# Data Formats

## Reading / References

- JSON's description[1]
- Twitter's Rest API[2]
- Python's JSON support[3]
- Python Dictionaries[4]
- More docs on Python Dictionaries[5]
- List comprehensions[6] for processing lists
- Twitter's search/tweets API[7]
- Tweet objects[8] describes the dictionary keys you should expect in an object. It's an important start point to understand the nested structure of the data.

### Practice questions on the reading

- Pick 5 important pieces of information that you may find in a tweet, and describe how you would access them.

## Notes

In this section we will discuss the process of accessing JSON information as served via Web services. We will do so in the particular case of the Twitter API, and more specifically we will look at tweets that mention the "user" HanoverCollege. The steps are roughly the following:

### Preparing the request

1. Create an appropriate "application account" for the service you want to use, if that service requires something like that. For Twitter in particular you would go to https://apps.twitter.com/. You will need to create a Twitter account first, and then create a new "application".
2. The application will typically provide you with a "key" and a "secret". In Twitter you will need to go to your specific application's page, and look at the "keys and access tokens" page. While you are there, you may want to also alter the access

[1] http://json.org/
[2] https://dev.twitter.com/rest/public
[3] https://docs.python.org/3/library/json.html
[4] https://docs.python.org/3/tutorial/datastructures.html#dictionaries
[5] https://docs.python.org/3/library/stdtypes.html#typesmapping
[6] https://docs.python.org/2/tutorial/datastructures.html#tut-listcomps
[7] https://dev.twitter.com/rest/reference/get/search/tweets
[8] https://dev.twitter.com/overview/api/tweets

level to be “read only”, meaning that your application is not permitted to post new tweets, only to read existing tweets.

3. You need to store these pieces of information in a location that can be accessed by your scripts, but that is otherwise not visible to the world. A good way to do that is to create a JSON file in your work directory, that your scripts can access. We will call ours keys.json, and this is how it would look:

```
{
   "twitter": {
      "key": "Your key here",
      "secret": "Your secret here"
   }
}
```

4. From your script, you need to first read your access information and then authenticate yourself with the application in question:

```
#!/usr/bin/python3

# Loading libraries needed for authentication and requests
from requests_oauthlib import OAuth2Session
from oauthlib.oauth2 import BackendApplicationClient
import json

# In order to use this script, you must:
# - Have a Twitter account and create an app
# - Store in keys.json a property called "twitter" whose value is an
#     object with two keys, "key" and "secret"
with open('keys.json', 'r') as f:
   keys = json.loads(f.read())['twitter']

twitter_key = keys['key']
twitter_secret = keys['secret']

# We authenticate ourselves with the above credentials
# We will demystify this process later
client = BackendApplicationClient(client_id=twitter_key)
oauth = OAuth2Session(client=client)
token = oauth.fetch_token(token_url='https://api.twitter.com/oauth2/token',
                          client_id=twitter_key,
                          client_secret=twitter_secret)

# At this point the "oauth" object can be used to make requests
```

5. At the end of the above lines, the “oauth” object represents an authenticated client that you can use to make further requests. Since many of the requests will have a URL in common, you should create some variables for that purpose:

```
# Base url needed for all subsequent queries
base_url = 'https://api.twitter.com/1.1/'

# Particular page requested. The specific query string will be appended to that.
page = 'search/tweets.json'

# Depending on the query we are interested in, we append the necessary string
```

```
# As you read through the twitter API, you'll find more possibilities
req_url = base_url + page + '?q=%40HanoverCollege&tweet_mode=extended'
```

6. We then perform a request, and process the result as JSON:

```
# We perform a request. Contains standard HTTP information
response = oauth.get(req_url)

# Read the query results
results = json.loads(response.content.decode('utf-8'))
```

7. Typically the "results" dictionary will contain some metadata that tells you information about the request, as well as the actual results. This is a standard Python dictionary[9] that you can examine. It is very useful at this point to be using Python interactively.

### Processing the request

You now have a dictionary that contains the important information you want to access. Let us look at its keys:

```
>>> results.keys()
dict_keys(['search_metadata', 'statuses'])
```

The search_metadata entry contains information about the search performed. It sounds like statuses contains what we need.

```
>>> type(results['statuses'])
<class 'list'>
```

Ok so that is a list. We could access the first element and take a look at it:

```
type(results['statuses'][0])    # It's a dictionary
results['statuses'][0].keys()   # Its keys
```

Ok that is a dictionary, and we can look at its keys, and it seems there are a number of possibilities there. For example we can access the tweet's text via:

```
results['statuses'][0]['full_text']
```

Or we can see if someone has retweeted or favorited it:

```
results['statuses'][0]['retweeted']
results['statuses'][0]['favorited']
```

The possibilities are endless. Most of the times you would want to iterate over the statuses list:

```
for status in results['statuses']:
    dosomething(status)
```

[9] https://docs.python.org/3/tutorial/datastructures.html#dictionaries

Or you can operate on the list of statuses via map and filter (see the docs[10]). Or even better, use list comprehensions[11]:

```
texts = [tweet['text'] for tweet in results['statuses']]
many_mentions = [tweet['text'] for tweet in results['statuses']
                 if len(tweet['entities']['user_mentions']) > 2]
```

[10] https://docs.python.org/2/library/functions.html
[11] https://docs.python.org/2/tutorial/datastructures.html#tut-listcomps