Data Collection

API (Application Programming Interface)

> An application programming interface (API) is a set of subroutine definitions, communication protocols, and tools for building software. In general terms, it is a set of clearly defined methods of communication among various components. A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer.

(1) Create a Twitter Developer Account and an App to Use the Twitter API

> The Twitter API is simply a set of URLs that take parameters. These URLs let you access many features of Twitter, such as posting a tweet or finding tweets that contain a word, etc.

> In order to collect tweets from Twitter API, we need to create a Twitter account and create an app (could be empty) to get credentials (consumer keys, consumer secret, access token, and access token secret) from Twitter.
>
> Twitter Application Management: https://apps.twitter.com/

Step 1: Developer Portal Overview

Step 2: Create an App

> To create an app, please click "Create App" button, and input the name of your app.

Step 3: Check Your Keys and Tokens

> Now you just created your twitter app. Please check your keys and tokens, and copy and paste them in your Notes or MS Word app.

Step 4: Check Your App on the Menu Bar

> You can access your app settings anytime. Plese click your app on the menu bar.

Step 5: Check Keys and Tokens through the Menu Bar

> Please click "Keys and tokens".

Step 6: Access the API Key & Secret

> If you may not save the API Key and Secret in Step 3, you can access them here. Please check your API key and API key secret, and copy and paste them in your Notes or MS Word app.

Step 7: Acess the Access Token & Secret

> Please check your access token and access token secret, and copy and paste them in your Notes or MS Word app.

(2) Create JSON File of Your Credentials

> Once you get the four credentials, create your own credential JSON file in the current working directory.

In [ ]:
```
credential = {"API_KEY": "VqR9HJemANJ893BekWXTxXWT7",
    "API_SECRET": "gfAdxfdXj3eP9xH1XzwkKlcs7tQ6x2q68yzQRP4gOzuDu3t0iF",
    "ACCESS_TOKEN": "1244356645393707024-YgjM3PKRbwc65oFmmgdyEFNSkhoHqF",
    "ACCESS_TOKEN_SECRET": "LJLnwmw65iMuZnR2ULsU5RQq8Edfu4f1oHUilW3If9TTe"
}

credential
```

In [ ]:
```
import json
outfile = open('twitter_credentials.json', 'w')
json.dump(credential, outfile)
outfile.close()
```

(3) Install twython Package

> Twitter API can be used in various languages. Within Python, there are many Python packages for Twitter API: https://developer.twitter.com/en/docs/developer-utilities/twitter-libraries
>
> We will use Twython: https://github.com/ryanmcgrath/twython
>
> However, this is not installed in Anaconda. So we need to install it using pip (a Python package manager): https://packaging.python.org/tutorials/installing-packages/#use-pip-for-installing
>
> You can also use conda (a package manager for Anaconda): https://conda.io/docs/user-guide/tasks/manage-pkgs.html

In [ ]:
```
!pip install twython
```

(4) Create a Twython Instance with Your Credentials from JSON file

> Now we will read credentials from the JSON file.
>
> Let's create a Twython instance with our credentials.

In [ ]:
```
from twython import Twython
import sys
```

In [ ]:
```
infile = open('twitter_credentials.json', 'r')
credentials = json.load(infile)
infile.close()

credentials
```

In [ ]:
```
# Create your own app to get consumer key and secret
API_KEY = credentials['API_KEY']
API_SECRET = credentials['API_SECRET']
```

In [ ]:
```
help(Twython)
```

In [ ]:
```
twitter = Twython(API_KEY, API_SECRET)
```

In [ ]:
```
dir(twitter)
```

(5) Let's Use the Twitter Standard Search API

> We will first use the Twitter search API:
> https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets

In [ ]:
```
res = twitter.search(q = "COVID")
```

In [ ]:
```
res
```

In [ ]:
```
type(res)
```

In [ ]:
```
len(res)
```

In [ ]:
```
res.keys()
```

In [ ]:
```
res['statuses']
```

In [ ]:
```
type(res['statuses'])
```

In [ ]:
```
len(res['statuses'])
```

In [ ]:
```
res['search_metadata']
```

In [ ]:
```
type(res['search_metadata'])
```

In [ ]:
```
res['statuses'][0]
```

In [ ]:
```
type(res['statuses'][0])
```

In [ ]:
```
res['statuses'][0].keys()
```

In [ ]:
```
res['statuses'][0]['text']
```

In [ ]:
```
for t in res['statuses']:
    print(t['text'])
```

In [ ]:
```
for item in res['statuses']:
    print(item['lang'])
```

In [ ]:
```
for t in res['statuses']:
    if t['lang'] == 'en':
        print(t['text'])
```

In [ ]:
```
outfile = open('tweet_search_COVID.json', 'w')
json.dump(res, outfile)
outfile.close()
```

In [ ]:
```
# To search more than 15 tweets (but up to 100 tweets), use the count parameter.

res = twitter.search(q = "COVID", count = 100)

for t in res['statuses']:
```

```
    if t['lang'] == 'en':
        print(t['text'])
```

In [ ]:
```
len(res['statuses'])
```

(6) Putting Things Together (Example #1)

> Please define a function 'call_twitter_search_api( )' that takes inputs of a keyword and a number of tweets, 1) searches the number of tweets using the keyword in Twitter, 2) creates a list of searched tweets, 3) prints a user name and his/her tweet, 4) prints a number of tweets. and 5) save the list of searched tweets to 'tweet_search_keyword_number.json'.

In [ ]:
```
def call_twitter_search_api(keyword, number):

    from twython import Twython
    import sys
    import json

    infile = open('twitter_credentials.json', 'r')
    credentials = json.load(infile)
    infile.close()

    twitter = Twython(credentials['API_KEY'], credentials['API_SECRET'])

    tweets = []

    for t in twitter.search(q = '{}'.format(keyword), count = '{}'.format(number))['sta
        user = t['user']['screen_name']
        text = t['text']
        print('{}\t{}'.format(user, text))
        tweets.append(t)
    print(len(tweets), 'tweets')

    outfile = open('tweet_search_{}_{}.json'.format(keyword, number), 'w')
    json.dump(tweets, outfile)
    outfile.close()
```

In [ ]:
```
call_twitter_search_api('CLEMSON', 20)
```

In [ ]:
```
infile = open('tweet_search_CLEMSON_20.json')
data = json.load(infile)
infile.close()
```

In [ ]:
```
print(data[0])
```

In [ ]:
```
from pprint import pprint
```

In [ ]:
```
pprint(data[0])
```

(7) Let's Use the Twitter Streaming API

> Now, we will use the Streaming API:
> https://twython.readthedocs.io/en/latest/usage/streaming_api.html

In [ ]:
```
from twython import TwythonStreamer
import sys
import json
```

In [ ]:
```
infile = open('twitter_credentials.json')
credentials = json.load(infile)
infile.close()

API_KEY = credentials['API_KEY']
API_SECRET = credentials['API_SECRET']
ACCESS_TOKEN = credentials['ACCESS_TOKEN']
ACCESS_TOKEN_SECRET = credentials['ACCESS_TOKEN_SECRET']
```

In [ ]:
```
dir(TwythonStreamer)
```

In [ ]:
```
help(TwythonStreamer)
```

In [ ]:
```
stream = TwythonStreamer(API_KEY, API_SECRET, ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
```

In [ ]:
```
dir(stream)
```

In [ ]:
```
help(stream.statuses)
```

> Now, we can filter realtime tweets with the Streaming API with following parameters:
>
> https://developer.twitter.com/en/docs/twitter-api/v1/tweets/filter-realtime/api-reference/post-statuses-filter

In [ ]:
```
stream.statuses.filter(track = 'covid')
```

> You may not have any outcome and your kernel would run forever.
>
> It's time to set your own streamer.

In [ ]:

```
from twython import TwythonStreamer

# We are inheriting from TwythonStreamer class

class MyStream(TwythonStreamer):
    '''our own subclass of TwythonStreamer'''

    # Overriding
    def on_success(self, data):
        if 'text' in data:
            print(data['text'])

    # Overriding
    def on_error(self, status_code, data):
        print(status_code)
```

In [ ]:

```
import json

infile = open('twitter_credentials.json')
credentials = json.load(infile)
infile.close()

API_KEY = credentials['API_KEY']
API_SECRET = credentials['API_SECRET']
ACCESS_TOKEN = credentials['ACCESS_TOKEN']
ACCESS_TOKEN_SECRET = credentials['ACCESS_TOKEN_SECRET']

stream = MyStream(API_KEY, API_SECRET, ACCESS_TOKEN, ACCESS_TOKEN_SECRET)

stream.statuses.filter(track = 'covid')
```

> You may have a lot of outcomes and your kernel would run forever.
>
> Now, it's time to think about how to stop it.
>
> What if you want to stop the streamer after getting 200 tweets?

In [ ]:

```
from twython import TwythonStreamer

# Set a container to append 200 tweets
container = []

# We are inheriting from TwythonStreamer class
class MyStream(TwythonStreamer):
    '''our own subclass of TwythonStreamer'''

    # Overriding
    def on_success(self, data):
        if 'text' in data:
            print(data['text'])
            # append tweets to the container
            container.append(data)

        # If you get enough tweets (e.g. 200), disconnect API.
```

```
        if len(container) == 200:
            self.disconnect()

    # Overriding
    def on_error(self, status_code, data):
        print(status_code)
```

In [ ]:

```
import json

infile = open('twitter_credentials.json')
credentials = json.load(infile)
infile.close()

API_KEY = credentials['API_KEY']
API_SECRET = credentials['API_SECRET']
ACCESS_TOKEN = credentials['ACCESS_TOKEN']
ACCESS_TOKEN_SECRET = credentials['ACCESS_TOKEN_SECRET']

stream = MyStream(API_KEY, API_SECRET, ACCESS_TOKEN, ACCESS_TOKEN_SECRET)

stream.statuses.filter(track = 'covid')
```

In [ ]:

```
len(container)
```

In [ ]:

```
container
```

> What if you want to save the 200 tweets into a JSON file when you search them?

In [ ]:

```
from twython import TwythonStreamer
import json

# Set a container to append 200 tweets
container = []

# We are inheriting from TwythonStreamer class
class MyStream(TwythonStreamer):
    '''our own subclass of TwythonStreamer'''

    # Overriding
    def on_success(self, data):
        if 'text' in data:
            print(data['text'])
            # Append tweets to the container
            container.append(data)

        # If you get enough tweets (e.g. 200), store it into JSON file and disconnect Al
        if len(container) == 200:
            self.store_json()
            self.disconnect()

    # Overriding
    def on_error(self, status_code, data):
        print(status_code)
```

```
    # New method to store tweets into JSON file
    def store_json(self):
        infile = open('tweet_stream.json', 'w')
        json.dump(container, infile)
        infile.close()
```

In [ ]:

```
import json

infile = open('twitter_credentials.json')
credentials = json.load(infile)
infile.close()

API_KEY = credentials['API_KEY']
API_SECRET = credentials['API_SECRET']
ACCESS_TOKEN = credentials['ACCESS_TOKEN']
ACCESS_TOKEN_SECRET = credentials['ACCESS_TOKEN_SECRET']

stream = MyStream(API_KEY, API_SECRET, ACCESS_TOKEN, ACCESS_TOKEN_SECRET)

stream.statuses.filter(track = 'covid')
```

In [ ]:

```
infile = open('tweet_stream.json')
tweets = json.load(infile)
infile.close()

tweets
```

In [ ]:

```
len(tweets)
```

(8) Putting Things Together (Example #2)

In [ ]:

```
# From the twython package, import the TwythonStreamer module
from twython import TwythonStreamer
import sys
import json

# Set a container to append 200 tweets
container = []

# We are inheriting from TwythonStreamer class
class MyStream(TwythonStreamer):
    '''our own subclass of TwythonStreamer'''

    # Overriding
    def on_success(self, data):
        if 'text' in data:
            print(data['text'])
            # append tweets to the container
            container.append(data)

        # If you get enough tweets (e.g. 200), store it into JSON file and disconnect AI
        if len(container) == 200:
            self.store_json()
            self.disconnect()
```

```
    # Overriding
    def on_error(self, status_code, data):
        print(status_code)
        self.disconnect()

    # New method to store tweets into JSON file
    def store_json(self):
        infile = open('tweet_stream_{}_{}.json'.format(keyword, len(container)), 'w')
        json.dump(container, infile)
        infile.close()

# Read Twitter credentials from json file and assign them to variables
infile = open('twitter_credentials.json')
credentials = json.load(infile)
infile.close()

API_KEY = credentials['API_KEY']
API_SECRET = credentials['API_SECRET']
ACCESS_TOKEN = credentials['ACCESS_TOKEN']
ACCESS_TOKEN_SECRET = credentials['ACCESS_TOKEN_SECRET']

# Twitter Streaming API needs all four credentials
stream = MyStream(API_KEY, API_SECRET, ACCESS_TOKEN, ACCESS_TOKEN_SECRET)

# Set a keyword
keyword = 'covid'

# Time to collect 200 tweets of the keyword
stream.statuses.filter(track = keyword)
```

In [ ]:

```
infile = open('tweet_stream_covid_200.json')
tweets = json.load(infile)
infile.close()

tweets
```

In [ ]:

```
len(tweets)
```

(9) Develop a Twitter Streaming Module (Example #3)

> What if you want to make a Twitter Streaming module from MyStream class, which can run in the terminal or command line or import it to use the MyStream class?

In [ ]:

```
# From the twython package, import the TwythonStreamer module
from twython import TwythonStreamer
import sys
import json

# Set a container to append 200 tweets
container = []

# We are inheriting from TwythonStreamer class
class MyStream(TwythonStreamer):
```

```
    '''our own subclass of TwythonStreamer'''

    # Overriding
    def on_success(self, data):
        if 'text' in data:
            print(data['text'])
            # Append tweets to the container
            container.append(data)

        # If you get enough tweets (e.g. 200), store it into JSON file and disconnect AP
        if len(container) == 200:
            self.store_json()
            self.disconnect()

    # Overriding
    def on_error(self, status_code, data):
        print(status_code)
        self.disconnect()

    # New method to store tweets into JSON file
    def store_json(self):
        infile = open('tweet_stream_{}_{}.json'.format(keyword, len(container)), 'w')
        json.dump(container, infile)
        infile.close()

# Read Twitter credentials from json file and assign them to variables
infile = open('twitter_credentials.json')
credentials = json.load(infile)
infile.close()

API_KEY = credentials['API_KEY']
API_SECRET = credentials['API_SECRET']
ACCESS_TOKEN = credentials['ACCESS_TOKEN']
ACCESS_TOKEN_SECRET = credentials['ACCESS_TOKEN_SECRET']

# Twitter Streaming API needs all four credentials
stream = MyStream(API_KEY, API_SECRET, ACCESS_TOKEN, ACCESS_TOKEN_SECRET)

# Set a keyword
keyword = 'covid'

# Time to collect 200 tweets of the keyword
stream.statuses.filter(track = keyword)
```

In [ ]:

```
!cat API_v1.py
```

In [ ]:

```
!python API_v1.py
```

In [ ]:

```
'''
!python API_v1.py covid, 300
'''
```

sys.argv

> To get command line arguments, use sys.argv.

In [ ]:
```
!cat test.py
```

In [ ]:
```
!python test.py argt1, argt2, argt3
```

Step 1: Keyword

> Let's change the Twitter Streaming Module to take a keyword.

In [ ]:
```
!cat API_v2.py
```

In [ ]:
```
!python API_v2.py covid
```

Step 2: Keyword and a Number of Tweets

> Let's change the Twitter Streaming Module to take a keyword and a number of tweets.

In [ ]:
```
!cat API_v3.py
```

In [ ]:
```
!python API_v3.py covid 10
```

Step 3: Error Handling

> What if a user may import the API_v3 module? What will happen? How do we prevent the error?

In [ ]:
```
import API_v3
```

In [ ]:
```
!cat API_v4.py
```

In [ ]:
```
!python API_v4.py covid 10
```

In [ ]:
```
import API_v4
```