IMPORTING DATA IN PYTHON II

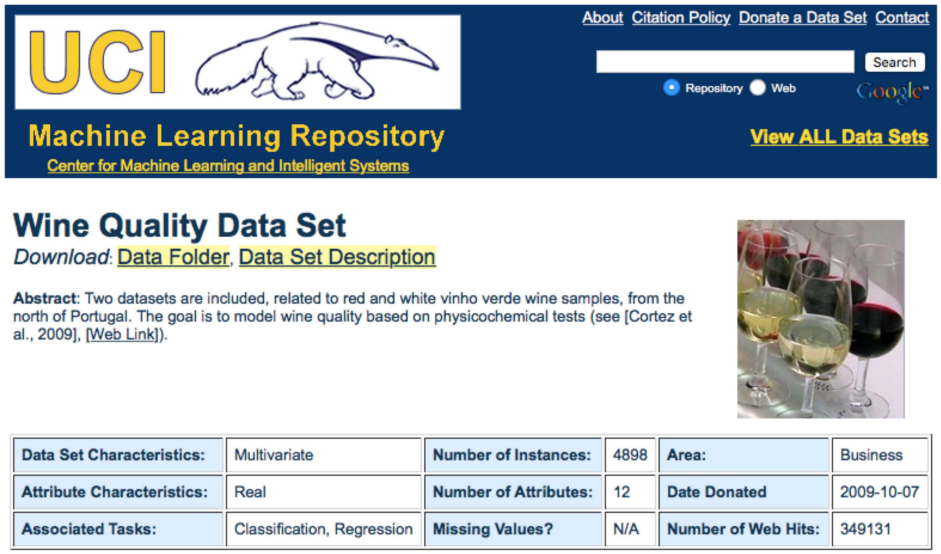
Importing flat files from the web

You're already great at importing!

- Flat files such as .txt and .csv
- Pickled files, Excel spreadsheets, and many others!
- Data from relational databases

- You can do all these locally
- What if your data is online?

Can you import web data?



- You can: go to URL and click to download files
- BUT: not reproducible, not scalable

You'll learn how to...

- Import and locally save datasets from the web
- Load datasets into pandas DataFrames
- Make HTTP requests (GET requests)
- Scrape web data such as HTML
- Parse HTML into useful data (BeautifulSoup)
- Use the urllib and requests packages

The urllib package

- Provides interface for fetching data across the web
- `urlopen()` - accepts URLs instead of file names

How to automate file download in Python

```
In [1]: from urllib.request import urlretrieve

In [2]: url = 'http://archive.ics.uci.edu/ml/machine-learning-
databases/wine-quality/winequality-white.csv'

In [3]: urlretrieve(url, 'winequality-white.csv')
Out[3]: ('winequality-white.csv', <http.client.HTTPMessage at
0x103cf1128>)
```

IMPORTING DATA IN PYTHON II

Let's practice!

IMPORTING DATA IN PYTHON II

HTTP requests to import files from the web

URL

- Uniform/Universal Resource Locator
- References to web resources
- Focus: web addresses
- Ingredients:
  - Protocol identifier - http:
  - Resource name - datacamp.com
- These specify web addresses uniquely

HTTP

- HyperText Transfer Protocol
- Foundation of data communication for the web
- HTTPS - more secure form of HTTP
- Going to a website = sending HTTP request
  - GET request
- `urlretrieve()` performs a GET request
- HTML - HyperText Markup Language

GET requests using urllib

```
In [1]: from urllib.request import urlopen, Request

In [2]: url = "https://www.wikipedia.org/"

In [3]: request = Request(url)

In [4]: response = urlopen(request)

In [5]: html = response.read()

In [6]: response.close()
```

GET requests using requests

- Used by “her Majesty's Government, Amazon, Google, Twilio, NPR, Obama for America, Twitter, Sony, and Federal U.S. Institutions that prefer to be unnamed”

GET requests using requests

- One of the most downloaded Python packages

```
In [1]: import requests

In [2]: url = "https://www.wikipedia.org/"

In [3]: r = requests.get(url)

In [4]: text = r.text
```

IMPORTING DATA IN PYTHON II

Let's practice!

IMPORTING DATA IN PYTHON II

Scraping the web in Python

HTML

- Mix of unstructured and structured data
- Structured data:
  - Has pre-defined data model, or
  - Organized in a defined manner
- Unstructured data: neither of these properties

```
<!DOCTYPE html>
<html>
    <head>
        <!-- SEO -->
    <meta charset="utf-8" />
<title>DataCamp: The Easy Way To Learn R &amp;amp; Data Science Online</title>
```

BeautifulSoup

- Parse and extract structured data from HTML

- Make tag soup beautiful and extract information

BeautifulSoup

```
In [1]: from bs4 import BeautifulSoup

In [2]: import requests

In [3]: url = 'https://www.crummy.com/software/BeautifulSoup/'

In [4]: r = requests.get(url)

In [5]: html_doc = r.text

In [6]: soup = BeautifulSoup(html_doc)
```

Prettified Soup

```
In [7]: print(soup.prettify())
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" "http://www.w3.org/TR/REC-html40
<html>
 <head>
  <meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
  <title>
   Beautiful Soup: We called him Tortoise because he taught us.
  </title>
  <link href="mailto:leonardr@segfault.org" rev="made"/>
  <link href="/nb/themes/Default/nb.css" rel="stylesheet" type="text/css"/>
  <meta content="Beautiful Soup: a library designed for screen-scraping HTML and XML." name="D
  <meta content="Markov Approximation 1.4 (module: leonardr)" name="generator"/>
  <meta content="Leonard Richardson" name="author"/>
 </head>
 <body alink="red" bgcolor="white" link="blue" text="black" vlink="660066">
  <img align="right" src="10.1.jpg" width="250"/>
  <br/>
  <p>
```

Exploring BeautifulSoup

- Many methods such as:

```
In [9]: print(soup.title)
<title>Beautiful Soup: We called him Tortoise because he taught
us.</title>
```

```
In [8]: print(soup.get_text())

Beautiful Soup: We called him Tortoise because he taught us.

You didn't write that awful page. You're just trying to get some
data out of it. Beautiful Soup is here to help. Since 2004, it's
been saving programmers hours or days of work on quick-turnaround
screen scraping projects.
```

Exploring BeautifulSoup

- `find_all()`

```
In [10]: for link in soup.find_all('a'):
   ....:     print(link.get('href'))
   ....:
bs4/download/
#Download
bs4/doc/
#HallOfFame
https://code.launchpad.net/beautifulsoup
https://groups.google.com/forum/?fromgroups#!forum/beautifulsoup
http://www.candlemarkandgleam.com/shop/constellation-games/
http://constellation.crummy.com/Constellation%20Games
%20excerpt.html
https://groups.google.com/forum/?fromgroups#!forum/beautifulsoup
https://bugs.launchpad.net/beautifulsoup/
http://lxml.de/
http://code.google.com/p/html5lib/
```

IMPORTING DATA IN PYTHON II

Let's practice!

IMPORTING DATA IN PYTHON II

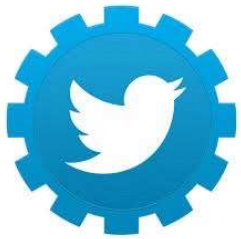
Introduction to APIs and JSONs

APIs

- Application Programming Interface
- Protocols and routines
  - Building and interacting with software applications

OMDb API

The Open Movie Database



JSONs

- JavaScript Object Notation
- Real-time server-to-browser communication
- Douglas Crockford
- Human readable

JSONs

```
{'Actors': 'Samuel L. Jackson, Julianna Margulies, Nathan
Phillips, Rachel Blanchard',
 'Awards': '3 wins & 7 nominations.',
 'Country': 'Germany, USA, Canada',
 'Director': 'David R. Ellis',
 'Genre': 'Action, Adventure, Crime',
 'Language': 'English',
 'Rated': 'R',
 'Released': '18 Aug 2006',
 'Runtime': '105 min',
 'Title': 'Snakes on a Plane',
 'Type': 'movie',
 'Writer': 'John Heffernan (screenplay), Sebastian Gutierrez
(screenplay), David Dalessandro (story), John Heffernan (story)',
 'Year': '2006',
 'imdbID': 'tt0417148',
 'imdbRating': '5.6',
 'imdbVotes': '114,668'}
```

Loading JSONs in Python

```
In [1]: import json

In [2]: with open('snakes.json', 'r') as json_file:
   ...:     json_data = json.load(json_file)

In [3]: type(json_data)
Out[3]: dict
```

Exploring JSONs in Python

```
In [4]: for key, value in json_data.items():
   ...:     print(key + ':', value)

Title: Snakes on a Plane
Country: Germany, USA, Canada
Response: True
Language: English
Awards: 3 wins & 7 nominations.
Year: 2006
Actors: Samuel L. Jackson, Julianna Margulies
Runtime: 105 min
Genre: Action, Adventure, Crime
imdbID: tt0417148
Director: David R. Ellis
imdbRating: 5.6
Rated: R
Released: 18 Aug 2006
```

IMPORTING DATA IN PYTHON II

Let's practice!

IMPORTING DATA IN PYTHON II

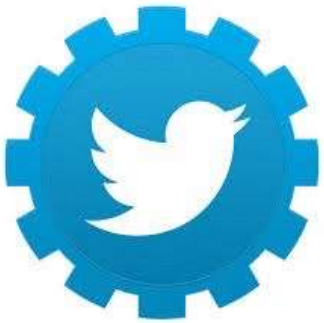
APIs and interacting with the world wide web

Herein, you'll learn

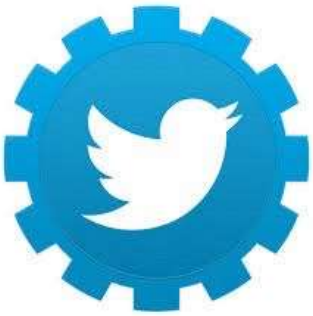
- What APIs are
- Why APIs are important
- In the exercises:
  - Connecting to APIs
  - Pulling data from APIs
  - Parsing data from APIs

What is an API?

- Set of protocols and routines
- Bunch of code
  - Allows two software programs to communicate with each other



APIs are everywhere



UBER Developers

Instagram API

Connecting to an API in Python

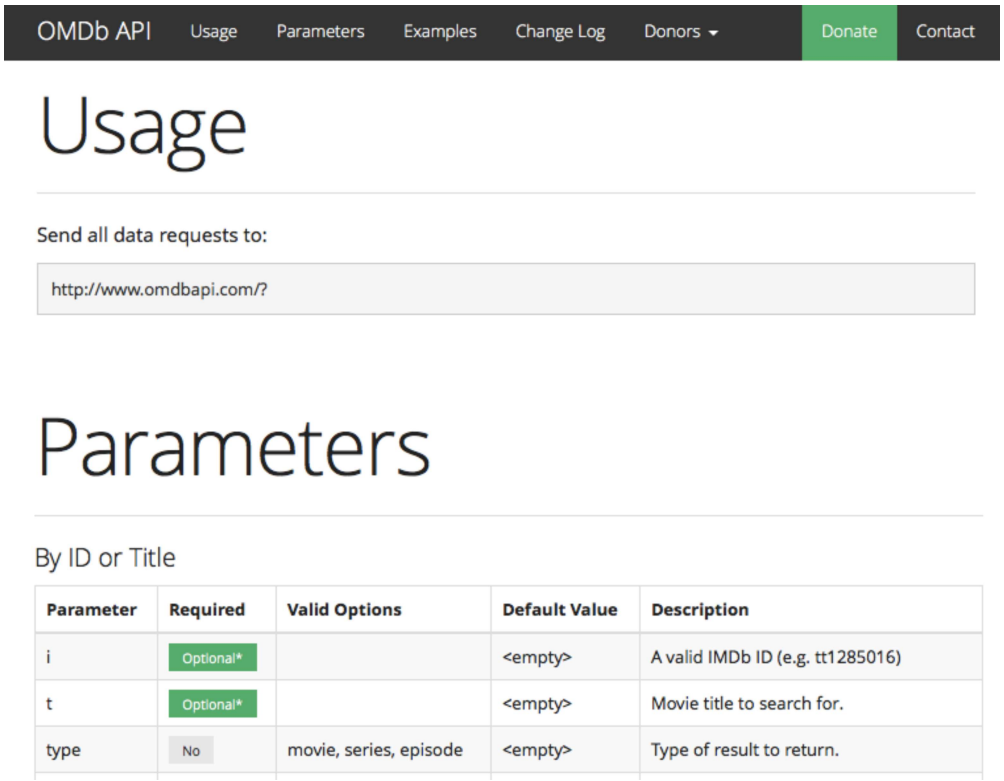
```
In [1]: import requests

In [2]: url = 'http://www.omdbapi.com/?t=hackers'

In [3]: r = requests.get(url)

In [4]: json_data = r.json()

In [5]: for key, value in json_data.items():
   ...:     print(key + ':', value)
```

What was that URL?

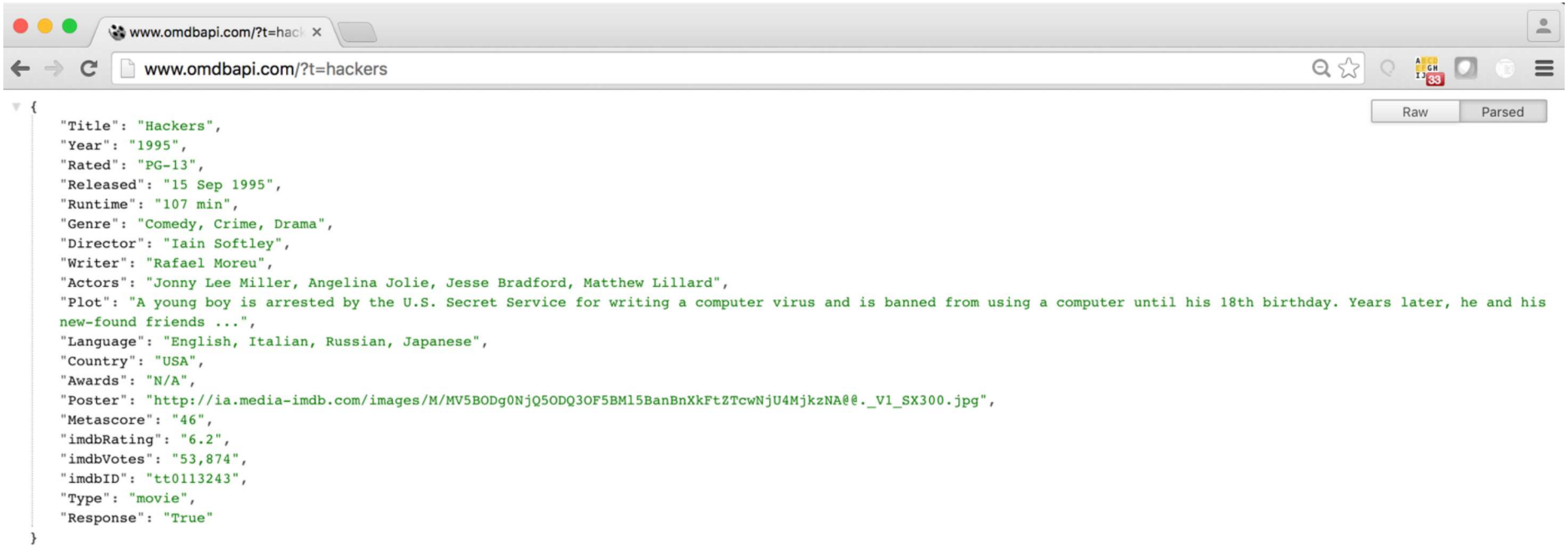
- http - making an HTTP request
- www.omdbapi.com - querying the OMDB API
- ?t=hackers
  - Query string
  - Return data for a movie with title (t) ‘Hackers’

```
'http://www.omdbapi.com/?t=hackers'
```

OMDb API


OMDb API
Usage
Parameters
Examples
Change Log
Donors
Donate
Contact
Usage
Send all data requests to:
http://www.omdbapi.com/?
Parameters
By ID or Title
Parameter
Required
Valid Options
Default Value
Description
i
Optional*
<empty>
A valid IMDb ID (e.g. tt1285016)
t
Optional*
<empty>
Movie title to search for.
type
No
movie, series, episode
<empty>
Type of result to return.

It's a regular URL!



IMPORTING DATA IN PYTHON II

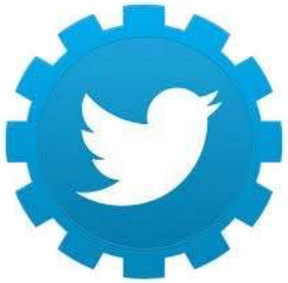
Let's practice!

IMPORTING DATA IN PYTHON II

The Twitter API and Authentication

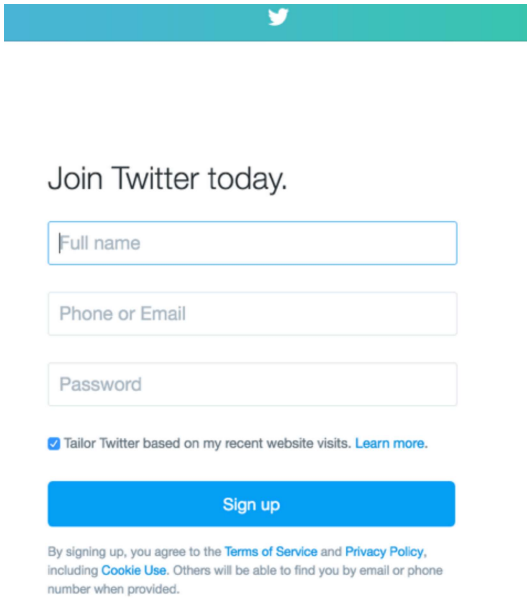
Herein, you'll learn

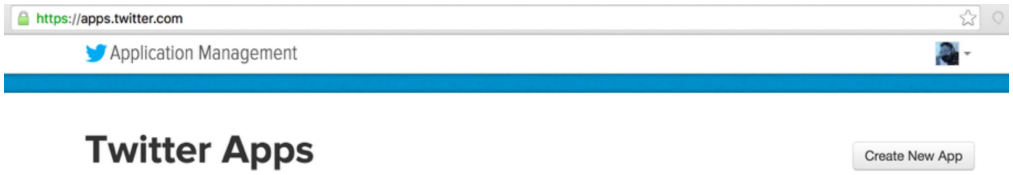
- How to stream data from the Twitter API
- How to filter incoming tweets for keywords
- About API Authentication and OAuth
- How to use the Tweepy Python package



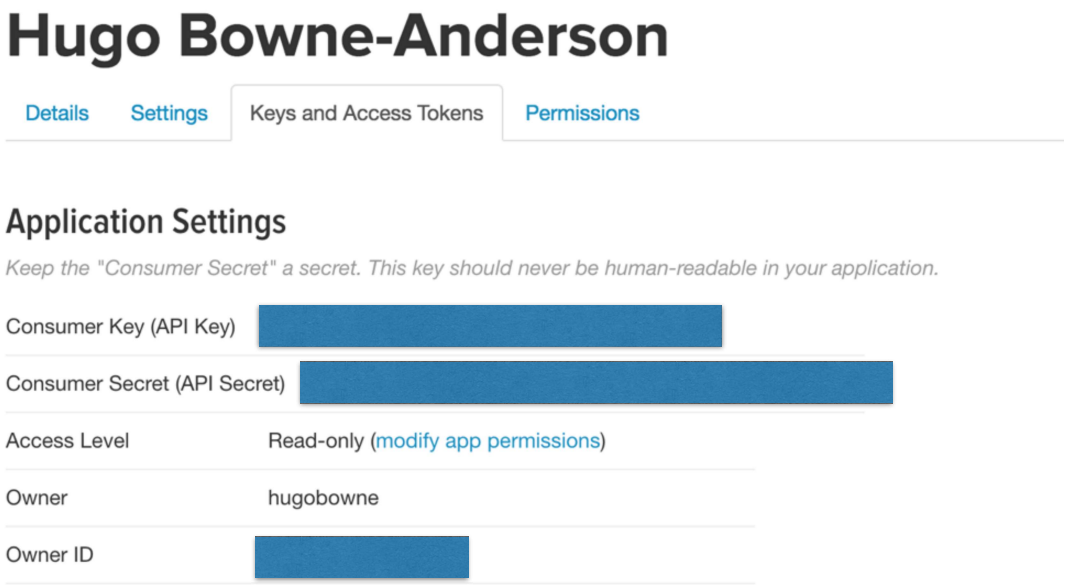
tweepy

Access the Twitter API





Access the Twitter API



Access the Twitter API

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	[redacted]
Access Token Secret	[redacted]
Access Level	Read-only
Owner	hugobowne
Owner ID	[redacted]

Twitter has a number of APIs

REST APIs

The REST APIs provide programmatic access to read and write Twitter data. Author a new Tweet, read author profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth; responses are available in JSON.

If your intention is to monitor or process Tweets in real-time, consider using the Streaming API instead.

Twitter has a number of APIs

The Streaming APIs

Overview

The Streaming APIs give developers low latency access to Twitter's global stream of Tweet data. A proper implementation of a streaming client will be pushed messages indicating Tweets and other events have occurred, without any of the overhead associated with polling a REST endpoint.

If your intention is to conduct singular searches, read user profile information, or post Tweets, consider using the REST APIs instead.

Twitter offers several streaming endpoints, each customized to certain use cases.

Public streams	Streams of the public data flowing through Twitter. Suitable for following specific users or topics, and data mining.
User streams	Single-user streams, containing roughly all of the data corresponding with a single user's view of Twitter.
Site streams	The multi-user version of user streams. Site streams are intended for servers which must connect to Twitter on behalf of many users.

Twitter has a number of APIs

GET statuses/sample

Returns a small random sample of all public statuses. The Tweets returned by the default access level are the same, so if two different clients connect to this endpoint, they will see the same Tweets.

Resource URL

`https://stream.twitter.com/1.1/statuses/sample.json`

Twitter has a number of APIs

Firehose

API Reference Documents

Streaming

GET statuses/firehose

This endpoint requires special permission to access.
Returns all public statuses. Few applications require this level of access. Creative use of a combination of other resources and various access levels can satisfy nearly every application use case.

Tweets are returned as JSONs

https://dev.twitter.com/overview/api/tweets

Field Guide

		The actual UTF-8 text of the status update. See twitter-text for details on what is currently considered valid characters. Example:
text	String	`"text":"Tweet Button, Follow Button, and Web Intents javascript now support SSL http:\/\/t.co\/9fbA0oYy ^TS"`

Tweets are returned as JSONs

https://dev.twitter.com/overview/api/tweets

Field Guide

lang	String	*Nullable.* When present, indicates a BCP 47 language identifier corresponding to the machine-detected language of the Tweet text, or "und" if no language could be detected. Example: `"lang": "en"`

Using Tweepy: Authentication handler

tw_auth.py

```
import tweepy, json

access_token = "..."
access_token_secret = "..."
consumer_key = "..."
consumer_secret = "..."

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

Tweepy: define stream listener class

st_class.py

```
class MyStreamListener(tweepy.StreamListener):
    def __init__(self, api=None):
        super(MyStreamListener, self).__init__()
        self.num_tweets = 0
        self.file = open("tweets.txt", "w")

    def on_status(self, status):
        tweet = status._json
        self.file.write(json.dumps(tweet) + '\n')
        tweet_list.append(status)
        self.num_tweets += 1
        if self.num_tweets < 100:
            return True
        else:
            return False
        self.file.close()
```

Using Tweepy: stream tweets!!

tweets.py

```
# Create Streaming object and authenticate
l = MyStreamListener()
stream = tweepy.Stream(auth, l)

# This line filters Twitter Streams to capture data by keywords:
stream.filter(track=['apples', 'oranges'])
```

IMPORTING DATA IN PYTHON II

Let's practice!

IMPORTING DATA IN PYTHON II

Final Thoughts

What you've learned:

- Importing text files and flat files
- Importing files in other formats
- Writing SQL queries
- Getting data from relational databases
- Pulling data from the web
- Pulling data from APIs

IMPORTING DATA IN PYTHON II

Congratulations!