# Social Media Tracking Documentation

Cedric Tan - Roar Sports

April 2019

**Abstract**

The point of this documentation is to outline the details of the Python program that will track the social media following of Click Agency clients. The actual data is transported into an Excel file where independent analysis is conducted.

# Contents

# 1 Introduction

This document aims to spell out the documentation of the Python programme that will automate social media tracking for Roar Sports. The program focuses on the key social media platforms for scraping follower data. Further analysis will be done with the excel content at a later stage. The following social media platforms are included in this documentation:

1. Facebook
2. Twitter
3. Twitch
4. Youtube
5. Instagram

Following this guide will help you understand how the program works.

# 2 Facebook

Beginning with the Facebook interface, the program will request access to the Facebook **Graph API** by authenticating with the website, making a request to access for the data of a particular page or profile and subsequently saving this profile data to a database where we can print it all out on excel.

# 3 Twitter

With Twitter, the method will utilise **Tweepy** which is a library that allows for quick access to the Twitter API.

## 3.1 Code Rundown

First we will begin with the dependencies required to gain the data:

```
1
2 from tweepy import OAuthHandler
3 from tweepy import API
4 from tweepy import Cursor
5 import sys
```

Here in the code above, you can see the rundown on each dependency:

- Tweepy is a Python library that we can use to access the Twitter API

- OAuthHandler will allow us to painlessly authorise access to get data from the API itself

- API is the main access point for Tweepy to get the data

- Cursor allows for pagination and access to large volumes of data that we might need to segment into sections i.e. pages

- Sys provides some basic functionality for the interpreter

Then we will set up the keys to authorise our access to the Twitter API:

```
1
2 # These are the keys from the app created on the
      developer account
3
4 consumer_key = "..."
5 consumer_secret = "..."
6 access_token = "..."
7 access_secret_token = "..."
8
9 # This begins the authorisation process with the
      Twitter API so that the program can authenticate it
      self with the website
10
11 auth = OAuthHandler(consumer_key, consumer_secret)
```

```
12  auth.set_access_token(access_token, access_secret_token
        )
13  auth_api = API(auth)
```

Then with all parts authorised, we can begin to get the data from the API itself through simple get commands.

```
1
2  # We will first get the Twitter handle of the account
        that we want to check out:
3
4  user = auth_api.get.user("barackobama")
5
6  # Having gotten the object with the get.user function,
        in this case Barack Obama's account, we can execute
        some functions on it like:
7
8  screen_name = user.screen_name # Take the user's screen
        name
9  follower_count = user.followers_count # Take the user's
        follower count
10
11 # Then printing these, we return values specific to the
        account
12
13 print (screen_name)
14 print (follower_count)
15
16 # This returns:
17    BarackObama
18    105699616 # 17/04/2019 12:07pm
```

## 3.2 Data Retrieval

From the previous section, we can just iterate the code over and over again from our database of Twitter Handles. This can either be hard coded into the system or taken from an Excel and imported in.

The iteration of code will be a simple for loop such that the output will show the each line of data specific to the Twitter handle. Taking the code from above, we can implement the loop below:

```
1  # Defining a dictionary
2  name_dictionary = ["name1", "name2", "name3", "name4"]
3
4  for name in name_dictionary:
```

```python
5    user = auth_api.get.user(name)
6    screen_name = user.screen_name
7    follower_count = user.followers_count
8
9    print(screen.name)
10   print(follower.count)
11
12 # This will return
13   name1
14   name1_follower_count
15   ...
16   name4
17   name4_follower_count
```

# 4  Twitch

Twitch has API authentication through your Twitch account with the same OAuth processes and application interfaces that are necessary for the API request.

## 4.1  Code Rundown

The dependencies for the Twitch API are much simpler, we will only use requests.

```
1
2 import requests
```

And from there we can simply call the API and authenticate by sending in a header with our application Client-ID.

We can begin by asking for some user information:

```
1
2 user_request = requests.get('https://api.twitch.tv/
    helix/users?login=' + user_name, headers={"Client-
    ID": "..."}
```

# 5  Youtube

The Google developer console makes the Youtube Data API very accessible - all you require is a Google account to get an API key that is valid once you activate the Youtube Data API package on your account.

## 5.1  Code Rundown

Again we will begin with the dependencies that we are looking to use. For Youtube we are simply going to request some data from the Youtube Data API **without using a particular software development kit** which makes it more simplistic, requiring less installation.

```python
import urllib.request
import json

key = "..."
```

The rundown:

- Urllib.request is the library used to draw requests from the actual API

- JSON allows us to read these objects and draw the important information we want from them

- The key is your personal authentication to gain access to the API

Then we can call on channel handles to get information on the amount of subscribers they have using the JSON object we have called:

```python
dictionary = ["pewdiepie","bgfilms", "rhettandlink2"]

for name in dictionary:
  # Requesting access to the API along with utilising
    the key to gain access

    data = urllib.request.urlopen("https://www.
    googleapis.com/youtube/v3/channels?part=statistics&
    forUsername="+name+"&key="+key).read()

    # Loading the JSON requested data and then
    searching for the subscriber count
```

```
10      subs = json.loads(data)["items"][0]["statistics"]["
        subscriberCount"]

11

12      # Printing the output
13      print(name + " has " + "{:,d}".format(int(subs)) +
        " subscribers")

14

15      # This will give us:

16

17      pewdiepie has 94,317,649 subscribers
18   bgfilms has 4,122,055 subscribers
19   rhettandlink2 has 15,294,165 subscribers
```

# 6 Object Oriented Structure

Having discussed the various APIs that we are going to use, the next step is to discuss what type of structure we will use to access the required information across the board. Probably the most scalable and efficient structure to use with the ability to call numerous functions from would be using classes.

The structure of a class provides a mean of bundling data and functionality together. Creating a class is creating a new type of object allowing *instances of that object* to be created again and again.

For the purposes of Social Media Tracking and Performance, we will create the influencer class which stores the data of each influencer in a new instance. From there, we can call functions as necessary on these influencers to gain information on their numbers through our APIs.

## 6.1 Code Rundown

Setting up the class is easy and we will give it a lot of variables related to the data that we want to collect.

```
1
2  class influencer:
3    def __init__(self, name, gender, gamer_tag, fb_tag,
       twitter_tag, yt_tag, insta_tag, twitch_tag):
4      self.name = name
5      self.gender = gender
6      self.gamer_tag = gamer_tag
7      self.fb_tag = fb_tag # Facebook Take
```

```
 8        self.twitter_tag = twitter_tag # Twitter Tag
 9        self.yt_tag = yt_tag # Youtube Channel Tag
10        self.insta_tag = insta_tag # Instagram Tag
11        self.twitch_tag = twitch_tag # Twitch Channel Tag
12
13     def return_name():
14        return self.name
```

From this, we can recognise the different social media accounts assigned to each along with some other basic information such as name and gender.

Following that, we can input some basic variables to store for the future and then subsequently call them with our return function that we had illustrated above. This is shown below:

```
 1
 2 # Defining the influencer
 3
 4 influencer_1 = influencer(Jack, Male, JackTTV, Jack_FB,
        Jack_Twitter, Jack_YT, Jack_Insta, Jack_Twitch)
 5
 6 # Now calling our return_name function
 7
 8 print (influencer_1.return_name)
 9
10 # This will give us 'Jack' in the console.
```

# 7    Data Structures

This section will focus on a simple datastructure that we will store our data in. Here we will use the pandas framework that is built on top of NumPy.

```
 1 # The standard way of importing dependencies
 2
 3 import numpy as np
 4 import pandas as pd
```