have access to an internet browser. The advantage to such a setup is that “you can develop and host your website or any other code directly from your browser without having to install software or manage your own server”.

Please note, if you are interested in using PythonAnywhere, especially for hosting your class project online, understand that it is **not required** that you buy a plan from them. The free plan should well enough for you. Once again, this class will not require you to use PythonAnywhere or to subscribe to one of their paid plans.

However, if you are interested in using this platform, please watch the following videos on how to use PythonAnywhere:

- Walkthrough detailing how to use PythonAnywhere to write a simple script
- A short video on how to deploy a Flask application on PythonAnywhere