Spotle AI-thon 2020

Level III - The Mental Health Of India During COVID

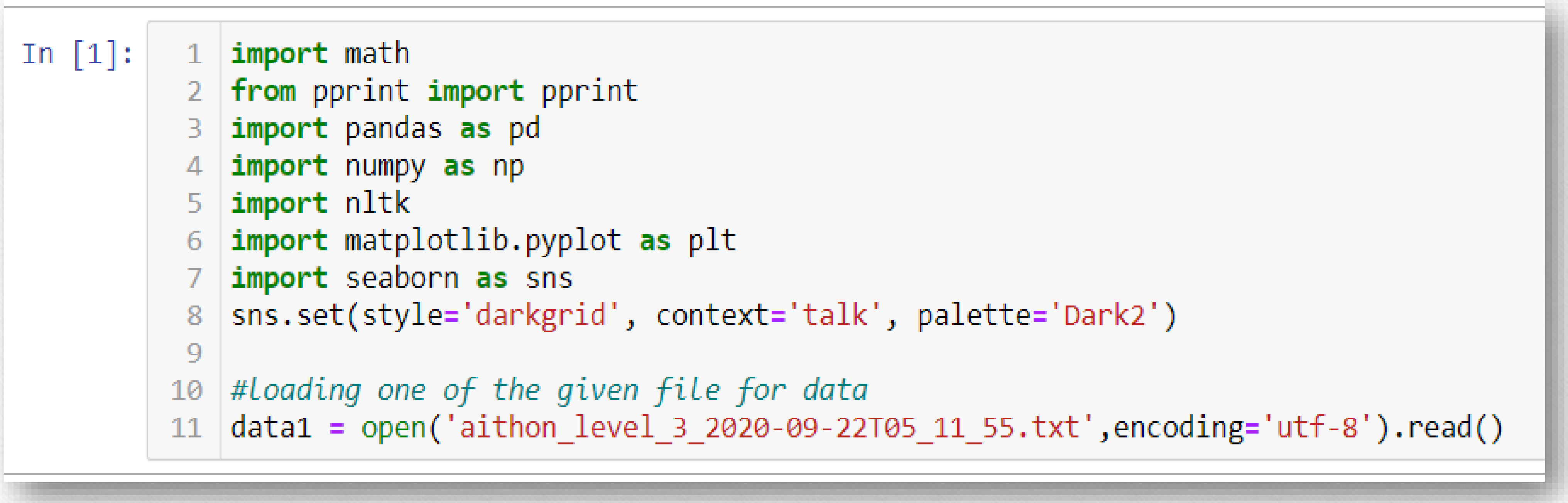
By: Team Furnace_Blasters
Shreyansh Gupta
Mayank Goel

Sentiment Analysis of tweets during pandemic

Resources used:

- **Python**
- **Basic Machine Learning**
- **Libraries** : pandas, numpy, nltk, matplotlib, seaborn

Starting with importing some files & loading our dataset.



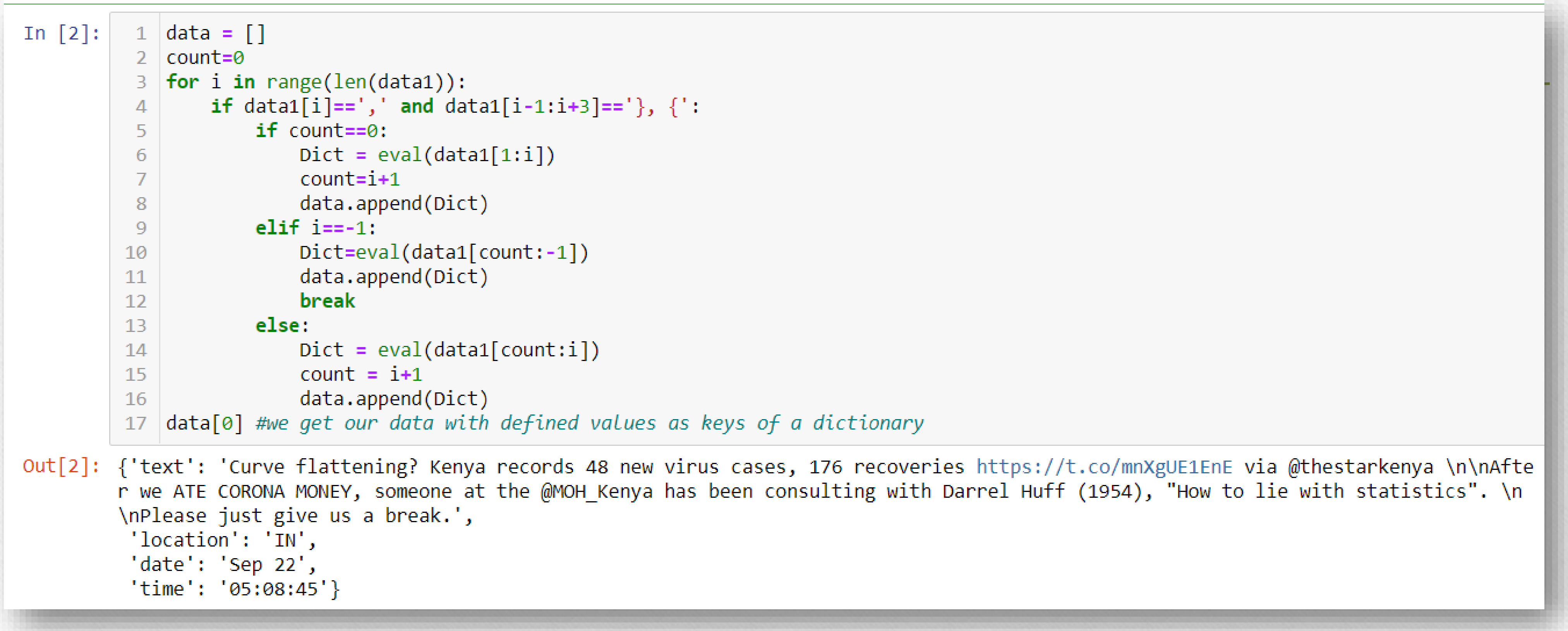
```python
import math
from pprint import pprint
import pandas as pd
import numpy as np
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style='darkgrid', context='talk', palette='Dark2')

#loading one of the given file for data
data1 = open('aithon_level_3_2020-09-22T05_11_55.txt',encoding='utf-8').read()
```

Two things important to be mentioned are pprint which is 'pretty-print' for printing JSON files and seaborn(sns) for beautifying the plotted matplotlib graphs and display

NLTK

- Before we get started with gathering the data, we will need to install the Natural Language Toolkit (NLTK) python package. To see how to install NLTK, you can go here: http://www.nltk.org/install.html. We'll need to open Python command line and run nltk.download() to grab NLTK's databases.
- All other needs will be specified when needed in upcoming slides.

Making our data worth visualizing and to work on it.



```python
data = []
count=0
for i in range(len(data1)):
    if data1[i]==',' and data1[i-1:i+3]=='}, {':
        if count==0:
            Dict = eval(data1[1:i])
            count=i+1
            data.append(Dict)
        elif i==-1:
            Dict=eval(data1[count:-1])
            data.append(Dict)
            break
        else:
            Dict = eval(data1[count:i])
            count = i+1
            data.append(Dict)
data[0] #we get our data with defined values as keys of a dictionary
```

```
{'text': 'Curve flattening? Kenya records 48 new virus cases, 176 recoveries https://t.co/mnXgUE1EnE via @thestarkenya \n\nAfte
r we ATE CORONA MONEY, someone at the @MOH_Kenya has been consulting with Darrel Huff (1954), "How to lie with statistics". \n
\nPlease just give us a break.',
 'location': 'IN',
 'date': 'Sep 22',
 'time': '05:08:45'}
```

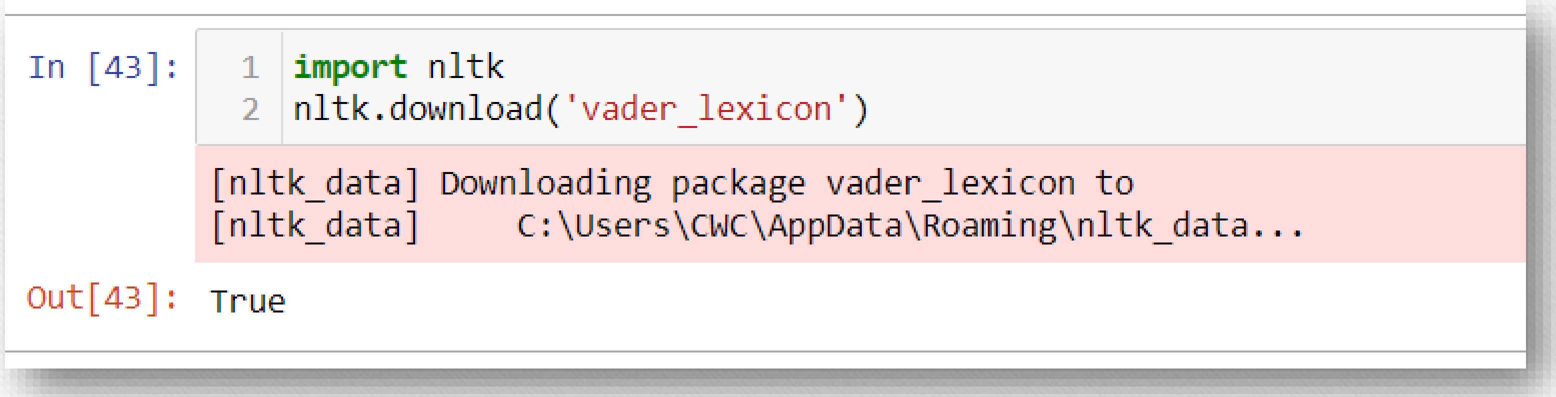
Labelling our data

NLTK’s built-in **Vader Sentiment Analyzer** will simply rank a piece of text as positive, negative or neutral using a lexicon of positive and negative words.

We can utilize this tool by first creating a **Sentiment Intensity Analyzer (SIA)** to categorize our headlines, then we'll use the `score` method to get the sentiment.

We'll append each sentiment dictionary to a results list, which we'll transform into a Dataframe as shown in upcoming slides.

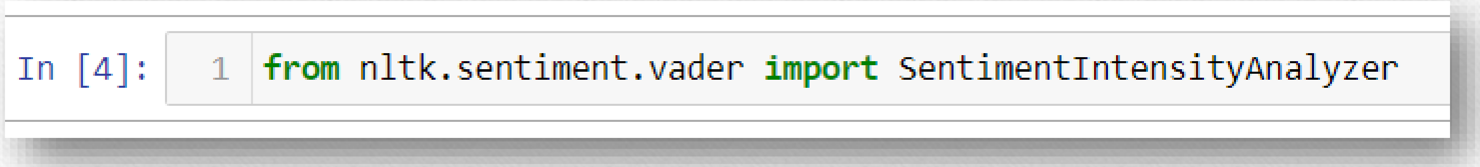
Installing needs.



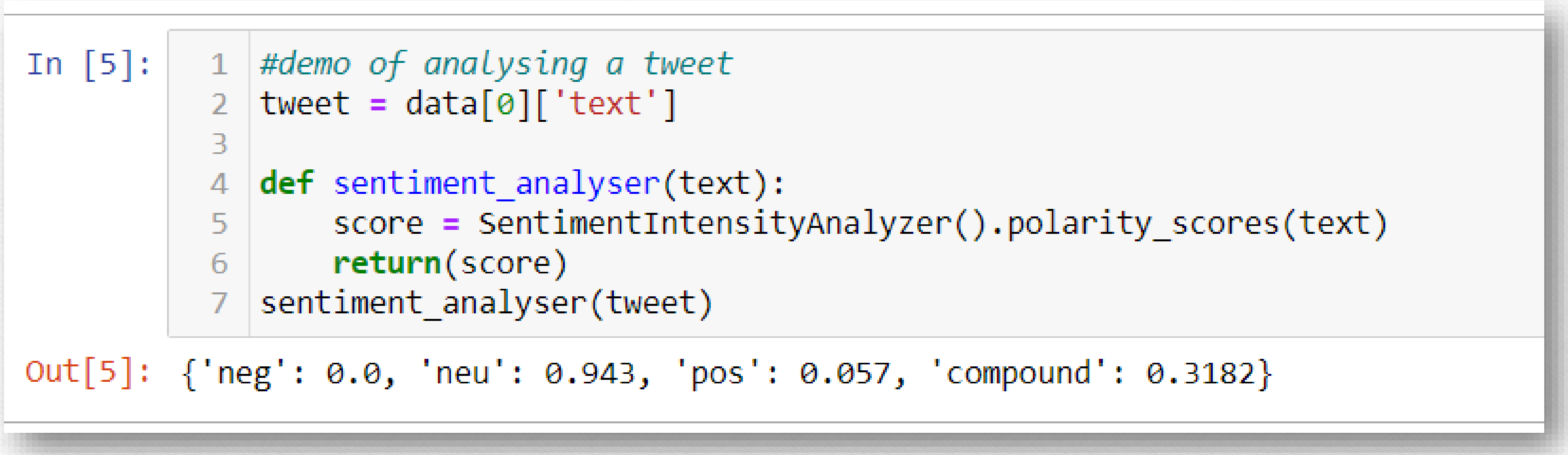
```
In [43]:
import nltk
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\CWC\AppData\Roaming\nltk_data...
```

```
Out[43]: True
```



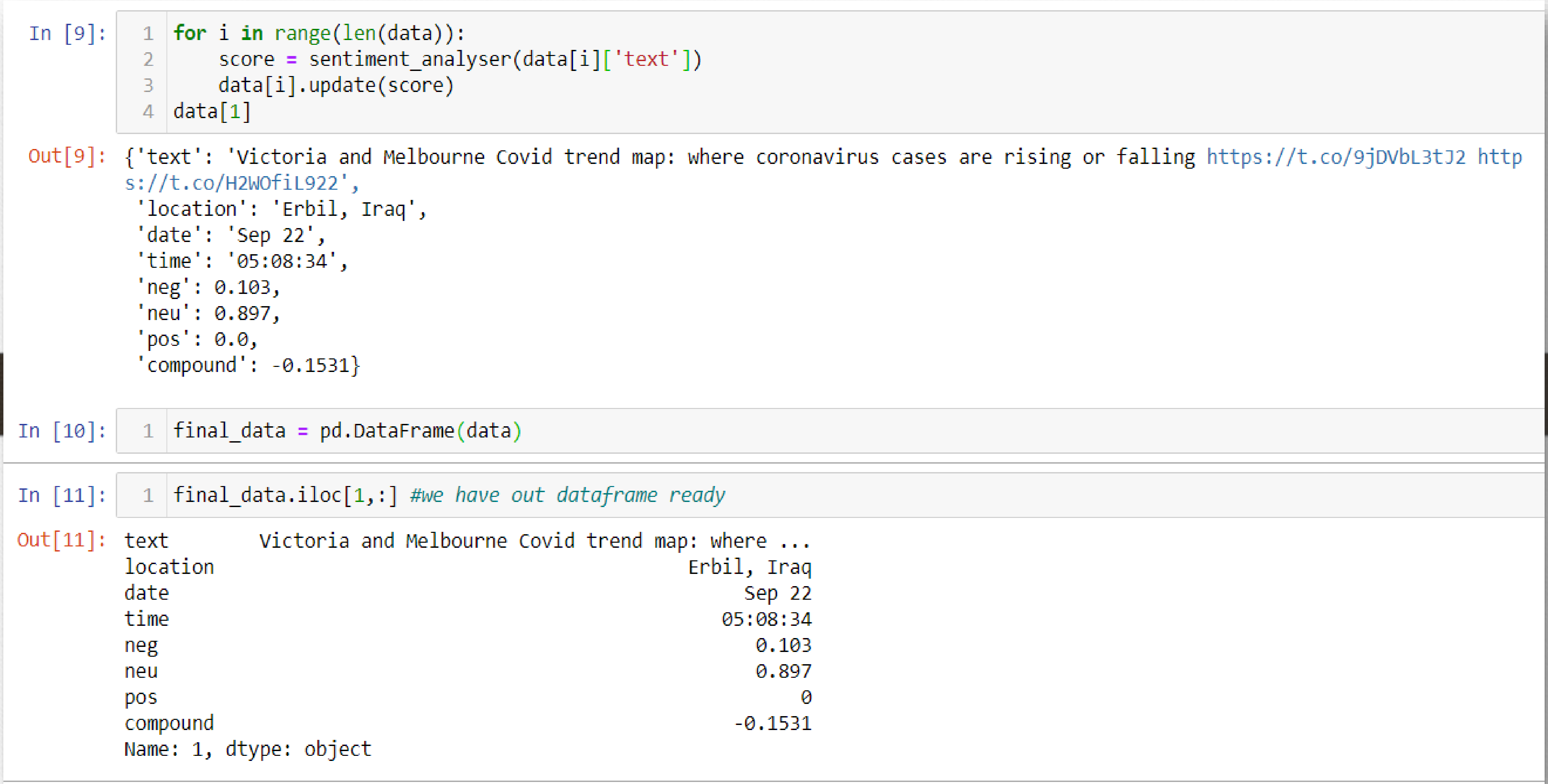
```
In [4]:
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```



```
In [5]:  #demo of analysing a tweet
         tweet = data[0]['text']

         def sentiment_analyser(text):
             score = SentimentIntensityAnalyzer().polarity_scores(text)
             return(score)
         sentiment_analyser(tweet)

Out[5]:  {'neg': 0.0, 'neu': 0.943, 'pos': 0.057, 'compound': 0.3182}
```

A dictionary of four columns from the sentiment scoring: Neu, Neg, Pos and compound. The first three represent the sentiment score percentage of each category in our headline, and the compound single number that scores the sentiment. `compound` ranges from -1 (Extremely Negative) to 1 (Extremely Positive).

We'll add these scores in our data and then make a Dataframe having columns as shown in next slide.



In [9]:
```
for i in range(len(data)):
    score = sentiment_analyser(data[i]['text'])
    data[i].update(score)
data[1]
```

Out[9]:
```
{'text': 'Victoria and Melbourne Covid trend map: where coronavirus cases are rising or falling https://t.co/9jDVbL3tJ2 http
s://t.co/H2WOfiL922',
 'location': 'Erbil, Iraq',
 'date': 'Sep 22',
 'time': '05:08:34',
 'neg': 0.103,
 'neu': 0.897,
 'pos': 0.0,
 'compound': -0.1531}
```

In [10]:
```
final_data = pd.DataFrame(data)
```

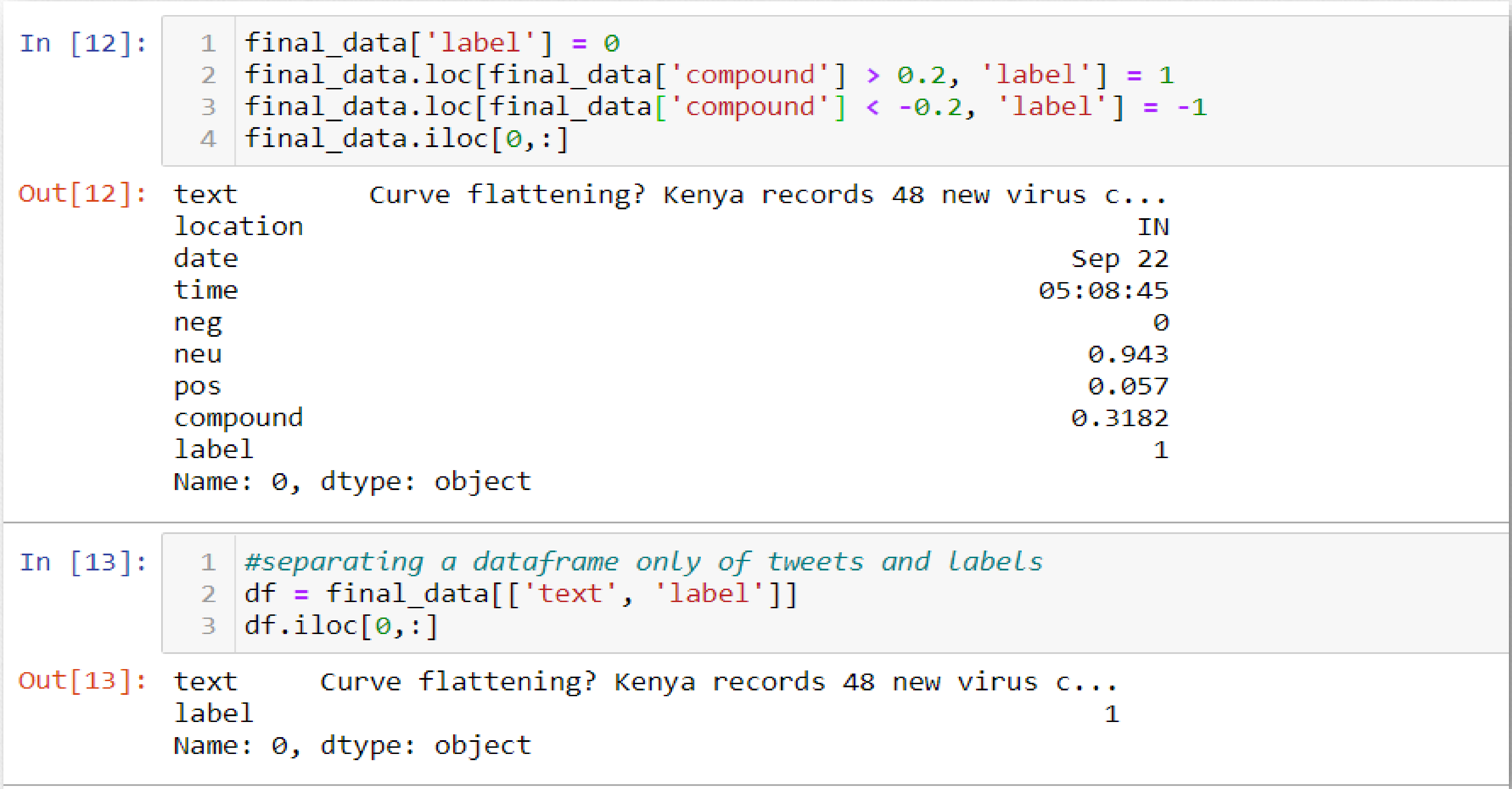
In [11]:
```
final_data.iloc[1,:] #we have out dataframe ready
```

Out[11]:
```
text        Victoria and Melbourne Covid trend map: where ...
location                                          Erbil, Iraq
date                                                   Sep 22
time                                                 05:08:34
neg                                                     0.103
neu                                                     0.897
pos                                                         0
compound                                              -0.1531
Name: 1, dtype: object
```

Labelling our data

We will consider posts with a compound value **greater than 0.2 as positive** and **less than -0.2 as negative**. There's some testing and experimentation that goes with choosing these ranges, and there is a trade-off to be made here. If you choose a higher value, you might get more compact results (less false positives and false negatives), but the size of the results will decrease significantly.

Let's create a positive label of 1 if the compound is greater than 0.2, and a label of -1 if compound is less than -0.2. Everything else will be 0.



```
In [12]: final_data['label'] = 0
         final_data.loc[final_data['compound'] > 0.2, 'label'] = 1
         final_data.loc[final_data['compound'] < -0.2, 'label'] = -1
         final_data.iloc[0,:]
```

```
Out[12]: text        Curve flattening? Kenya records 48 new virus c...
         location                                                   IN
         date                                                   Sep 22
         time                                                 05:08:45
         neg                                                         0
         neu                                                     0.943
         pos                                                     0.057
         compound                                               0.3182
         label                                                       1
         Name: 0, dtype: object
```

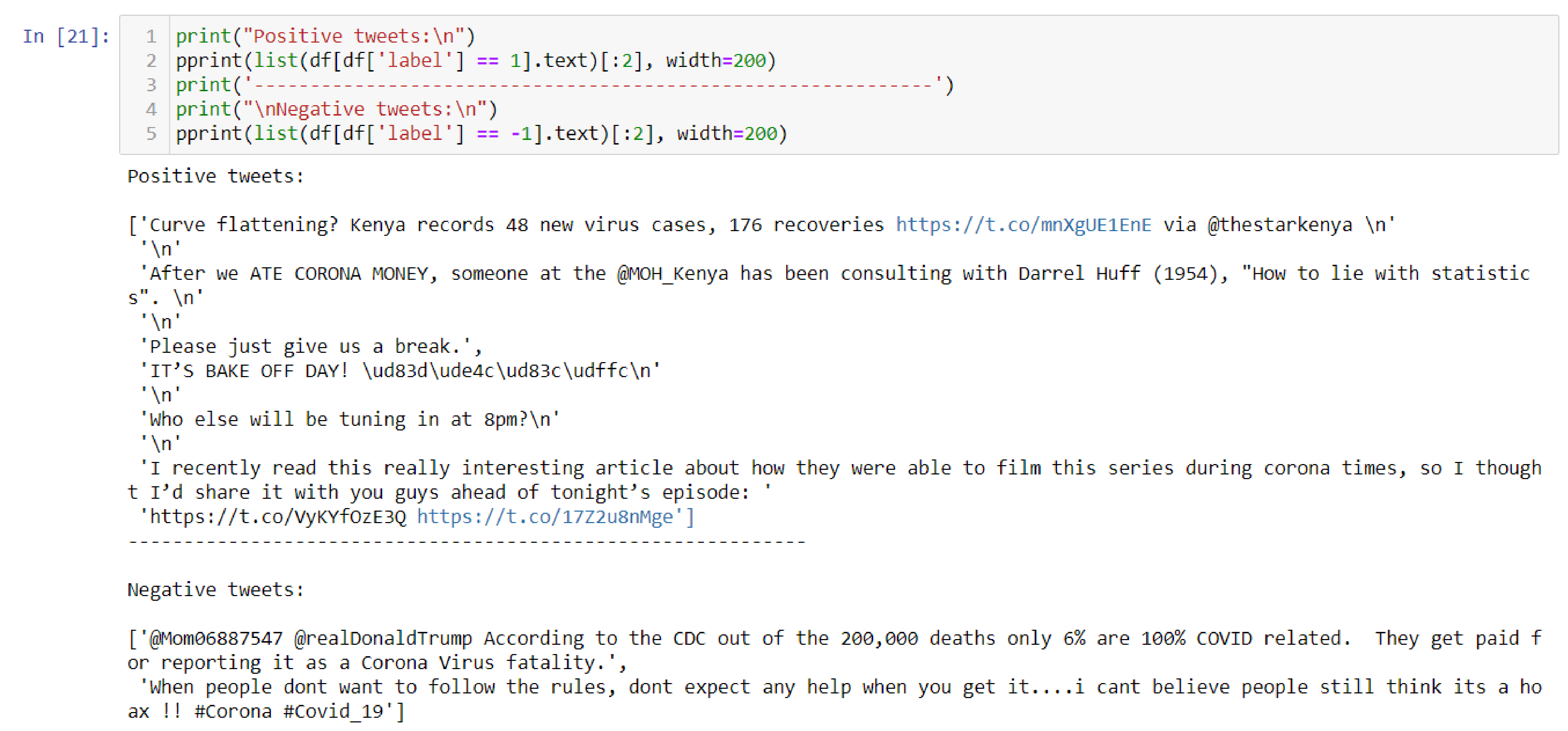
```
In [13]: #separating a dataframe only of tweets and labels
         df = final_data[['text', 'label']]
         df.iloc[0,:]
```

```
Out[13]: text     Curve flattening? Kenya records 48 new virus c...
         label                                                    1
         Name: 0, dtype: object
```

Dataset Info and Statistics

We have all the data we need to save, so let’s now see what this data wants to tell us!

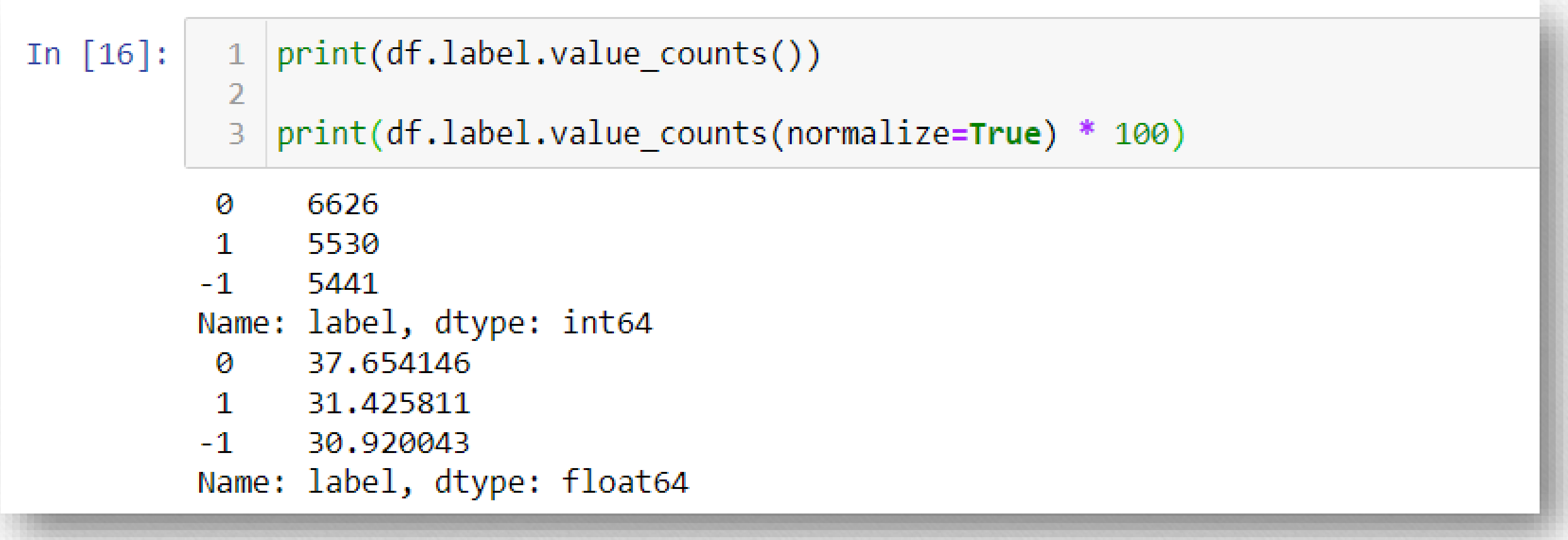
- Let’s look into some positive and negative tweets first!! – as per our model of classification.



```
In [21]:
print("Positive tweets:\n")
pprint(list(df[df['label'] == 1].text)[:2], width=200)
print('-------------------------------------------------------------')
print("\nNegative tweets:\n")
pprint(list(df[df['label'] == -1].text)[:2], width=200)
```

```
Positive tweets:

['Curve flattening? Kenya records 48 new virus cases, 176 recoveries https://t.co/mnXgUE1EnE via @thestarkenya \n'
 '\n'
 'After we ATE CORONA MONEY, someone at the @MOH_Kenya has been consulting with Darrel Huff (1954), "How to lie with statistic
s". \n'
 '\n'
 'Please just give us a break.',
 'IT’S BAKE OFF DAY! \ud83d\ude4c\ud83c\udffc\n'
 '\n'
 'Who else will be tuning in at 8pm?\n'
 '\n'
 'I recently read this really interesting article about how they were able to film this series during corona times, so I though
t I’d share it with you guys ahead of tonight’s episode: '
 'https://t.co/VyKYfOzE3Q https://t.co/17Z2u8nMge']
-------------------------------------------------------------

Negative tweets:

['@Mom06887547 @realDonaldTrump According to the CDC out of the 200,000 deaths only 6% are 100% COVID related.  They get paid f
or reporting it as a Corona Virus fatality.',
 'When people dont want to follow the rules, dont expect any help when you get it....i cant believe people still think its a ho
ax !! #Corona #Covid_19']
```

Now let's check how many total positives and negatives we have in this dataset:



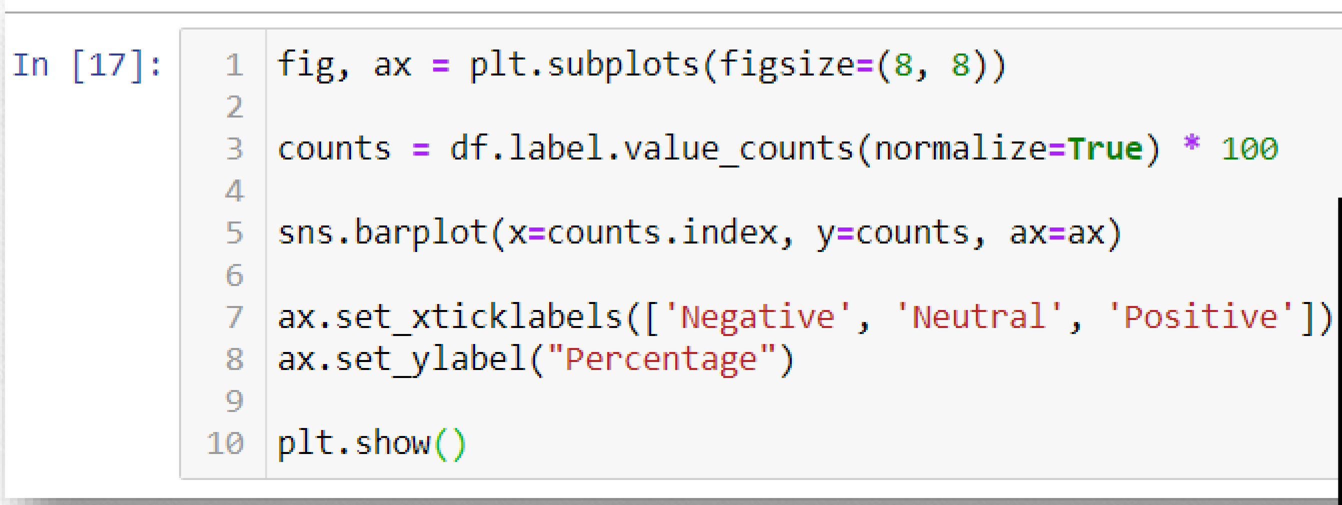
```
In [16]:
print(df.label.value_counts())

print(df.label.value_counts(normalize=True) * 100)
```

```
 0    6626
 1    5530
-1    5441
Name: label, dtype: int64
 0    37.654146
 1    31.425811
-1    30.920043
Name: label, dtype: float64
```

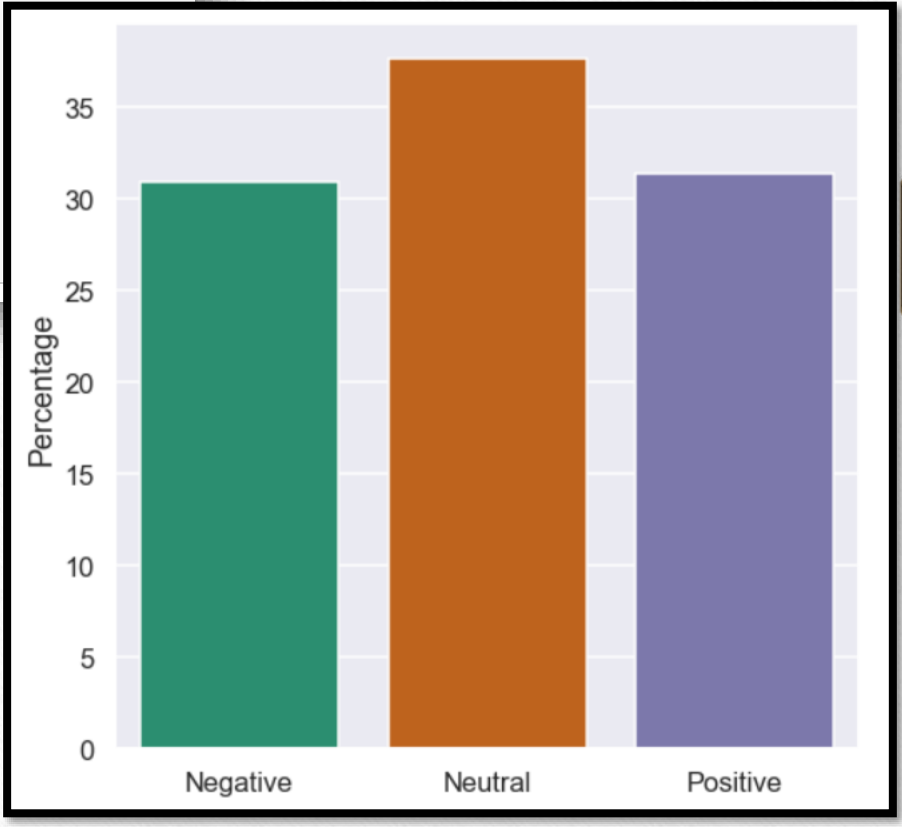
The first line gives us raw value counts of the labels, whereas the second line provides percentages with the normalize keyword.

The first line gives us raw value counts of the labels, whereas the second line provides percentages with the `normalize` keyword.

Visualizing mindsets from Tweets in Neg, Pov, Neu :



```
In [17]:
fig, ax = plt.subplots(figsize=(8, 8))

counts = df.label.value_counts(normalize=True) * 100

sns.barplot(x=counts.index, y=counts, ax=ax)

ax.set_xticklabels(['Negative', 'Neutral', 'Positive'])
ax.set_ylabel("Percentage")

plt.show()
```



The large number of neutral tweets is due to two main reasons:

1. The assumption that we made earlier where tweets with compound value between 0.2 and -0.2 are considered neutral. The higher the margin, the larger the number of neutral headlines.
2. We used general lexicon to categorize political news. The more correct way is to use a political-specific lexicon, but for that we would either need a human to manually label data, or we would need to find a custom lexicon already made.

What tweets Say?

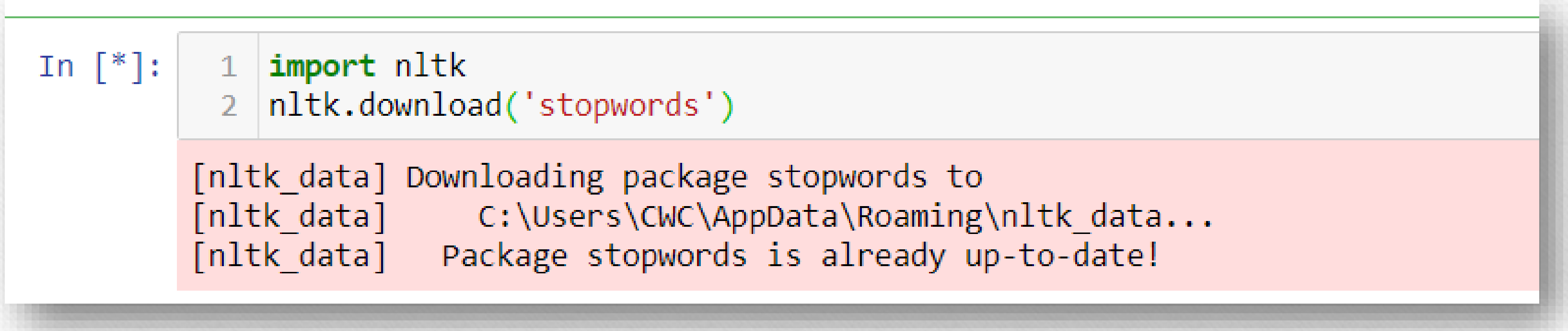
- An interesting observation is the number of negative tweets, which could be attributed to the media’s behavior, such as the exaggeration of titles for clickbait which is also a possibility which made people negative in minds and so our analyzer produced a lot of pressurized negatives.
- There's definitely places to explore for improvements, but let's move on for now and develop our analyzer for detecting emotions behind the positive and negative sense.

Emotion Analyzer

The different types of emotions we are using for analysis are namely:

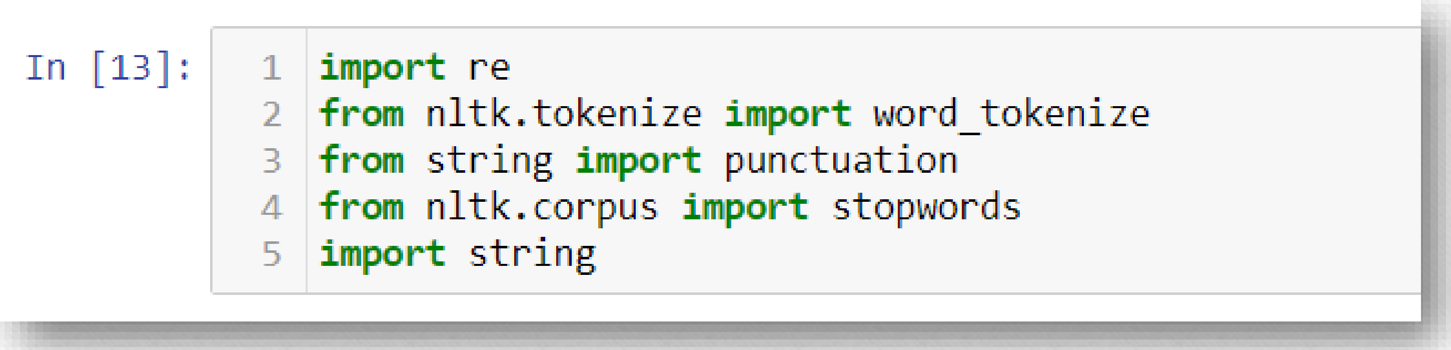
‘Fear’, ‘Happy’, ‘Sad’, ‘Attracted’, ‘Focused’, ‘Powerless’, ‘Hate’, ‘Loved’, ‘Cheated’, ‘Alone’, ‘Angry’, ‘Bored’, ‘Esteemed’, ‘Attached’, ‘Independent’, ‘Co-dependent’, ‘Embarrassed’, ‘Powerless’, ‘Surprise’, ‘Fearless’, ‘Safe’, ‘Adequate’, ‘Belittled’, ‘Apathetic’, ‘Obsessed’, ‘Anxious’ and some more which we have used in a separate file named ‘‘emotions.txt’’

Installing and Importing our needs!!



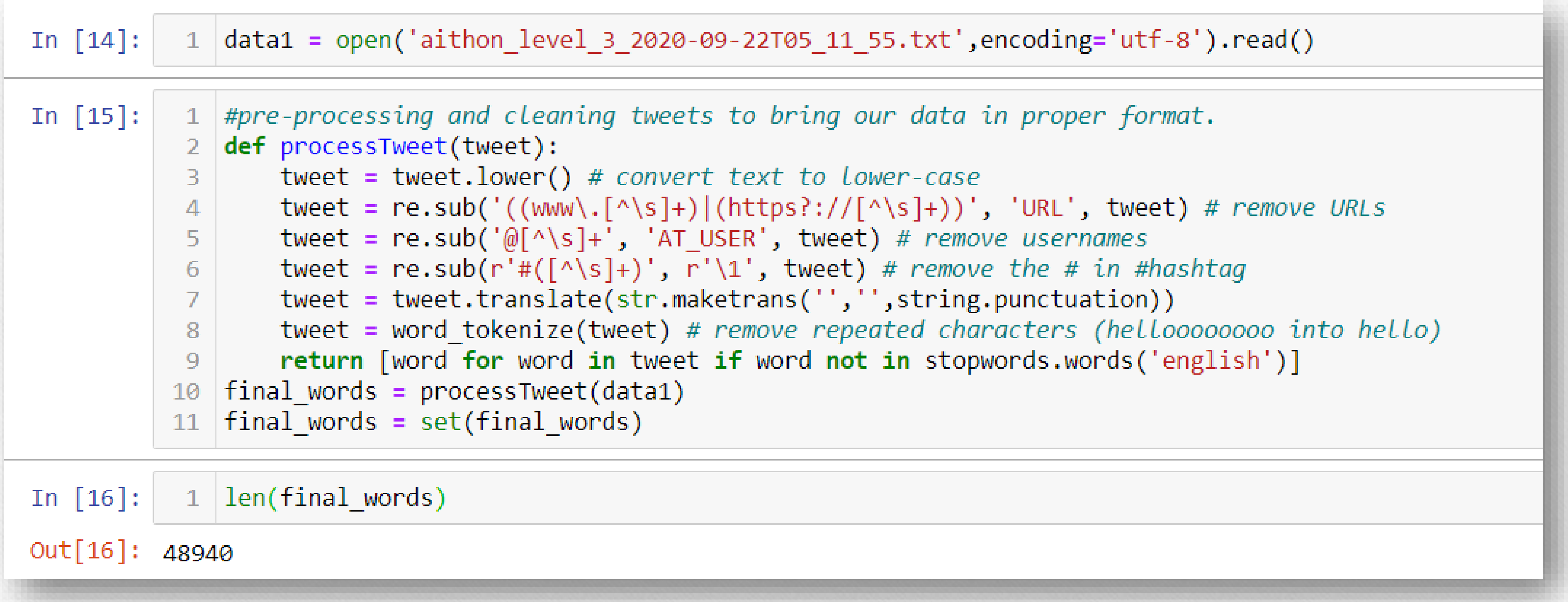
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\CWC\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```



```
import re
from nltk.tokenize import word_tokenize
from string import punctuation
from nltk.corpus import stopwords
import string
```

Loading and Processing our Data

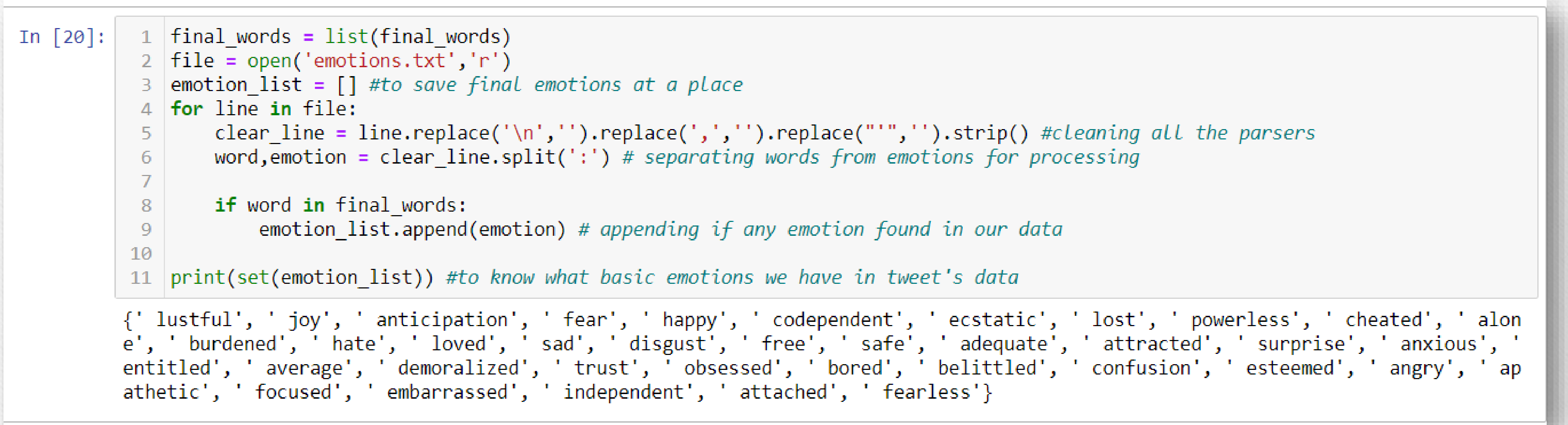


```
In [14]: data1 = open('aithon_level_3_2020-09-22T05_11_55.txt',encoding='utf-8').read()
```

```
In [15]: #pre-processing and cleaning tweets to bring our data in proper format.
         def processTweet(tweet):
             tweet = tweet.lower() # convert text to lower-case
             tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', 'URL', tweet) # remove URLs
             tweet = re.sub('@[^\s]+', 'AT_USER', tweet) # remove usernames
             tweet = re.sub(r'#([^\s]+)', r'\1', tweet) # remove the # in #hashtag
             tweet = tweet.translate(str.maketrans('','',string.punctuation))
             tweet = word_tokenize(tweet) # remove repeated characters (helloooooooo into hello)
             return [word for word in tweet if word not in stopwords.words('english')]
         final_words = processTweet(data1)
         final_words = set(final_words)
```

```
In [16]: len(final_words)

Out[16]: 48940
```

Emotion-set found in our Dataset

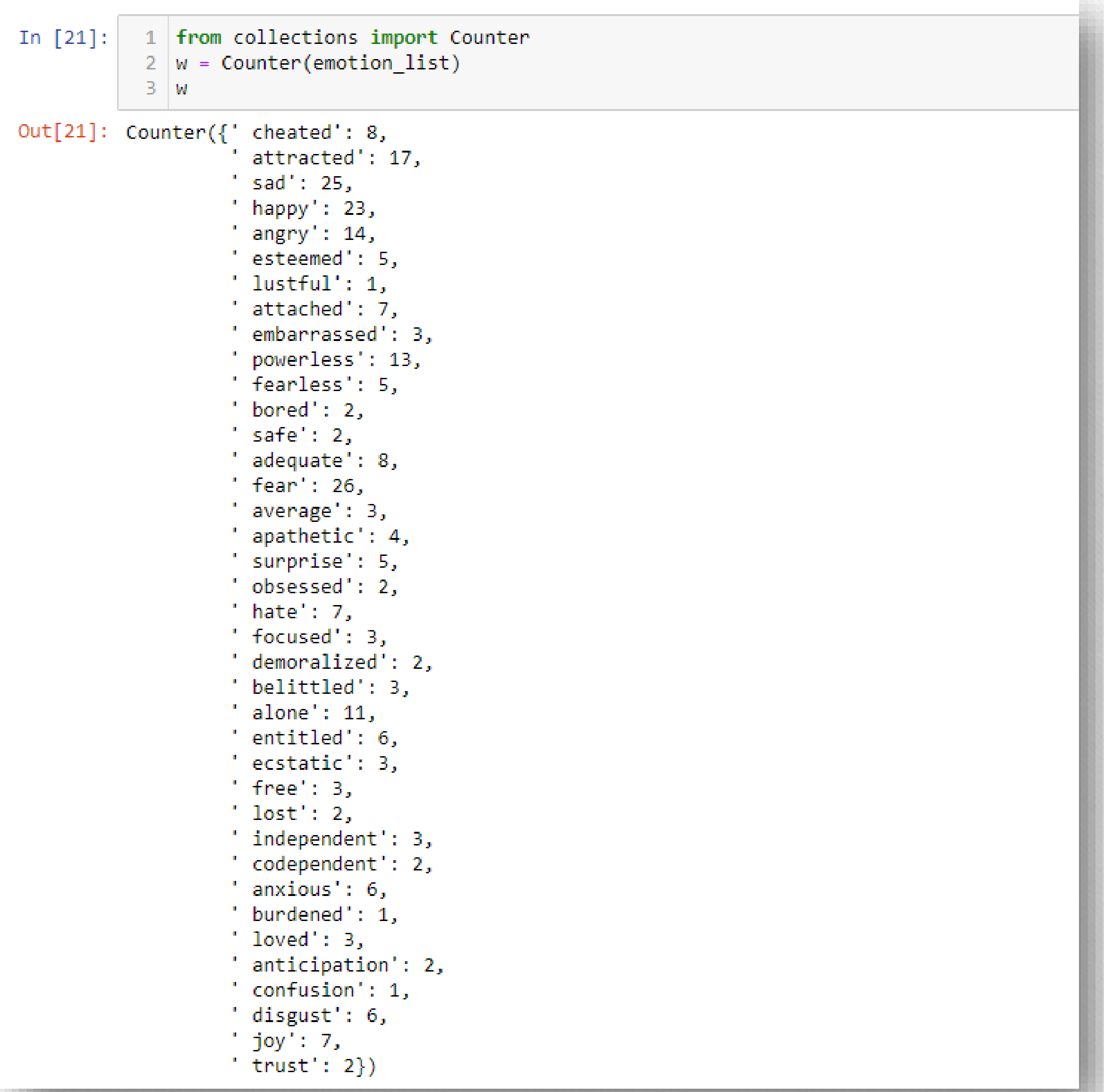


In [20]:

```
final_words = list(final_words)
file = open('emotions.txt','r')
emotion_list = [] #to save final emotions at a place
for line in file:
    clear_line = line.replace('\n','').replace(',','').replace("'",'').strip() #cleaning all the parsers
    word,emotion = clear_line.split(':') # separating words from emotions for processing

    if word in final_words:
        emotion_list.append(emotion) # appending if any emotion found in our data

print(set(emotion_list)) #to know what basic emotions we have in tweet's data
```

```
{' lustful', ' joy', ' anticipation', ' fear', ' happy', ' codependent', ' ecstatic', ' lost', ' powerless', ' cheated', ' alon
e', ' burdened', ' hate', ' loved', ' sad', ' disgust', ' free', ' safe', ' adequate', ' attracted', ' surprise', ' anxious', '
entitled', ' average', ' demoralized', ' trust', ' obsessed', ' bored', ' belittled', ' confusion', ' esteemed', ' angry', ' ap
athetic', ' focused', ' embarrassed', ' independent', ' attached', ' fearless'}
```

Counting Emotions to plot them.

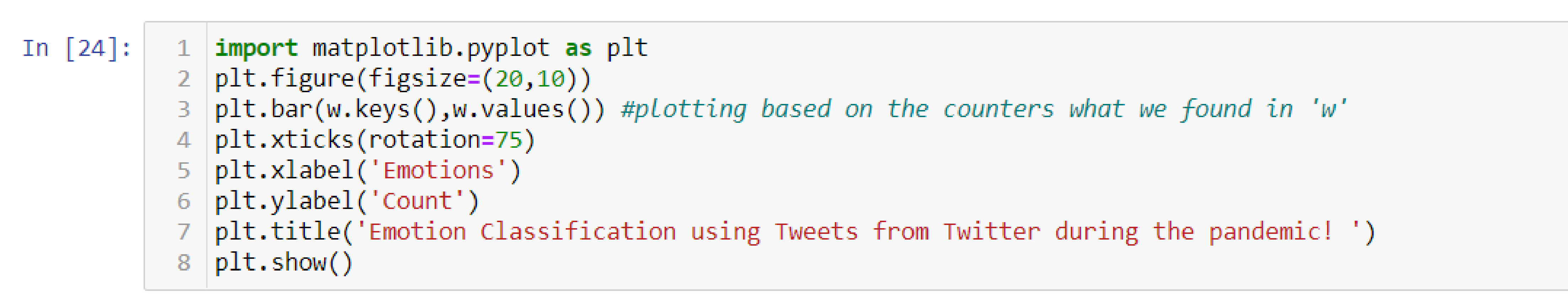


In [21]:
```
from collections import Counter
w = Counter(emotion_list)
w
```

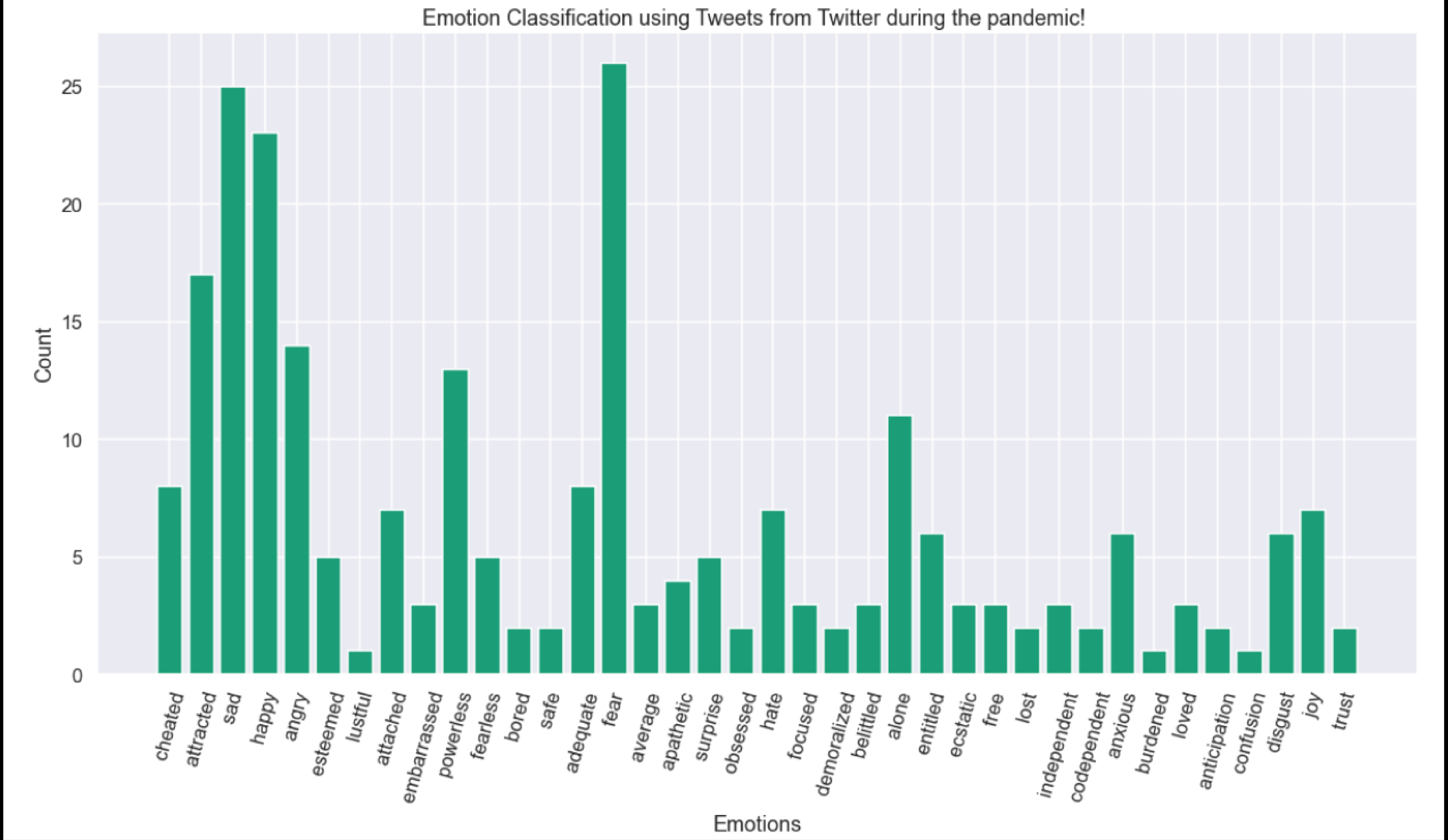
Out[21]:
```
Counter({' cheated': 8,
         ' attracted': 17,
         ' sad': 25,
         ' happy': 23,
         ' angry': 14,
         ' esteemed': 5,
         ' lustful': 1,
         ' attached': 7,
         ' embarrassed': 3,
         ' powerless': 13,
         ' fearless': 5,
         ' bored': 2,
         ' safe': 2,
         ' adequate': 8,
         ' fear': 26,
         ' average': 3,
         ' apathetic': 4,
         ' surprise': 5,
         ' obsessed': 2,
         ' hate': 7,
         ' focused': 3,
         ' demoralized': 2,
         ' belittled': 3,
         ' alone': 11,
         ' entitled': 6,
         ' ecstatic': 3,
         ' free': 3,
         ' lost': 2,
         ' independent': 3,
         ' codependent': 2,
         ' anxious': 6,
         ' burdened': 1,
         ' loved': 3,
         ' anticipation': 2,
         ' confusion': 1,
         ' disgust': 6,
         ' joy': 7,
         ' trust': 2})
```

Plotting our data in bars.

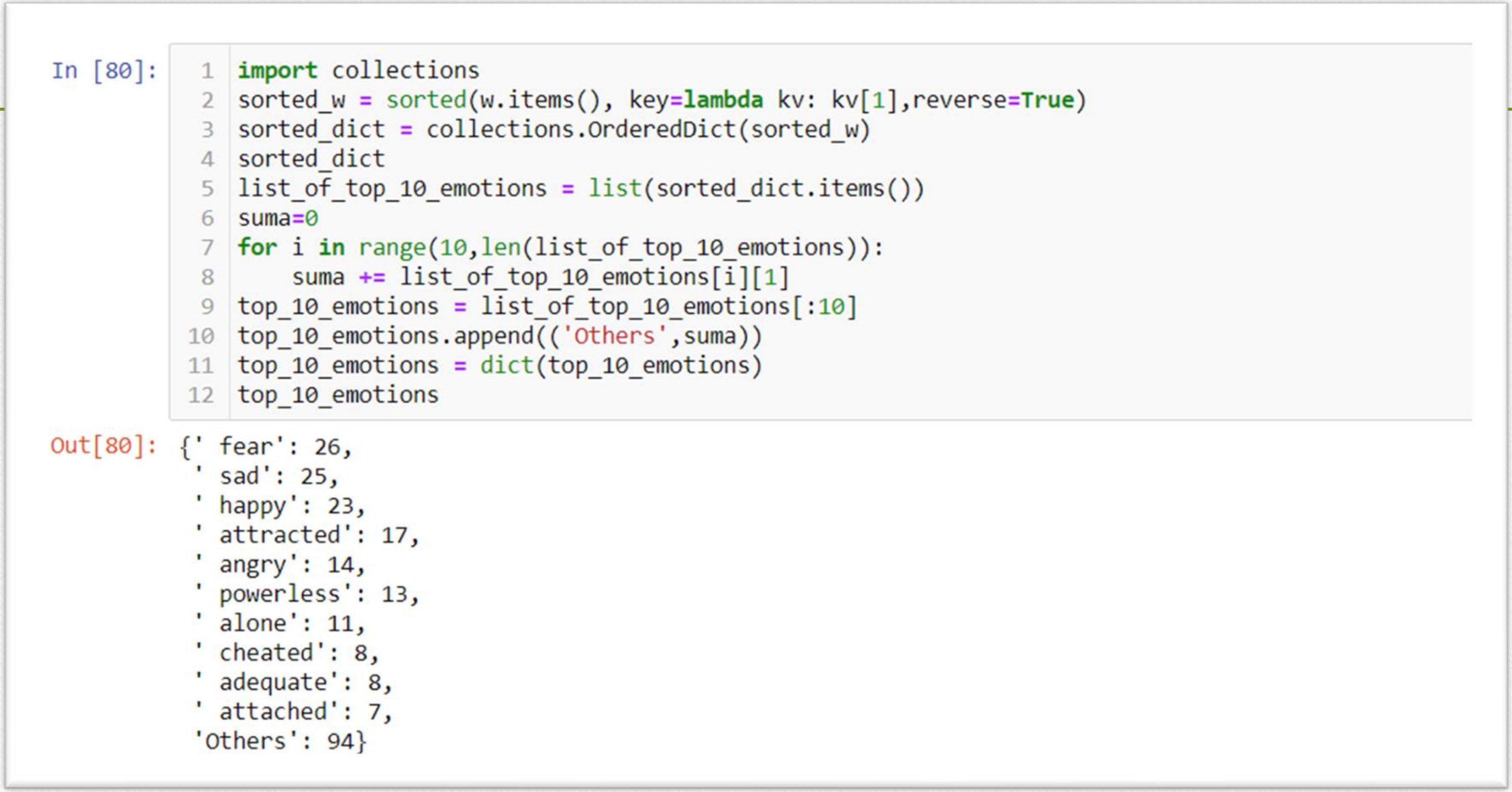
By this plot we can analyse how people feel and portray their mental health in the statements/gestures they make on Twitter which was the ultimate goal of our work.



```python
import matplotlib.pyplot as plt
plt.figure(figsize=(20,10))
plt.bar(w.keys(),w.values()) #plotting based on the counters what we found in 'w'
plt.xticks(rotation=75)
plt.xlabel('Emotions')
plt.ylabel('Count')
plt.title('Emotion Classification using Tweets from Twitter during the pandemic! ')
plt.show()
```


Emotion Classification using Tweets from Twitter during the pandemic!
Count
0
5
10
15
20
25
cheated
attracted
sad
happy
angry
esteemed
lustful
attached
embarrassed
powerless
fearless
bored
safe
adequate
fear
average
apathetic
surprise
obsessed
hate
focused
demoralized
belittled
alone
entitled
ecstatic
free
lost
independent
codependent
anxious
burdened
loved
anticipation
confusion
disgust
joy
trust
Emotions

Analysing for top 10 Emotions



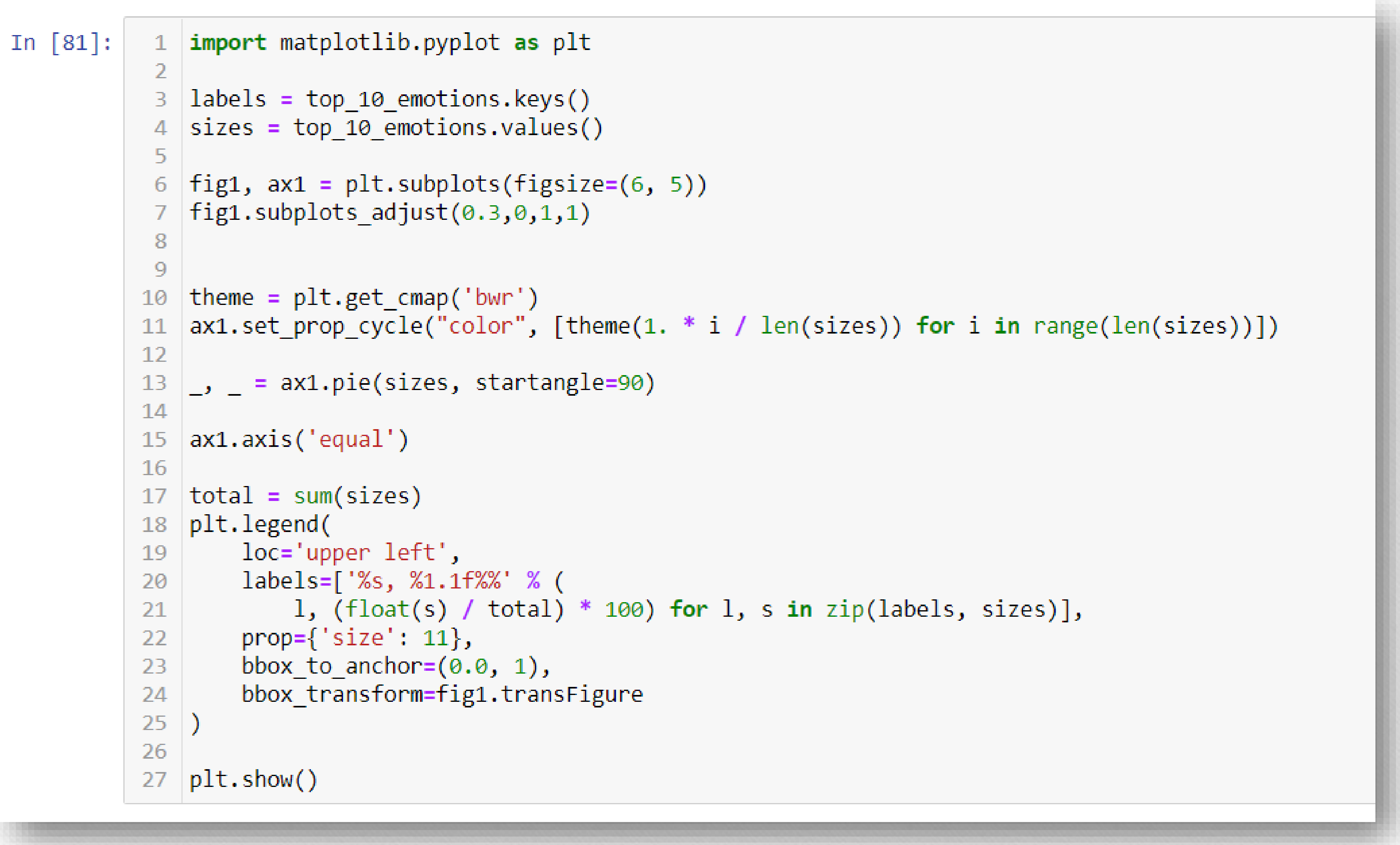
In [80]:

```python
import collections
sorted_w = sorted(w.items(), key=lambda kv: kv[1],reverse=True)
sorted_dict = collections.OrderedDict(sorted_w)
sorted_dict
list_of_top_10_emotions = list(sorted_dict.items())
suma=0
for i in range(10,len(list_of_top_10_emotions)):
    suma += list_of_top_10_emotions[i][1]
top_10_emotions = list_of_top_10_emotions[:10]
top_10_emotions.append(('Others',suma))
top_10_emotions = dict(top_10_emotions)
top_10_emotions
```

Out[80]:

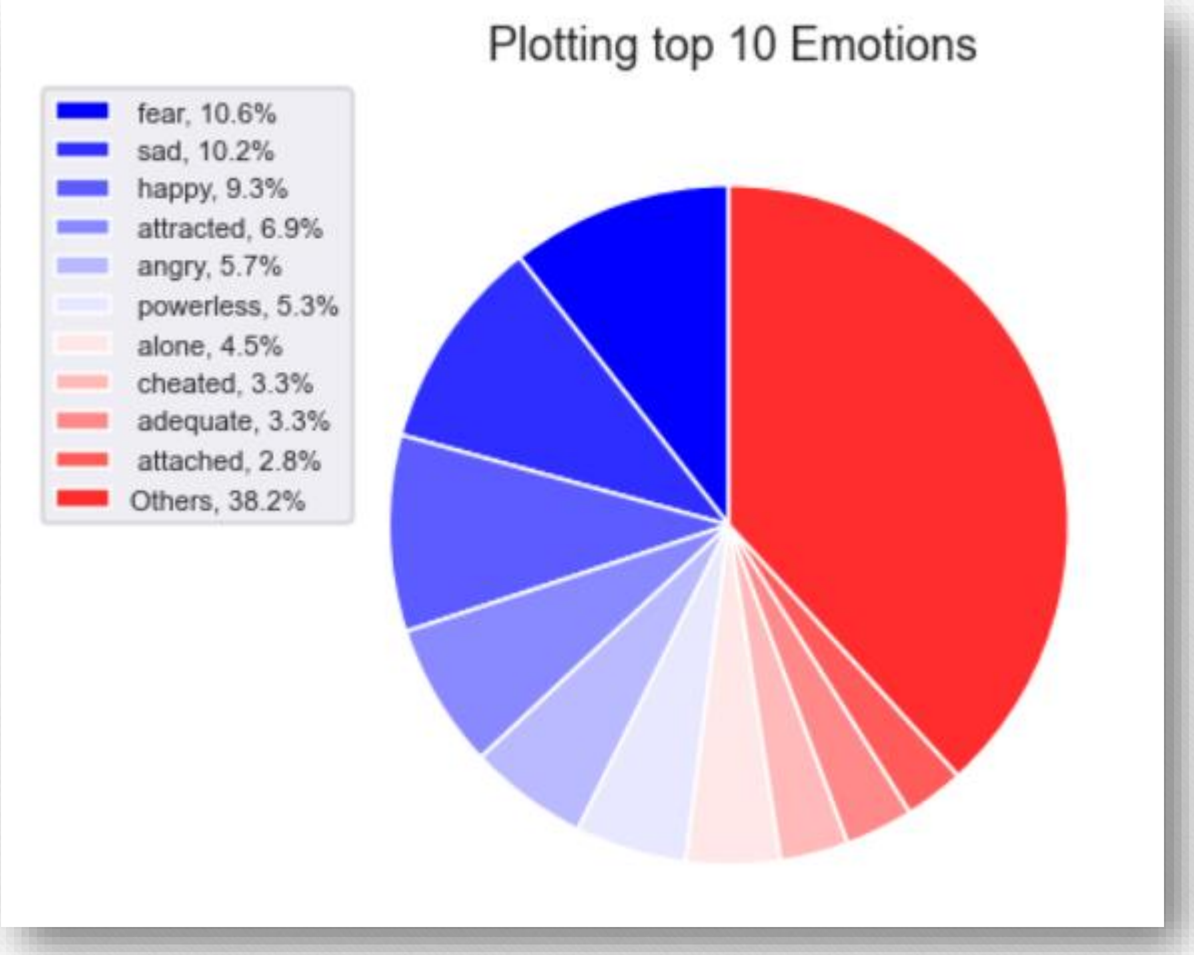
```
{' fear': 26,
 ' sad': 25,
 ' happy': 23,
 ' attracted': 17,
 ' angry': 14,
 ' powerless': 13,
 ' alone': 11,
 ' cheated': 8,
 ' adequate': 8,
 ' attached': 7,
 'Others': 94}
```

Plotting a Pie!



In [81]:

```
import matplotlib.pyplot as plt

labels = top_10_emotions.keys()
sizes = top_10_emotions.values()

fig1, ax1 = plt.subplots(figsize=(6, 5))
fig1.subplots_adjust(0.3,0,1,1)


theme = plt.get_cmap('bwr')
ax1.set_prop_cycle("color", [theme(1. * i / len(sizes)) for i in range(len(sizes))])

_, _ = ax1.pie(sizes, startangle=90)

ax1.axis('equal')

total = sum(sizes)
plt.legend(
    loc='upper left',
    labels=['%s, %1.1f%%' % (
        l, (float(s) / total) * 100) for l, s in zip(labels, sizes)],
    prop={'size': 11},
    bbox_to_anchor=(0.0, 1),
    bbox_transform=fig1.transFigure
)

plt.show()
```



By this we can conclude:

- The top emotion during the pandemic was fear with 10.6% of total emotions in our dataset.
- Second, Third top emotions are ‘Sad’ and ‘Happy’ with 10.2% and 9.3% resp.
- Mentally, people are hit hard by the pandemic and we tried to demonstrate as much as we could.

Thank You!