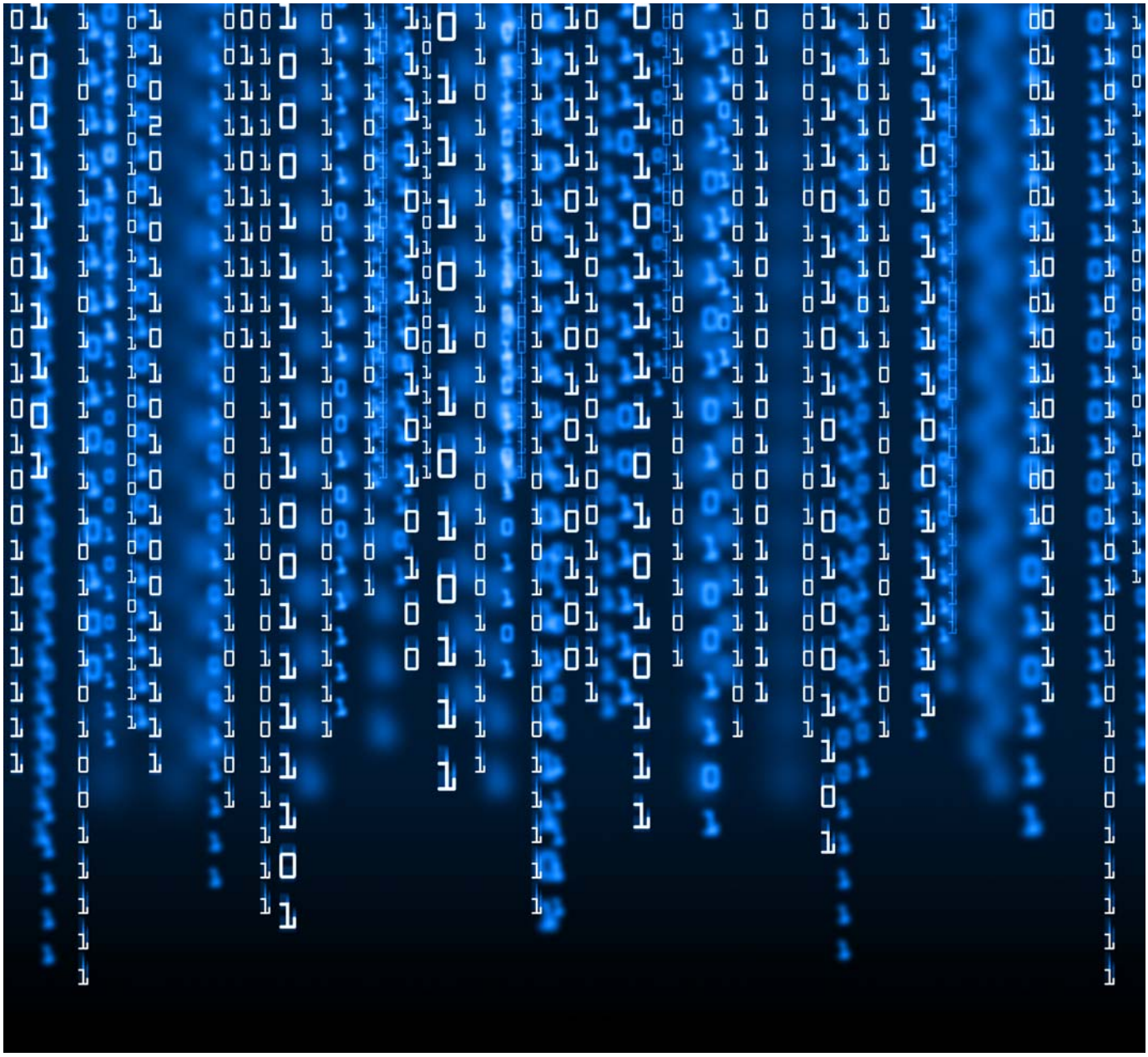
MAY 20, 2015

Python Code Tutorial

APRIL 21, 2014 BY GREGORY SAXTON 10 COMMENTS



I often get requests to explain how I obtained the data I used in a particular piece of academic research. I am always happy to share my code along with my data (and frankly, I think academics who are unwilling to share should be forced to take remedial Kindergarten). The problem is, many of those who would like to use the code don't know where to start. There are too many new steps involved for the process to be accessible. So, I'll try to walk you through the basic steps here through periodic tutorials.

To start, Python is a great tool for grabbing data from the Web. Generally speaking, you'll get your data by either accessing an API (Application Programming Interface) or by 'scraping' the data off a webpage. The easiest scenario is when a site makes available an API. Twitter is such a site. Accordingly, as an introductory example I'll walk you through the basic steps of using Python to

access the Twitter API, read and manipulate the data returned, and save the output.

In any given project I will run a number of different scripts to grab all of the relevant data. We'll sta with a simple example. This script is designed to grab the information on a set of Twitter users. First, as stated above, what we're doing to get the data is tapping into the Twitter API. For our purposes, think of the Twitter API as a set of routines Twitter has set up for allowing us to access specific chunks of data. I use Python for this, given its many benefits, though any programming language will work. If you are really uninterested in programming and have more limited data needs, you can use NodeXL (if you're on a Windows machine) or other services for gathering the data. If you do go the Python route, I highly recommend you install Anaconda Python 2.7 — it's free, it works on Mac and PC, and includes most of the add-on packages necessary for scientific computing. In short, you pick a programming language and learn some of it and then develop cod that will extract and process the data for you. Even though you can start with my code as a base, is still useful to understand the basics, so I highly recommend doing some of the many excellent tutorials now available online for learning how to use and run Python. A great place to start is Codeacademy.

Contents [hide]

Accessing the Twitter API

Almost all of my Twitter code grabs data from the Twitter API. The first step is to determine which part of the Twitter API you'll need to access to get the type of data you want — there are different API methods for accessing information on tweets, retweets, users, following relationships, etc. Th code we're using here plugs into the users/lookup part of the Twitter API, which allows for the bul downloading of Twitter user information. You can see a description of this part of the API here, along with definitions for the variables returned. Here is a list of the most useful of the variables returned by the API for each user (modified descriptions taken from the Twitter website):

FIELD	DESCRIPTION

FIELD	DESCRIPTION
created_at	The UTC datetime that the user account was created on Twitter.
description	The user-defined UTF-8 string describing their account.
entities	Entities which have been parsed out of the url or description fields defined by the user.
favourites_count	The number of tweets this user has favorited in the account's lifetime. British spelling used in the field name for historical reasons.
followers_count	The number of followers this account currently has. We can also get a list of these followers by using different parts of the API.
friends_count	The number of users this account is following (AKA their "followings"). We can also get a list of these friends using other API methods.
id	The integer representation of the unique identifier for this User. This number is greater than 53 bits and some programming languages may have difficulty/silent defects in interpreting it. Using a signed 64 bit integer for storing this identifier is safe. Use id_str for fetching the identifier to stay on the safe side. See Twitter IDs, JSON and Snowflake.
id_str	The string representation of the unique identifier for this User. Implementations should use this rather than the large, possibly un-consumable integer in id.
lang	The BCP 47 code for the user's self-declared user interface language.
listed_count	The number of public lists that this user is a member of.
location	The user-defined location for this account's profile. Not necessarily a

FIELD	DESCRIPTION
	location nor parseable.
name	The name of the user, as they've defined it. Not necessarily a person's name.
screen_name	The screen name, handle, or alias that this user identifies themselves with. screen_names are unique but subject to change. Use id_str as a user identifier whenever possible. Typically a maximum of 15 characters long, but some historical accounts may exist with longer names.
statuses_count	The number of tweets (including retweets) issued by the user to date.
time_zone	A string describing the Time Zone this user declares themselves within
url	A URL provided by the user in association with their profile.
withheld_in_countries	When present, indicates a textual representation of the two-letter country codes this user is withheld from. See New Withheld Content Fields in API Responses.
withheld_scope	When present, indicates whether the content being withheld is the "status" or a "user." See New Withheld Content Fields in API Responses.

Second, beginning in 2013 Twitter made it more difficult to access the API. Now OAuth authentication is needed for almost everything. This means you need to go on Twitter and create an ‘app.’ You won’t actually use the app for anything — you just need the password and authentication code. You can create your app here. For more detailed instructions on creating the app take a look at this presentation.

Third, as a Python 'wrapper' around the Twitter API I use Twython. This is a package that is an add-on to Python. You will need to install this as well as simplejson (for parsing the JSON data th is returned by the API). Assuming you installed Anaconda Python, the simplest way is to use pip. On a Mac or Linux machine, you would simply open the Terminal and type *pip install Twython* and *pip install simplejson.*

The above steps can be a bit of a pain depending on your familiarity with UNIX, but you'll only have to do them once. It may take you a while. But once they're all set up you won't need to do it again.

Understanding the Code

At the end of this post I'll show the entire script. For now, I'll go over it in sections. The first line in the code is the shebang — you'll find this in all Python code.

```
1 #!/usr/bin/env python
```

Lines 3 – 10 contain the docstring — also a Python convention. This is a multi-line comment that describes the code. For single-line comments, use the # symbol at the start of the line.

```
1 """
2
3 Use Twitter API to grab user information from list of organizations;
4 export text file
5
6 Uses Twython module to access Twitter API
7
8 """
```

Next we'll import several Python packages needed to run the code.

```
1 import sys
2 import string
3 import simplejson
4 from twython import Twython
```

In lines 18-22 we will create day, month, and year variables to be used for naming the output file.

```
1  import datetime
2  now = datetime.datetime.now()
3  day=int(now.day)
4  month=int(now.month)
5  year=int(now.year)
```

Modify the Code

There are two areas you'll need to modify. First, you'll need to add your OAuth tokens to lines 26-30.

```
1  t = Twython(app_key='APP_KEY', #REPLACE 'APP_KEY' WITH YOUR APP KEY, ETC
2      app_secret='APP_SECRET',
3      oauth_token='OAUTH_TOKEN',
4      oauth_token_secret='OAUTH_TOKEN_SECRET')
```

Second, you'll need to modify lines 32-35 with the ids from your set of Twitter users. If you don't have user_ids for these, you can use screen_names and change line 39 to 'screen_name = ids'

```
1  ids = "4816,9715012,13023422, 13393052,  14226882,  14235041, 14292458,
2      15029174, 15474846, 15634728, 15689319, 15782399, 15946841, 16116519
3      16315120, 16566133, 16686673, 16801671, 41900627, 42645839, 42731742
4      48073289, 48827616, 49702654, 50310311, 50361094,"
```

Line 39 is where we actually access the API and grab the data. If you've read over the description of users/lookup API, you know that this method allows you to grab user information on up to 100 Twitter IDs with each API call.

```
1  users = t.lookup_user(user_id = ids)
```

Understanding JSON

Now, a key step to this is understanding the data that are returned by the API. As is increasingly common with Web data, this API call returns data in JSON format. Behind the scenes, Python ha grabbed this JSON file, which has data on the 32 Twitter users listed above in the variable *ids*. Each user is an *object* in the JSON file; objects are delimited by left and right curly braces, as shown here for one of the 32 users:


```
{
  "id": 171314974,
  "id_str": "171314974",
  "name": "Global Partnership ",
  "screen_name": "GPforEducation",
  "location": "Washington, DC",
  "url": "http://www.globalpartnership.org",
  "description": "We work with partners in nearly 60 low-income countri
  "protected": false,
  "followers_count": 23712,
  "friends_count": 335,
  "listed_count": 391,
  "created_at": "Tue Jul 27 02:17:01 +0000 2010",
  "favourites_count": 231,
  "utc_offset": -18000,
  "time_zone": "Eastern Time (US & Canada)",
  "geo_enabled": true,
  "verified": true,
  "statuses_count": 4394,
  "lang": "en",
  "contributors_enabled": false,
  "is_translator": false,
  "profile_background_color": "127CB8",
  "profile_background_image_url": "http://a0.twimg.com/profile_backgrou
  "profile_background_image_url_https": "https://si0.twimg.com/profile_
  "profile_background_tile": false,
  "profile_image_url": "http://pbs.twimg.com/profile_images/2479080709/
  "profile_image_url_https": "https://pbs.twimg.com/profile_images/2479
  "profile_banner_url": "https://pbs.twimg.com/profile_banners/17131497
  "profile_link_color": "000000",
  "profile_sidebar_border_color": "FFFFFF",
  "profile_sidebar_fill_color": "FFFFFF",
  "profile_text_color": "4A494A",
  "profile_use_background_image": true,
  "default_profile": false,
  "default_profile_image": false,
  "following": null,
  "follow_request_sent": null,
  "notifications": null
},
```


JSON output can get messy, so it's useful to bookmark a JSON viewer for formatting JSON outpu

What you're seeing above is 38 different variables returned by the API — one for each row — an arranged in *key: value* (or *variable: value*) pairs. For instance, the value for the *screen_name* variable for this user is *GPforEducation*. Now, we do not always want to use all of these variables so what we'll do is pick and label those that are most useful for us.

So, we first initialize the output file, putting in the day/month/year in the file name, which is useful you're regularly downloading this user information:

```
1  outfn = "twitter_user_data_%i.%i.%i.txt" % (now.month, now.day, now.year
```

We then create a variable with the names for the variables (columns) we'd like to include in our output file, open the output file, and write the header row:

```
1  fields = "id screen_name name created_at url followers_count friends_cou
2      favourites_count listed_count \
3      contributors_enabled description protected location lang expanded_ur
4
5  outfp = open(outfn, "w")
6  outfp.write(string.join(fields, "\t") + "\n")  # header
```

Recall that in line 39 we grabbed the user information on the 32 users and assigned these data to the variable *users*. The final block of code in lines 55-90 loops over each of these IDs (each one a different object in the JSON file), creates the relevant variables, and writes a new row of output. Here's the first few rows:

```
1  for entry in users:
2      #CREATE EMPTY DICTIONARY
3      r = {}
4      for f in fields:
5          r[f] = ""
6      #ASSIGN VALUE OF 'ID' FIELD IN JSON TO 'ID' FIELD IN OUR DICTIONARY
7      r['id'] = entry['id']
8      #SAME WITH 'SCREEN_NAME' HERE, AND FOR REST OF THE VARIABLES
9      r['screen_name'] = entry['screen_name']
```

If you compare this code to the raw JSON output shown earlier, what we're doing here is creating an empty Python dictionary, which we'll call 'r', to hold our data for each user, creating variables

called *id* and *screen_name*, and assigning the values held in the *entry['id']* and *entry['screen_name']* elements of the JSON output to those two respective variables. This is all placed inside a Python for loop — we could have called 'entry' anything so long as we're consistent.

Now let's put the whole thing together. To recap, what this entire script does is to loop over each of the Twitter accounts in the *ids* variable — and for each one it will grab its profile information and add that to a row of the output file (a text file that can be imported into Excel, etc.). The filename given to the output file varies according to the date. Now you can download this script, modify the lines noted above, and be on your way to downloading your own Twitter data!

```
#!/usr/bin/env python

"""

Use Twitter API to grab user information from list of organizations;
export text file

Uses Twython module to access Twitter API

"""

import sys
import string
import simplejson
from twython import Twython

#WE WILL USE THE VARIABLES DAY, MONTH, AND YEAR FOR OUR OUTPUT FILE NAME
import datetime
now = datetime.datetime.now()
day=int(now.day)
month=int(now.month)
year=int(now.year)


#FOR OAUTH AUTHENTICATION -- NEEDED TO ACCESS THE TWITTER API
t = Twython(app_key='APP_KEY', #REPLACE 'APP_KEY' WITH YOUR APP KEY, ETC., IN THE NEXT 4 LINES
    app_secret='APP_SECRET',
    oauth_token='OAUTH_TOKEN',
    oauth_token_secret='OAUTH_TOKEN_SECRET')

#REPLACE WITH YOUR LIST OF TWITTER USER IDS
ids = "4816,9715012,13023422, 13393052,  14226882,  14235041, 14292458, 14335586, 14730894,\
    15029174, 15474846, 15634728, 15689319, 15782399, 15946841, 16116519, 16148677, 16223542,\
    16315120, 16566133, 16686673, 16801671, 41900627, 42645839, 42731742, 44157002, 44988185,\
    48073289, 48827616, 49702654, 50310311, 50361094,"

#ACCESS THE LOOKUP_USER METHOD OF THE TWITTER API -- GRAB INFO ON UP TO 100 IDS WITH EACH API CALL
#THE VARIABLE USERS IS A JSON FILE WITH DATA ON THE 32 TWITTER USERS LISTED ABOVE
users = t.lookup_user(user_id = ids)
```

```
40
41 #NAME OUR OUTPUT FILE - %i WILL BE REPLACED BY CURRENT MONTH, DAY, AND YEAR
42 outfn = "twitter_user_data_%i.%i.%i.txt" % (now.month, now.day, now.year)
43
44 #NAMES FOR HEADER ROW IN OUTPUT FILE
45 fields = "id screen_name name created_at url followers_count friends_count statuses_count \
46     favourites_count listed_count \
47     contributors_enabled description protected location lang expanded_url".split()
48
49 #INITIALIZE OUTPUT FILE AND WRITE HEADER ROW
50 outfp = open(outfn, "w")
51 outfp.write(string.join(fields, "\t") + "\n")  # header
52
53 #THE VARIABLE 'USERS' CONTAINS INFORMATION OF THE 32 TWITTER USER IDS LISTED ABOVE
54 #THIS BLOCK WILL LOOP OVER EACH OF THESE IDS, CREATE VARIABLES, AND OUTPUT TO FILE
55 for entry in users:
56     #CREATE EMPTY DICTIONARY
57     r = {}
58     for f in fields:
59         r[f] = ""
60     #ASSIGN VALUE OF 'ID' FIELD IN JSON TO 'ID' FIELD IN OUR DICTIONARY
61     r['id'] = entry['id']
62     #SAME WITH 'SCREEN_NAME' HERE, AND FOR REST OF THE VARIABLES
63     r['screen_name'] = entry['screen_name']
64     r['name'] = entry['name']
65     r['created_at'] = entry['created_at']
66     r['url'] = entry['url']
67     r['followers_count'] = entry['followers_count']
68     r['friends_count'] = entry['friends_count']
69     r['statuses_count'] = entry['statuses_count']
70     r['favourites_count'] = entry['favourites_count']
71     r['listed_count'] = entry['listed_count']
72     r['contributors_enabled'] = entry['contributors_enabled']
73     r['description'] = entry['description']
74     r['protected'] = entry['protected']
75     r['location'] = entry['location']
76     r['lang'] = entry['lang']
77     #NOT EVERY ID WILL HAVE A 'URL' KEY, SO CHECK FOR ITS EXISTENCE WITH IF CLAUSE
78     if 'url' in entry['entities']:
79         r['expanded_url'] = entry['entities']['url']['urls'][0]['expanded_url']
80     else:
81         r['expanded_url'] = ''
82     print r
83     #CREATE EMPTY LIST
84     lst = []
85     #ADD DATA FOR EACH VARIABLE
86     for f in fields:
87         lst.append(unicode(r[f]).replace("\/", "/"))
88     #WRITE ROW WITH DATA IN LIST
89     outfp.write(string.join(lst, "\t").encode("utf-8") + "\n")
90
91 outfp.close()
```

twitter-user-data.py hosted with ♥ by **GitHub** **view raw**

Share

FILED UNDER: FEATURED, PYTHON, TWITTER
TAGGED WITH: PYTHON, TWITTER

COMMENTS

amy says
May 12, 2014 at 11:38 pm

Excellent script!, but it returns an error:

lst.append(unicode(r[f]).replace(“/”, “/”))

^

IndentationError: unexpected indent

Reply

Gregory Saxton says
May 13, 2014 at 12:01 am

Hi Amy — Thanks! A line of code was missing right before that line. Sorry about that. I’ve updated it and it should work now.

Reply

Matt Kushin says

June 5, 2014 at 10:57 pm

Greg. Thanks so much for posting this. I am brand new to Python and a bit confused about what the API “wrapper” does. I found Tweepy which seems to deal with Twitter differently than the Twython… though I’m not sure I quite understand. Can you explain what the wrapper is? Thanks so much!

Reply

Gregory Saxton says
June 10, 2014 at 11:45 pm

Hi Matt, you’re welcome! To understand this, first look at line 39 in the above code:

users = t.lookup_user(user_id = ids)

This is the line that makes the call to the Twitter API. The specific API we’re tapping into here is the users/lookup API, which you can read about here:

http://dev.twitter.com/docs/api/1/get/users/lookup

Why we refer to Twython as a “wrapper” around the API is that you are spared from directly making the OAuth2 “handshake” with Twitter and then separately making a potentially complicated “GET” call to access the data, e.g., https://api.twitter.com/1/users/lookup.json?screen_name=twitterapi,twitter&include_entities=true

In practice, by using the Twython wrapper around the API in this fashion you can typically save a few steps. Moreover, Twitter can changes its requirements for how you access the API, and the developers of a package such as Twython will take care of this for you in its updates.

Hope this helps!

Reply

JJ says
October 18, 2014 at 7:31 pm

Hi, many thanks for your code! It really helped to better understand how to get data out of Twitter. i was wondering if you would know how to collect tweets from the beginning of twitter up to today. I need to collect about 600 tweets and would like to have the whole history. I have been trying to do this for a while, but nothing seems to work! James.

Reply

Gregory Saxton says
October 18, 2014 at 7:42 pm

Hi James — you're welcome! Are you looking to download 600 tweets from a single Twitter user? If so, the Twitter API will let you get the last 3,200 tweets sent by any user. If that's what you need, I can point you in the right direction. And if that's not what you're looking for let me know and I'll try to help.

Reply

Putt says
October 26, 2014 at 1:07 pm

Hey Greg,

I am a total beginner to Python, and I'm doing an academic project on political campaign. I need to have full set of tweets from 6 different accounts during Feb – June this year and also full sets of tweets from a certain hashtags. Can I apply this to that projects?

Any answer and help will be very much appreciated!

Reply

Gregory Saxton says
October 26, 2014 at 10:33 pm

Hi — I'm happy to help. You can definitely do what you're suggesting! To help, I just created a brief post that puts all of my tutorials in context and helps point you in the right direction.

The tutorial above that you're commenting on is intended to serve as an introduction to how to access the Twitter API and then read the JSON data that is returned. This gets you the account-level data for a Twitter user (such as when the account was created), but not the actual tweets. That is a more difficult process, but not a huge leap once you've made it through all of the above steps. In short, take a look at the new blog post, start working through the steps listed, and you'll be on your way to getting the data you want.

Cheers,

Greg

Reply

Vikrant Hole says
January 31, 2015 at 4:57 pm

Hi, I am ME student and is doing academic project on finding events in football match using tweets. Can you suggest me from where I can download tweets of at least one match.

Reply

Gregory Saxton says
February 2, 2015 at 2:39 am

Hi Vikrant, a couple of questions. First, am I correct in assuming you're interested in downloading tweets that relate to a *conversation* around a particular football match? If so, you would like want to identify those hashtags that are used. Alternatively, you could simply download tweets that "mention" the two teams playing in the match. In either case, you'll want to follow the steps outlined in this post. Best of luck in your research!

Reply

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Comment

POST COMMENT

Search this website…

RECENT POSTS

iPython Notebook and PANDAS Cookbook

Random Python Recipes

Python Tutorials for Downloading Twitter Data

How to Download Tweets with a Specific Hashtag

Setting up Your Computer to Use My Python Code for Downloading Twitter Data

RELATED POSTS

Your First Steps with Python: Part I -- Running your First Python Code

Python: Where to Start?

FEATURED POSTS

Python Tutorials for Downloading Twitter Data

I often get requests to explain how I obtained the data I … [Read More...]

Python: Where to Start?

For the complete beginner, getting up and running in a new … [Read More...]

Does Twitter Matter?

Twitter is not the Gutenberg Press. The 'Big Data' … [Read More...]

ARCHIVES

April 2015

November 2014

October 2014

September 2014

May 2014

April 2014

E-MAIL SIGN-UP

Every time I post something new to my blog, receive it free by email.

First Name

Last Name

E-Mail Address

GO

No spam.

CONTACT INFORMATION

Gregory D. Saxton
331 Baldy Hall, Dept. of Communication
University at Buffalo, SUNY
Buffalo, NY 14260
gdsaxton@buffalo.edu

RECENT POSTS

TAG CLOUD

academia academic research arnova14 conference hashtags ica iPython PANDAS PhD_studies python replication research social media socialmedia tutorial Twitter