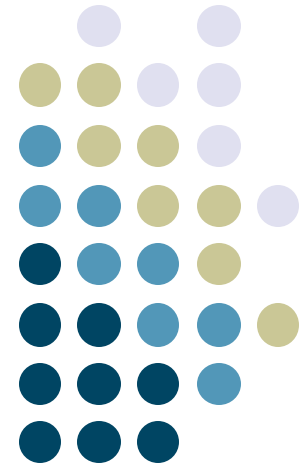


Accessing Web Files in Python



**Georgia
Tech**

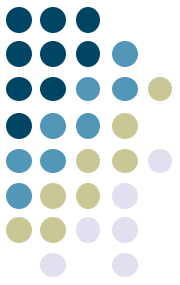


John Stasko

CS 6452

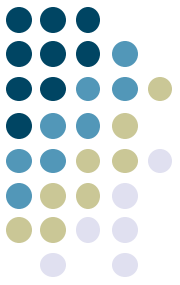
Prototyping Interactive Systems

Learning Objectives



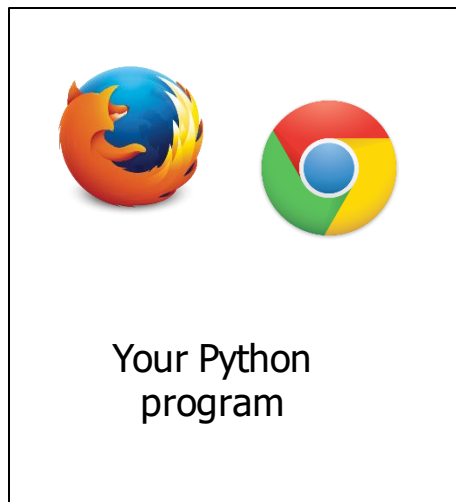
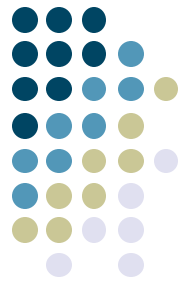
- Understand simple web-based model of data
- Learn how to access web page content through Python
- Understand web services & API architecture/model
- See how to access Twitter web API

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



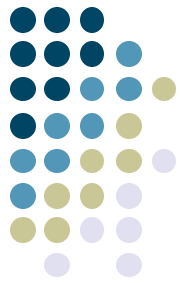
Client

Asks for the resources



Server

Holds the resources



URL: Uniform Resource Locator

`http://www.xyz.com/people.html`



Protocol to use to
access the resource

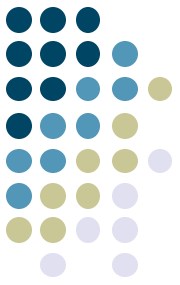


Domain name of server
that provides resource



Resource to access

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>

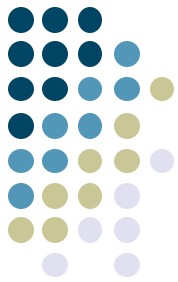
<HR>

<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>
```

Georgia
Tech



Python Access (Simple)



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

Example

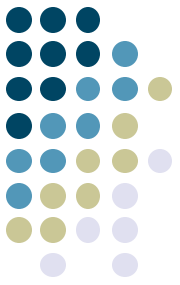


```
import urllib
import urllib.request

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

Try It

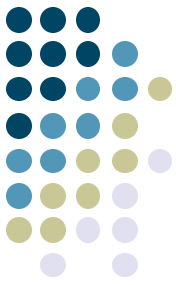
openURL.py program from t-square



```
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

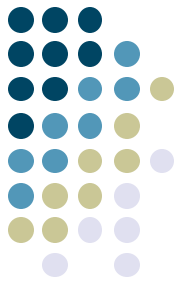
- `geturl()` — return the URL of the resource retrieved, commonly used to determine if a redirect was followed
- `info()` — return the meta-information of the page, such as headers, in the form of an `email.message_from_string()` instance (see Quick Reference to HTTP Headers)
- `getcode()` — return the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

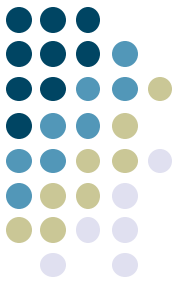
Raises `URLError` on protocol errors.

From Python doc



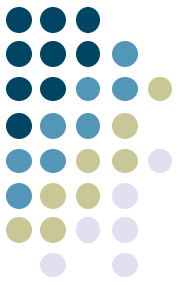
More powerful method

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**

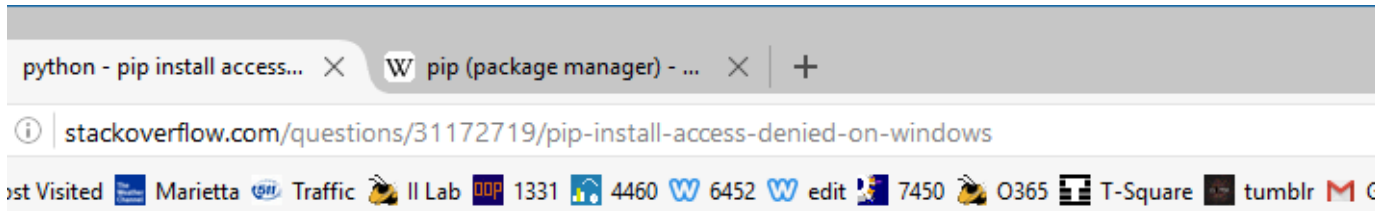
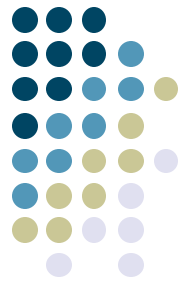
- `pip install requests`

- **Windows**

- `python -m pip install requests`

- **Likely to have a problem**

Windows Problem Fix



▲ Change your Python installation folder's security permissions by:

5



1. Open a Python shell
2. Go to task manager
3. Find the python process
4. Right-click and open location
5. The folder will open in explorer, go up a directory
6. Right-click the folder and select properties
7. Click the security tab and hit 'edit'
8. Add everyone and give them full control
9. Save your changes

If you open `cmd` as admin; then you can do the following:

If Python is set in your `PATH`, then:

```
python -m pip install mitmproxy
```


Try it



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

Response is an object with many fields

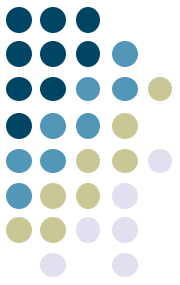
```
dir(response)
```

Shows those fields

See `status_code`, `headers`, `text`

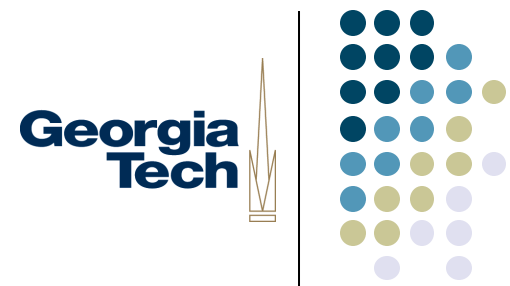
e.g., `response.status_code`

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

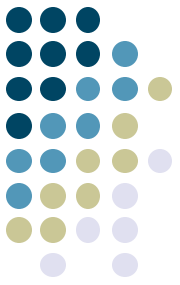
Web Scraping



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

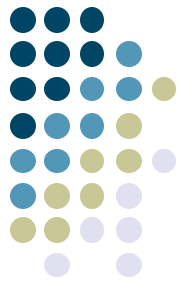
- Can be quite tricky

An Easier Way?



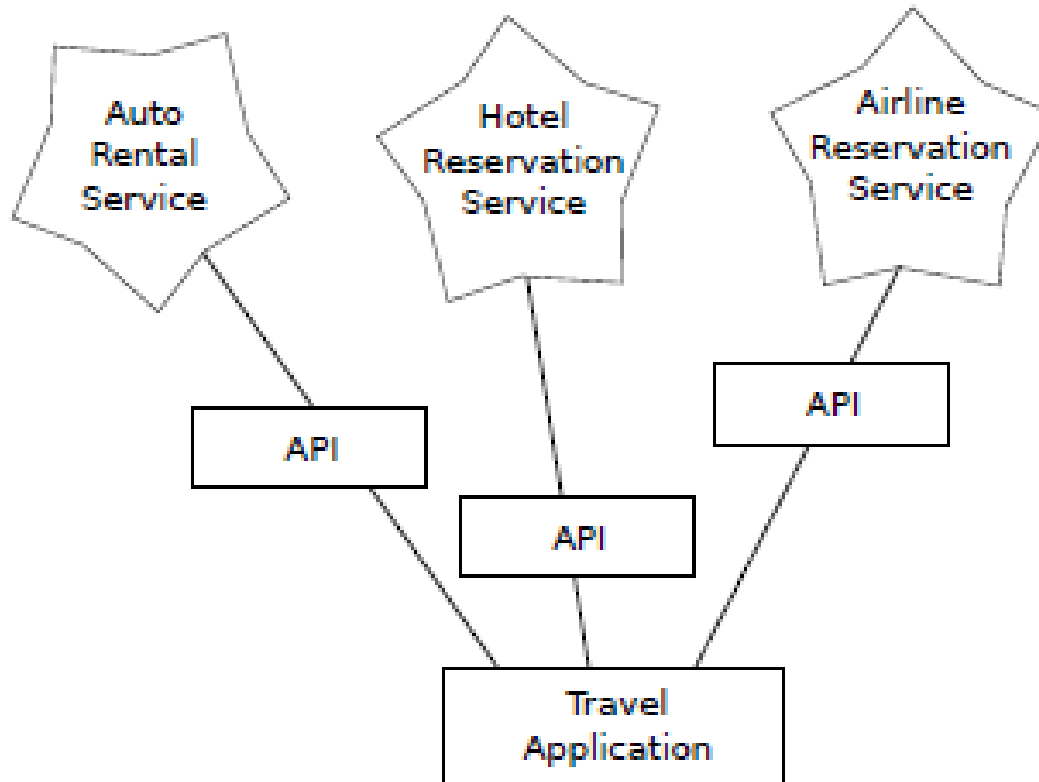
- Websites realized that they have useful data for people
- They have published APIs (Application Programmer Interfaces) that provide the data more directly
- Many websites have this
 - e.g., New York Times, Yelp, Twitter, Flickr, Foursquare, Instagram, LinkedIn, Vimeo, Tumblr, Google Books, Facebook,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

Web APIs



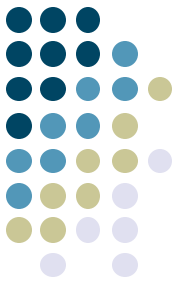
- A site makes a set of services available to other applications
- When we write out program to make use of a set of services from other, we're defining a Service-Oriented Architecture (SOA)

Example



From Severance p.160

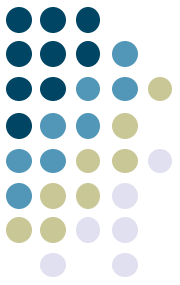
Example: Twitter



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

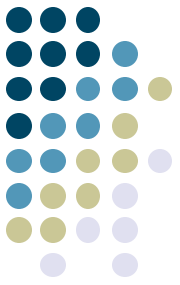
```
pip install tweepy
```

Pause I



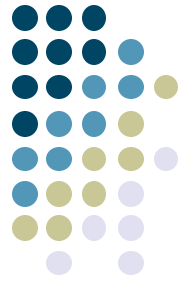
- **WARNING:** With these web APIs, you need to be careful
- Could write a python program that keeps calling the API to get data in a tight for loop
 - If lots of people did this, could bring down the web server (denial of service attack)
 - They block you from doing that, ie, shut you down

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



The screenshot shows a web browser window displaying the Twitter Developer documentation page for OAuth. The browser's address bar shows the URL `https://dev.twitter.com/oauth/overview/introduction`. The page title is "Using OAuth | Twitter Dev...". The page content is organized into sections: "Using OAuth Introduction", "Client libraries", and "More information".

Using OAuth

Introduction

Understanding the way OAuth works can help create and debug applications which use Twitter's API. To use OAuth, an application must:

- [Obtain access tokens](#) to act on behalf of a user account.
- [Authorize all HTTP requests](#) it sends to Twitter's APIs.

The following pages will cover exactly how to obtain authorization through OAuth. If the process sounds like it is beyond the scope of your integration, consider using [Web Intents](#), which do not need to use access tokens to interact with the Twitter API.

Client libraries

Most developers will not need to work with the details of OAuth, since Twitter Client Libraries already implement the protocol. It is strongly recommended to use one of these libraries.

- [Twitter libraries with OAuth](#) lists libraries known to work with Twitter.
- [Single user OAuth with examples](#) shows code examples for some libraries.

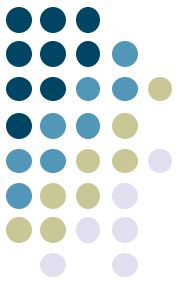
More information

- [OAuth FAQ](#) for questions which are asked frequently.
- [Which authorization path should I choose?](#) for more details about supported authorization methods.
- [The application permission model](#) for information about setting permissions for your applications.

The left sidebar of the page contains a navigation menu with the following items:

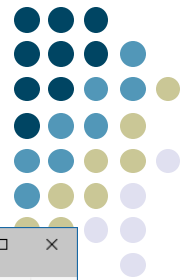
- Overview
- Using OAuth
- OAuth FAQ
- Authentication & Authorization
- Authorizing a request
- Percent encoding parameters
- Creating a signature
- Application Permission Model
- Application Permission Model FAQ
- Single-user OAuth with Examples
- Tokens from dev.twitter.com
- GET `oauth/authenticate`
- GET `oauth/authorize`
- POST `oauth/access_token`
- POST `oauth/request_token`
- POST `oauth2/invalidate_token`
- POST `oauth2/token`
- Application-Only Authentication
- 3-Legged OAuth
- PIN-Based OAuth
- xAuth

Accessing an API



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

Getting Access Tokens



python - pip install access... x Tokens from dev.twitter.c... x +

https://dev.twitter.com/oauth/overview/application-owner-access-tokens

twitter access token

Most Visited Marietta Traffic Il Lab 1331 4460 6452 edit 7450 0365 T-Square tumblr Gmail FB IH ihmga Jazz Golf Chase WF Library Piazza VIS Overleaf alt6452

/ Developers / Documentation / OAuth

Tokens from dev.twitter.com

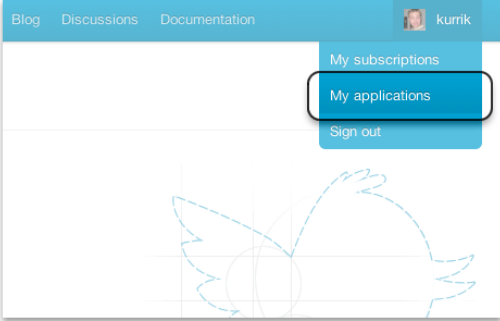
Overview

The [dev.twitter.com](#) application control panel offers the ability to generate an OAuth access token for the owner of the application. This is useful if:

- Your application only needs to make requests on behalf of a single user (for example, establishing a connection to the Streaming API).
- You wish to test API functionality from a single account before worrying about the 3-Legged OAuth flow.

Generating a token

Start by visiting "My applications" page by navigating to [apps.twitter.com](#), or hovering over your profile image in the top right hand corner of the site and selecting "My applications":



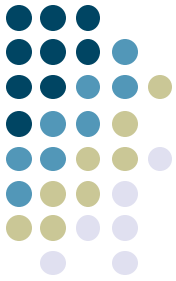
Getting Access Tokens



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

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

Getting Access Tokens



The screenshot shows a browser window with the URL `https://apps.twitter.com/app/new`. The page title is "Application Management" and the main heading is "Create an application". The form is divided into two main sections: "Application Details" and "Developer Agreement".

Application Details

- Name ***: A text input field. Below it, a note reads: "Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max."
- Description ***: A text input field. Below it, a note reads: "Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max."
- Website ***: A text input field. Below it, a note reads: "Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens. (If you don't have a URL yet, just put a placeholder here but remember to change it later.)"
- Callback URL**: A text input field. Below it, a note reads: "Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank."

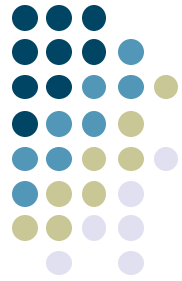
Developer Agreement

Yes, I have read and agree to the [Twitter Developer Agreement](#).

Twitter



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



Nice Tutorial

The screenshot shows a web browser displaying the Tweepy documentation page. The browser tabs include 'Getting started — tweepy ...', 'Create an application | Tw...', 'Twitter / Settings', 'Encoding of Python stdout...', and 'Twitter API Tutorial'. The address bar shows 'tweepy.readthedocs.io/en/v3.5.0/getting_started.html'. The page content includes a navigation sidebar on the left with a search bar and a list of topics: 'Getting started', 'Introduction', 'Hello Tweepy', 'API', 'Models', 'Authentication Tutorial', 'Code Snippets', 'Cursor Tutorial', 'API Reference', 'tweepy.api — Twitter API wrapper', 'tweepy.error — Exceptions', and 'Streaming With Tweepy'. The main content area is titled 'Getting started' and contains an 'Introduction' section with the text: 'If you are new to Tweepy, this is the place to begin. The goal of this tutorial is to get you set-up and rolling with Tweepy. We won't go into too much detail here, just some important basics.' Below this is a 'Hello Tweepy' section with a code block:

```
import tweepy

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)

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

Below the code block, the text reads: 'This example will download your home timeline tweets and print each one of their texts to the console. Twitter requires all requests to use OAuth for authentication. The [Authentication Tutorial](#) goes into more details about authentication.'

The 'API' section states: 'The API class provides access to the entire twitter RESTful API methods. Each method can accept various parameters and return responses. For more information about these methods please refer to [API Reference](#).'

The 'Models' section is partially visible at the bottom of the page.

Example Program (part I)



```
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()

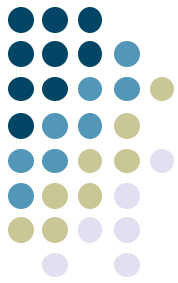
# Get the User object for twitter...
user = api.get_user('yourtwitterID')

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

main()
```

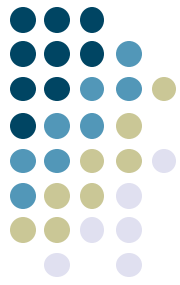
Let's Try It

Georgia
Tech



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)

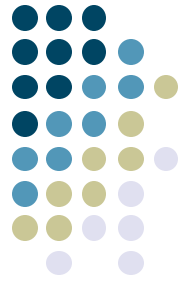
if tweet_count <= 0:
    break
```

Others

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



Very Nice Tutorial



Twitter's 404 error page -- the Fail Whale

Twitter API tutorial

by Wei Xu [Follow @ccoweiwu](#) (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:

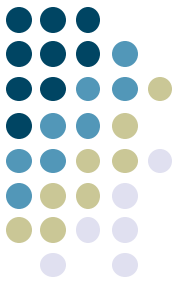
```
$ python setup.py --help
$ python setup.py build
$ python setup.py install
```

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, Birdoo 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

