Objective. To learn how to scrape data from the web.

Before we launch into our projects, many of which involve web-based data, I thought we would spend a little time learning how to automatically grab data from a web page. To that end, we'll be investigating the time-series data on the popularity of baby names in the Census and in Google's digitally scanned library.¹ Your tasks this week include developing a class to hold name distributions, a function to scrape data from a web page, and to develop an interesting name-based plot, using pyplot.

Setting up the environment.
Experiments this week will require the use of Python's matplotlib and requests modules in a virtual environment. First, let's activate the virtual environment and verify those modules have been installed:

```
$ cd ~/cs135
$ source bin/activate
(cs135)$ pip install matplotlib
Requirement already satisfied...
...several lines elided...
(cs135)$ pip install requests
Collecting requests
  Downloading requests-2.9.1-py2.py3-none-any.whl (501kB)
    100% |################################| 501kB 390kB/s
Installing collected packages: requests
Successfully installed requests-2.9.1
```

Recall, also, we had some problems with matplotlib until we edited the matplotlib run commands file. Verify its content:

```
(cs135)$ cat ~/.matplotlib/matplotlibrc
backend: Agg
```

Now, clone the starter kit for this lab (assuming you're 18xyz):

```
(cs135)$ git clone ssh://18xyz@gala.cs.williams.edu/~cs135/18xyz/lab7.git lab7
```

Now, we're ready to play.

Gathering baby name data.
First, we'll gather the database for baby names from the Social Security Administration. Go to

https://www.ssa.gov/oact/babynames/limits.html

and download the file linked by “National Data”, into your lab7/names directory (likely this is accomplished in your browser with control-click on the link, selecting Save link as…). You may be familiar with “zip” files: they're bundles of compressed files. From the Terminal window you can create

¹Special thanks to Brent Heeringa for this lab.
The files are uncompressed into a subdirectory, names. Do not add any files to this subdirectory (in particular, don't put Python source there), and do not add this directory to your repository.

This database includes the counts of first names of applicants for Social Security cards, sorted by year and sex. Names of babies born in 1990, for example, will be counted in yob1990.txt. This is a CSV file where each line is of the form:

```
Name,Sex,Count
```

where Sex is M (male) or F (female), and count is the number of occurrences. The file is sorted first by sex, then by decreasing frequency. Any name that occurs fewer than 5 times may surrender personal data and is, thus, not tallied.

Notice that some names are popular for both boys and girls (e.g. Robbie), so there may be multiple entries in any particular year.

In the lab7 directory, you'll see the start of a module, baby.py, that includes the beginnings of a name database class, nameDB. Please read the starter file carefully, and fill in the methods marked pass. To ensure things are working correctly, you might consider adding document-based testing code and uncommenting the testing code at the bottom of the file.

Using Google's NGram service.

A side-effect of Google digitally scanning entire libraries is the development of databases of the adjacencies of n specific words. These are called n-grams (or bi-grams, for n = 2, tri-grams, etc.). You can make queries against these databases with Google's NGram server:

```
https://books.google.com/ngrams
```

At this entry page, you'll notice you can specify one or more strings, and you can identify a range of years, a corpus, case sensitivity, and a level of data smoothing. Run some experiments. For example, you'll notice that Amherst appears in texts more frequently than Williamstown.

You can generate your own queries by hand. Suppose we're interested in comparing (in a case sensitive manner) Amherst College and Williams College. Using the English corpus (corpus 15), and considering the range 1793-2013, we have:

```
https://books.google.com/ngrams/graph?
  content=Amherst+College%2CWilliams+College&
  year_start=1793&
  year_end=2013&
  corpus=15&
  smoothing=3
```

(I've printed the long URL over several lines to highlight the structure; in reality, the URL is one long string.) The question mark (?) indicates that we're making a query. All spaces in the search-for strings
are converted to + marks, and the content strings are separated by a comma (%2C). All other items are specified as key=value pairs, separated by ampersands (&).

We'll use the requests module to perform these queries from within Python. It's a versatile package, well documented at

    http://docs.python-requests.org

In its most basic use, we can perform queries with code similar to:

```python
url = "https://books.google...."
req = requests.get(url)
t = req.text
```

That's it!

The result is a request response, which is the entire content of the page that was fetched. (You can, from within a typical browser, look at this content with View Source.) Much of this text we can ignore, but we can glean the plotted data from a that part of the response that has the form:

```javascript
var data = [...elided text...];
```

The data between (and including) the brackets can be evaluated as a list of dictionaries that describe the time-series data. Experiment!

Find the ngram.py file. This contains two procedures, request and parse. The request procedure takes a list of ngrams and a start and end year. You should form an appropriate URL for the request to Google and return the text of the response.

The parse method can be used to parse the text of the response, returning a dictionary that maps Ngram strings to a list containing a percentage for each year. These percentages are comparable and can be used to plot relative frequencies of ngrams.

### Plotting the data.

Define a function, plot(filename, db, ngdb, names, years) in baby.py. This function plots one or two plots to filename that document the relative frequencies of names from (1) the Social Security database as described by a baby.nameDB, db, and/or (2) the Ngram database returned from Google over the same years.

I encourage you to continue to play with the pyplot package. Again, it is described at:

    http://matplotlib.org/users/pyplot_tutorial.html

To plot more than one plot in the image, look at descriptions of subplots (the tutorial covers this in Working with multiple figures and axes).

Document your plot in the README file in the lab7 directory.

### Turning in.

When you're finished, make sure you push changes for README, baby.py, ngram.py, a plot of your own design.

*