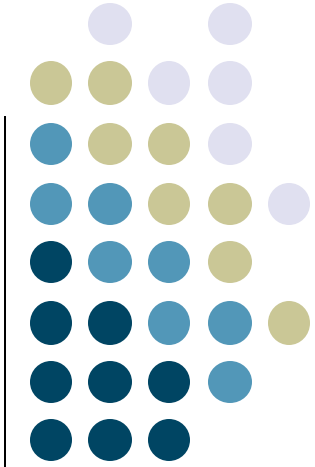
Accessing Web Files in Python

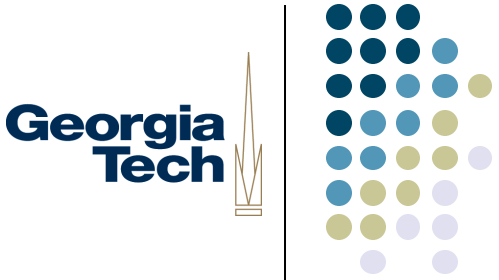


John Stasko

CS 6452

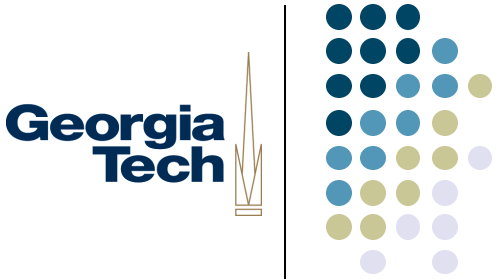
Prototyping Interactive Systems

Learning Objectives



- Understand simple web-based model of data
- Learn how to access web page content through Python
- Understand web services & API architecture/model
- See how to access Twitter web API

Data Files



- Last time we learned how to open, read from, and write to CSV and JSON files that are already on your computer
- Today, we get those files from the internet

Client - Server

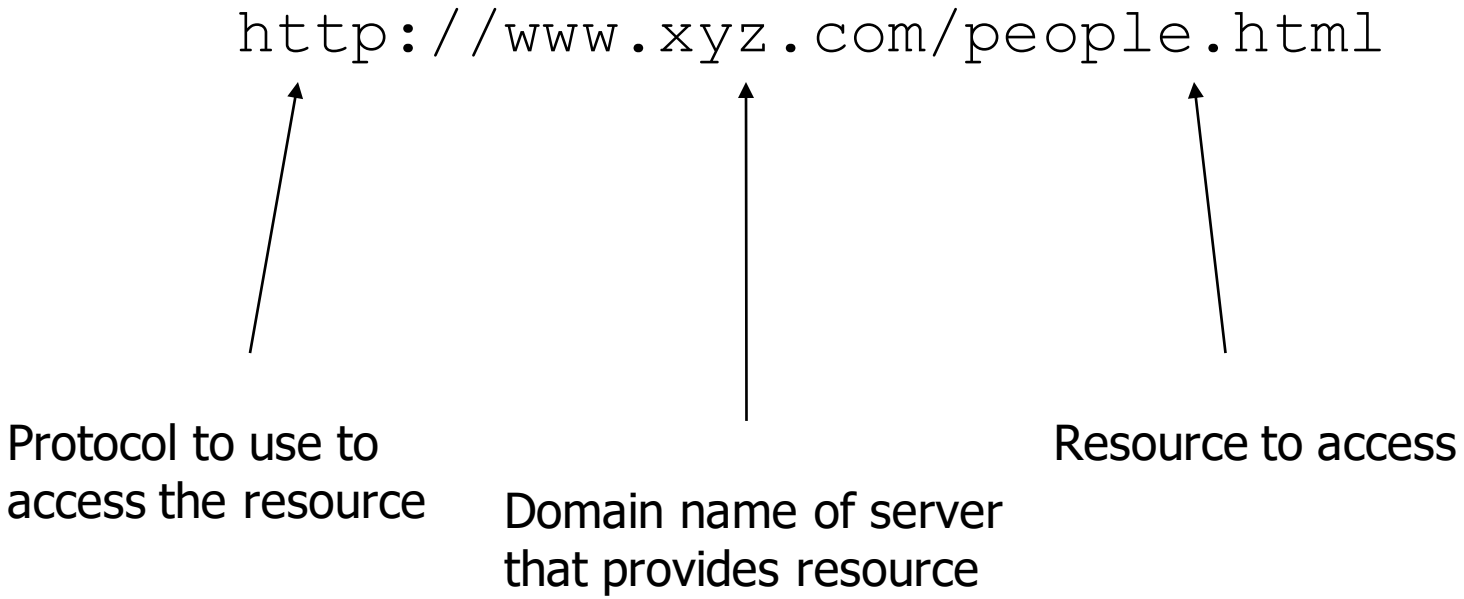
Client

Asks for the resources

Server

Holds the resources

URL: Uniform Resource Locator



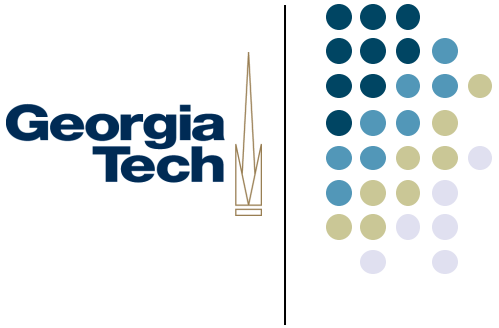
Notes

- Not every computer connected to the internet can serve data
  - Must be running software that knows http (or ftp) to be a server
  - Typically there's a special server directory. Only files in there can be accessed.

HTML

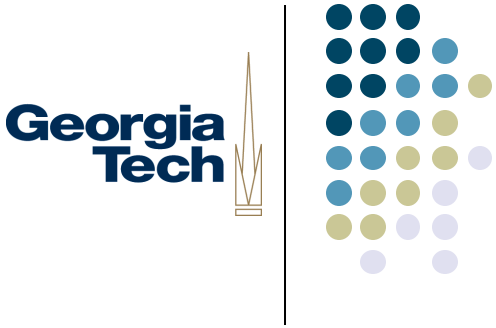
```
<HTML>

<HEAD>
<TITLE>CS 7450 Homework 1</TITLE>
</HEAD>

<BODY BGCOLOR=white>

<TABLE>
<TR>
<TD WIDTH=33% ALIGN=LEFT> <I>Due August 29</I>
<TD WIDTH=34% ALIGN=CENTER> <A
   HREF=http://www.cc.gatech.edu/~stasko/7450>
   CS 7450 - Information Visualization</A>
<TD WIDTH=33% ALIGN=RIGHT> <I>Fall 2016</I>
</TR>
</TABLE>

<HR>

<CENTER>
<H2> Homework 1: Data Exploration and Analysis </H2>
</CENTER>

<P>The purpose of this assignment is to provide you with some experience
exploring and analyzing data <b>without</b> using an information
visualization system.  Below is a data set (that can be imported into
Excel) about cereals.  You should explore and analyze this data using
Excel or simply by hand (drawing pictures is fine), but do not use any
visualization tools.  Your goal here is to perform an exploratory
analysis of the data set, to better understand the data
set and its characteristics, and to develop insights about the cereal
data.</P>

</BODY>
</HTML>
```

Python Access (Simple)



- Use `urllib` module
  - `urllib.urlopen` function to open resource
  - `read` function to get data

Example



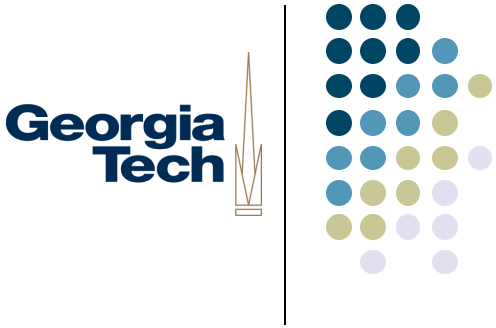
```
import urllib
import urllib.request

connect = urllib.request.urlopen("http://www.cnn.com")
content = connect.readlines()
connect.close()
print(content[0:20])
```

Try It

openURL.py program from t-square

```
import urllib
import urllib.request

target = input("URL to open? ")
connect = urllib.request.urlopen(target)
content = connect.readlines()
connect.close()
print(content[0:20])
```

`urlopen` info



This function always returns an object which can work as a context manager and has methods such as

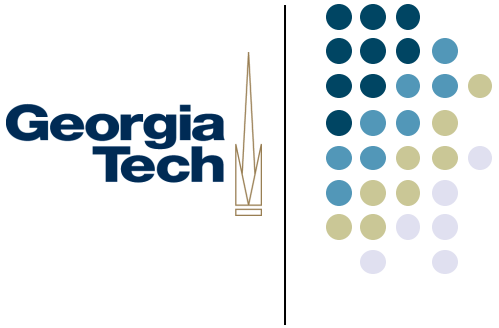
- geturl() — return the URL of the resource retrieved, commonly used to determine if a redirect was followed
- info() — return the meta-information of the page, such as headers, in the form of an email.message_from_string() instance (see Quick Reference to HTTP Headers)
- getcode() – return the HTTP status code of the response.

For HTTP and HTTPS URLs, this function returns a http.client.HTTPResponse object slightly modified. In addition to the three new methods above, the msg attribute contains the same information as the reason attribute — the reason phrase returned by server — instead of the response headers as it is specified in the documentation for HTTPResponse.

For FTP, file, and data URLs and requests explicitly handled by legacy URLopener and FancyURLopener classes, this function returns a urllib.response.addinfourl object.

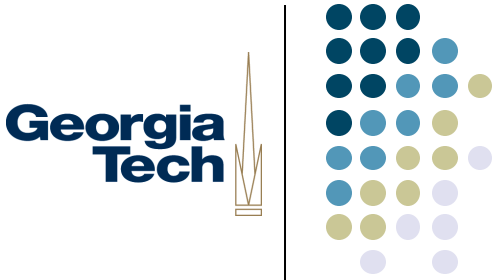
Raises URLError on protocol errors.

From Python doc



More powerful method

requests Library



- Not part of standard python distribution
- Part of anaconda

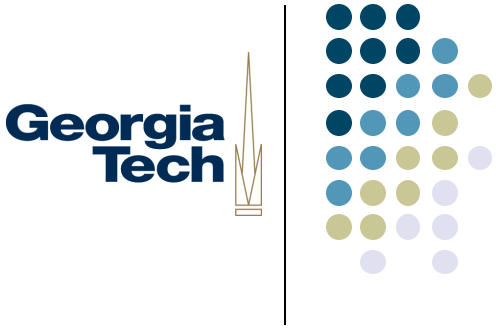
- If you don't have anaconda, must install requests
  - Use pip

pip

- Package management system used to install and manage software packages written in Python

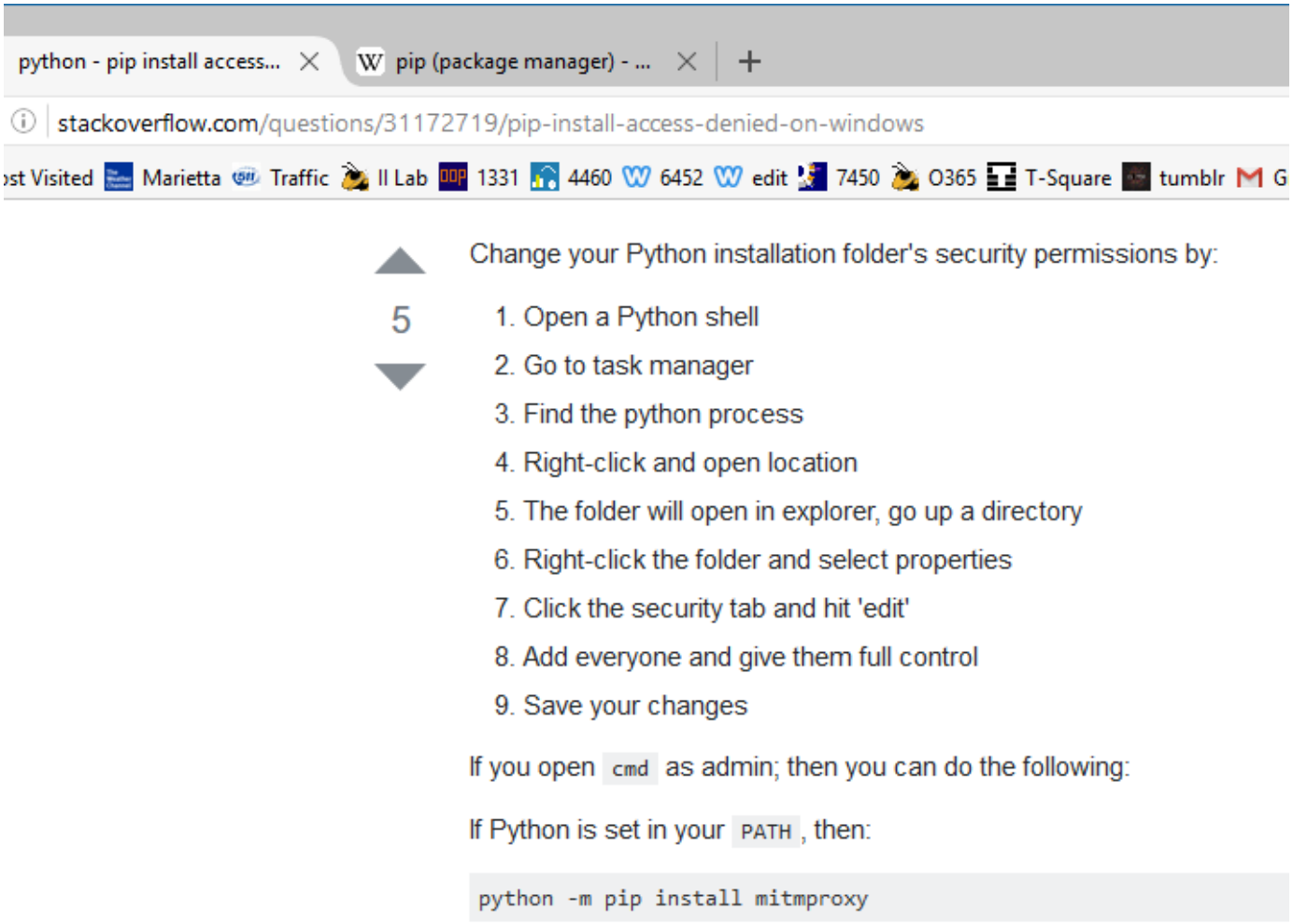
```
pip  install package_name

pip uninstall package_name
```

How-to

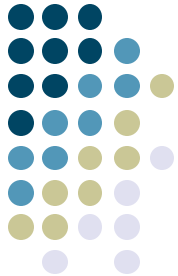


- Mac
  - `pip install requests`
- Windows
  - `python -m pip install requests`
  - Likely to have a problem

Windows Problem Fix



Try it



```
import requests
response = requests("http://www.gatech.edu")
```

Response is an object with many fields

```
dir(response)
```

Shows those fields

See `status_code, headers, text`

e.g., `response.status_code`

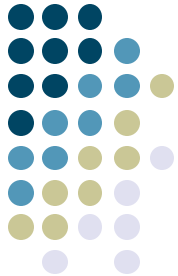
Accessing Webpage Data

- You now can get any webpage and read the code/data on it
  - For example, a page may have a table of data values
  - You will need to parse all the HTML text to get the contents of the table

Web Scraping

- Tools that assist you to go pull in (scrape) the data sitting on webpages
  - BeautifulSoup
  - Scrapy
- Can be quite tricky

An Easier Way?

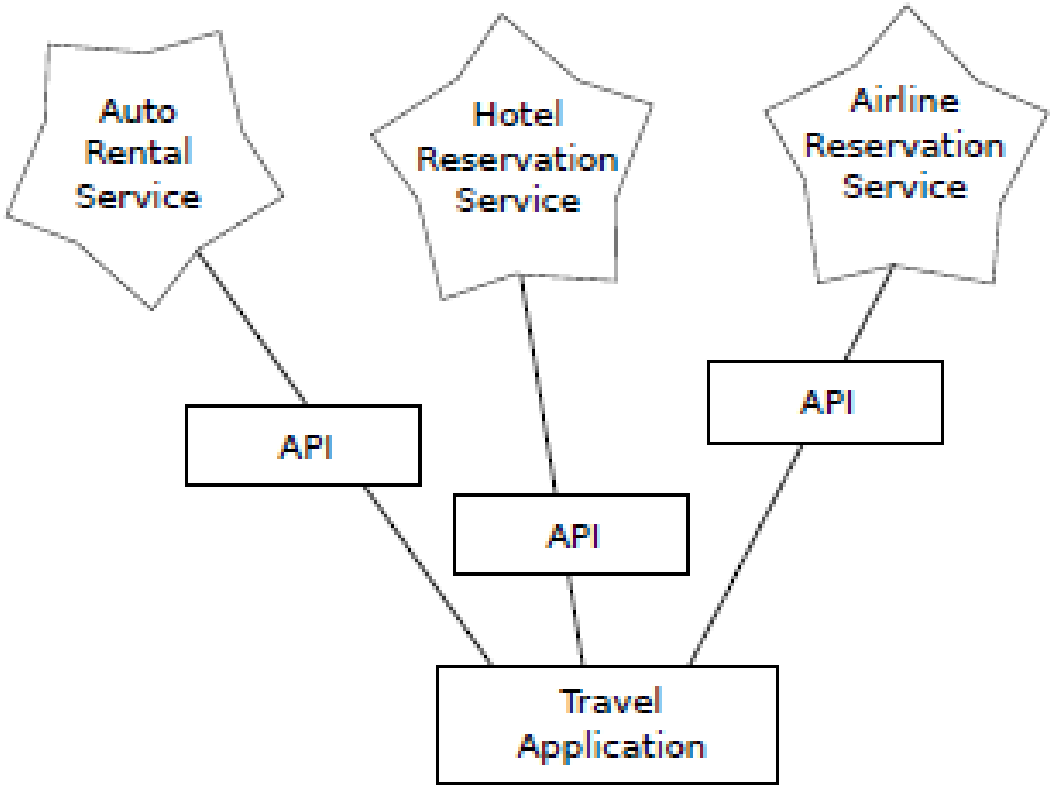


- Websites realized that they have useful data for people
- They have published APIs (Application Programmer Interfaces) that provide the data more directly
- Many websites have this
  - e.g., New York Times, Yelp, Twitter, Flickr, Foursquare, Instagram, LinkedIn, Vimeo, Tumblr, Google Books, Facebook, Google+, YouTube, Rotten Tomatoes

Web APIs

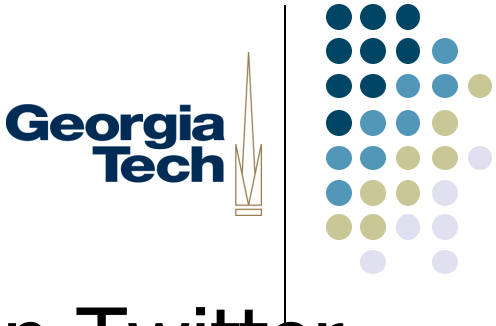
- A site makes a set of services available to other applications
- When we write out program to make use of a set of services from other, we're defining a Service-Oriented Architecture (SOA)

Example



From Severance p.160

Example: Twitter



- Tweepy is an easy-to-use Python Twitter library
- Allows you to get latest tweets from your timeline

```
pip install tweepy
```

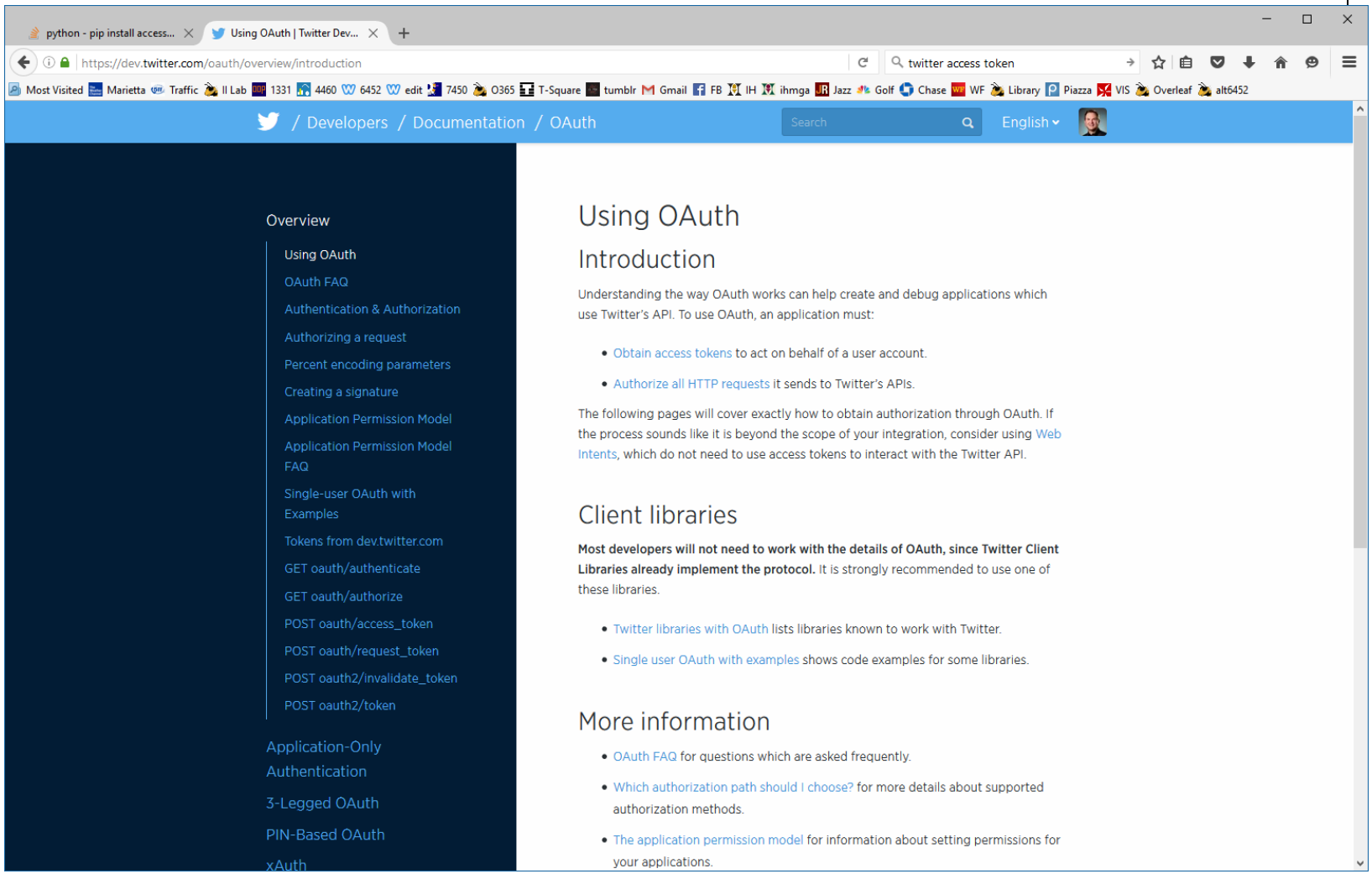
Pause I

- WARNING: With these web APIs, you need to be careful
- Could write a python program that keeps calling the API to get data in a tight for loop
  - If lots of people did this, could bring down the web server (denial of service attack)
  - They block you from doing that, ie, shut you down

Pause 2

- You must respect the limits to requests put on by these websites
  - eg, 15 requests in 15 minutes
- If you don't, then you may find your (or your organization's) access to the parent website shut off

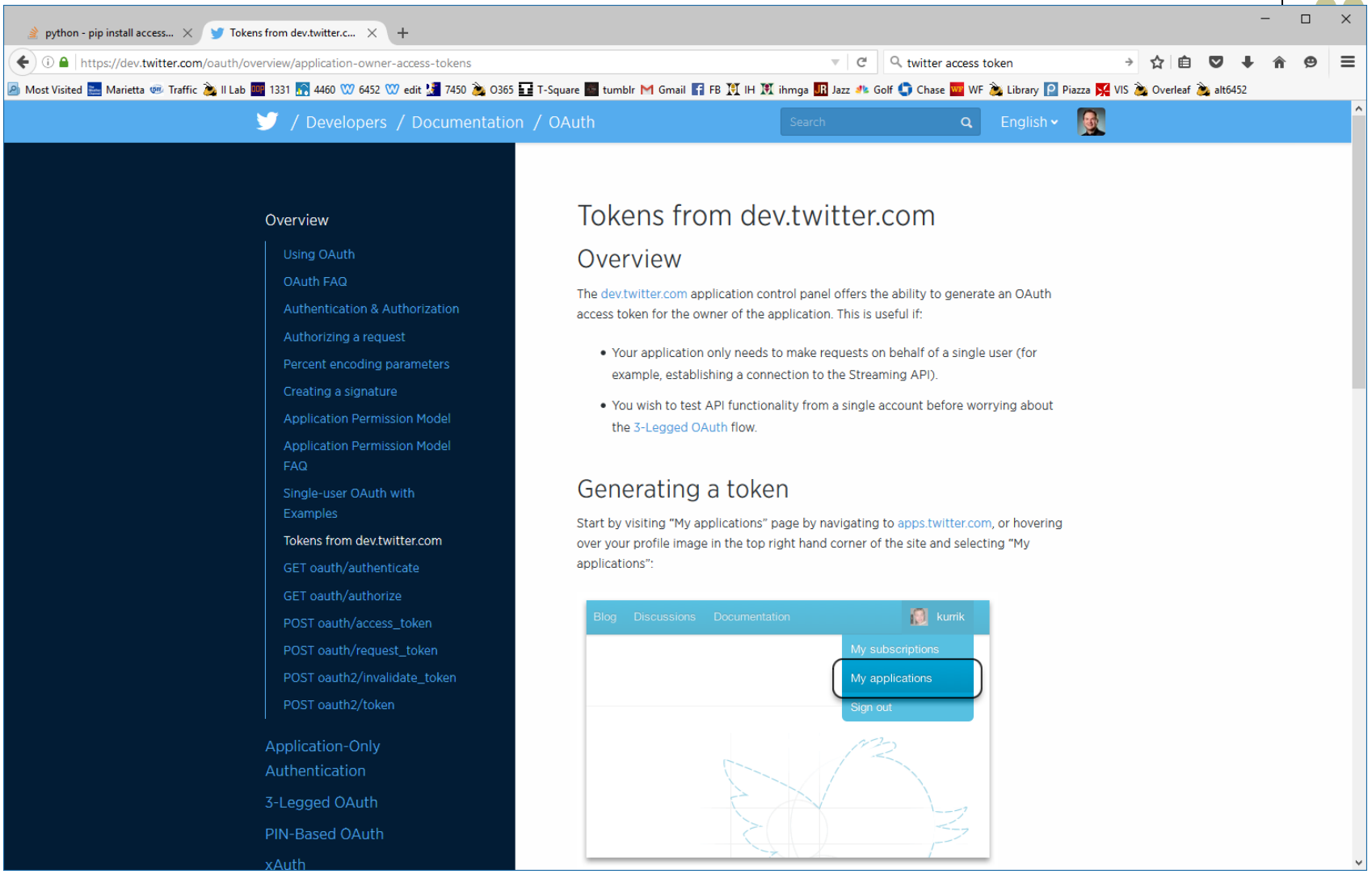
Twitter API Info



Accessing an API

- They don't let in any old riff-raff
- You must get permission, ie, accesss tokens
- Unique to each user (you)
  - That way they can monitor & track who's accessing their site

Getting Access Tokens

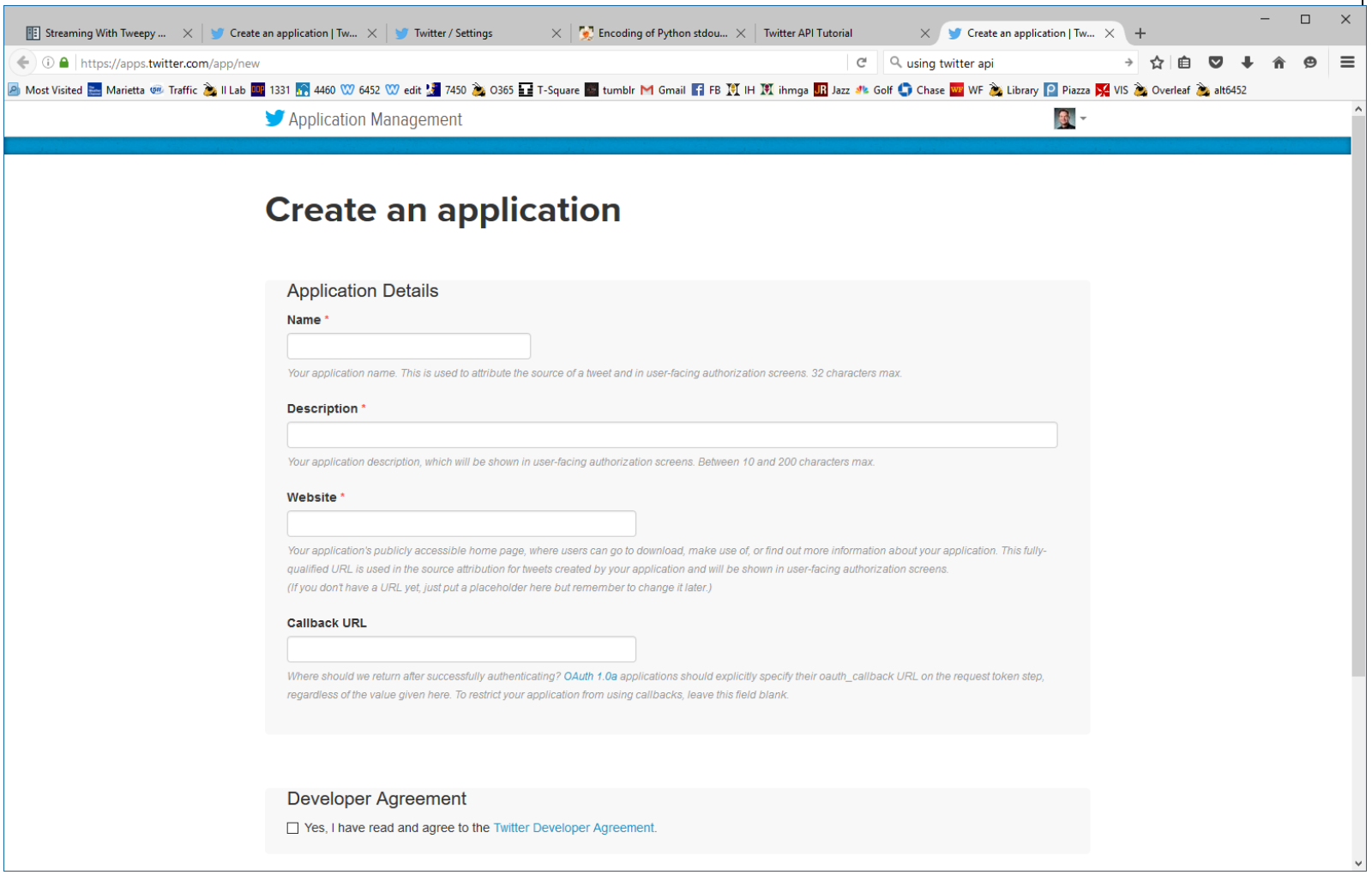


Getting Access Tokens

Go to `https://apps.twitter.com/`

Will need to make a Twitter app
You have to fill out forms and names

Getting Access Tokens

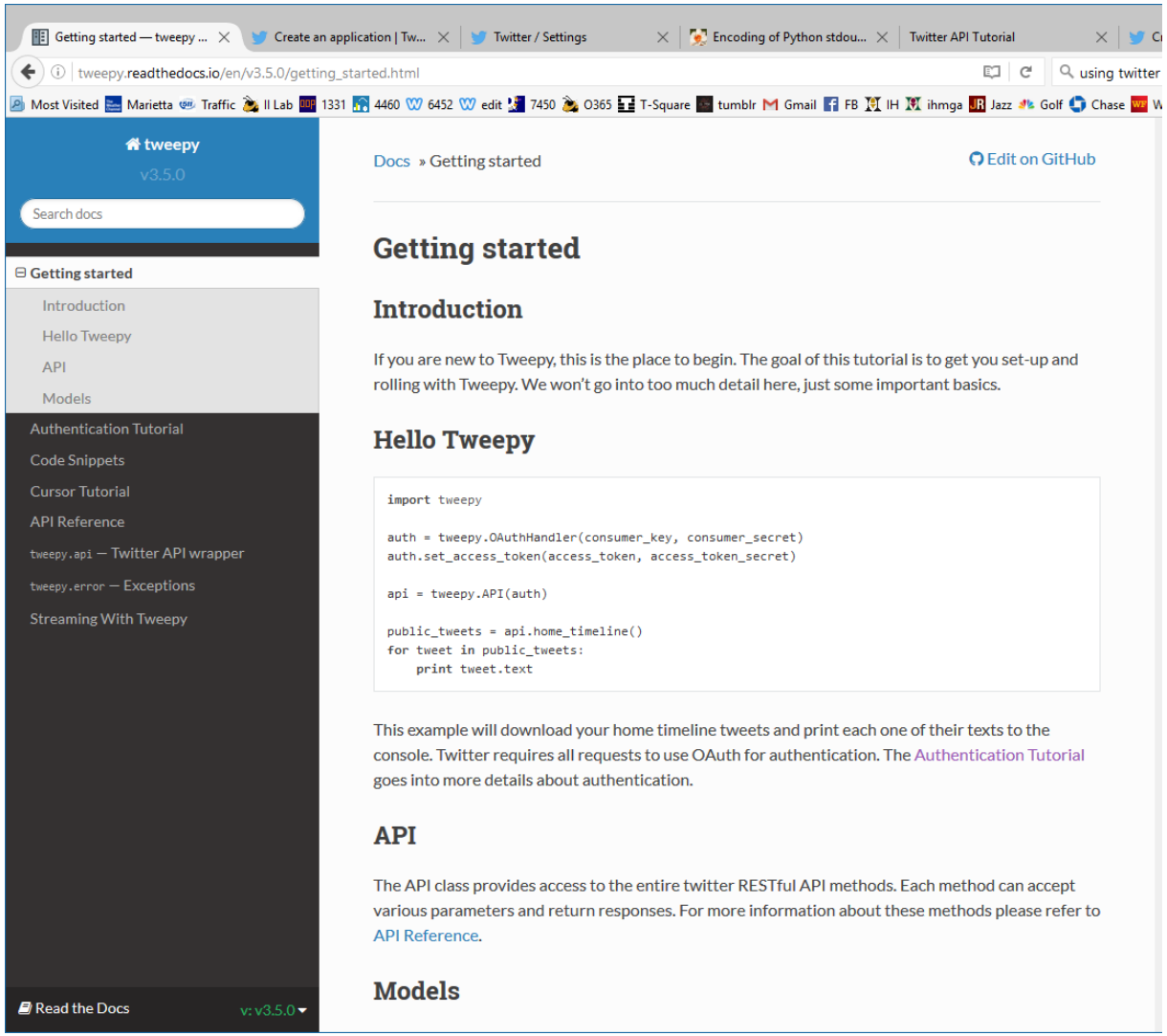


Twitter

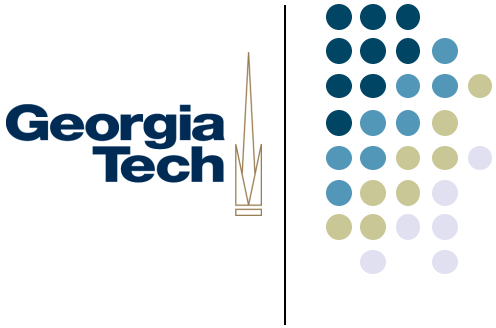
- Need
  - `access_token`
  - `access_token_secret`
  - `consumer_key`
  - `consumer_secret`

http://tweepy.readthedocs.io/en/v3.5.0/getting_started.html

Nice Tutorial

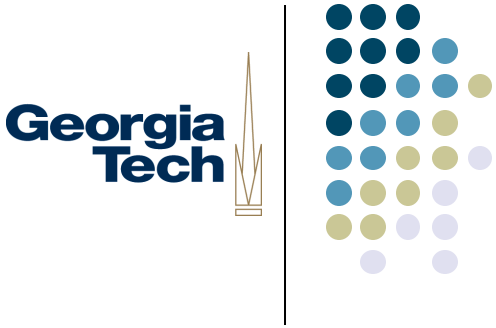


Example Program (part 1)



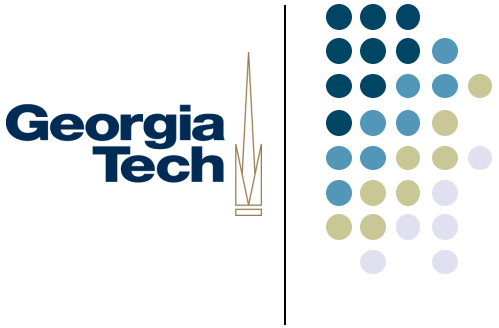
```
import tweepy
import sys
import codecs

access_token = "yours_here"
access_token_secret = "yours_here"

consumer_key = "yours_here"
consumer_secret = "yours_here"


def main():
    # some junk to get weird chars to print out OK on your terminal
    if sys.stdout.encoding != 'UTF-8':
        sys.stdout = codecs.getwriter('utf-8')(sys.stdout.buffer, 'strict')
    if sys.stderr.encoding != 'UTF-8':
        sys.stderr = codecs.getwriter('utf-8')(sys.stderr.buffer, 'strict')

    # Pass your credentials
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    # … continued on next page
```

Example Program (part 2)



```
    # … continued from previous page
    api = tweepy.API(auth)

    public_tweets = api.home_timeline()
    for tweet in public_tweets:
        print(tweet.text)
        print()

    # Get the User object for twitter...
    user = api.get_user('yourtwitterID')

    print(user.screen_name)
    print(user.followers_count)
    for friend in user.friends():
       print(friend.screen_name)


main()
```

Let's Try It



Stream API

Getting a live (dampened) stream of Tweets

```
# Do authentication stuff…

# Initiate the connection to Twitter Streaming API
twitter_stream = TwitterStream(auth=oauth)

# Get a sample of the public data flowing through Twitter
iterator = twitter_stream.statuses.sample()

# Print each tweet in the stream to the screen
# Here we set it to stop after getting 1000 tweets.
# You don't have to set it to stop, but can continue running
# the Twitter API to collect data for days or even longer.
tweet_count = 1000
for tweet in iterator:
    tweet_count -= 1
    # Twitter Python Tool wraps the data returned by Twitter
    # as a TwitterDictResponse object.
    # We convert it back to the JSON format to print/score
    print json.dumps(tweet)

    # The command below will do pretty printing for JSON data, try it out
    # print json.dumps(tweet, indent=4)

    if tweet_count <= 0:
        break
```

Others

- Search API
  - Can search by #terms
- Trends API
  - Can grab different trends

Very Nice Tutorial

Social Media & Text Analytics | Syllabus | Twitter API Tutorial | Homework

Twitter's 404 error page -- the Fail Whale

Twitter API tutorial

by Wei Xu Follow @cocoweixu (Ohio State University)

last updated Feb 28, 2016; originally written July 1, 2015

1. Getting Twitter API keys

To start with, you will need to have a Twitter account and obtain credentials (i.e. API key, API secret, Access token and Access token secret) on the Twitter developer site to access the Twitter API, following these steps:

- Create a Twitter user account if you do not already have one.
- Go to https://apps.twitter.com/ and log in with your Twitter user account. This step gives you a Twitter dev account under the same name as your user account.
- Click "Create New App"
- Fill out the form, agree to the terms, and click "Create your Twitter application"
- In the next page, click on "Keys and Access Tokens" tab, and copy your "API key" and "API secret". Scroll down and click "Create my access token", and copy your "Access token" and "Access token secret".

2. Installing a Twitter library

We will be using a Python library called Python Twitter Tools to connect to Twitter API and downloading the data from Twitter. There are many other libraries in various programming languages that let you use Twitter API. We choose the Python Twitter Tools for this tutorial, because it is simple to use yet fully supports the Twitter API.

Download the Python Twitter tools at https://pypi.python.org/pypi/twitter.

Install the Python Twitter Tools package by typing in commands:

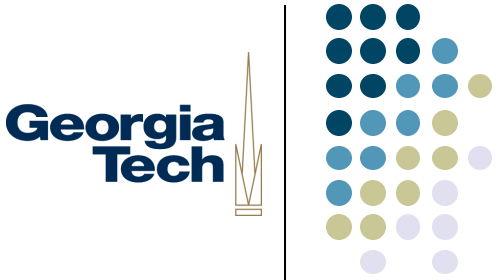
```
$ python setup.py --help
$ python setup.py build
$ python setup.py install
```

3. Connecting to Twitter Streaming APIs

The Streaming APIs give access to (usually a sample of) all tweets as they published on Twitter. On average, about 6,000 tweets per second are posted on Twitter and you (normal dev users) will get a small proportion (<=1%) of it. The Streaming APIs are one of the two types of Twitter APIs. The other one called REST APIs (we will talk about later in this tutorial), which is more suitable for singular searches, such as searching historic tweets, reading user profile information, or posting Tweets. The Streaming API **only** sends out real-time tweets, while the Search API (one of the popular REST APIs) gives historical tweets up to about a week with a max of a couple of hundreds. You may request elevated access (e.g. Firehose, Retweet, Link, Birddog or Shadow) for more data by contacting Twitter's API support.

http://socialmedia-class.org/twittertutorial.html

Learning Objectives



- Understand simple web-based model of data
- Learn how to access web page content through Python
- Understand web services & API architecture/model
- See how to access Twitter web API

Next Time

- Visualizing data with Pandas