



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?

UCI Machine Learning Repository  
Center for Machine Learning and Intelligent Systems

## Wine Quality Data Set

Download: [Data Folder](#), [Data Set Description](#)

**Abstract:** Two datasets are included, related to red and white vinho verde wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests (see [Cortez et al., 2009], [[Web Link](#)]).



Data Set Characteristics:	Multivariate	Number of Instances:	4898	Area:	Business
Attribute Characteristics:	Real	Number of Attributes:	12	Date Donated	2009-10-07
Associated Tasks:	Classification, Regression	Missing Values?	N/A	Number of Web Hits:	349131

- 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

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <!-- SEO -->
5     <meta charset="utf-8" />
6   <title>DataCamp: The Easy Way To Learn R & Data Science Online</title>
```



# BeautifulSoup

- Parse and extract structured data from HTML



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.

## [Beautiful Soup](#)

"A tremendous boon." -- Python411 Podcast

[ [Download](#) | [Documentation](#) | [Hall of Fame](#) | [Source](#) | [Discussion group](#) ]

If Beautiful Soup has saved you a lot of time and money, the best way to pay me back is to check out [Constellation Games](#), my sci-fi novel about alien video games.

You can [read the first two chapters for free](#), and the full novel starts at 5 USD. Thanks!

*If you have questions, send them to [the discussion group](#). If you find a bug, [file it](#).*

Beautiful Soup is a Python library designed for quick turnaround projects like screen-scraping. Three features make it powerful:



- 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="DCP-Description">
    <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">
    
    <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

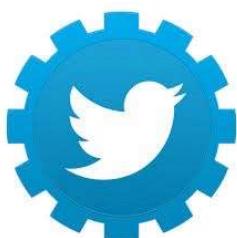


# APIs

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

## OMDb API

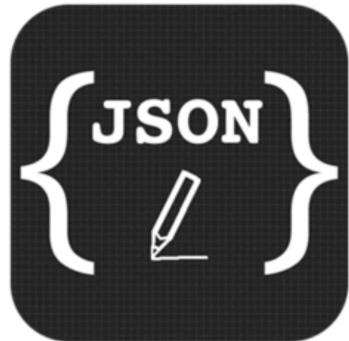
The Open Movie Database





# 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



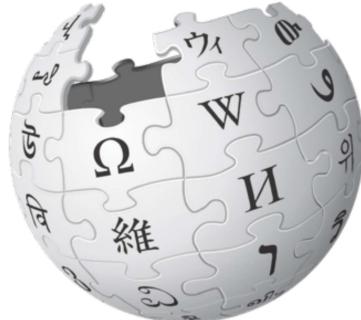
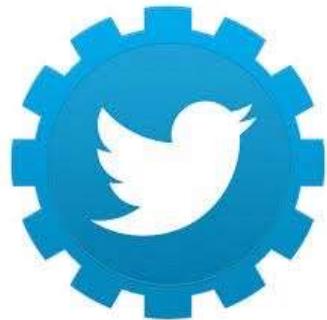
# Herein, you'll learn

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



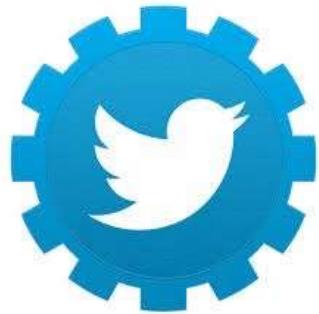
# What is an API?

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





# APIs are everywhere





# 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

## Usage

Send all data requests to:

<http://www.omdbapi.com/>

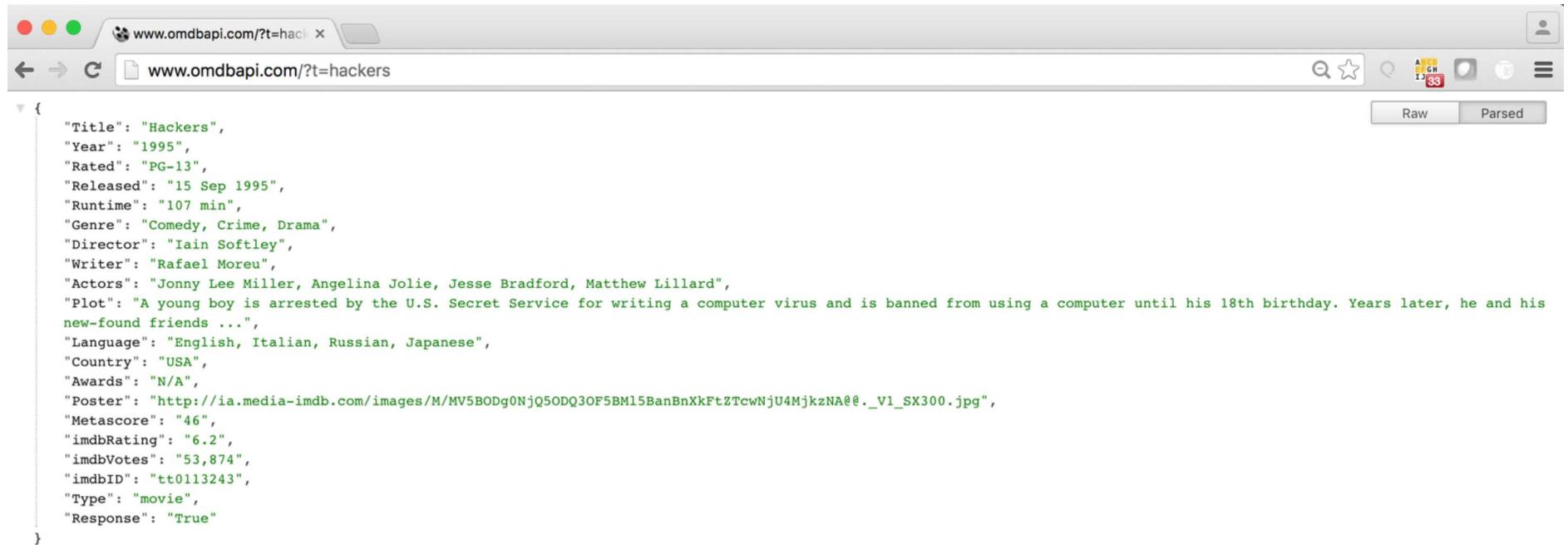
## Parameters

### 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!



A screenshot of a web browser window. The address bar shows "www.omdbapi.com/?t=hackers". The main content area displays a JSON object representing the movie "Hackers". The JSON structure includes fields like Title, Year, Rated, Released, Runtime, Genre, Director, Writer, Actors, Plot, Language, Country, Awards, Poster, Metascore, imdbRating, imdbVotes, imdbID, Type, and Response. The "Poster" field contains a URL to a movie poster image.

```
{
  "Title": "Hackers",
  "Year": "1995",
  "Rated": "PG-13",
  "Released": "15 Sep 1995",
  "Runtime": "107 min",
  "Genre": "Comedy, Crime, Drama",
  "Director": "Iain Softley",
  "Writer": "Rafael Moreu",
  "Actors": "Jonny Lee Miller, Angelina Jolie, Jesse Bradford, Matthew Lillard",
  "Plot": "A young boy is arrested by the U.S. Secret Service for writing a computer virus and is banned from using a computer until his 18th birthday. Years later, he and his new-found friends ...",
  "Language": "English, Italian, Russian, Japanese",
  "Country": "USA",
  "Awards": "N/A",
  "Poster": "http://ia.media-imdb.com/images/M/MV5BODg0NjQ5ODQ3OF5BMl5BanBnXkFtZTcwNjU4MjkzNA@@._V1_SX300.jpg",
  "Metascore": "46",
  "imdbRating": "6.2",
  "imdbVotes": "53,874",
  "imdbID": "tt0113243",
  "Type": "movie",
  "Response": "True"
}
```



IMPORTING DATA IN PYTHON II

**Let's practice!**



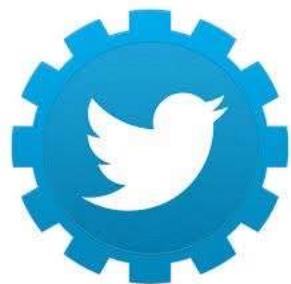
IMPORTING DATA IN PYTHON II

# The Twitter API and Authentication



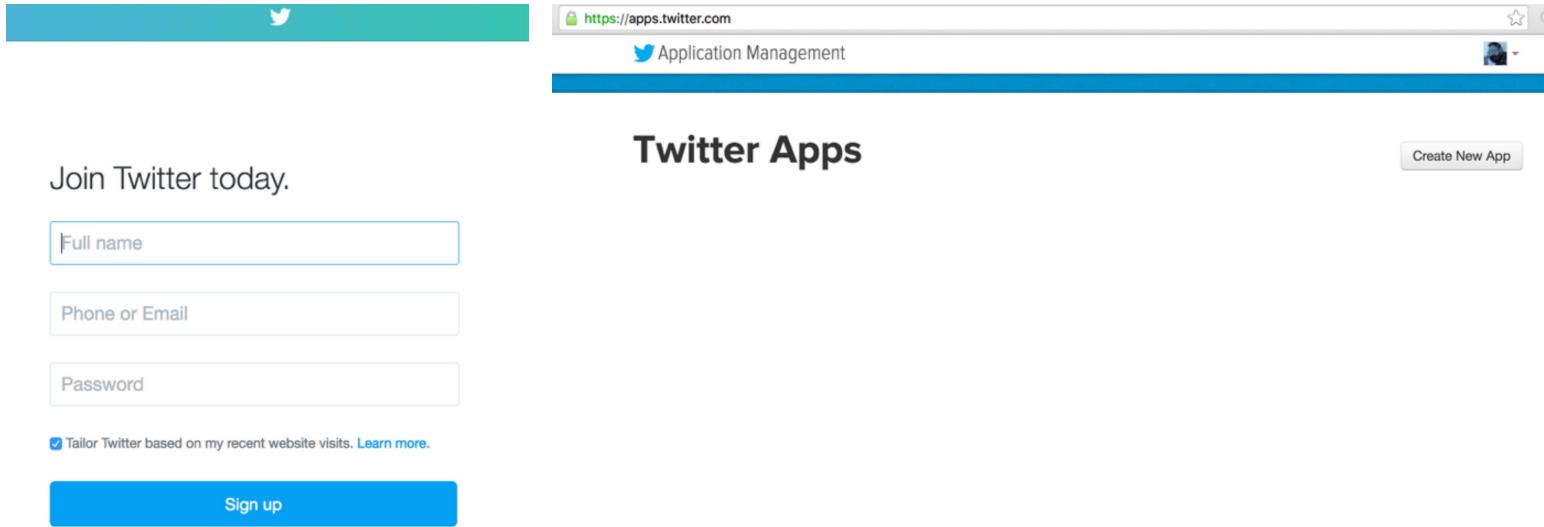
# Herein, you'll learn

- 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





# Access the Twitter API



The screenshot shows the Twitter Apps sign-up page. At the top, there's a navigation bar with the Twitter logo, the URL <https://apps.twitter.com>, and a search icon. Below the navigation is a blue header bar with the text "Twitter Apps". On the left, there's a sidebar with the text "Join Twitter today." and three input fields for "Full name", "Phone or Email", and "Password". To the right of these fields is a checkbox for "Tailor Twitter based on my recent website visits." followed by a "Sign up" button. In the bottom left corner, there's a note about agreeing to the Terms of Service and Privacy Policy.

Join Twitter today.

Full name

Phone or Email

Password

Tailor Twitter based on my recent website visits. [Learn more.](#)

Sign up

By signing up, you agree to the [Terms of Service](#) and [Privacy Policy](#), including [Cookie Use](#). Others will be able to find you by email or phone number when provided.



# Access the Twitter API

## Hugo Bowne-Anderson

Details    Settings    Keys and Access Tokens

Permissions

### Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)

[REDACTED]

Consumer Secret (API Secret)

[REDACTED]

Access Level

Read-only ([modify app permissions](#))

Owner

hugobowne

Owner ID

[REDACTED]



# 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



Access Token Secret



Access Level

Read-only

Owner

hugobowne

Owner ID





# 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.

<b>Public streams</b>	Streams of the public data flowing through Twitter. Suitable for following specific users or topics, and data mining.
<b>User streams</b>	Single-user streams, containing roughly all of the data corresponding with a single user's view of Twitter.
<b>Site streams</b>	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/9f  
bA0oYy ^TS"
```



# Tweets are returned as JSONs

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

## Field Guide

lang	String	<p><i>Nullable.</i> When present, indicates a <a href="#">BCP 47</a> language identifier corresponding to the machine-detected language of the Tweet text, or “und” if no language could be detected.</p>
------	--------	---

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!