TP de Langages de script n° 3 : introduction aux listes et ensembles

Exercice 1 : Listes.

Une liste python (type `list`) est une collection ordonnée d'éléments non nécessairement de mêmes natures. Elle est délimitée par des crochets (`[ ]`) et ses éléments sont séparés par des virgules. Par exemple `[0, 's', ['abc',4], {'e':'exercice','p':'probleme'}]` est une liste; `[[1, 2, 3], 1, [1]]` est aussi une liste.

Comme les chaînes de caractères, les listes font partie de ce que l'on appelle les *séquences* en python. Elles sont un outil utile et puissant comme nous le verrons dans les semaines futures.

Ce que nous avons vu sur les chaînes de caractères s'applique également aux listes: extraction d'un élément, d'une suite contiguë d'éléments, signification du mot-clef `in`, etc.

1. Soit `L` une liste. Que se passe-t-il si on affecte une valeur à `L[1:3]`? à `L[1:]`? à `L[:]`?
2. Ecrivez une fonction `inversion` qui demande à l'utilisateur de saisir une ligne de texte et renvoie une liste des mots dans l'ordre inverse (on supposera sans faire de vérification que l'utilisateur ne saisit que des lettres ou des espaces). Pour découper une chaine de caractères en une liste de mots, vous pourrez utiliser la méthode split.
3. Ecrivez une fonction `inverse` qui prend en argument un mot et retourne le mot mirroir.
4. Ecrivez une fonction `palindrome` qui prend en argument un texte et affiche les mots de ce texte qui sont des palindromes. Ecrivez au préalable une fonction `decoupe` qui prend en argument un texte et retourne une liste formée des mots du texte dans leur ordre d'apparition.
   Dans un premier temps, on suppose que deux mots sont séparés par un espace. Vous pourrez utiliser la fonction `split`. Ensuite on généralisera aux signes de ponctuation en utilisant la fonction `translate` du type `str` et la constante `string.punctuation`, chaîne des caractères ASCII de ponctuation.
5. Ecrivez une fonction `inclus` qui prend en argument deux listes d'entiers et dit si les éléments de la première liste apparaissent dans la deuxième, dans le même ordre.

Exercice 2 : occurrences de mots

1. Ecrivez une fonction qui compte le nombre de mots différents apparaissant dans une chaîne de caractères passée en argument. Vous pourrez utiliser la fonction `decoupe` de la question 4 de l'exercice 1.
2. Ecrivez une fonction qui affiche le nombre d'occurrences de chaque mot apparaissant dans une chaîne de caractères passée en argument.
3. Reprenez 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. Reprenez les deux premières questions en utilisant des ensembles (`set`) ainsi que la fonction `count` du type `str` pour la question 2.

Exercice 3 : traduction automatique (suite)

On souhaite compléter le programme de traduction automatique écrit au TP2.
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
```

1. Mettez au point une représentation adaptée pour ce nouveau cas, puis réécrivez les fonctions `read_lexic` et `write_lexic`.
2. Réécrivez 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`).
3. Ecrivez la fonction `invert_lexic` prenant un lexique et renvoyant le lexique inversé.