**Analyse des sentiments**

**Introduction :**

Le but de ce travail est de réaliser un programme qui reçoit un texte puis détecte si ce texte est positive ou négative. Pour le réaliser, on va utiliser une base de données qui contient 5000 tweets positifs et 5000 tweets négatifs. Et on va utiliser ces tweets pour apprendre à un modèle de classifier les tweets entre négatifs et positifs. Ce travail va être ensuite utilisé pour classifier des commentaires de Youtube entre négatifs et positifs.

Le fait d’analyser et de comprendre des tweets et les séparer entre positifs et négatifs relève du domaine du traitement du langage naturel : *Natural Language Processing (NLP)*.

Python dispose de plusieurs librairies qui aident à analyser les données textuelles. Dans ce projet, on va utiliser NLTK : *Natural Language Toolkit.*

Le fichier jupyter est nommé Sentiments.ipynb.

Le fichier python est nommé Sentiments.py.

**Installation de NLTK et préparation des données :**

On installe NLTK par la commande suivante  : pip install nltk.

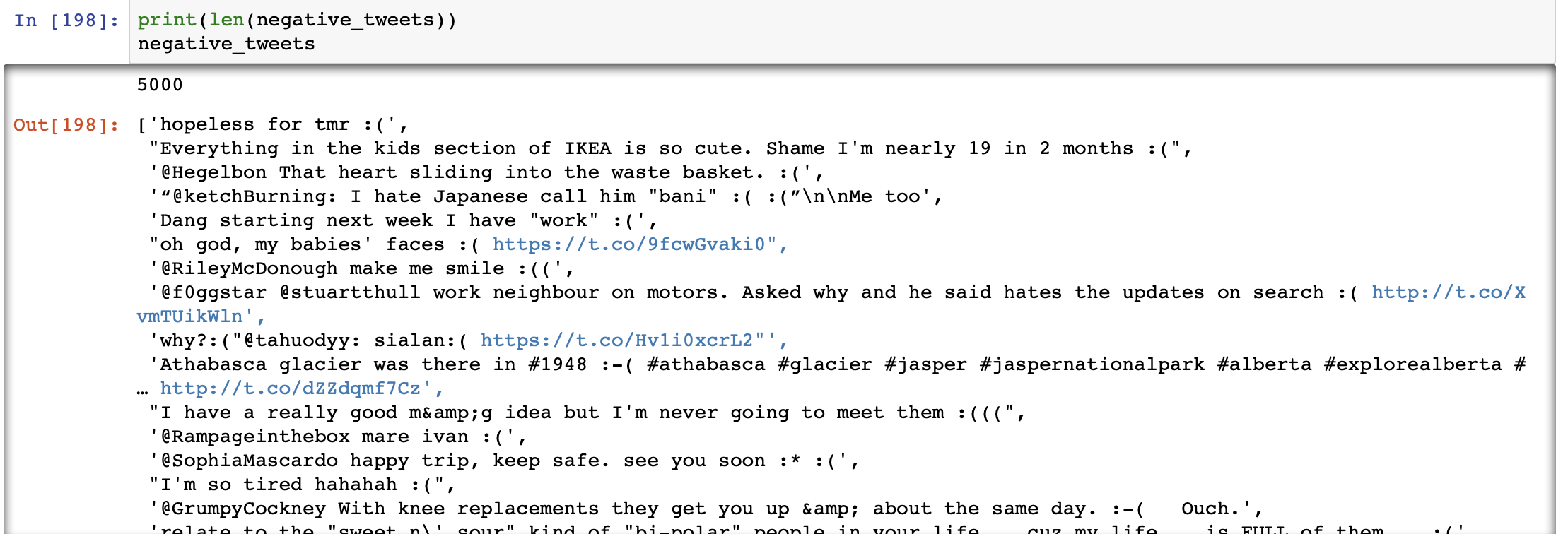
NLTK dispose de plusieurs échantillons de tweets. Et on peut importer ces échantillons, qu’ils ont un sentiment positif, négatif ou non définis.

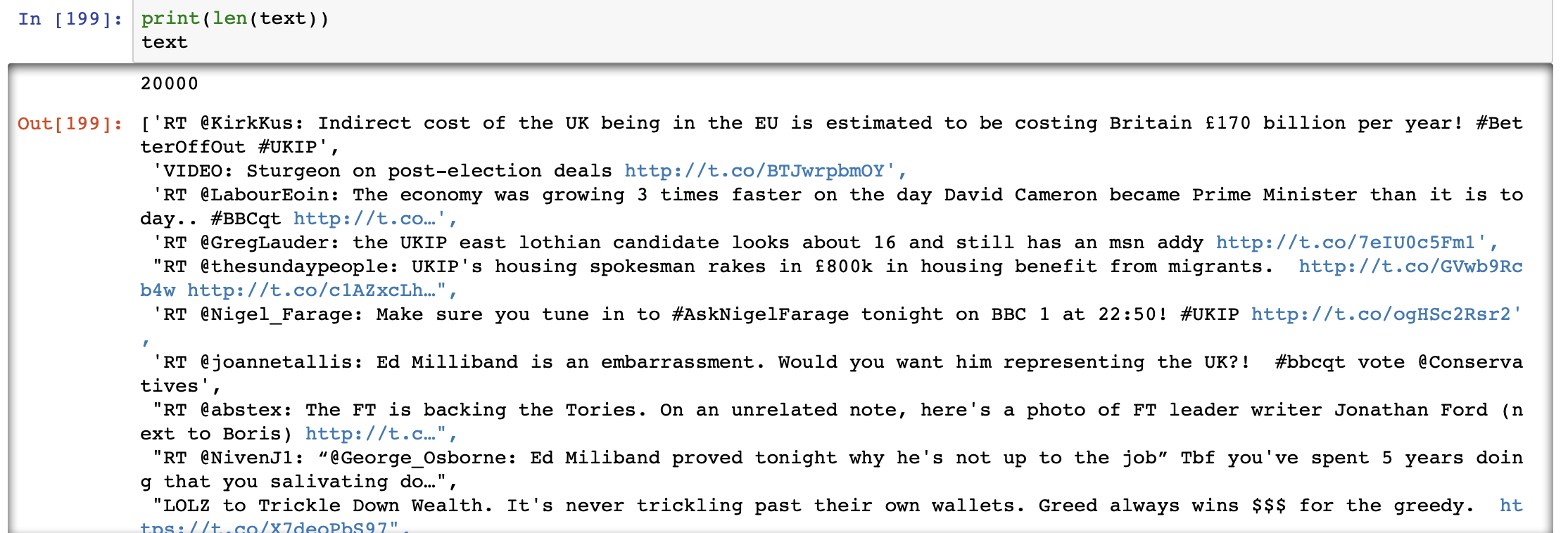


Pour chacun de ces échantillons, on affiche combien de tweets il contient, et ces tweets :

Pour les tweets positifs : 

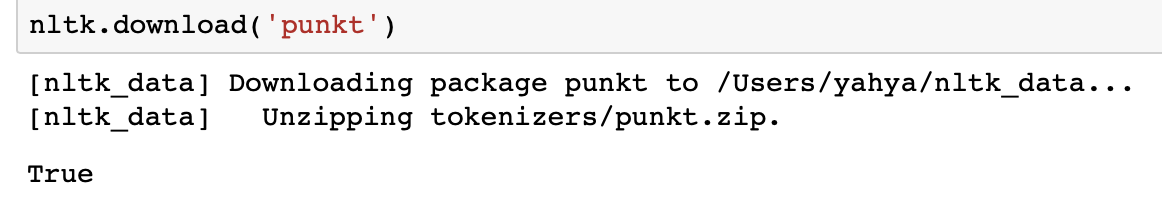
Pour les tweets négatifs :



Pour les tweets sans sentiment définis :

Pour faciliter le travail du classifieur, il faut qu’on organise les tweets. On fait cela en faisant en sorte que chaque tweet est rendu comme une liste avec chaque mot un élément de la liste.

Nltk dispose de la méthode .tokenized qui prend une phrase et la sépare automatiquement.

Pour utiliser cette fonction, on télécharge d’abord le module punkt :

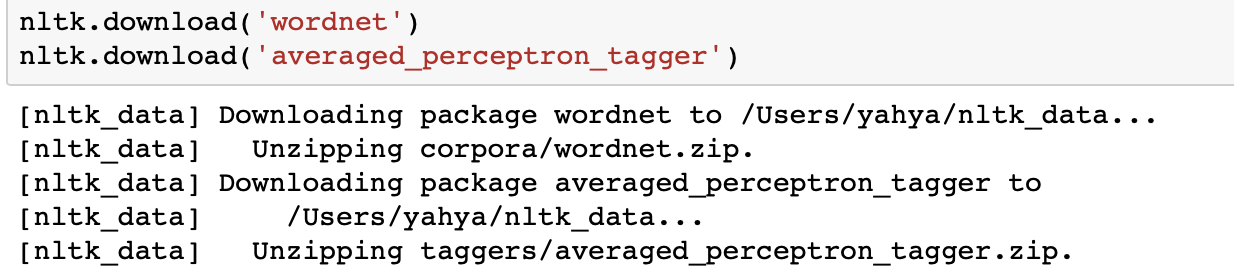


La fonction rend chaque échantillon comme une liste de liste.

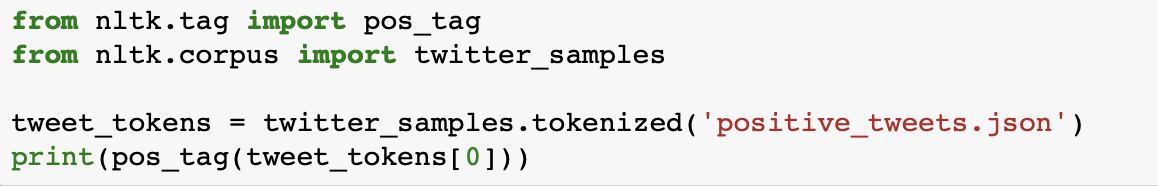
Après on doit rendre les mots en leur forme basique, par exemple un verbe conjugué est rendu à l’infinitif. Pour faire cela, *nltk* dispose du module *wordnet* qui est une base de données de la langue anglaise et qui donc aide à détermine la forme basique d’un mot.

Et aussi du module *averaged\_perceptron\_tagger* qui détermine le contexte du mot dans la phrase.

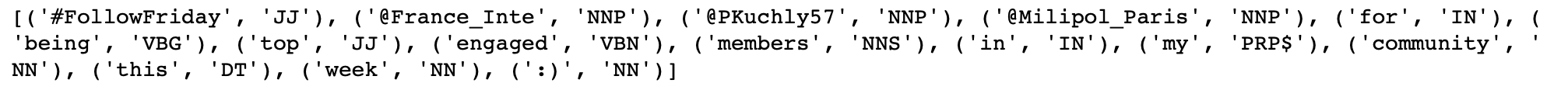
On télécharge ces deux modules :



Puis, on peut importer la fonction pos\_tag qui nous donne le contexte de chaque mot d’une phrase, par exemple : nom, verbe, adjectif…



On utilisant la fonction sur une phrase, ça nous retourne :



Chaque mot est suivit de son contexte.

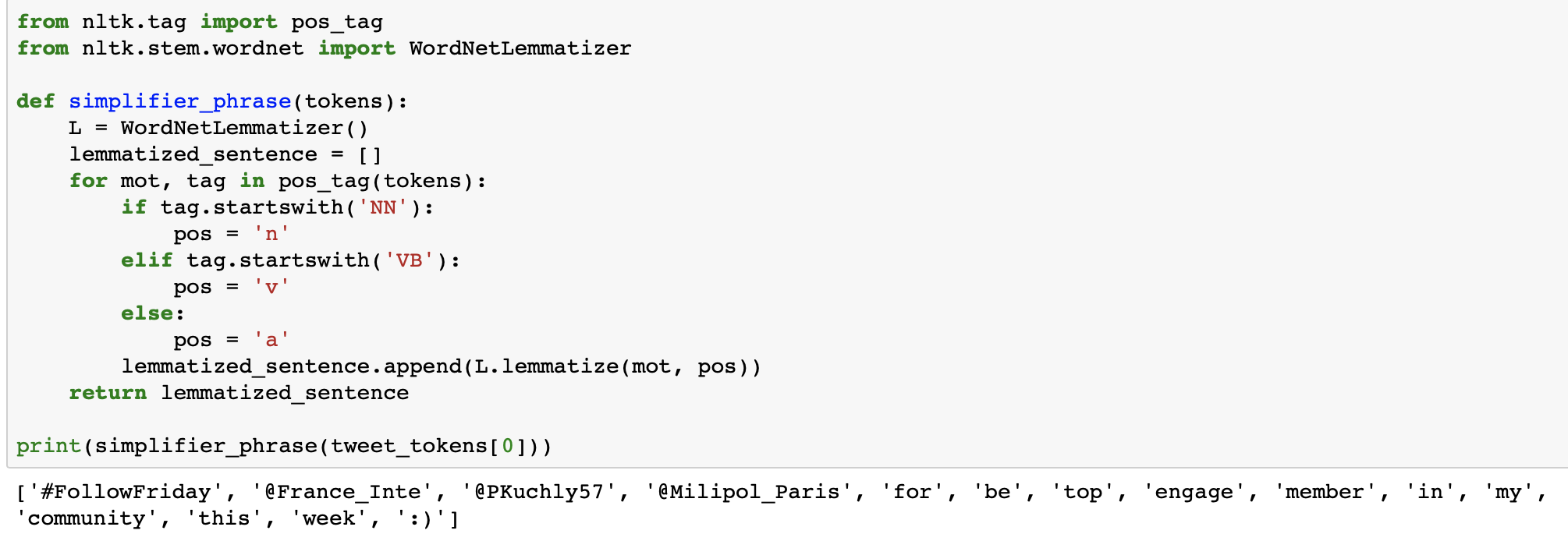
‘JJ’ représente adjectif.

Si un tag commence par ‘NN’ alors généralement il représente un nom.

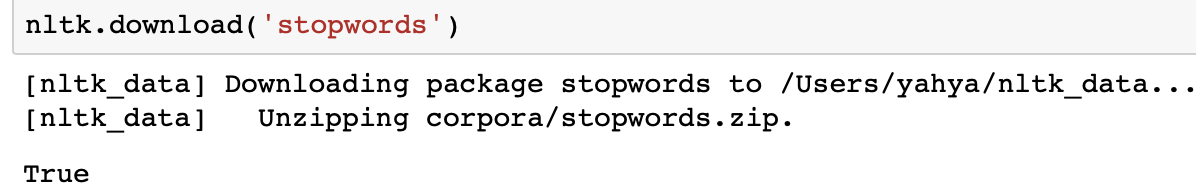
Et si il commence par un ‘VB’ alors il représente un verbe.

Le module *WordNetLemmatizer* de Nltk dispose de la fonction *lemmatize* qui reçoit un mot et sa position (si nom, verbe ou adjectif) et retourne sa forme basique.

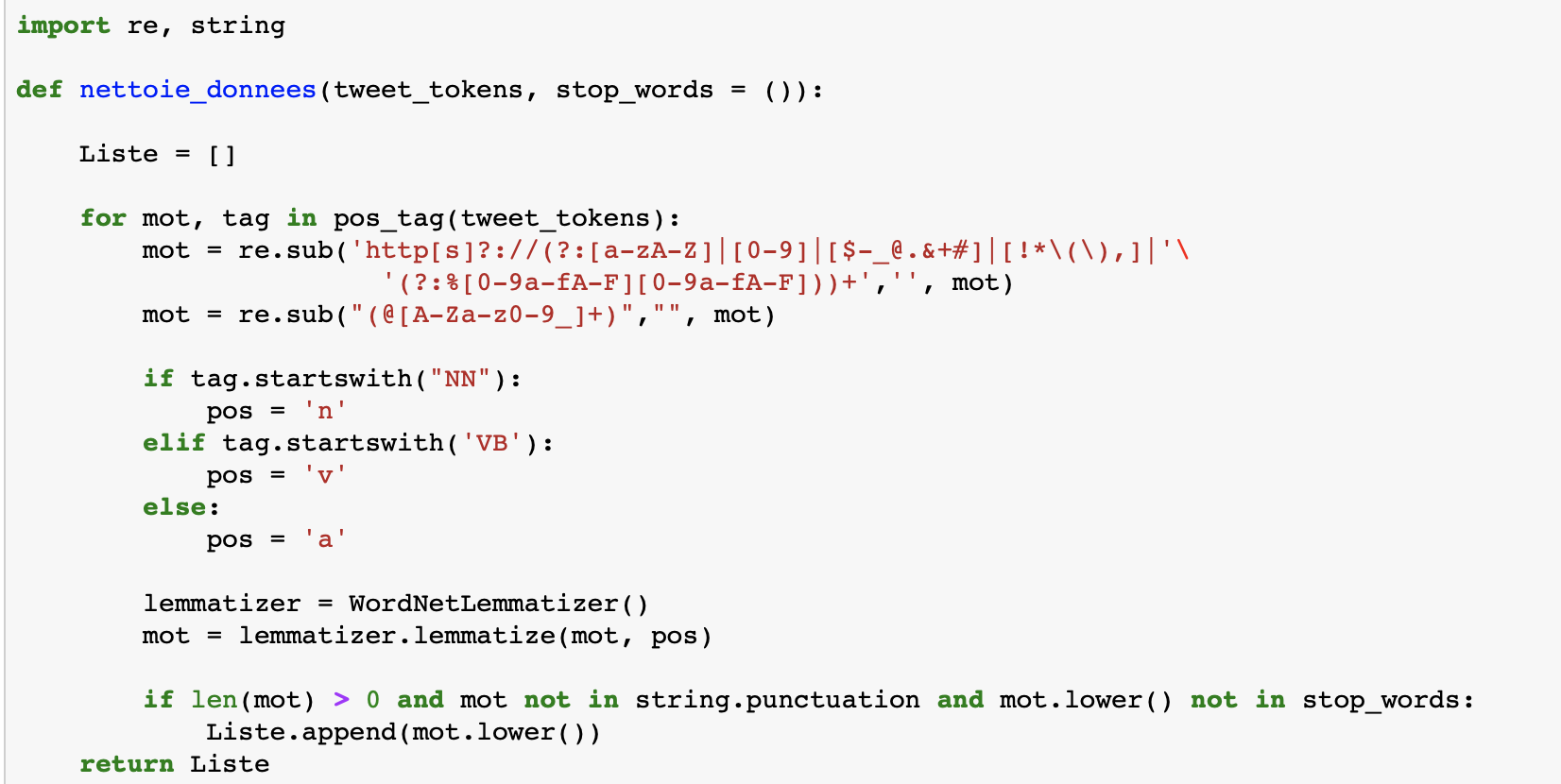
On crée alors la fonction simplifier phrase quireçoit une phrase déjà séparé puis rend tous les mots de la phrase en leurs forme basique et on la teste sur une phrase :

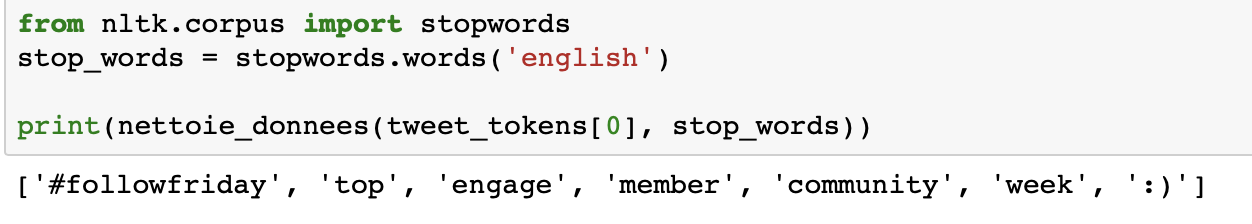


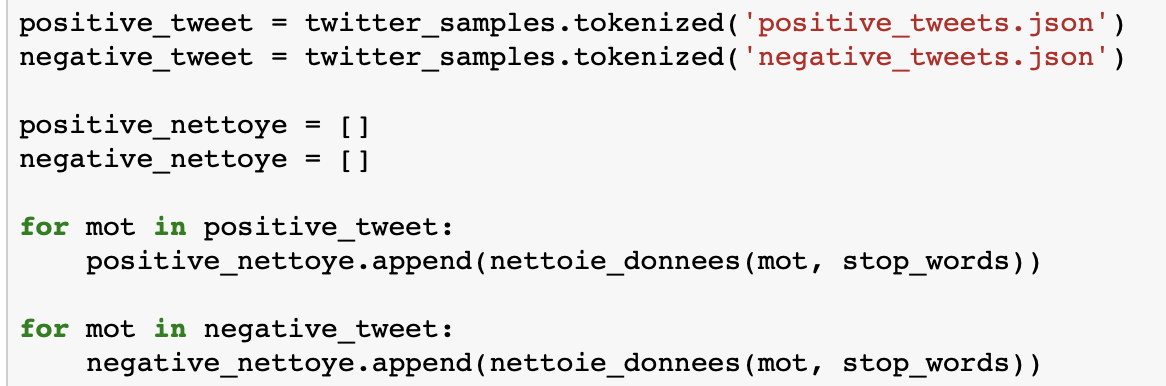
Après on nettoie nos données, on enlevant les hyperliens, les noms d’utilisateurs Twitter précédés par @, les ponctuations, les caractères spéciaux ou les mots inutiles ( par exemple « and » ou « or » ….).

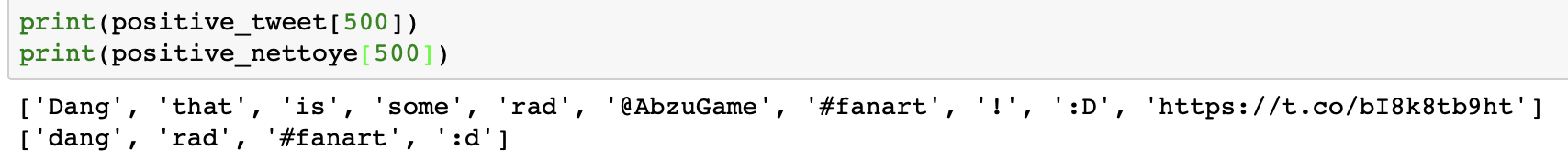
Ces mots sont regroupés dans une libraire de nltk nommée stopwords. On l’installe :

Pour optimiser, on crée une fonction qui à la fois nettoie les données et rend les mots en leur forme basique.

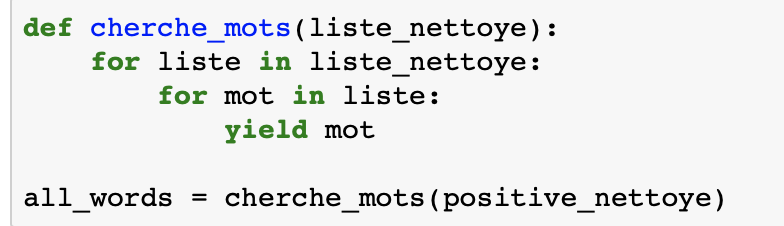
On appelle cette fonction nettoie\_donnees : 

On teste la fonction sur une phrase : 

Après, on fait ce travails sur les échantillons positifs et négatifs, c.-à-d. on sépare chaque tweet mot par mot, et après on les nettoie : 

Pour voir la différence, on choisit un tweet au hasard : 

Si on veut voir quels sont les mots les plus fréquents dans l’échantillon positif nettoyé :

On crée d’abord une fonction qui cherche tous les mots dans l’échantillon: 

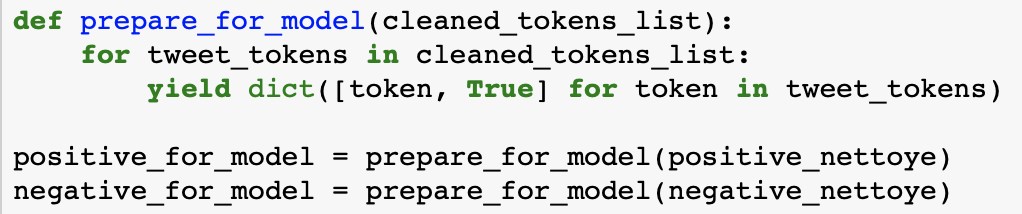
Puis on importe le module FreqDist de nltk :



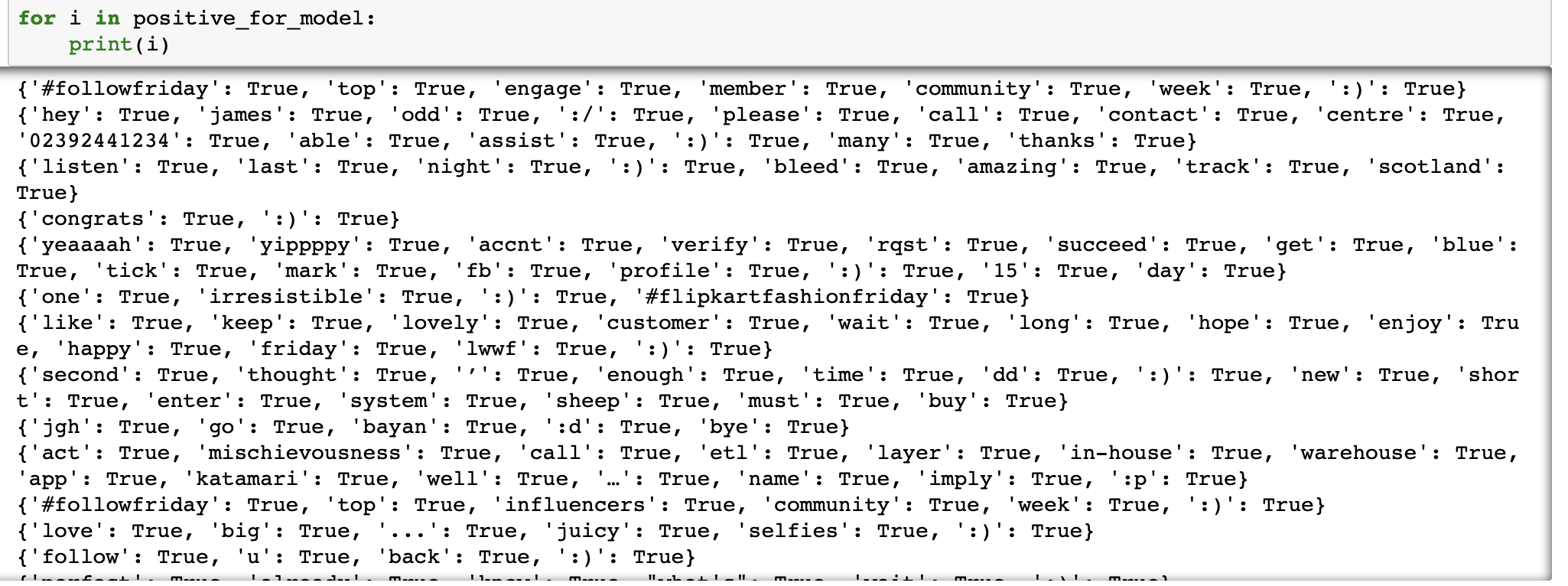
On connaît alors quels sont les mots les plus fréquents.

**Préparation du modèle :**

On prépare les données en les transformant en des dictionnaires en préparation à l’utilisation du classifieur de Bayes.



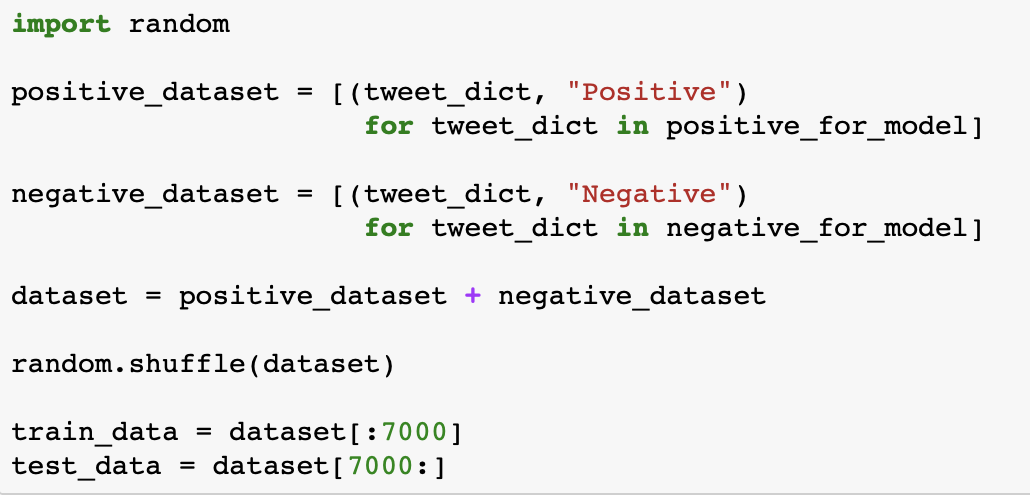
Les échantillons deviennent alors sous forme d’un générateur de dictionnaire avec comme clé les mots et comme valeur True :



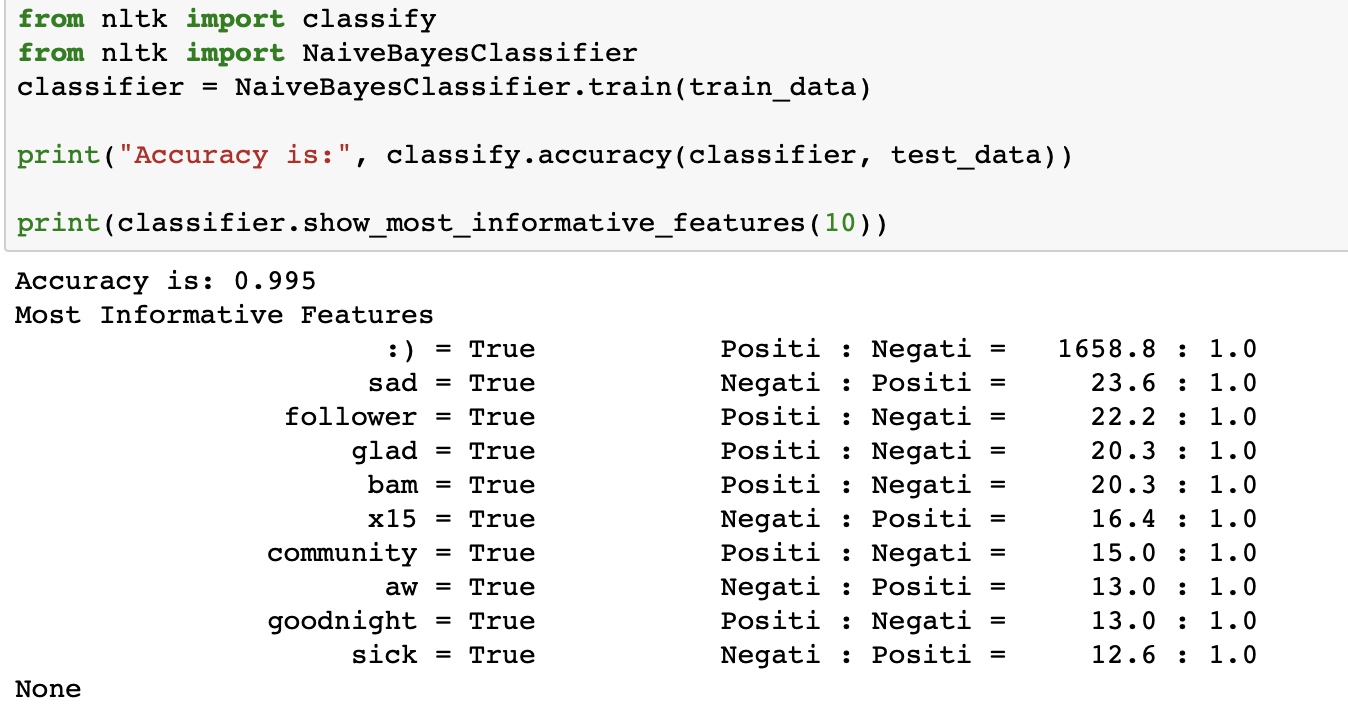
Après, on crée notre *Dataset* qui contient les données positifs et ceux négatifs.

Pour éviter d’avoir du biais, on mélange le champ de données de manière aléatoire.

Après on le sépare entre données d’entrainement du modèle et données d’essai avec le ratio 70 :30.



Après on importe le classifieur, et on l’utilise :



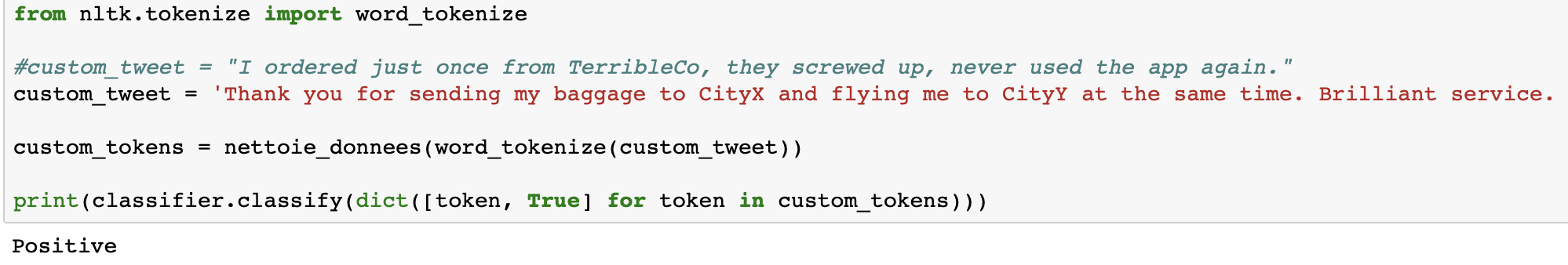
On a alors que le taux de précision sur les données d’essai est de 99.5%

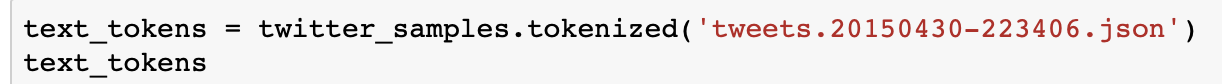
Et on a aussi les mots qui ont une préférence élevée pour un des sentiments :

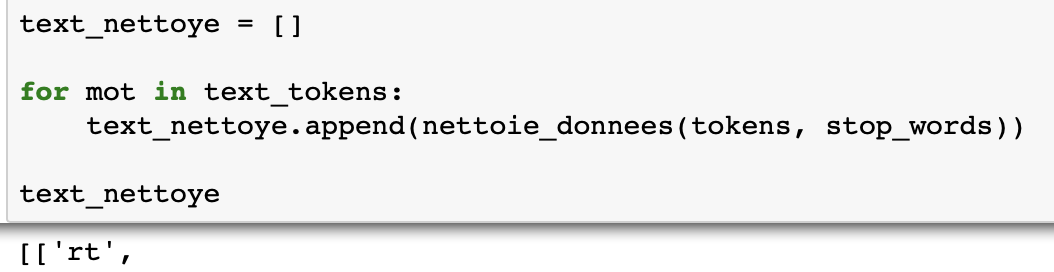
Par exemple : :) a 1650 apparitions positifs pour chaque apparition négatifs.

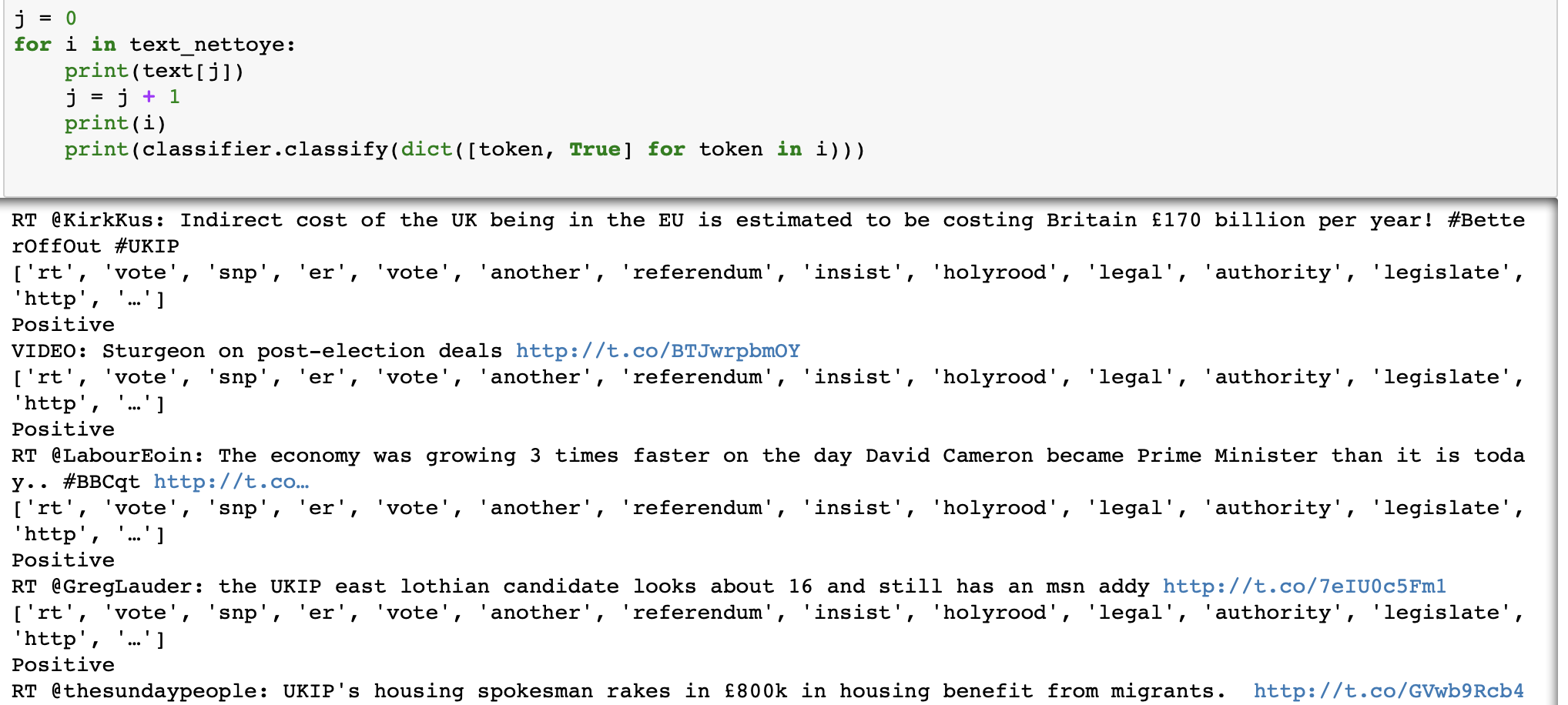
L’un des défauts des classifieurs est qu’ils sont incapables de comprendre du sarcasme,

Par exemple, il catégorise cette phrase comme positif, alors qu’elle est négatif :



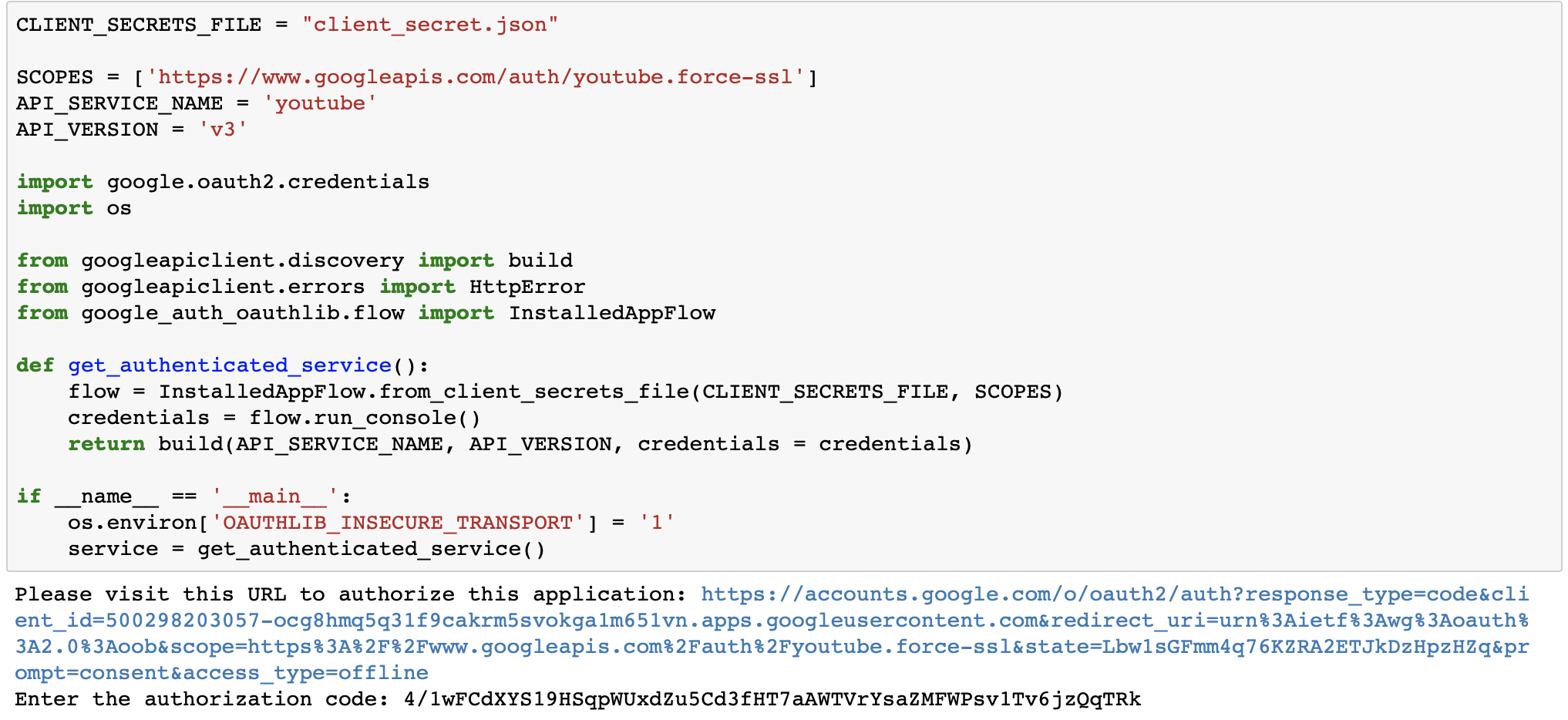
On prépare l’échantillon non définis pour l’entrer dans le classifieur : 



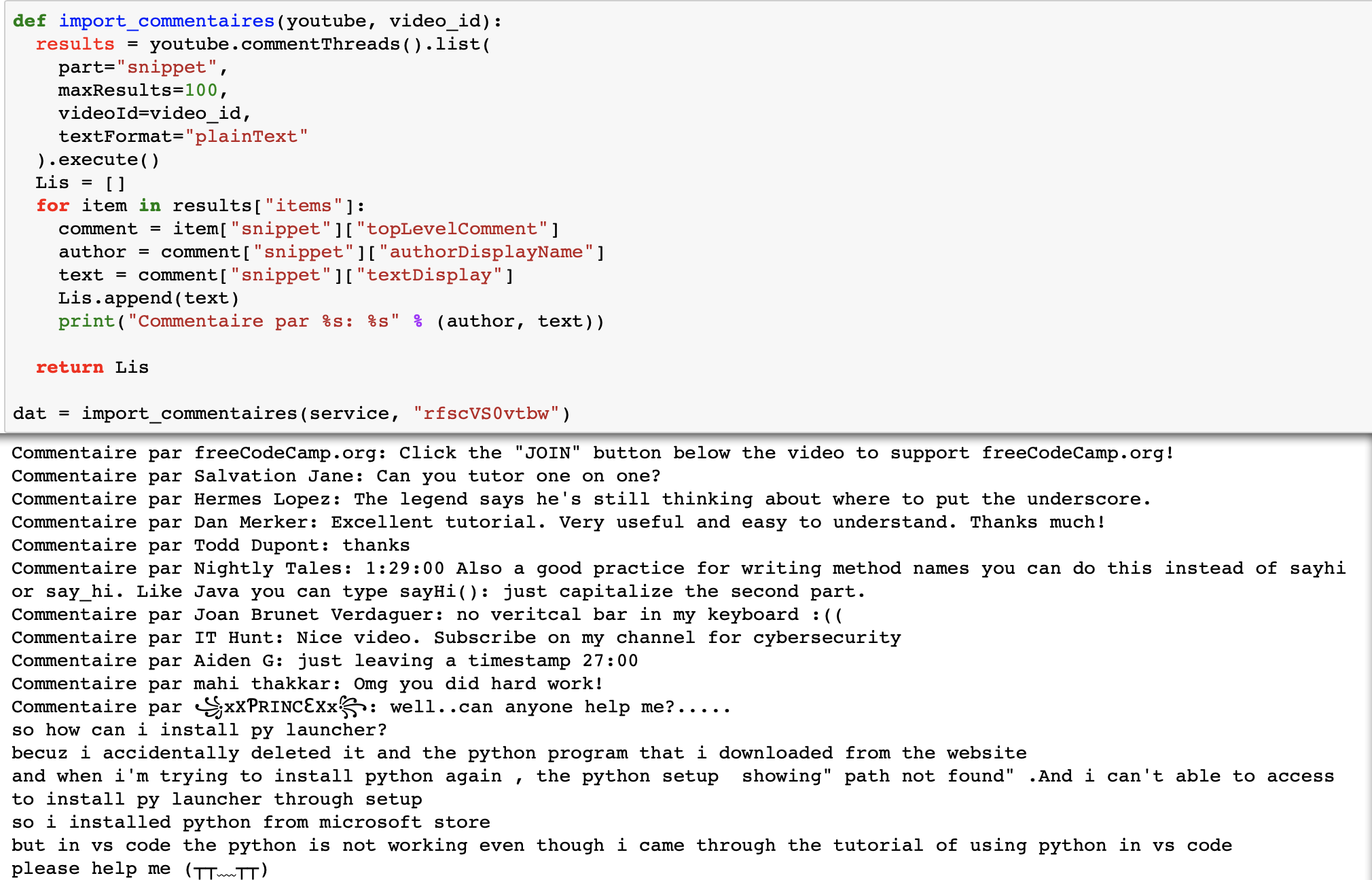


**Utilisation pour commentaires Youtube :**

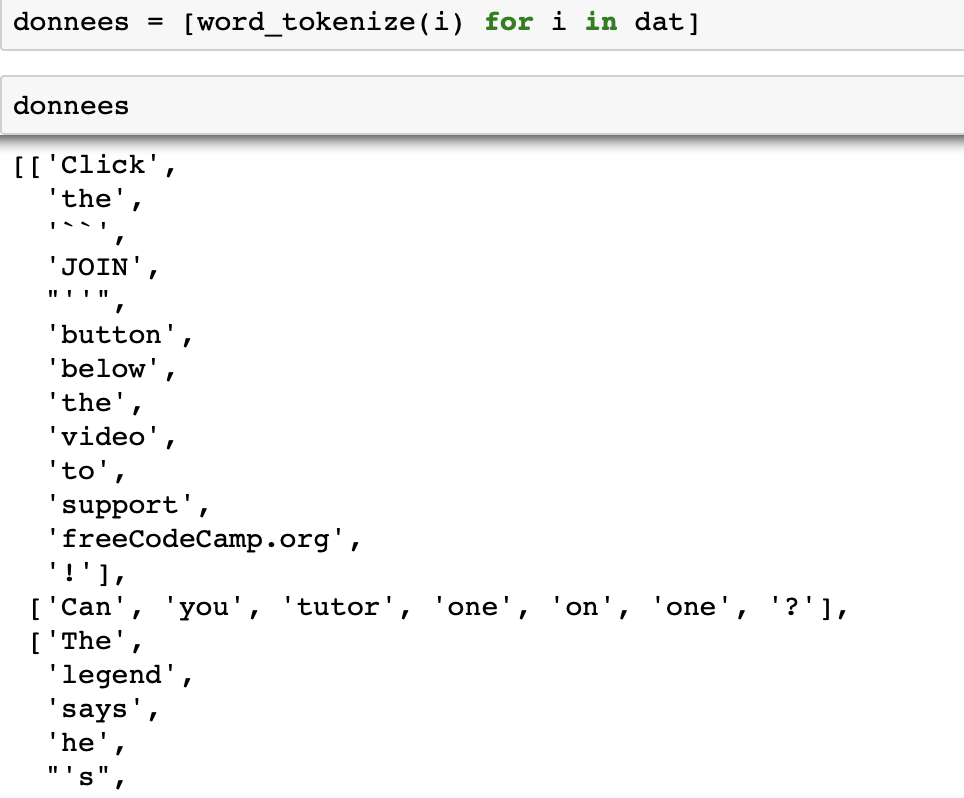
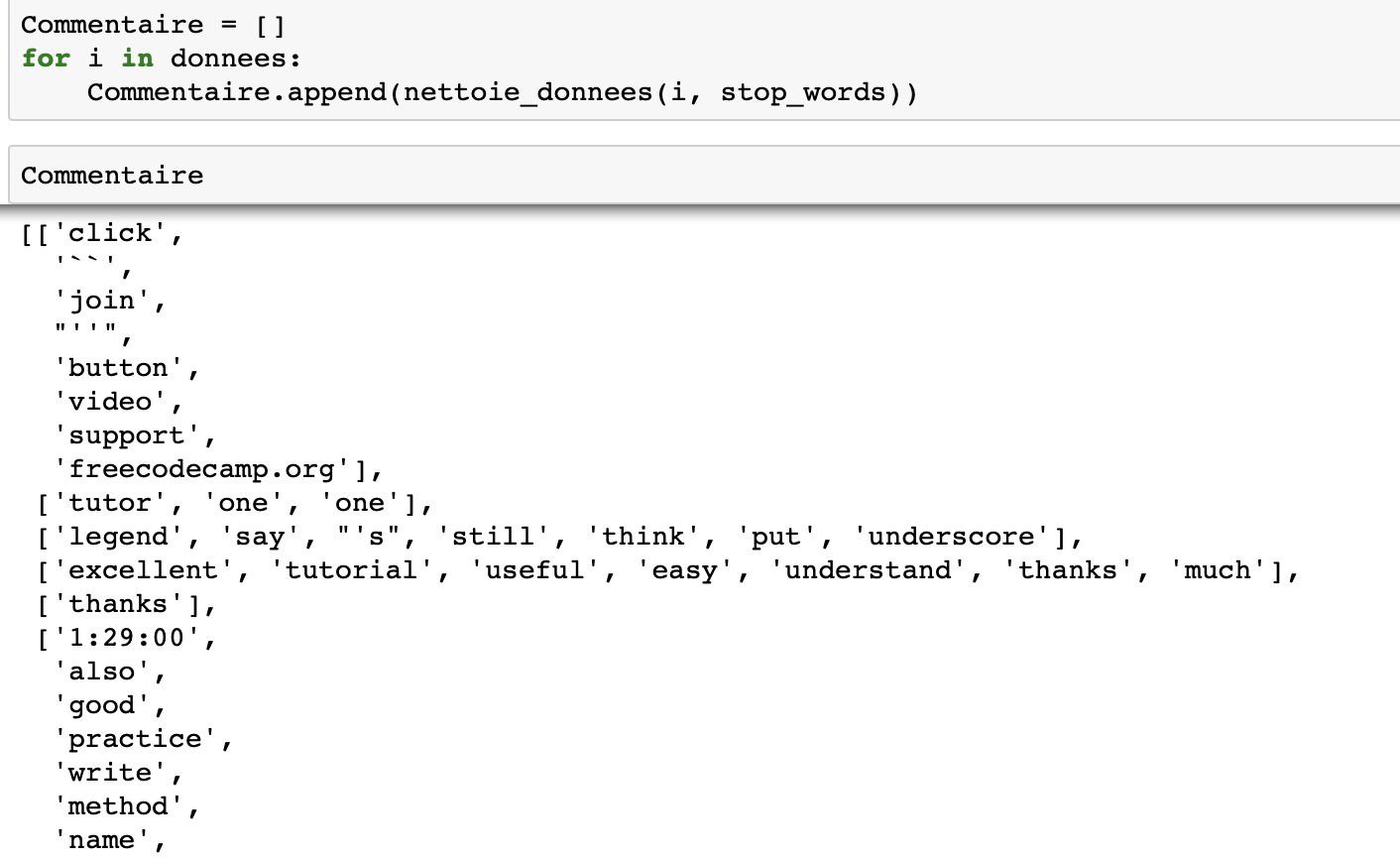
On se connecte d’abord à Youtube Data Api :



Puis on importe les commentaires d’une vidéo :



Puis on sépare chaque commentaire mot par mot, puis on le nettoie :



Après, on utilise le classifieur pour déterminer si le commentaire est positif ou négatif :

