TP : Analyse de dictionnaire

2 BCPST 2 - TP Informatique

CONSEIL : à chaque nouvelle question faites une nouvelle cellule en tapant "#%%" .
Vous pourrez ainsi traiter toutes les questions dans un même fichier.

Objectif du TP :
- Lire un fichier texte
- Etudier des données

Dans ce TP on propose d'extraire des informations depuis la liste des mots français. Ces mots sont regroupés dans un fichier texte `liste_mots.txt` nécessaire pour faire ce TP.
Pour rappel on peut lire un fichier texte à l'aide de la commande python suivante :

```
texte = open("document.txt", "r")
```

`"document.txt"` est le nom du fichier à ouvrir, `"r"` (read) est le mode de lecture (avec `"w"` (write) ou `"a"` (append) pour le mode d'écriture, mais pas utile dans ce TP).

Dans ce TP, la liste de mots présente des accents, pour que Python parvienne à lire le ficher sans erreur, il faut donc ajouter l'entrée `encoding='utf-8'` au sein de la commande `open` :

```
texte = open("document.txt", "r" ,
    encoding='utf-8')
```

Pour lire ligne à ligne le fichier texte, on rappelle la commande :

```
L = texte.readlines()
```

Qui crée une liste `L` contenant toutes chaînes de caractères formant les lignes du document.
Enfin on rappelle qu'il est impératif de fermer le fichier texte lorsque les informations ont été utilisées. On fait la commande :

```
texte = close()
```

I Etape 1 : Récupération et "nettoyage" des données

Q.1. On va commencer par récupérer la liste des mots dans une liste `L` en suivant les indications ci-dessus.

Vous pouvez constater en affichant `L[2]` dans la zone de droite sur votre logiciel que les passages à la ligne notés `'\n'` sont toujours dans les éléments de `L`.

Q.2. Modifier `L` afin que ses éléments ne terminent plus par le caractère `'\n'`.

En fouillant un peu la liste `L`, vous verrez que des mots sont insatisfaisants pour notre étude, notamment les mots composés. On peut les repérer avec le caractère `'-'`.

Q.3. Eliminer de la liste `L` tous les mots composés.

Avant de passer à la suite, je vous invite à taper la commande ci-dessous :

```
import unicodedata

L = [unicodedata.normalize("NFC",
    mot) for mot in L]
```

Cela sert à ce que les accents soient bien codés dans notre liste.

II Etape 2 : Analyse des données

Q.4. Trouver la taille la plus grande des mots de `L` et afficher les mots qui ont cette taille.

Q.5. Trouver les mots de `L` qui possèdent le caractère "à".

Q.6. Trouver les mots de `L` qui contiennent le plus grand nombre de lettre `'t'`.
Indication : On pourra préalablement coder une fonction `compte_T(m)` qui compte le nombre de `'t'` dans un mot `m`.

Q.7. Trouver les mots de `L` qui contiennent toutes les voyelles.
Indication : On considère que les voyelles (avec accents possibles) sont les caractères : aàâ,eéëèê, oô, iïî, uùû, y.

Q.8. Trouver les mots de `L` les plus longs composés uniquement de voyelles.

Q.9. On va appeler "score de voyelle" d'un mot le nombre le plus élevé de voyelle identiques de ce mot. (Par exemple "banane" a un score de 2 parce qu'il contient 2 'a').
Afficher tous les mots de L qui ont le score de voyelle le plus élevé. *Indication :* On pourra préalablement coder une fonction `score_voyelle(m)` qui retourne le score de voyelle d'un mot `m`.

Q.10. Trouver les mots de `L` qui contiennent le nombre maximal de consonnes consécutives.
Indication : On pourra préalablement coder une fonction `consonnes_consécutives(m)` qui retourne le nombre maximal de consonnes consécutives d'un mot `m`.