Langages de script
TP n° 2 : Listes et fichiers

Coordonnées des enseignants
Peter Habermehl <Peter.Habermehl@liafa.univ-paris-diderot.fr.fr>
Sophie.Laplante <Sophie.Laplante@liafa.univ-paris-diderot.fr.fr>
Anne Micheli <Anne.Micheli@liafa.univ-paris-diderot.fr.fr>

Documents à télécharger
Tous les énoncés et les documents à télécharger seront disponibles sur didel `http://didel.script.univ-paris-diderot.fr` (le code du cours est LS6). Inscrivez-vous si ce n'est déjà fait pour faciliter la communication avec l'équipe enseignante.

Version de Python Nous utiliserons la version 3.4.2.

a) Les 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', sys.argv[1]]` 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.
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.

Exercice 1 : listes

1. Écrire 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).
   Indication : penser à la méthode `split` du type `str`
2. Modifier la fonction `inversion` afin qu'elle retourne la chaîne de caractères composée des mots, dans l'ordre inverse, de la ligne de texte entrée par l'utilisateur.
   Indication : penser à la méthode `join` du type `str`
3. Écrire 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 : premières mutations de listes

1. Grâce à des mutations de listes, produire les listes suivantes :
   a. la liste des multiples de 5 ou 7 inférieurs à 200 ;
   b. la sous-liste des éléments de rang pair d'une liste donnée ;
   Indication : penser à la fonction `questions`
   c. la liste des couples `[(l1, m1), (l2, m2), ...]` étant données les listes de même longueur `[l1, l2, ...]` et `[m1, m2, ...]`.
   Indication : penser à la fonction `zip`

2. Écrire une fonction `diviseurs` qui trouve tous les diviseurs d'un entier `n` passé en argument. Utiliser cette fonction pour créer la liste de tous les nombres premiers jusqu'à `n`.

Exercice 3 : encore des mutations de listes

Dans cet exercice, on demande d'utiliser le mécanisme de mutation de listes.

1. Écrire une fonction `sans_e` qui prend en argument une liste de mots et retourne la liste de ceux qui ne contiennent pas la lettre `'e'`.

2. Écrire une fonction `anti_begue` qui prend en argument une chaîne de caractères et en crée une nouvelle dans laquelle ont été supprimés les mots consécutifs identiques. La fonction devra transformer le texte d'origine en liste de mots avant tout traitement et transformer la liste nouvellement obtenue en chaîne de caractères.
   On remarque que si on zippe la liste des mots de la phrase avec la liste des mots décalée de 1, les mots répétés se retrouvent ensemble.

```
>>> begue
['ceci', 'est', 'un', 'un', 'test']
>>> begue_decale
['est', 'un', 'un', 'test']

>>> list(zip(begue, begue_decale))
[('ceci', 'est'), ('est', 'un'), ('un', 'un'), ('un', 'test')]
```

   Il suffit alors de choisir les bons mots (attention au dernier mot de la phrase...).

Exercice 4 : Mot sans cube

On considère les mots sur un alphabet à deux lettres $\{a, b\}$. On dit qu'un mot (c'est-à-dire une suite de lettres) possède un cube s'il possède un facteur de la forme uuu où u est un mot non vide. Par exemple *baaa* et *bababa* possèdent un cube alors que *bababb* est sans cube. On peut montrer qu'il existe un mot infini sans cube (mais ce n'est pas l'objet de cet exercice). Nous allons être plus modestes et écrire une fonction qui teste si un mot est sans cube

1. Écrire une fonction `prefixes` qui renvoie la liste des préfixes d'un mot de tailles inférieures à une taille passée en argument.

2. Écrire une fonction `est_sans_prefixe_cube` qui prend en argument une chaîne de caractères et détermine si elle possède un cube comme préfixe.

3. Écrire une fonction `est_sans_cube` qui prend en argument une chaîne de caractères et détermine si elle est sans cube. Pour cela il suffit tester les suffixes de l'argument grâce à la fonction précédente.

b) Les fichiers

PYTHON gère bien le codage le plus moderne, UTF-8, et il serait dommage de ne pas en profiter. Pour cela il faut faire quelques réglages, si nécessaire, de son environnement de travail :

- Ajouter `export LANG="fr_FR.UTF-8"` dans son fichier .bashrc, puis redémarrer la session de travail.
- Régler le terminal sur le codage UTF-8. Vérifier que ça marche en affichant un fichier codé en UTF-8 (par exemple `vers-queneau.txt`) sur le terminal.
- Vérifier que l'éditeur de texte gère bien UTF-8 et sauvegarde les fichiers dans ce codage. Si nécessaire faire des réglages.

Dans les exercices suivants, vous devrez écrire des scripts commentés pour la documentation. Ils devront également faire ce qui est demandé lorsqu'ils seront exécutés en tant que script principal.

Exercice 5 : recherche dans un fichier

Écrire un script PYTHON qui affiche toutes les lignes d'un fichier contenant un certain mot. La référence du fichier et le mot sont passés en argument.
La fonction de recherche peut s'écrire en une seule ligne en utilisant une liste en compréhension si le script s'applique à des fichiers pas trop gros.

Exercice 6 : la disparition

Écrire un script PYTHON qui, prenant en entrée un nom de fichier, en crée une "copie" dans laquelle toutes les occurrences du caractère `e` ont été supprimées.
Tester le script sur le fichier `vers-queneau.txt`.

Exercice 7 : extraction d'information

Le fichier `inscrits.csv` contient une liste d'étudiants avec un certains nombres d'informations séparées par des points virgules.

1. Écrire un script PYTHON qui extrait la liste des mails des étudiants.

2. Écrire un script PYTHON qui crée un fichier `emargement.csv`, où :
   - la 1ère ligne contient les mots `Prenom` et `Nom` séparés par une virgule,
   - chaque autre ligne du fichier contient le prénom et le nom d'un étudiant, dans cet ordre, séparés par une virgule. Il faut que le prénom et le nom soient en minuscule excepté la première lettre qui doit être une majuscule.

3. Modifier le script pour que le fichier `emargement.csv` soit trié en fonction du nom, puis du prénom.