**API Task 1**

*Get the twitter content of the tweet*[*https://twitter.com/Google/status/1257326183101980673*](https://twitter.com/Google/status/1257326183101980673)*and store it in a flat-file.  
Download the video file from the tweet*[*https://twitter.com/Google/status/1257326183101980673*](https://twitter.com/Google/status/1257326183101980673)*and store it in a folder.  
Get the number of retweets for the tweet*[*https://twitter.com/Google/status/1257326183101980673*](https://twitter.com/Google/status/1257326183101980673)*and store it in the same file.  
Get the retweeters ID for the tweet*[*https://twitter.com/Google/status/1257326183101980673*](https://twitter.com/Google/status/1257326183101980673)*and store it in the same file.  
  
Now use the values in the flat file as base-lined ground truth and assert them for the next test run.*

**Authentication:**

The authentication method used here is ***OAuth2.0 Bearer Token (Application only authentication)*** since the aim is to GET the tweet contents.

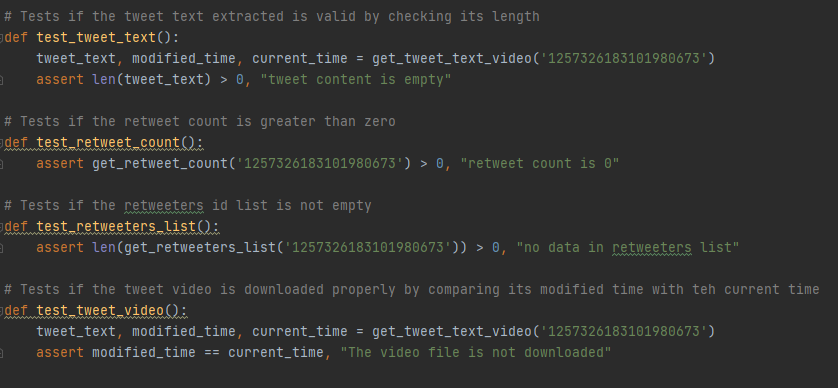
Refer GetAuthenticationHeader() from APITask1.py file

Note: Authentication flow explained in the code in their respective methods

**Assertions:**

To assert the tweet results, separate set of test functions are used as below,

1. To test the tweet text, an assertion to check its length is added. In case if the tweet text is not extracted then the assertion error “tweet content empty” is reported
2. To test the retweet count, an assertion to check if it is not zero is added. In the given tweet the retweet count is not zero so I have added an assertion to check if the retweet count is greater than zero
3. To validate the retweeters id list, assertion is added to check if the list is empty
4. To test if the video is downloaded properly, an assertion to compare if the last modified time is equal to the current time



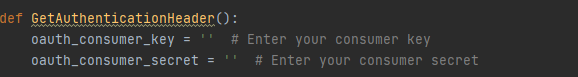
**File Reference:**

APITask1.py 🡪 Main file to achieve task1

DownloadVideo.py 🡪 Download the video file from tweet

**Execution:**

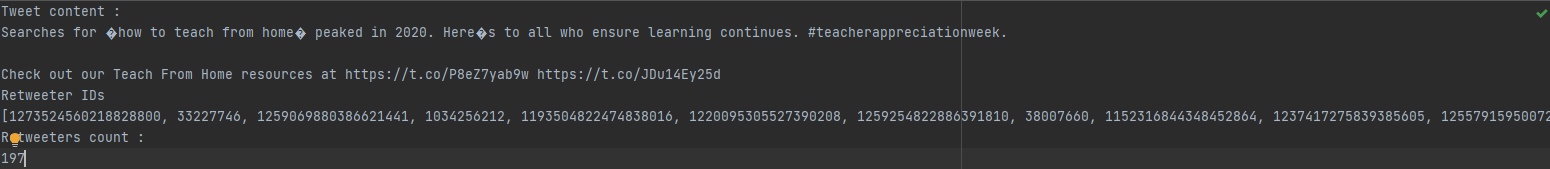
1. Enter the credentials in GetAuthenticationHeader() method of APITask1.py

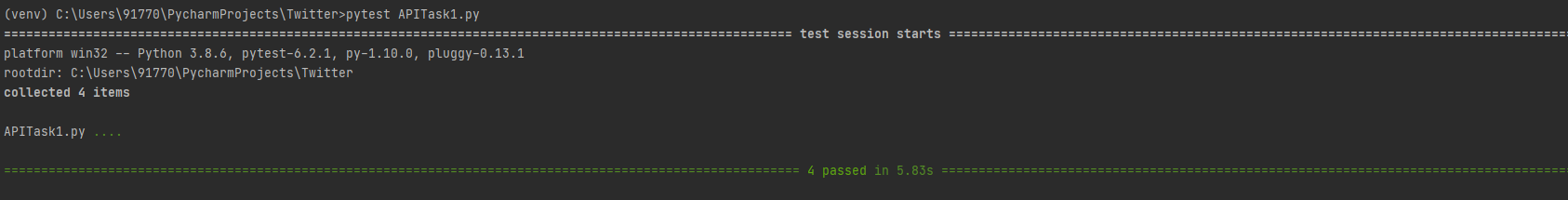


1. Run APITask1.py

**Results:**

APITask1.txt





**API Task 2**

*Have your twitter login details in a file and use the same to achieve the following. Make sure you remove your login details before sending it to us.  
The tests should run if we give our login details in the file.  
Make a new tweet with the text "We welcome you to MSD family :)"  
Now retweet the same tweet.  
Now get the retweet count & retweeters ID and validate the correctness of the data.  
Now revert the previous retweet (un retweet the above tweet) and get the retweet count & retweeters ID and validate the correctness of the data.  
Now finally delete the tweet.*

**Authentication:**

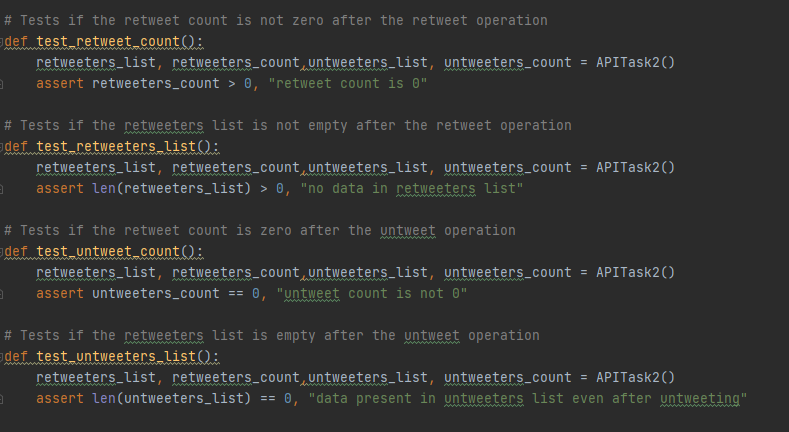
The authentication method used here is ***OAuth1.0a*** for posting, retweeting, untweeting, deleting the tweet and ***OAuth2.0 Bearer Token - Application only authentication*** for getting the retweeters count and list

Note:

1. Have your credentials in credentials.json
2. Authentication flow explained in the code in their respective methods in ***GetAuthentication.py***

**Assertions:**

1. Test if the retweet count is not zero after the retweet operation
2. Test if the retweeters list is not empty after the retweet
3. Test if the retweet count is zero after the untweet operation
4. Test if the retweeters list is empty after the untweet



**File Reference:**

APITask2.py 🡪 Main file to achieve task2

GetAuthentication.py 🡪 To create a authentication header for the API requests (POST)

PostTweet.py 🡪 To post a tweet

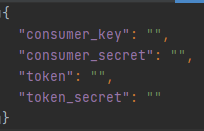
Retweet.py 🡪 to retweet a tweet

Untweet.py 🡪 to untweet a tweet

DeleteTweet.py 🡪 to delete the tweet

**Execution:**

1. Add the credentials in the credentials.json



1. Run APITask2.py

**Results**

