Twitter and While Loops

April 18 2016

Today

- Using GeCoords.py to extract geo locations to write a KML file
- Getting Twitter posts using *tweepy*
- While loops
- More powerful functions to get Twitter posts
  - Using while loops

Twitter

- You know what it is
  - 140-character text, plus:
    - Favorite/retweet count
    - Geographical coordinates
  - Each user has a timeline
    - *You can query a user's timeline*
  - All tweets are accessible
    - *You can search for specific terms*

We're doing both

Let's install tweepy

- External Python module
  - Help us using the Tweeter API
    - (API: Application Programming Interface)

- Not in the default installation
  - Install with "`sudo easy_install tweepy`" or "pip install tweepy" in your terminal (Mac OS terminal)
    - A little more cumbersome on Windows
    - Feel free to request help from us installing packages

Let’s install tweepy

hmendes — bash — 80×24

```
Honey Badger's MacBook Pro $ sudo easy_install tweepy
```

Authentication Keys

- We need to “authenticate” on Twitter…
  - Tell them who we are
  - We get “keys” to use API functionality
    - Do *not* share your application keys with anyone else!
  - Let’s do it together

Authentication Keys

- Go to https://apps.twitter.com
- Click on: Create New App

Authentication Keys

Application Details

Name *

CS931app

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

My tweeter app for CS931

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

http://cs.brown.edu/courses/csci0931/

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about yo
qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization scre
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth_callback URL or
regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

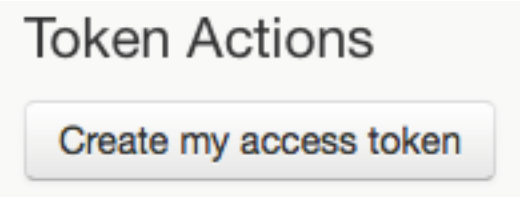
Authentication Keys

- Look for “Application Settings”, “Consumer Key”, and then click on:

  – “manage keys and access tokens”

Authentication Keys

- Click on “Create My Access Token” button



Authentication Keys

- You now have this:

Consumer Key (API Key)	xshlkPT	...
Consumer Secret (API Secret)	xKYAdm	...
Access Token	5783	...
Access Token Secret	MBZ	...

Download `tw1.py`

Copy/paste it there

Run the program!

Keys will be quite large, and differ from this first letters

Do *NOT* share this with anyone! (It encodes your password.)

Really

- Do NOT share your authentication info
- Or you'll expose your Twitter account!

tw1.py (Searching)

```
import tweepy
import codecs

# Consumer keys and access tokens, used for authentication
consumer_key = ''
consumer_secret = ''
access_token = ''
access_token_secret = ''

# Gives you back an api object to interact with
def initialize():
  auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
  auth.set_access_token(access_token, access_token_secret)

  api = tweepy.API(auth)

  return api
```

tw1.py (Searching)

```
# Main program
api = initialize()

# Gives back a list of tweet objects
# q: query to perform
# count: number of tweets to collect (maximum 100)
# geocode: area to search (lat,lng,radius)
results = api.search(q="exam", count=100,
                             geocode="41.8262,-71.4032,10mi")

for result in results:
  print('Text: ', ascii(result.text))
  print('User: ', ascii(result.user.screen_name))
  print('# Retweets: ', result.retweet_count)
  print('# Favorites: ', result.favorite_count)
```

tw1.py (Searching)

```
# Main program
api = initialize()

# Gives back a list of tweet objects
# q: query to perform
# count: number of tweets to collect (maximum 100)
# geocode: area to search (lat,lng,radius)
results = api.search(q="exam", count=100,
                     geocode="41.8262,-71.4032,10mi")
```

Empty string if you don't care

```
for result in results:
  print('Text: ', ascii(result.text))
  print('User: ', ascii(result.user.screen_name))
  print('# Retweets: ', result.retweet_count)
  print('# Favorites: ', result.favorite_count
```

`tw2.py` (Querying Timelines)

In our webpage, you also find `tw2.py`, which *queries the timeline* for a *specific user*.

Copy/paste your authentication information there, and it's ready to use

tw2.py (Querying Timelines)

```
# Main program
api = initialize()

# Gives back a list of tweet objects
# screen_name: user name of the queried timeline
# count: number of tweets to collect (maximum 100)
results = api.user_timeline(screen_name='BrownUniversity',
                                                    count=100)

for result in results:
  print('Text: ', ascii(result.text))
  print('User: ', ascii(result.user.screen_name))
  print('# Retweets: ', result.retweet_count)
  print('# Favorites: ', result.favorite_count
```

Only 100?

- Well, 100 per query
  - You can collect more than 100, going back a week
    - Limitation imposed by Twitter
- If you want 850 tweets, you need to tell Twitter this:
    - Hey Twitter, give me 100 tweets (one query)
    - Hey Titter, give me 100 more, older than those I got before
    - …
    - Hey Titter, give me 100 more, older than those I got before
    - Hey Titter, give me 50 more, older than those I got before

How do we do this?

- Let’s learn a new Python trick

While Loops

- A *Python* language construct
  - You can use in *any* Python program
- It's a loop construct
  - Repeats the body of the loop while the specified *condition* evaluates to *True*

```
def example1():
    text = ''
    while text != "stop":
        text = input('give a new value for text: ')
        print('text is now: ', text)
```

While Loops

Consider this example:

```
def example2():
  myList = [10, 20, 30, 40, 50, 60] # len is 6

  pos = 0
  while myList[pos] < 35 and pos < len(myList):
    print(myList[pos])
    pos = pos + 1
```

While Loops

- First time: `pos = 0`
  - `myList[pos] < 35` (10 < 35)
  - `pos < len(myList)` (0 < 6)

```
def example2():
  myList = [10, 20, 30, 40, 50, 60] # len is 6

  pos = 0
  while myList[pos] < 35 and pos < len(myList):
    print(myList[pos])
    pos = pos + 1
```

While Loops

- First time: `pos = 0`
  - `myList[pos] < 35` (10 < 35)
  - `pos < len(myList)` (0 < 6)
- True
  - Enter the loop
    - print(`myList[pos]`) (10)
    -

```
def example2():
  myList = [10, 20, 30, 40, 50, 60] # len is 6

  pos = 0
  while myList[pos] < 35 and pos < len(myList):
    print(myList[pos])
    pos = pos + 1
```

While Loops

- First time: `pos = 0`
  - `myList[pos] < 35` (10 < 35)
  - `pos < len(myList)` (0 < 6)
- True
  - Enter the loop
    - print(`myList[pos]`) (10)
    - makes `pos = 1`

```
def example2():
  myList = [10, 20, 30, 40, 50, 60] # len is 6

  pos = 0
  while myList[pos] < 35 and pos < len(myList):
    print(myList[pos])
    pos = pos + 1
```

While Loops

- Second time: `pos = 1`
  - `myList[pos] < 35` (20 < 35)
  - `pos < len(myList)` (1 < 6)

```
def example2():
  myList = [10, 20, 30, 40, 50, 60] # len is 6

  pos = 0
  while myList[pos] < 35 and pos < len(myList):
    print(myList[pos])
    pos = pos + 1
```

While Loops

- Second time: `pos = 1`
  - `myList[pos] < 35` (20 < 35)
  - `pos < len(myList)` (1 < 6)
- True
  - Enter the loop
    - print(`myList[pos]`) (20)
    -

```
def example2():
  myList = [10, 20, 30, 40, 50, 60] # len is 6

  pos = 0
  while myList[pos] < 35 and pos < len(myList):
    print(myList[pos])
    pos = pos + 1
```

While Loops

- Second time: `pos = 1`
  - `myList[pos] < 35` (20 < 35)
  - `pos < len(myList)` (1 < 6)
- True
  - Enter the loop
    - print(`myList[pos]`) (20)
    - makes `pos = 2`

```
def example2():
  myList = [10, 20, 30, 40, 50, 60] # len is 6

  pos = 0
  while myList[pos] < 35 and pos < len(myList):
    print(myList[pos])
    pos = pos + 1
```

While Loops

- Third time: `pos = 2`
  - `myList[pos] < 35` (30 < 35)
  - `pos < len(myList)` (2 < 6)

```
def example2():
  myList = [10, 20, 30, 40, 50, 60] # len is 6

  pos = 0
  while myList[pos] < 35 and pos < len(myList):
    print(myList[pos])
    pos = pos + 1
```

While Loops

- Third time: `pos = 2`
  - `myList[pos] < 35` (30 < 35)
  - `pos < len(myList)` (2 < 6)
- True
  - Enter the loop
    - print(`myList[pos]`)(30)
    -

```
def example2():
  myList = [10, 20, 30, 40, 50, 60] # len is 6

  pos = 0
  while myList[pos] < 35 and pos < len(myList):
    print(myList[pos])
    pos = pos + 1
```

While Loops

- Third time: `pos = 2`
  - `myList[pos] < 35` (30 < 35)
  - `pos < len(myList)` (2 < 6)
- True
  - Enter the loop
    - print(`myList[pos]`)(30)
    - makes `pos = 3`

```
def example2():
  myList = [10, 20, 30, 40, 50, 60] # len is 6

  pos = 0
  while myList[pos] < 35 and pos < len(myList):
    print(myList[pos])
    pos = pos + 1
```

While Loops

- Forth time: `pos = 3`
  - `myList[pos] < 35` (40 < 35) ----- false
  - `pos < len(myList)` (2 < 6) ----- still true

```
def example2():
  myList = [10, 20, 30, 40, 50, 60] # len is 6

  pos = 0
  while myList[pos] < 35 and pos < len(myList):
    print(myList[pos])
    pos = pos + 1
```

While Loops

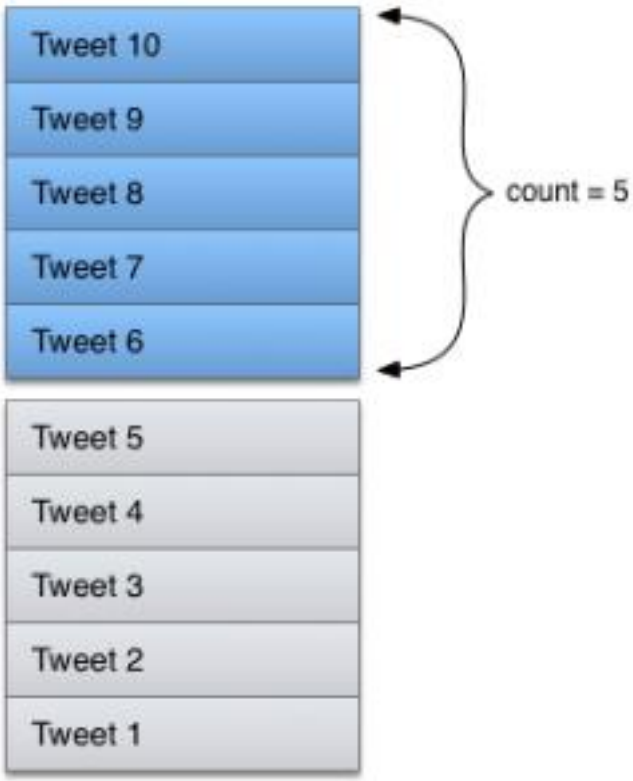
- Forth time: `pos = 3`
  - `myList[pos] < 35` (40 < 35) ----- false
  - `pos < len(myList)` (2 < 6) ----- still true
- False
  - DO NOT enter the loop

```
def example2():
  myList = [10, 20, 30, 40, 50, 60] # len is 6

  pos = 0
  while myList[pos] < 35 and pos < len(myList):
    print(myList[pos])
    pos = pos + 1
```

Using while loops to get more tweets

If you ask for 5 tweets (either from a timeline or based on a search term), Twitter will give you the 5 most recent ones

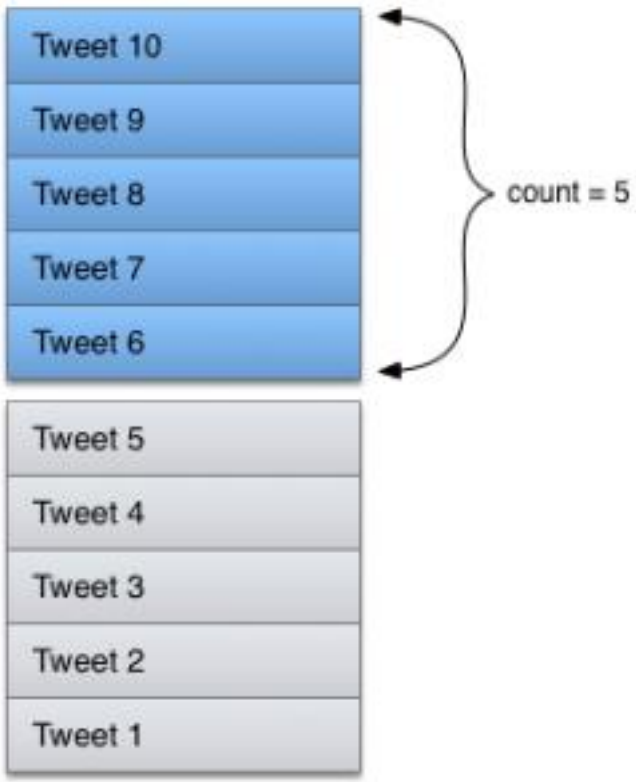


Newer tweets have bigger IDs

Using while loops to get more tweets

If you want 10 tweets…

Say you may only ask for 5 at a time…



1) Get your new 5 tweets

2) Look at them

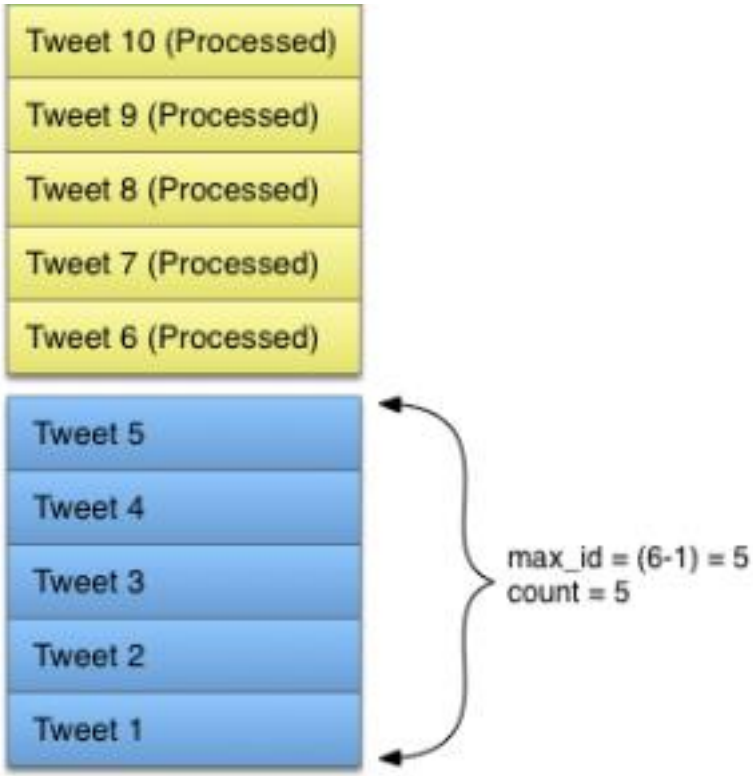
3) Get their minimum ID **(6)**

4) Ask again for tweets with ID 5 or less

Using while loops to get more tweets

If you want 10 tweets…

Now ask again for 5 tweets more



1) Get your new 5 tweets

2) Look at them

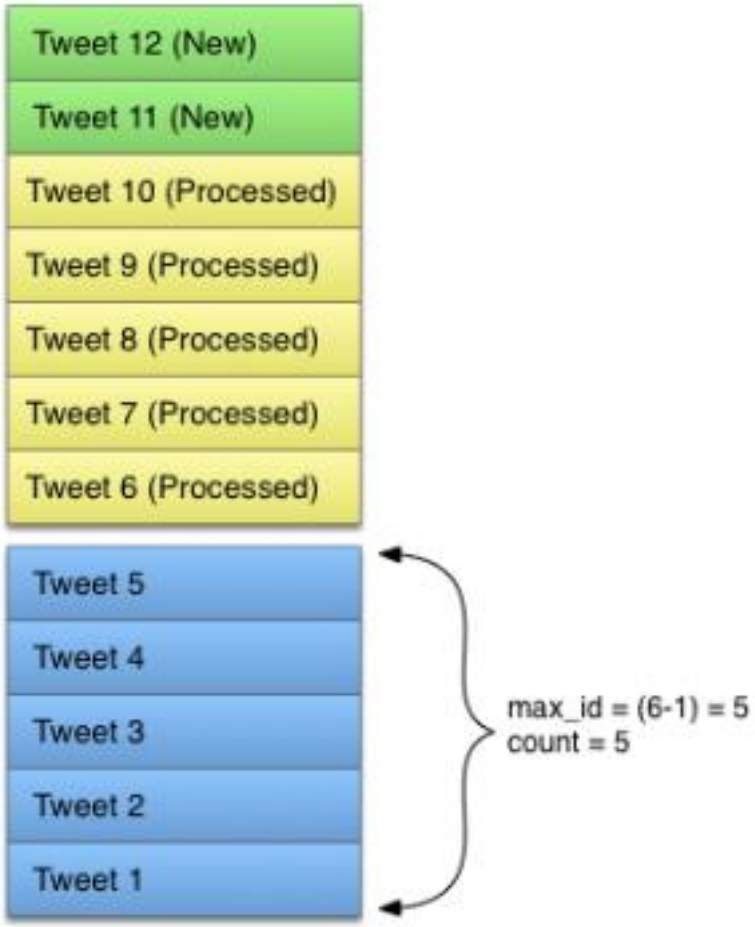
3) Get their minimum ID **(6)**

4) Ask again for tweets with ID 5 or less

Using while loops to get more tweets

If you want 10 tweets…

Now ask again for 5 tweets more



1) Get your new 5 tweets

2) Look at them

3) Get their minimum ID **(6)**

4) Ask again for tweets with ID 5 or less

Of course, new tweets may show up

`twcol.py`
(`tw1.py` and `tw2.py` on while loops)

- Ask for 100 tweets
- *If* we got nothing, return nothing
- *Else*, get the minimum ID
- *While* we are *not done*
  - Ask for 100 more tweets (older than the minimum ID)
  - Declare it *done* if we
    - (i) get nothing (all appropriate tweets have been consumed) *or*
    - (ii) the total exceeds the amount requested
  - Make sure to update the minimum ID

- Generates CSV files with the obtained tweets

`twcol.py`
(`tw1.py` and `tw2.py` on while loops)

- Check the file in our course webpage
  - You have *Python-maturity* to understand it ☺