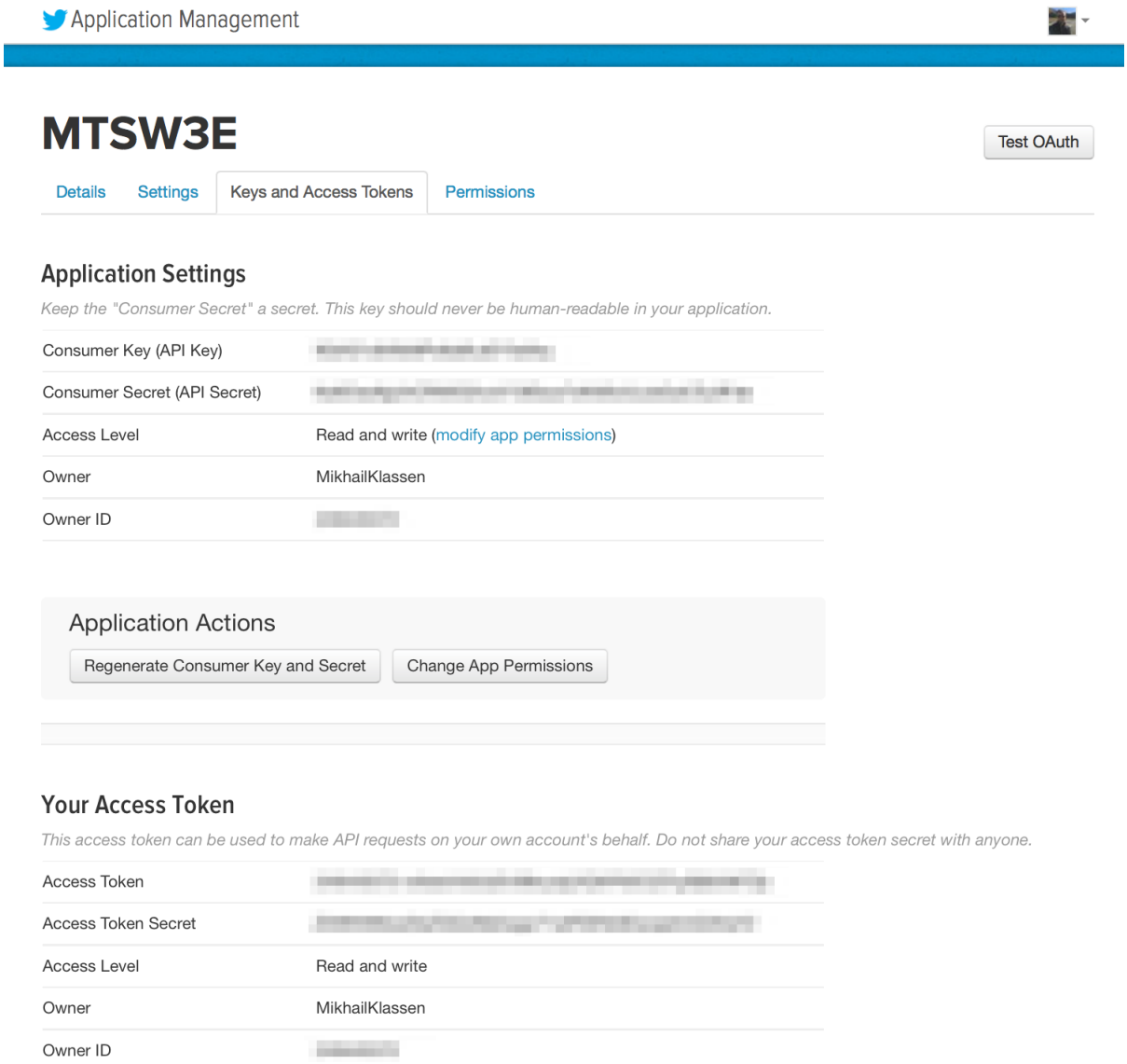
Mining Twitter

- Reference Chapter 1 of Mining the Social Web, 3rd Edition
- Safari books online library

Note Some of the code snippets in the book got depreciated Fixes has been done in this example

- Twitter implements OAuth 1.0A as its standard authentication mechanism, and in order to use it to make requests to Twitter's API, you'll need to go to https://developer.twitter.com/en/apps (https://developer.twitter.com/en/apps) and create a sample application.
- It is possible that Twitter no longer supports sandboxed applications and you may need to submit a request for permission to develop an app on Twitter.
- There are four primary identifiers you'll need to note for an OAuth 1.0A workflow: consumer key, consumer secret, access token, and access token secret.
- Note that you will need an ordinary Twitter account in order to login, create an app, and get these credentials.



- Please do not use the access token provided below those are mine.

- You need to use yours.

- pip install tweepy

Authorizing an application to access Twitter account data

In [1]:
```
#import libraries that you need
import tweepy
from tweepy import OAuthHandler, Stream
from credentials import *
# Go to https://developer.twitter.com/en/apps to create an app and get valu
# for these credentials, which you'll need to provide in place of these
# empty string values that are defined as placeholders.
# See https://developer.twitter.com/en/docs/basics/authentication/overview/
# for more information on Twitter's OAuth implementation.



auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)

twitter_api = tweepy.API(auth,  wait_on_rate_limit=True)

# Nothing to see by displaying twitter_api except that it's now a
# defined variable

print(twitter_api)
```

```
<tweepy.api.API object at 0x7ff368248760>
```

Very import links

- http://docs.tweepy.org/en/v3.5.0/api.html#API.trends_place (http://docs.tweepy.org/en/v3.5.0/api.html#API.trends_place)

Example 1-2. Retrieving trends

Exploring Trending Topics

- We are going to Twitter for the topics that are currently trending worldwide, but keep in mind that the API can easily be parameterized to constrain the topics to more specific locales if you feel inclined to try out some of the possibilities.
- The device for constraining queries is via Yahoo! GeoPlanet's Where On Earth (WOE) ID system, which is an API unto itself that aims to provide a way to map a unique identifier to any named place on Earth (or theoretically, even in a virtual world).
- We are going to collect a set of trends for both the entire world and just the United States.

In [4]:
```
# The Yahoo! Where On Earth ID for the entire world is 1.
# See https://developer.twitter.com/en/docs/trends/locations-with-trending-

WORLD_WOE_ID = 1
US_WOE_ID = 23424977

# Prefix ID with the underscore for query string parameterization.
# Without the underscore, the twitter package appends the ID value
# to the URL itself as a special case keyword argument.

world_trends = twitter_api.get_place_trends(id=WORLD_WOE_ID)
us_trends = twitter_api.get_place_trends(id=US_WOE_ID)

print(world_trends)
print("********************")
print(us_trends)
```

```
[{'trends': [{'name': '#BamBamxโหนกระแส', 'url': 'http://twitter.com/se
arch?q=%23BamBamx%E0%B9%82%E0%B8%AB%E0%B8%99%E0%B8%81%E0%B8%A3%E0%B8%B
0%E0%B9%81%E0%B8%AA', 'promoted_content': None, 'query': '%23BamBamx%E
0%B9%82%E0%B8%AB%E0%B8%99%E0%B8%81%E0%B8%A3%E0%B8%B0%E0%B9%81%E0%B8%A
A', 'tweet_volume': 374871}, {'name': '#AmbedkarJayanti', 'url': 'htt
p://twitter.com/search?q=%23AmbedkarJayanti', 'promoted_content': None,
'query': '%23AmbedkarJayanti', 'tweet_volume': 133782}, {'name': 'Ret e
L7', 'url': 'http://twitter.com/search?q=%22Ret+e+L7%22', 'promoted_con
tent': None, 'query': '%22Ret+e+L7%22', 'tweet_volume': 43453}, {'nam
e': 'Freddy Rincón', 'url': 'http://twitter.com/search?q=%22Freddy+Rin
c%C3%B3n%22', 'promoted_content': None, 'query': '%22Freddy+Rinc%C3%B3
n%22', 'tweet_volume': 64001}, {'name': 'Filipe Ret', 'url': 'http://tw
itter.com/search?q=%22Filipe+Ret%22', 'promoted_content': None, 'quer
y': '%22Filipe+Ret%22', 'tweet_volume': 28765}, {'name': 'Rwanda', 'ur
l': 'http://twitter.com/search?q=Rwanda', 'promoted_content': None, 'qu
ery': 'Rwanda', 'tweet_volume': 50105}, {'name': '#HalodocRamadan', 'ur
l': 'http://twitter.com/search?q=%23HalodocRamadan', 'promoted_conten
t': None, 'query': '%23HalodocRamadan', 'tweet_volume': None}, {'name':
'PA e Scooby', 'url': 'http://twitter.com/search?q=%22PA+e+Scooby%22',
```

In [5]:
```
for trend in world_trends[0]['trends']:
    print(trend['name'])
```

```
#BamBamxโหนกระแส
#AmbedkarJayanti
Ret e L7
Freddy Rincón
Filipe Ret
Rwanda
#HalodocRamadan
PA e Scooby
#Baisakhi
L7nnon
#インターネット老人音楽祭
インターネットのヲタク
じゅじゅステ
三浦涼介
磯丸水産
Daily Quordle 80
Pelicans
Chivas
HP 108MP 2jutaan
San Lorenzo
Prabowo Semakin Mantab
Dipercaya Karena Kerjanya
Bangun Indonesia Tangguh
ナナミン
ENDYMION
GUCCIの財布
Water Fight With Patrick
五条先生
luan city no bbb
Coloso
Matheus Fernandes
Foto Malam Terbaik
Indian Constitution
OUR BELOVED HARUTO
Leaño
महावीर जयंती
グリムグリモア
Herb Jones
あと2話
発売延期
Willax
Maundy Thursday
りょんくん
アイカツ
ARTHUR AGUIAR NO YOUTUBE
ユイナ先輩
9 pro
Never Goodbye
Happy New Year
syudouさん
```

In [6]:
```
for trend in us_trends[0]['trends']:
    print(trend['name'])
```

```
#TheKardashians
Pelicans
#SnowfallFX
#AEWDynamite
Spurs
Daily Quordle 80
#dreamOUT
#GoAvsGo
Herb Jones
Moskva
CJ McCollum
Pels
Black Sea
Louie
Zion
Murray
Hornets
Spider-Man 3
One Direction Heardle
Ingram
Keldon
Hawks
Vassell
Jose Alvarado
MacKinnon
Miles Bridges
Freddy Rincón
Vladdy
Chivas
TDKR
Tabata
john sterling
Great Khali
Willie Green
The Dark Knight Rises
Samoa Joe
Frei
makar
Warchant
Khloe
The Avs
Suzuki
Neptune
Lonnie
poeltl
Satnam Singh
NYCFC
Abbott Elementary
Jethro Tull
Popovich
```

- The pattern for using the twitter module is simple and predictable:

- instantiate the Twitter class with an object chain corresponding to a base URL and then invoke methods on the object that correspond to URL contexts.
- For example, twitter_api.trends_place(id=WORLD_WOE_ID) initiates an HTTP call to GET https://api.twitter.com/1.1/trends/available.json (https://api.twitter.com/1.1/trends/available.json)
- Example "url": "http://where.yahooapis.com/v1/place/2458410" (http://where.yahooapis.com/v1/place/2458410"),
- "woeid": 2458410
  - https://api.twitter.com/1.1/trends/place.json?id=1 (https://api.twitter.com/1.1/trends/place.json?id=1).
- Note the URL mapping to the object chain that's constructed with the twitter package to make the request and how query string parameters are passed in as keyword arguments.
- To use the twitter package for arbitrary API requests, you generally construct the request in that kind of straightforward manner.
- *** add more explanation **

Example 1-3. Displaying API responses as pretty-printed JSON

In [7]:
```
#Displaying API responses as pretty-printed JSON
#Please refer to the following link to understand the json.dumps feature mo
#https://www.w3schools.com/python/python_json.asp

import json

print(json.dumps(world_trends, indent=1))
print()
print(json.dumps(us_trends, indent=1))
```


```
[
 {
  "trends": [
   {
    "name": "#BamBamx\u0e42\u0e2b\u0e19\u0e01\u0e23\u0e30\u0e41\u0e2a",
    "url": "http://twitter.com/search?q=%23BamBamx%E0%B9%82%E0%B8%AB%E
0%B8%99%E0%B8%81%E0%B8%A3%E0%B8%B0%E0%B9%81%E0%B8%AA",
    "promoted_content": null,
    "query": "%23BamBamx%E0%B9%82%E0%B8%AB%E0%B8%99%E0%B8%81%E0%B8%A3%E
0%B8%B0%E0%B9%81%E0%B8%AA",
    "tweet_volume": 374871
   },
   {
    "name": "#AmbedkarJayanti",
    "url": "http://twitter.com/search?q=%23AmbedkarJayanti",
    "promoted_content": null,
    "query": "%23AmbedkarJayanti",
    "tweet_volume": 133782
   },
```


Example 1-4

- Demonstrates how to use a Python list comprehension to parse out the names of the trending topics from the results that were previously queried, cast those lists to sets, and compute the

setwise intersection to reveal the common items between them.

- Please refere to the following link to understand list comprehension more
- https://docs.python.org/2/tutorial/datastructures.html#list-comprehensions (https://docs.python.org/2/tutorial/datastructures.html#list-comprehensions)

In [8]:
```
world_trends_set = set([trend['name']
                        for trend in world_trends[0]['trends']])

us_trends_set = set([trend['name']
                     for trend in us_trends[0]['trends']])

common_trends = world_trends_set.intersection(us_trends_set)

print(common_trends)
```

```
{'Freddy Rincón', 'Pelicans', 'Daily Quordle 80', 'Chivas', 'Herb Jones'}
```

In [ ]: