**Authentication:**In order to fetch tweets through Twitter **API**, one needs to register an App through their twitter account. Follow these steps for the same:

* Open this[link](https://apps.twitter.com/) and click the button: ‘Create New App’
* Fill the application details. You can leave the callback url field empty.
* Once the app is created, you will be redirected to the app page.
* Open the ‘Keys and Access Tokens’ tab.
* Copy ‘Consumer Key’, ‘Consumer Secret’, ‘Access token’ and ‘Access Token Secret’.

### **Creating a Twitter App**

In order to extract tweets for a posterior analysis, we need to access to our Twitter account and create an app. The website to do this is <https://apps.twitter.com/>. (If you don't know how to do this, you can follow this [tutorial video](https://www.youtube.com/watch?v=BOA7SD_09Qk) to create an account and an application.)

From this app that we're creating we will save the following information in a script called credentials.py:

* Consumer Key (API Key)
* Consumer Secret (API Secret)
* Access Token
* Access Token Secret

An example of this script is the following:

*# Twitter App access keys for @user*

*# Consume:*

CONSUMER\_KEY = ''

CONSUMER\_SECRET = ''

*# Access:*

ACCESS\_TOKEN = ''

ACCESS\_SECRET = ''

The reason of creating this extra file is that we want to export only the value of this variables, but being unseen in our main code (our notebook). We are now able to consume Twitter's API. In order to do this, we will create a function to allow us our keys authentication. We will add this function in another cell of code and we will run it:

Example 1

**Implementation:**

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| import re  import tweepy  from tweepy import OAuthHandler  from textblob import TextBlob    **class TwitterClient(object):**      '''      Generic Twitter Class for sentiment analysis.      '''      def \_\_init\_\_(self):          '''          Class constructor or initialization method.          '''          # keys and tokens from the Twitter Dev Console          consumer\_key = 'XXXXXXXXXXXXXXXXXXXXXXXX'          consumer\_secret = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXX'          access\_token = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXX'          access\_token\_secret = 'XXXXXXXXXXXXXXXXXXXXXXXXX'            # attempt authentication          try:              # create OAuthHandler object              self.auth = **OAuthHandler**(consumer\_key, consumer\_secret)              # set access token and secret              self.auth.set\_access\_token(access\_token, access\_token\_secret)              # create tweepy API object to fetch tweets              self.api = **tweepy.API**(self.auth)          except:              print("Error: Authentication Failed")        def clean\_tweet(self, tweet):          '''          Utility function to clean tweet text by removing links, special characters          using simple regex statements.          '''          return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])                                      |(\w+:\/\/\S+)", " ", tweet).split())        def **get\_tweet\_sentiment**(self, tweet):          '''          Utility function to classify sentiment of passed tweet          using textblob's sentiment method          '''          # create TextBlob object of passed tweet text          analysis = TextBlob(self.**clean\_tweet**(tweet))          # set sentiment          if analysis.sentiment.polarity > 0:              return 'positive'          elif analysis.sentiment.polarity == 0:              return 'neutral'          else:              return 'negative'        def get\_tweets(self, query, count = 10):          '''          Main function to fetch tweets and parse them.          '''          # empty list to store parsed tweets          tweets = []            try:              # call twitter api to fetch tweets              fetched\_tweets = self.api.search(q = query, count = count)                # parsing tweets one by one              for tweet in fetched\_tweets:                  # empty dictionary to store required params of a tweet                  parsed\_tweet = {}                    # saving text of tweet                  parsed\_tweet['text'] = tweet.text                  # saving sentiment of tweet                  parsed\_tweet['sentiment'] = self.**get\_tweet\_sentiment**(tweet.text)                    # appending parsed tweet to tweets list                  if tweet.retweet\_count > 0:                      # if tweet has retweets, ensure that it is appended only once                      if parsed\_tweet not in tweets:                          tweets.append(parsed\_tweet)                  else:                      tweets.append(parsed\_tweet)                # return parsed tweets              return tweets            except tweepy.TweepError as e:              # print error (if any)              print("Error : " + str(e))    def main():      # creating object of TwitterClient Class  **api = TwitterClient()**      # calling function to get tweets      tweets = **api.get\_tweets**(query = 'Donald Trump', count = 200)        # picking positive tweets from tweets      ptweets = [tweet for tweet in tweets if tweet['sentiment'] == 'positive']      # percentage of positive tweets      print("Positive tweets percentage: {} %".format(100\*len(ptweets)/len(tweets)))      # picking negative tweets from tweets      ntweets = [tweet for tweet in tweets if tweet['sentiment'] == 'negative']    # percentage of negative tweets      print("Negative tweets percentage: {} %".format(100\*len(ntweets)/len(tweets)))      # percentage of neutral tweets      print("Neutral tweets percentage: {} % \          ".format(100\*len(tweets - ntweets - ptweets)/len(tweets)))        # printing first 5 positive tweets      print("\n\nPositive tweets:")      for tweet in ptweets[:10]:          print(tweet['text'])        # printing first 5 negative tweets      print("\n\nNegative tweets:")      for tweet in ntweets[:10]:          print(tweet['text'])    if \_\_name\_\_ == "\_\_main\_\_":      # calling main function      main() |

Here is how a sample output looks like when above program is run:

Positive tweets percentage: 22 %

Negative tweets percentage: 15 %