## Problem 0: Query Twitter with Python

Use the urllib and json libraries in python to access the basic twitter search API and return JSON data. The url is:

http://search.twitter.com/search.json?q=microsoft

The format of the result is *JSON*, which stands for JavaScript Object Notation. It is a simple format for representing nested structures of data --- lists of lists of dictionaries of lists of .... you get the idea.

As you might imagine, it is fairly straightforward to convert JSON data into a Python data structure. Indeed, there is a convenient library to do so, called json, which we will use. Twitter provides only **partial documentation for understanding this data format.** But it's not difficult to deduce the structure.

The response is a dictionary representing the entire resultset. The "results" key corresponds holds the actual tweets; each tweet is another dictionary. Write a program to print out the text of every tweet.

Generalize your program to fetch 10 pages of results. Note that you can return a different page of results by passing an additional argument in the url:

http://search.twitter.com/search.json?q=microsoft&page=2

What to turn in: Report the number of tweets returned by this query.

## Problem 1: Write a script to calculate sentiment for a term

Download and unzip the file **python_assignment.tar.gz**.

**Part A**

To access the live stream, you will need to install the **oauth2 library** so you can properly authenticate.

The steps below will help you set up your twitter account to be able to access the live 1% stream.

- Create a twitter account if you do not already have one.
- Go to **https://dev.twitter.com/apps** and log in with your twitter credentials.
- Click "create an application"

- Fill out the form and agree to the terms. Put in a dummy website if you don't have one you want to use.
- On the next page, scroll down and click "Create my access token"
- Copy your "Consumer key" and your "Consumer secret" into twitterstream.py
- Click "Create my access token." You can **Read more about Oauth authorization.**
- Open twitterstream.py and set the variables corresponding to the consumer key, consumer secret, access token, and access secret.
- Run the following and make sure you see data flowing.
- $ python twitterstream.py

You can pipe the output to a file, wait a few minutes, then kill the program to generate a sample.

**Part B**

In this part you will be creating a script that computes the sentiment for a term in a file of tweets.

You are provided with a skeleton file, sent.py, which can be executed using the following command with the proper arguments:

```
$ python sent.py <term> <sentiment_file> <tweet_file>
```

The file AFINN-111.txt contains a list of sentiment scores. Each line in the file contains a word or phrase followed by a score. See the file AFINN-README.txt for more information.

Fill out sent.py by writing a function that takes a term and returns a sentiment score using the sentiment_file and tweet_file.

This paper may be useful in developing a sentiment metric:

> O'Connor, B., Balasubramanyan, R., Routedge, B., & Smith, N. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. (ICWSM), May 2010.

Python hint that you may or may not need in your solution: If you want to concatenate a whole list of strings by a delimiter, you can use the join method of a string. Note that you call join as a method of the delimiter you wish to use, not as a method of the list you wish to concatenate.

```
>>> ",".join(["separated", "by", "commas"]
"separated,by,commas"
>>> "".join(["a","b","c"])
"abc"
>>> " ".join(["a","b","c"])
"a b c"
```

What to turn in: sent.py

## Problem 2: Word Frequencies

Write a script, count_english.py, with a function, twitter_word_frequency(), that computes the word frequency histogram of your Twitter live stream data.

twitter_word_frequency() should return a dictionary mapping words to frequencies.

What to turn in: count_english.py

## Problem 3: Which State is happiest?

Using the live stream, write a script, happiest_state.py, with a function, happiest_state() to deduce the state from which each tweet was submitted.

happiest_state() should return a string representation of the happiest state.

Some tweets are geotagged, and you can use this. There are other ways that may be simpler but perhaps less accurate. Anything is fine, but document and defend your assumptions.

The live stream has a slightly different format from the response to the query you have been using. In this file, each line is a Tweet object, as **described in the twitter documentation**.

Note: Not every tweet dictionary will have a text key -- real data is dirty. Be prepared to debug, and feel free to throw out tweets that your code can't handle to get something working.

The file saturday_afternoon.json contains a 5-minute sample of about 1% of the complete twitter stream, which may be useful during testing.

What to turn in: happiest_state.py

## Problem 4: Top 10 hash tags

Write a script, top_tags.py, with a function, top_tags(), that computes the ten hash tags with the highest frequency from your Twitter live stream data.

top_tags() should return a ten-element mapping from hash_tags to frequencies.

What to turn in: top_tags.py