We follow these 3 major steps in our program:

- Authorize twitter API client.
- Make a GET request to Twitter API to fetch tweets for a particular query.
- Parse the tweets. Classify each tweet as positive, negative or neutral.

Now, let us try to understand the above piece of code:

- First of all, we create a **TwitterClient** class. This class contains all the methods to interact with Twitter API and parsing tweets. We use **__init__** function to handle the authentication of API client.
- In **get_tweets** function, we use:

  ```
  fetched_tweets = self.api.search(q = query, count = count)
  ```

  to call the Twitter API to fetch tweets.

- In **get_tweet_sentiment** we use textblob module.

  ```
  analysis = TextBlob(self.clean_tweet(tweet))
  ```

  TextBlob is actually a high level library built over top of NLTK library. First we call **clean_tweet** method to remove links, special characters, etc. from the tweet using some simple regex.
  Then, as we pass **tweet** to create a **TextBlob** object, following processing is done over text by textblob library:

  - Tokenize the tweet ,i.e split words from body of text.
  - Remove stopwords from the tokens.(stopwords are the commonly used words which are irrelevant in text analysis like I, am, you, are, etc.)
  - Do POS(part of speech) tagging of the tokens and select only significant features/tokens like adjectives, adverbs, etc.
  - Pass the tokens to a **sentiment classifier** which classifies the tweet sentiment as positive, negative or neutral by assigning it a polarity between -1.0 to 1.0 .

  Here is how **sentiment classifier** is created:

  - **TextBlob** uses a Movies Reviews dataset in which reviews have already been labelled as positive or negative.
  - Positive and negative features are extracted from each positive and negative review respectively.
  - Training data now consists of labelled positive and negative features. This data is trained on a Naive Bayes Classifier.

  Then, we use **sentiment.polarity** method of **TextBlob** class to get the polarity of tweet between -1 to 1.
  Then, we classify polarity as:

  ```
  if analysis.sentiment.polarity > 0:
         return 'positive'
  elif analysis.sentiment.polarity == 0:
  ```

```
        return 'neutral'
else:
        return 'negative'
```

- Finally, parsed tweets are returned. Then, we can do various type of statistical analysis on the tweets. For example, in above program, we tried to find the percentage of positive, negative and neutral tweets about a query.