IAE 101 – Introduction to Digital Intelligence

Documentation for simple_twit, v2.1

0. What is simple_twit?
simple_twit is an interface that has been wrapped around the Tweepy Python library package for accessing Twitter's Application Programming Interface (API v2). The purpose of simple_twit is to give users a simpler, safer, and more restricted way to access some of the methods inside the Tweepy library and so access Twitter functionality from inside their programs.

1. Dependencies
simple_twit depends upon Tweepy in order to function. The Tweepy library must be installed before simple_twit can be used. Tweepy can be installed with the following commands run from the command-line:

On Windows:

1. Open an instance of CMD – the command prompt:

2. At the prompt enter: `py -3 -m pip install tweepy --user`

On Mac:

1. Open an instance of the terminal.

2. At the command line enter: `python3 -m pip install tweepy --user`

2. Tweepy Data Model and Twitter's Response Data

A. Response Objects: https://docs.tweepy.org/en/stable/response.html

Whenever you submit a successful request to Twitter's API (to post a tweet, or search for tweets, or get information about a user, etc.), data is sent back by Twitter to your program in the form of a JSON structure called a Response. Tweepy takes this data and creates a Response object (actually a *named tuple*, if you want to be technical about it, but you can treat it like an object) to represent this response data in your program.

So, every simple_twit method returns this Response object generated by Tweepy to contain the data sent back by Twitter. If you want to print out or use the data sent back by Twitter, you need to access that data through the fields of the Response object.

Every response may contain several fields: data, includes, error, and meta. For our purposes, the two most important fields are the ***data*** field and the ***meta*** field.

- The data field contains the actual response data. If you posted a tweet, it will contain the details about the tweet you successfully posted; if you are searching for tweets or users, then the data field will contain the collection (usually a Python list) of elements returned as the results of the search. So, when you use the response in your program you will frequently be using the data field of the Response object.

- The meta field contains information about your request and about the data. The nature of the metadata varies for different kinds of requests and responses. For example, if you use the search_tweets() function, the meta field will contain the ID of the newest tweet in the response and the oldest tweet in the response—this information can be used to shape future searches so that they don't include redundant results.

B. Tweet objects: https://docs.tweepy.org/en/stable/v2_models.html#tweet

Also see: https://developer.x.com/en/docs/x-api/data-dictionary/object-model/tweet

These are objects that represent a tweet. Tweet objects created using simple_twit functions should contain the following fields:

- data:
  The JSON data representing the tweet.
  Type: dict

- id:
  The unique identifier of the requested Tweet.
  Type: int (can also use as str)

- text:
  The actual UTF-8 text of the Tweet.
  Type: str

- author_id:
  The unique identifier of the User who posted this Tweet.
  Type: int (can also use as str)

- created_at:
  Creation time of the Tweet.
  Type: datetime.datetime

- in_reply_to_user_id:
  If the represented Tweet is a reply, this field will contain the original Tweet's author ID. This will not necessarily always be the user directly mentioned in the Tweet.
  Type: int (can also used as str)

- lang
  Language of the Tweet, if detected by Twitter. Returned as a BCP47 language tag.
  Type: str

- public_metrics:
  Public engagement metrics for the Tweet at the time of the request.
  Type: dict

C. User objects: https://docs.tweepy.org/en/stable/v2_models.html#user

Also see: https://developer.x.com/en/docs/x-api/data-dictionary/object-model/user

- data:
  The JSON data representing the user.
  Type: dict

- id:
  The unique identifier of the user.
  Type: int (can also use as string)

- name:
  The name of the user, as they've defined it on their profile. Not necessarily a person's name. Typically capped at 50 characters, but subject to change.
  Type: str

- username:
  The Twitter screen name, handle, or alias that this user identifies themselves with. Usernames are unique but subject to change. Typically a maximum of 15 characters long, but some historical accounts may exist with longer names.
  Type: str

- created_at
  The UTC datetime that the user account was created on Twitter.
  Type: datetime.datetime

- description:
  The text of this user's profile description (also known as bio), if the user provided one.
  Type: str

- pinned_tweet_id:
  Unique identifier of this user's pinned Tweet.
  Type: int (can also use as str)

- public_metrics
  Contains details about activity for this user.
  Type: dict

- url:
  The URL specified in the user's profile, if present.
  Type: str

3. simple_twit Functions

A. General Functions

simple_twit.create_client(api_key, api_key_secret)

This is the initialization function for the simple_twit module. This function must be called before calling any other simple_twit functions.

This function will authorize your program to make requests to Twitter's API through Tweepy on behalf of a specific Twitter user. create_client() will look for some user authorization credentials in a file called "twitterbot.config". If that file exists, then it will read the access token and access token secret from the file and use them to complete the authorization process. If not, then it will launch a web browser to complete the authorization process, and then store the new access token and secret in the file "twitterbot.config".

When authorization is complete, create_client() will create and initialize a Tweepy Client object and return it. This Tweepy Client object must be passed as the first argument to other simple_twit functions in order for them to make authorized requests to Twitter's API.

Parameters:

api_key:
a string; first element of the developer credentials; credentials are distributed by the instructor, but you may also request your own from twitter; also known as an ***Consumer key***.

api_key_secret:
a string; second element of the developer credentials; credentials are distributed by the instructor, but you may also request your own from twitter; also known as an ***Consumer Secret***.

Return Value: An initialized Tweepy Client object

simple_twit.version()

This will print out and return a string stating the version of the simple_twit module being used.

Parameters: None
Return Value: A string stating the current version number.

B. Tweet Functions

simple_twit.send_tweet(client, msg)

Updates the authenticating user's current status, also known as Tweeting. If the text duplicates a recent tweet posted by the home user, then this function will return an error. If the user has posted too many tweets recently, then this function will return an error.
Otherwise, the return value is a Tweet object representing the tweet just posted by this function call.

Parameters:
client:
A Tweepy client object; default value is None
msg:
A string containing the status update text to be posted to Twitter; default value is an empty string.
Return Value:
A Response object containing information about the tweet just posted.

simple_twit.send_reply_tweet(client, msg, tweet_id)

Same as send_tweet(), but in reply to another tweet. The return value is a Response object containing information about the reply tweet just posted by this function call.

Parameters:
client:
A Tweepy Client object; default value is None
msg:
A string containing the status update text to be posted to twitter; default value is an empty string.
tweet_id:
An int that is a unique identifier for the tweet to which this is a reply.

Return Value: A Response object containing information about the new tweet.

simple_twit.send_media_tweet(client, msg, filename)

Same as send_tweet(), but the status update can include an image. Return value is a Response object containing information about the tweet just posted by this function call.

Parameters:
client:
A Tweepy client object; default value is None
msg:
A string containing the status update text to be posted to Twitter; default value is an empty string.
filename:
A string naming an image file. The image will be included in the status update posted to twitter.

Return Value: A Response object containing information about the new tweet.

simple_twit.send_reply_media_tweet(client, msg, tweet_id, filename)

Same as send_reply_tweet(), but the status update can include an image. Return value is a Response object containing information about the tweet just posted by this function call.

Parameters:
client:
A Tweepy Client object; default value is None
msg:
A string containing the status update text to be posted to Twitter; default value is an empty string.
tweet_id:
An int that is a unique identifier for the tweet to which this is a reply.
filename:
A string naming an image file. The image will be included in the status update posted to twitter.

Return Value:
A Response object containing information about the new tweet.

simple_twit.retweet(client, tid)

Reposts ("retweets") a tweet. Requires the id of the tweet you are retweeting.

Parameters:
client:
A Tweepy Client object; default value is None
tid:
This argument is the numerical identifier for the tweet that will be retweeted by the home user; default value is None
Return Value:
A Response object containing information about the retweet.

simple_twit.get_tweet(client, tid)

Returns a Response object that contains information about a specific tweet as specified by the **tid** parameter.

Parameters:
client:
A Tweepy Client object; default value is None
tid:
This argument is the numerical identifier for the tweet whose information will be retrieved; default value is None
Return Value:
A Response object containing information about the specified tweet.

simple_twit.get_retweeters(client, tid, count)

Return a Response object whose **data** field contains a list of up to *count* many User objects each of who reposted the tweet specified by **tid**.

Parameters:
client:
A Tweepy Client object; default value is None
tid:
This argument is the numerical identifier for the tweet whose retweeters are being retrieved; default value is None
count:
The number of Users to retrieve; it must be an integer between 1 and 10; the default value is 1.
Return Value:
A Response object containing a list of the retweeting users.

C. Timeline Functions

simple_twit.get_home_timeline(client, count)

Returns the most recent tweets, including retweets, posted by the authenticating user and accounts that user follows. This is the equivalent of /timeline/home on the Web. The number of tweets returned is determined by the value of the count parameter. Tweets are returned as a list of Tweet objects in the data field of the Response object.

Parameters:
client:
A Tweepy Client object; default value is None
count:
An integer that specifies the number of tweets to be returned from the user's home timeline; integer must be between 1 and 10; default value is 1.
Return Value:
A Response object containing a list of Tweet objects in its **data** field.

simple_twit.get_users_tweets(client, user_name, count)

Returns the most recent tweets posted by the specified user. We identify the user by their user name. We specify how many recent tweets to return through the count parameter.

Parameters:

client:

A Tweepy Client object; default value is None

user_name:

A string giving the user name of the user whose recent tweets will be returned; default value is an empty string.

count:

An integer specifying how many recent tweets will be returned from the user's timeline; integer must be between 5 and 10; default value is 5.

Return Value: A Response object containing a list of Tweet objects in its **data** field.

simple_twit.get_users_mentions(client, user_name, count)

Returns the most recent mentions of the user specified by the **user_name** parameter, including retweets.

Parameters:

client:

A Tweepy Client object; default value is None

user_name:

A string giving the user name of the user whose mentions will be returned; default value is an empty string.

count:

An integer that specifies the number of tweets mentioning the specified user; the number must be between 5 and 10; default value is 5.

Return Value:

A Response object containing a list of Tweet objects in its **data** field that mention the specified user.

D. User Functions

simple_twit.get_user(client, *, user_name, user_id)

Returns a Response object that contains information about the specified user. **Notes:** You must either pass an argument to the user_name parameter or the user_id parameter, but not both. In addition, you cannot leave them both empty. Further, you must use the parameter name when assigning to either of these parameters. See the posted examples.

Parameters:
- client:
  A Tweepy Client object; default value is None
- user_name:
  A string giving the account name or screen name of a user; default value is an empty string.
- user_id:
  A unique numerical identifier that picks out a specific user; default value is an empty string.

Return Value:
A Response object containing a User object in its **data** field containing information about the specified Twitter account holder.

E. Search Functions

simple_twit.search(client, query, count)

Returns a Response object containing a collection of Tweets in its **data** field responsive to the query string.

Parameters:
- client:
  A Tweepy Client object; default value is None
- query:
  A string describing what you want to search for; default value is an empty string.
- count:
  An integer representing how many search results to return; must be an integer between 10 and 20; default value of 10.

Return Value:
A Response object containing a collection of Tweets as the response to the search.

4. Guidelines for Using simple_twit and Tweepy

4A. Rate Limiting

Twitter places restrictions on how many times an application can make requests. These limits are imposed in 15 minute windows—so there is a maximum number of requests per 15 minute interval.

If you exceed the rate limits, your requests will be blocked until the next interval opens (after 15 minutes. If you try to evade the rate limits, Twitter will ban the application from accessing its API.

In addition, there are restrictions on how many posts we can make and how much data (tweets, users, etc.) we can request from twitter per month. The limit is large-ish, but with almost 500 students using the same credentials we can use them up quick if you are not careful.

You are using my application credentials to access Twitter. Please be careful. Keep your tests small and try to space them out. Do not spam Twitter's API with requests.

IMPORTANT: For now, do not put function calls to Twitter's API inside loops.

IMPORTANT: If you use loops, you must slow them down by using the `time.sleep()` function to force your program to wait before executing any more code.

-For example `time.sleep(900)` will force your program to wait 900 seconds (15 minutes) before executing the next line of code.

4B. Acceptable Behavior

Do not post profane, lewd, obscene, malicious, hateful, bigoted, or harassing content to Twitter.

Be very careful about how you interact with other users.

You must follow all of twitter's policies for the use of their API, especially their policies regarding automated behavior:

https://help.x.com/en/rules-and-policies/x-automation