TP de Langages de script n° 4 : dictionnaires et ensembles

Exercice 1 : occurrences de mots

1. Écrire une fonction qui compte le nombre de mots différents apparaissant dans une chaîne de caractères passée en argument.

   Dans un premier temps, on suppose que deux mots sont séparés par un espace. Ensuite on généralisera aux signes de ponctuation classiques (`,;.?!:'`).
2. Écrire une fonction qui affiche le nombre d'occurrences de chaque mot apparaissant dans une chaîne de caractères passée en argument.
3. Reprendre les deux fonctions précédentes de manière à ce qu'elles prennent en argument une chaîne de caractères correspondant au nom d'un fichier texte à analyser.
4. Reprendre les deux premières questions en utilisant des ensembles (`set`) ainsi que la fonction `count` de la classe `str` pour la question 2.

Exercice 2 : traduction automatique

On souhaite écrire un programme de traduction automatique simpliste. On représente un lexique de traduction par un fichier, dont chaque ligne est de la forme `mot:traduction` où `traduction` est un mot unique, supposé sans espaces.

1. Écrire une fonction `read_lexic` prenant comme argument le nom d'un fichier et renvoyant un dictionnaire Python dont chaque couple (clé, valeur) correspond à un mot du lexique et à sa traduction.
2. Écrire une fonction `translate_word` prenant un mot et un lexique et renvoyant sa traduction selon ce lexique.
3. Écrire une fonction `translate_sentence` prenant une chaîne de caractères et un lexique et renvoyant sa traduction. Les mots non traduits dans le lexique seront laissés inchangés.
4. Écrire une fonction `write_lexic` réciproque de la fonction `read_lexic`. **Attention**, on souhaite que les entrées soient classées par ordre alphabétique.
5. Écrire la fonction `invert_lexic` prenant un lexique et renvoyant le lexique inversé (on supposera que tous les mots et toutes les traductions sont distincts).

On suppose maintenant qu'un mot peut avoir plusieurs traductions possibles et que deux mots peuvent avoir la même traduction. Par exemple, dans un fichier lexique, on pourra avoir :

```
mot1:traduction1,traduction2,traduction3
mot2:traduction3,traduction4
```

6. Mettre au point une représentation adaptée pour ce nouveau cas, puis réécrire les fonctions `read_lexic` et `write_lexic`.
7. Réécrire les fonctions `translate_word` et `traduire_phrase` en tirant une traduction au sort si plusieurs sont possibles (on pourra utiliser la fonction `choice` du module `random`).
8. Réécrire la fonction `invert_lexic`.

Exercice 3 : un début de cryptanalyse

Un ancienne façon de chiffrer des messages consiste à procéder par substitutions de lettres. Par exemple, le chiffrement de César consiste à effectuer une rotation de 3 lettres sur l'alphabet. Mais cela peut se généraliser à une permutation quelconque sur l'alphabet.

Pour cryptanalyser un tel chiffrement on se base sur des tables d'occurrences des lettres dans la langue du message à décrypter. Un tel fichier `occurrences` est donné pour le français sous la forme `lettre:pourcentage` pour indiquer que la lettre `lettre` apparaît `pourcentage` fois sur 100 lettres dans cette langue.

Avant cela nous allons commencer par chiffrer le texte qui servira d'exemple.

1. Écrire une fonction qui enlève les espaces, les apostrophes et les signes de ponctuation d'un texte.
2. À partir du fichier `clef`, créer le dictionnaire permettant le chiffrement du texte donné.
3. Chiffrer le texte.
4. Écrire une fonction `count_occurrences` qui, à partir d'un fichier texte donné, crée un dictionnaire des couples (lettres, pourcentage).
5. Écrire une fonction similaire à `read_lexicon` d'exercice précédent qui crée un dictionnaire des couples (lettres, pourcentage) à partir d'`occurrences`.
6. À partir de ces deux dictionnaires, on crée un (lettre de `occurrences`, lettre du texte) qui permettrait de décrypter ce texte s'il suivait exactement les pourcentages de la langue.
7. À partir de ce dictionnaire, écrire la fonction qui remplace les lettres du texte par celles qui leurs correspondent dans le dictionnaire.
8. Afficher le texte décrypté.

On voit bien que cette façon de cryptanalyser n'est malheureusement pas suffisante ... Pour aller plus loin, il faut prendre en compte les bigrammes, c'est-à-dire les couples de lettres.