Chapter 10

Building a Twitter Bot

Summary: Twitter is a valuable publishing tool and can be used to automate the dissemination of interesting stories. We'll use Python to build your own.

Skills you will learn:

1. Creating an official Twitter app as a Twitter developer
2. Building an app that will automatically connect to Twitter and publish tweets on your behalf

The first thing you'll need is a text editor. See the tutorial **Choosing a Code Editor**. This tutorial relies on a basic understanding of the Python programming language. Python is introduced in Chapter 9 of *The Data Journalist*, and its tutorials.

You will need a Twitter account to which you are authorized to post content.

Twitter lets you use an API (application programming interface) to post to your account without having to go through the login screen every time.

You just need an API key and an account token, which are the equivalent of a username and password in the programming world.

Let's take care of the account token first.

Go to https://dev.twitter.com/oauth/overview/application-owner-access-tokens
Once you're logged in, go to https://apps.twitter.com and register yourself a new app.

Click “Keys and Access Tokens” to get yourself an API key, making note of both your public and private keys. Then scroll down that same page and get yourself an access token.

Note: **Do not share access tokens with anyone. They are a password to your account and should be treated as private information.**

Once you have your keys and access tokens, it's time to let Python work its magic.

First: There’s one bug that we need to fix that addresses an encryption issue that is still in some versions of Python. Run this in your console to make sure everything is updated properly.

```
$ pip install pyopenssl ndg-httpsclient pyasn1
  $ pip install --upgrade ndg-httpsclient
```

Python is good at many things and hooking into web-based APIs is definitely one of them. We could hook into the Twitter API without the help of an external package, but it’s easier to use what’s already available and what someone else has already debugged.

Tweepy is that package. In your Mac terminal or Windows Command window, enter the following at a prompt:

```
pip install tweepy
```

Keep an eye out for errors. Depending on your machine, the version of Python that you have running on it, and the other packages that are already installed, it may fail. If it does fail, try this:

```
pip install tweepy --upgrade --ignore-installed six
```

You may also wish to create a virtual environment using Python’s virutalenv package. Once you create a virtual environment, the packages you install in it will not have any impact on packages installed in another virtual environment. Creating virtual environments is beyond the scope of this book, but you can read the virtualenv documentation at https://virtualenv.pypa.io/en/stable/

Install Tweepy and it will handle much of the request formatting and the setup for posting methods so we can get on with the fun part. The fun part starts now. Open a new file in your preferred text editor.

First, we import Tweepy and the encryption components of a library called urllib3 so that we can make secure calls to Twitter over an encrypted connection:

```
import tweepy
import urllib3.contrib.pyopenssl
urllib3.contrib.pyopenssl.inject_into_urllib3()
```

So we can now add our API keys to our code, storing them in their own variables so we can keep using them later.

```
# Establish Twitter authorization.
# Need to sign up for API key at apps.twitter.com

TWEEPY_CONSUMER_KEY = #'YOUR_API_KEY'
TWEEPY_CONSUMER_SECRET = #'YOUR_SECRET_API_KEY'
TWEEPY_ACCESS_TOKEN = #'YOUR_ACCESS_TOKEN'
TWEEPY_ACCESS_TOKEN_SECRET = #'YOUR_SECRET_ACCESS_TOKEN'
```

I'm using all caps in this example. You can use whatever your heart desires. These are just variable names. Copy and paste your API and access tokens into the relevant spots inside quotation marks so they can be used later.

The first thing we have to do is initialize the app by connecting it to your Twitter account with our API key and our access token. This will make sure that we actually have all the permissions in place to post to this account.

```
# Setup authorizations with tweepy
auth1 = tweepy.auth.OAuthHandler(TWEEPY_CONSUMER_KEY,
TWEEPY_CONSUMER_SECRET)
```

```
auth1.set_access_token(TWEEPY_ACCESS_TOKEN,
TWEEPY_ACCESS_TOKEN_SECRET)
api = tweepy.API(auth1)
```

Again, I'm using all caps here. You can use whatever your heart desires. These are just variable names.

Now we can set up a function that will actually use that authorization. We have the variable "api" available to us now, which is a fully authorized and ready-to-go Twitter application just waiting for our commands. You can see the full list of available commands on the Tweepy website here: http://tweepy.readthedocs.io/en/v3.5.0/api.html

```
def tweetit(statusUpdate):
        api.update_status(status=statusupdate)
        print statusupdate
```

Notice the "def" keyword, the colon and the tabs that make this a function.

Now we just have give ourselves a status update and call the function:

```
statusupdate = "This is a test Tweet from our new Python
bot!"
tweetit(statusUpdate)
```

Now go check your Twitter account and you should see that tweet posted in place.

There are a slew of ways to control how this works. Look at the python event scheduler at https://docs.python.org/2/library/sched.html if you want to post at regular times. You can also look at the time.sleep() method to simply wait a given amount of time between your posts.

You can use Python lists [ ] to set up messages to Tweet or you can even draw from a spreadsheet with the very powerful CSV module available at https://docs.python.org/2/library/csv.html.