**Download Twitter Data**

**Use the strings from your Twitter app webpage to populate these four variables:**

consumer\_key = "IQ03DPOdXz95N3rTm2iMNE8va"

consumer\_secret = "0qGHOXVSX1D1ffP7BfpIxqFalLfgVIqpecXQy9SrUVCGkJ8hmo"

access\_token = "867193453159096320-6oUq9riQW8UBa6nD3davJ0SUe9MvZrZ"

access\_secret = "5zMwq2DVhxBnvjabM5SU2Imkoei3AE6UtdeOQ0tzR9eNU"

**Now we use the configured authentication information to connect to Twitter's API:**

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_secret)

api = tweepy.API(auth)

print("Connected to Twitter!")

**Now that we are connected to Twitter, let's do a brief check that we can read tweets by pulling the first few tweets from our own timeline (or the account associated with your Twitter app) and printing them:**

# Get tweets from our timeline

public\_tweets = api.home\_timeline()

# print the first five authors and tweet texts

for tweet in public\_tweets[:5]:

print (tweet.author.screen\_name, tweet.author.name, "said:", tweet.text)

**Now that we're connected, we can search Twitter for specific keywords with relative ease just like you were using Twitter's search box. While this search only goes back 7 days and/or 1,500 tweets (whichever is less), it can be powerful if an event you want to track just started.**

**Note that you might have to deal with paging if you get lots of data. Twitter will only return you one page of up to 100 tweets at a time.**

# Our search string

queryString = "earthquake"

# Perform the search

matchingTweets = api.search(queryString)

print ("Searched for:", queryString)

print ("Number found:", len(matchingTweets))

# For each tweet that matches our query, print the author and text

print ("\nTweets:")

for tweet in matchingTweets:

print (tweet.author.screen\_name, tweet.text)

**As mentioned, Twitter serves results in pages. To get all results, we can use Tweepy's Cursor implementation, which handles this iteration through pages for us in the background.**

# Lets find only media or links about earthquakes

queryString = "earthquake (filter:media OR filter:links)"

# How many tweets should we fetch? Upper limit is 1,500

maxToReturn = 100

# Perform the search, and for each tweet that matches our query,

# print the author and text

print ("\nTweets:")

for status in tweepy.Cursor(api.search, q=queryString).items(maxToReturn):

print (status.author.screen\_name, status.text)

**Up to this point, all of our work has been retrospective. An event has occurred, and we want to see how Twitter responded over some period of time.**

**To follow an event in real time, Twitter and Tweepy support Twitter streaming. Streaming is a bit complicated, but it essentially lets of track a set of keywords, places, or users.**

**To keep things simple, I will provide a simple class and show methods for printing the first few tweets. Larger solutions exist specifically for handling Twitter streaming.**

**You could take this code though and easily extend it by writing data to a file rather than the console. I've marked where that code could be inserted.**

# First, we need to create our own listener for the stream that will stop after a few tweets

class LocalStreamListener(tweepy.StreamListener):

# Max number of tweets

maxTweetCount = 10

# Set current counter

def \_\_init\_\_(self):

tweepy.StreamListener.\_\_init\_\_(self)

self.currentTweetCount = 0

# For writing out to a file

self.filePtr = None

# Create a log file

def set\_log\_file(self, newFile):

if ( self.filePtr ):

self.filePtr.close()

self.filePtr = newFile

# Close log file

def close\_log\_file(self):

if ( self.filePtr ):

self.filePtr.close()

# Pass data up to parent then check if we should stop

def on\_data(self, data):

print (self.currentTweetCount)

tweepy.StreamListener.on\_data(self, data)

if ( self.currentTweetCount >= self.maxTweetCount ):

return False

# Increment the number of statuses we've seen

def on\_status(self, status):

self.currentTweetCount += 1

# Could write this status to a file instead of to the console

print (status.text)

# If we have specified a file, write to it

if ( self.filePtr ):

self.filePtr.write("%s\n" % status.\_json)

# Error handling below here

def on\_exception(self, exc):

print (exc)

def on\_limit(self, track):

"""Called when a limitation notice arrives"""

print ("Limit", track)

return

def on\_error(self, status\_code):

"""Called when a non-200 status code is returned"""

print ("Error:", status\_code)

return False

def on\_timeout(self):

"""Called when stream connection times out"""

print ("Timeout")

return

def on\_disconnect(self, notice):

"""Called when twitter sends a disconnect notice"""

print ("Disconnect:", notice)

return

def on\_warning(self, notice):

print ("Warning:", notice)

"""Called when a disconnection warning message arrives"""

**Now we set up the stream using the listener above:**

Stream based on keywords:

listener = LocalStreamListener()

localStream = tweepy.Stream(api.auth, listener)

localStream.filter(track=['earthquake', 'disaster'])

Stream based on users:

listener = LocalStreamListener()

localStream = tweepy.Stream(api.auth, listener)

# List of screen names to track

screenNames = ['bbcbreaking', 'CNews', 'bbc', 'nytimes']

# Twitter stream uses user IDs instead of names so we must convert

userIds = []

for sn in screenNames:

user = api.get\_user(sn)

userIds.append(user.id\_str)

localStream.filter(follow=userIds)

Stream based on location:

listener = LocalStreamListener()

localStream = tweepy.Stream(api.auth, listener)

# Specify coordinates for a bounding box around area of interest. In this case, we use San Francisco

swCornerLat = 36.8

swCornerLon = -122.75

neCornerLat = 37.8

neCornerLon = -121.75

boxArray = [swCornerLon, swCornerLat, neCornerLon, neCornerLat]

localStream.filter(locations=boxArray)

# Say we want to write these tweets to a file

listener.set\_log\_file(codecs.open("tweet\_log.json", "w", "utf8"))

# Close the log file

listener.close\_log\_file()