```
In [28]: #warnings :)
         import warnings
         warnings.filterwarnings('ignore')

In [13]: #Creating bunch of sentences
         raw_docs = ["I am writing some very basic english sentences",
         "I'm just writing it for the demo PURPOSE to make audience understand the basics .",
         "The point is to _learn HOW it works_ on #simple # data."]

In [14]: #importing nltk package
         import nltk

In [15]: #nltk.download()

         #python -m nltk.downloader all
```

```
#display the csv file
pd.set_option ('display.max_colwidth', 300)
merged_df = pd.read_csv(r'merged.csv', encoding='latin1', sep='delimiter', engine='python')
merged_df.columns = ['text']
merged_df
```

By using the command: *merged_df.text.duplicated().sum()*, the number of tweets that are double is visible, and those are a total of '56.739'. Thus, meaning that from the '58.454' tweets, only '1.715' are originals. The originals tweets are being displayed and transferred to a new data frame named 'cleaned_df'. This is done for practical reasons and to be easier saved in a new csv file that will contain only the originals and will be used for the rest of the data cleaning process. The name of this csv is 'cleaned.csv'.

```
#display the original tweets
merged_df.drop_duplicates(keep='first')
```

0	Study: #Bots amplified pandemic #misinformation on #socialmediaÂ https://t.co/SUQQJWwCDE via @medcitynews #digitalhealth #medicalchatbot #ehealthInfo #...
1	"RT @MonarchinMN: New Report Confirms How #Iranâ□□s Regime #FakeNews And #Propagand...
2	https://t.co/BuWaIlsLPxÂ #infosec #cybersec #threat...
3	@condesm @jgnaredo @Katsuragui Seguro x ser un criadero de #bots y propagar #Fake...
4	"RT @carlos2015aceve: Lo dijo @JhonnyKandanga en @Notic...
...	
58318	"@mtgreenee Republican lies ...
58319	Republican conspiracy theories ...
58320	Republican insurrection u...
58321	Republicans are spreading more COVID misinformation than ever before. Like no one has eve...
58322	* Please RT * Vote Democrat for Ame...

1715 rows × 1 columns

```
cleaned_df = merged_df.drop_duplicates(keep='first')
```

```
#save to csv
cleaned_df.to_csv('cleaned.csv',encoding='utf-8', index =False)
csv = 'cleaned.csv'
```

Step 1 - convert to lower case

In [17]:
```
import string
raw_docs = [doc.lower() for doc in raw_docs]
print(raw_docs)
```

```
['i am writing some very basic english sentences', "i'm just writing it for the demo pu
audience understand the basics .", 'the point is to _learn how it works_ on #simple # d
```

```
text = re.sub(r'RT[\s]+', '', text) # Removing RT
text = re.sub(r'https?:\/\/\S+', '', text) # Remove the hyper link
```

Step 2 - Tokenization

In [18]:
```
# word tokenize
from nltk.tokenize import word_tokenize
tokenized_docs = [word_tokenize(doc) for doc in raw_docs]
print(tokenized_docs)

print("##########################################################################################")

#Sentence tokenization

from nltk.tokenize import sent_tokenize
sent_token = [sent_tokenize(doc) for doc in raw_docs]
print(sent_token)
```

Step 3 - Punctuation Removal

In [19]:
```
# Removing punctuation
import re
regex = re.compile('[%s]' % re.escape(string.punctuation)) #see documentation here: http://docs.python

tokenized_docs_no_punctuation = []

for review in tokenized_docs:
    new_review = []
    for token in review:
        new_token = regex.sub(u'', token)
        if not new_token == u'':
            new_review.append(new_token)

    tokenized_docs_no_punctuation.append(new_review)

print(tokenized_docs_no_punctuation)
```

Step 4 - Removing Stopwords

In [20]:
```
# Cleaning text of stopwords
from nltk.corpus import stopwords

tokenized_docs_no_stopwords = []

for doc in tokenized_docs_no_punctuation:
    new_term_vector = []
    for word in doc:
        if not word in stopwords.words('english'):
            new_term_vector.append(word)

    tokenized_docs_no_stopwords.append(new_term_vector)

print(tokenized_docs_no_stopwords)
```

```
[['writing', 'basic', 'english', 'sentences'], ['writing', 'demo', 'purpose', 'make',
derstand', 'basics'], ['point', 'learn', 'works', 'simple', 'data']]
```

Step 5- Stemming and Lemmatization

In [21]:
```
# Stemming and Lemmatization
from nltk.stem.porter import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

porter = PorterStemmer()
wordnet = WordNetLemmatizer()

preprocessed_docs = []

for doc in tokenized_docs_no_stopwords:
    final_doc = []
    for word in doc:
        final_doc.append(porter.stem(word))
        #final_doc.append(wordnet.lemmatize(word))

    preprocessed_docs.append(final_doc)

print(preprocessed_docs)
```

→ Stemming

→ Lemmatisieren