Will Monroe
CS 109

Supplementary Handout #3
July 2, 2017

Getting started with Python

While we will not require you to use a specific programming language in CS 109 this quarter, we highly recommend programming in Python for probability applications. Python has become the de-facto standard language for doing machine learning, and it is easy to learn if you've previously programmed in another imperative/object-oriented language like Java or C++.

Installation

If you haven't installed Python before, we recommend the Anaconda distribution:

```
https://www.continuum.io/downloads
```

Anaconda comes by default with several useful and tricky-to-install libraries: `numpy` and `scipy` offer efficient math operations on vectors and matrices, plus some handy math functions that aren't in the standard library (particularly `scipy`'s functions for dealing with commonly used probability distributions); and `matplotlib` gives you a way of making plots and visualizations. If you've used MATLAB before, you'll find that `numpy` plus `matplotlib` gives you a very close substitute (albeit a little more verbosely).

You can also try to install these libraries yourself on a previously installed version of Python. At a command line:

```
python get-pip.py
pip install numpy scipy matplotlib
```

The first line is only necessary if you do not have `pip` already installed. If you are using Python 3, you may need to replace `python` with `python3` and `pip` with `pip3` in the above commands.

Python 2 and Python 3

Python has two major versions that are both in widespread use. The differences between Python 2 and Python 3 are small but lead to incompatibilities in code between the two versions. For the purposes of this class, it does not matter which version you use.

By far the most common reason one might want to use Python 2 is because one wants to use the occasional old library that doesn't support Python 3; certain things about strings, files, and Unicode are also a bit harder to learn in Python 3. However, Python 3 also has a number of really cool new language features and standard libraries that can make up for these problems.[1]

[1] `http://python-3-for-scientists.readthedocs.io/en/latest/python3_user_features.html`

Running Python programs

Unlike when you might have used Eclipse in 106A or Visual Studio/XCode in 106B, Python doesn't have a single widely-used editor just for it. Many people simply write their Python programs in general-purpose programmers' text editors (Vim, Emacs, Sublime Text, Atom, Notepad++) and run them from the command line.

Suppose you write a Python program and save it as `/home/myusername/cs109/helloworld.py`. You can run it by opening up a terminal, switching to that directory (`cd /home/myusername/cs109`) and running

```
python helloworld.py
```

If you just run `python` instead (not providing a file name), it will give you an **interactive shell**. In the interactive shell, you can type in Python commands line by line and see the results immediately! This can be very handy when you all you need is a one- or two-line program.

You might want to check that `scipy` is installed by typing this line into the interactive shell:

```
import scipy.stats; print(scipy.stats.norm.cdf(2.0))
```

It should display a number around 0.977.

Troubleshooting

- **When I try to run something, I get `python: command not found` or `"python" is not recognized as an internal or external command, operable program or batch file.`** Python probably isn't in your `PATH` variable. With Mac and Linux, you can edit `~/.bash_profile` and add

  ```
  PATH="$PATH:(path to Python installation)/bin"
  ```

  With Windows, Anaconda recommends using the Anaconda Command Line rather than the usual Windows Command Prompt; this should take care of any `PATH` issues. However, if necessary, the `PATH` variable can be found in the Advanced System Settings tool (instructions: `https://www.computerhope.com/issues/ch000549.htm`).
- **When I try to import `scipy`, I get `ImportError: No module named scipy`.** First make sure you are either using Anaconda or have already used `pip` to successfully install `scipy`. If this seems to have worked but you still can't import `scipy`, you might have more than one version of Python, and the library got installed to the other version. You can determine where your Python executable is by running the following Python code:

  ```
  import sys
  print(sys.executable)
  ```

You can then force `pip` to install there. First, find out where `pip` is:

```
which pip
```

Suppose the code above tells you that Python is located at `/home/myusername/.local/bin/python`, and `pip` is at `/usr/bin/pip`. You can install `scipy` to that python using

```
/home/myusername/.local/bin/python /usr/bin/pip install scipy
```

Useful libraries

Python's best feature is its broad-coverage set of standard libraries, which allow you to do many things in one or two lines that take dozens in other languages. In this class, you'll want to make use of two specific libraries having to do with probability: `random`, which is built-in and contains random number generators; and `scipy.stats`, which is a separate package and contains functions that return values of probability distributions (we'll cover those starting in Week 3).

Here's an overview of the most useful functions in each library, plus `math` for good measure:

`math.sqrt(x)`	Computes $\sqrt{x}$
`math.exp(x)`	Computes e^x
`math.factorial(n)`	Computes $n!$
`random.randint(a, b)`	Returns a random integer between a and b, inclusive
`random.random()`	Returns a uniform float in the interval [0, 1)
`random.choice(seq)`	Draws one element of a sequence, equally likely
`random.sample(seq, k)`	Draws k elements without replacement
`random.shuffle(seq)`	Shuffles a sequence, in-place
`random.gauss(mean, std)`	Draws from a normal distribution note: standard deviation, not variance!
`random.expovariate(lambd)`	Draws from an exponential distribution
`random.betavariate(a, b)`	Draws from a beta distribution
`scipy.special.binom(n, m)`	Computes $\binom{n}{m}$ (as a `float`)
`scipy.stats.binom(n, p)`	Binomial distribution
`scipy.stats.norm(n, p).pmf(x)`	Probability mass function (PMF) of binomial distribution
`scipy.stats.binom(n, p).cdf(x)`	Cumulative distribution function (CDF) of binomial distribution
`scipy.stats.poisson(lambd)`	Poisson distribution
`scipy.stats.geom(p)`	Geometric distribution

`scipy.stats.norm(mean, std)`	Normal distribution note: standard deviation, not variance!
`scipy.stats.norm(mean, std).pdf(x)`	Probability density function (PDF) of normal distribution
`scipy.stats.norm(mean, std).cdf(x)`	Cumulative distribution function (CDF) of normal distribution
`scipy.stats.expon(0, scale=1 / lambd)`	Exponential distribution
`scipy.stats.beta(a, b)`	Beta distribution