Licence Fondamentale Sciences de l'informatique	Classes : LGLSI-2
TP n°2 Techniques d'indexation et recherche multimédia	
Objectifs	Indexation textuelle

TP2: INDEXATION TEXTUELLE

L'objectif de ce TP2 et du TP suivant (le TP3) est de réaliser un système de recherche d'information complet. Celui-ci comprendra deux modules principaux :

1. **le module d'indexation textuelle (TP2)**
2. **le module de recherche (TP3)**

A la fin du TP3, l'étudiant sera évalué sur la totalité du travail (TP2 + TP3). La note obtenue après cette évaluation sera considérée comme sa note de TP.

Ce TP sera réalisé en deux séances, séance 1 (exercice 1 et exercice2) et séance 2 (exercice3).

Avant de commencer, il faut ajouter la librairie python nltk :

- Dans un terminal, lancez la commande 'pip install --user nltk'
- Dans un terminal, lancez python
- Taper la commande 'import nltk'
- Taper la commande 'nltk.download()'
- A partir de l'onglet 'All packages', sélectionner 'punkt' et 'snowball_data'

Partie 1: INDEXATION:

EXERCICE 1:

Executer les commandes suivantes et interpreter les résultats obenus:

Splitting Tokens:

```
>>> "This is a test for text-based search".split()
>>> import re
>>> re.split(r"\W", "This is a test for text-based search")
```

Licence Fondamentale Sciences de l'informatique	Classes : LGLSI-2
TP n°2 Techniques d'indexation et recherche multimédia	
Objectifs	Indexation textuelle

Case Sensitivity

```
>>> "Paris is the capital of France".split()
>>> "Paris is the capital of France".lower().split()
```

Stemming

```
>>> from nltk.stem.porter import PorterStemmer
>>> stemmer = PorterStemmer()
>>> [stemmer.stem(token) for token in ["developed", "develop", "developing"]]
```

Stop Words

```
>>> tokens = "This is a test for text-based search".lower().split()
>>> [t for t in tokens if t not in ["a", "for", "is", "this"]]
```

Filtering

```
>>> import re
>>> tokens = "1234 test1234 rejected".lower().split()
>>> [t for t in tokens if re.match(r"^[a-z]+$", t)]
```

EXERCICE 2:

On considère la liste des stop-words suivantes:

"a", "an", "and", "are", "as", "at", "be", "but", "by", "for", "if", "in", "into", "is", "it", "no", "not", "of", "on", "or", "such", "that", "the", "their", "then", "there", "these", "they", "this", "to", "was", "will", "with"

Ecrire un programme qui permet de lire un fichier **fich-test.txt** à partir du repertoire courant et d'effectuer les operations suivantes:

- Extraction des mots
- Suppression des stop-words
- Réecriture des mots en minuscule
- Supprime les valeurs numeriques et les dates
- Racinisation (stemming) en utilisant l'algorithme de Porter
- Enregistrement du résultat dans un fichier nommé **sortie.txt**

Licence Fondamentale Sciences de l'informatique	Classes : LGLSI-2
TP n°2 Techniques d'indexation et recherche multimédia	
Objectifs	Indexation textuelle

EXERCICE 3: Pondération et Fichier Inversé

Dans cette partie, on souhaite construire un fichier inversé à partir d'un ensemble de fichiers.txt disponibles dans le dossier "documents" (à récupérer à partir du Drive). Il vous est demandé de:

- Ecrire une fonction qui permet de retourner pour chaque document, la liste des mots stem (résultat de l'étape précedente)
- Ecrire une fonction qui permet de retourner pour chaque document, un dictionnaire comprenant les mots stem ainsi que leurs frequences d'apparition (TF mesure simple).
- Ecrire une fonction qui permet de retourner un dictionnaire global (pour tous les fichiers) comprenant les mots stem ainsi que leurs frequences d'apparition TF dans chaque document.
- Ecrire une fonction qui permet de retourner le poids W de chaque mot stem par rapport à chaque document aprés avoir calculer son IDF en utilisant la formule simple.
- Ecrire une fonction qui permet de construire le fichier inversé. Cette fonction génère un fichier nommé "fich-inv.txt" ayant la structure suivante: stem----doc(i)-----W(i)…….
- Ecrire un programme qui permet d'indexer le corpus "documents"