Objective. To build course-description derived clouds of words, scaled to reflect relative word frequency.

In our last lab of the semester, we'll write code to generate word clouds from department course descriptions. We'll be building up skills for regular expression matching.

Setting up the environment.
Experiments this week will require the use of Python's cython and wordcloud modules. Activate the virtual environment and install those modules:

```
$ cd ~/cs135
$ source bin/activate
(cs135)$ pip install cython
(cs135)$ pip install wordcloud
```

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

```
(cs135)$ git clone ssh://18xyz@gala.cs.williams.edu/~cs135/18xyz/lab8.git lab8
```

Let's play.

Gathering course descriptions.
We'll be making heavy use of this year's Course Catalog web site:

```
http://catalog.williams.edu/1617/catalog.php
```

If we select the listing of courses from the CSCI department for this spring (term 1173) we go to:

```
http://catalog.williams.edu/1617/catalog.php?strm=1173&subj=CSCI
```

(Please go to this page.) You'll notice that all the courses the CS department currently teaches are listed on this page. From this page we can go to a particular page. Here is CS135:

```
http://catalog.williams.edu/1617/catalog.php?strm=1173&crsid=020081
```

On this page we see lots of information about the course, including, at the top, the course description. View the source for this page by selecting View Source off the View/Developer menu.

In the file catacloud.py, we'll modify and write code that will help us glean the course descriptions for all a department's courses. First, you'll find code I've written for building a dictionary of departmental course offerings.

```
offerings = subjDict(subj='CSCI',term=1173)
```

Entries of this dictionary are keyed by course numbers, returning a subdictionary with the following keys:

'subj' A string the identifies the department (e.g. 'CSCI').

'number' The int course number associated with this course (e.g. 134).
'title' The title of the course (a string).
'cid' The PeopleSoft ID associated with this course (an int).
'term' The PeopleSoft term ID associated with this offering (an int).

Read through and make sure you understand how subjDict works.

1. Your task is to complete the routine courseDescr, which will return the course description associated with a particular course. You might use it in the following way:

   print(courseDescr(subj='CSCI',number='135',term=1173))

   If a course does not have a description, return an empty string. You will notice that this method has the ability to reuse a subject dictionary if one has been computed before. This allows you to march through the dictionary and visit all the course pages without reinterpreting the subject listings each time.

   Note that if you query the course catalog too quickly, it will reject your request. You can detect this by looking for the string Error:ORA on a page you request. If this happens, wait a second (use time.sleep(1)) and try again.

2. You may find it useful to write another method that collects all the course descriptions for an entire department. Be aware that some courses do not have descriptions. Quietly ignore this.

Building a Cloud.

We’ll use the wordcloud package, by Andreas Mueller. You can find this package described on this page:

http://amueller.github.io/word_cloud/

We’ll use the WordCloud class from this package to generate a word cloud from descriptions of some collection of courses.

In catacloud.py you’ll see how to build a word cloud from the College’s Mission Statement in the routine missionCloud. Calling this routine generates a png image of words that occur in the statement, each sized based on word frequency. Running catacloud.py as a script will drop this cloud in mission.png. Try it. Notice that there is an element of randomness: every time you run the script a different word cloud is generated.

1. Write a method, cloudify, that takes, perhaps, a subject and a semester and generates a word cloud from the courses offered in that subject during that semester. You can design the wordcloud however you wish (see the page:

   http://amueller.github.io/word_cloud/references.html

   for the wordcloud API), but I’d like you to save the result as a lossless png file.

2. You may find it helpful to interpret the arguments to your script to allow the user to specify an alternative subject or term.


Turning in.
When you’re finished, push up changes to README, catacloud.py, and your png file.

*