La lecture et l'écriture de fichiers en Python

Romain Tavenard

1 Lecture de fichier textuel

1. Écrivez une fonction qui prend en entrée un nom de fichier et retourne le nombre de mots non vides contenus dans le fichier en question (on suppose que les mots sont séparés par des espaces).
2. Utilisez le module `os` et la fonction codée précédemment pour afficher, fichier par fichier, le nombre de mots des fichiers de votre répertoire `"data"` dont l'extension est `".txt"`.

2 Écriture de fichier textuel

3. Écrivez une fonction qui prend en entrée un nom de fichier et une chaîne de caractères et écrit la chaîne dans le fichier indiqué.
4. Écrivez une fonction qui prend en entrée un nom de fichier et une liste de chaînes de caractères et écrit chaque chaîne de la liste dans une nouvelle ligne du fichier indiqué.

2.1 L'écriture dans la console

Vous avez déjà écrit dans la console à l'aide de la fonction `print`. Plus précisément, cette fonction écrit sur la sortie standard. Il existe deux autres flux standards. Le premier, l'entrée standard, est celui duquel vous lisez des chaînes de caractères entrées par les utilisateurs lorsque vous utilisez la fonction `input`. Enfin, le dernier flux standard est l'erreur standard, qui est le flux qui doit être utilisé pour écrire les messages d'erreur générés par vos programmes. Pour choisir le flux de sortie que vous souhaitez utiliser, il faut importer le module `sys`, puis utiliser la syntaxe suivante :

```
sys.stdout.write(chaine_a_ecrire)  # Sortie standard
sys.stderr.write(chaine_a_ecrire)  # Erreur standard
```

Ainsi, ces flux sont utilisés de la même manière que les objets `file` (sauf qu'on ne les ouvre jamais).

5. Écrivez une fonction qui prend en entrée une liste de chaînes de caractères et écrit chaque chaîne de la liste dans une nouvelle ligne sur la sortie standard.
6. Écrivez une fonction qui prend en entrée une liste de chaînes de caractères et écrit chaque chaîne de la liste dans une nouvelle ligne sur l'erreur standard.

3 Fichers texte structurés

3.1 Fichiers CSV

7. Écrivez une fonction qui retourne le nombre de lignes et de colonnes (le nombre de colonnes d'un fichier CSV est égal au nombre maximum de champs des lignes de ce fichier) d'un fichier CSV dont le nom est fourni en argument.
8. Appliquez cette fonction pour déterminer les nombres de lignes et colonnes de chacun des fichiers de votre répertoire `"data"` dont l'extension est `".csv"`.
9. Reprenez la première question du TD sur les dictionnaires et adaptez la pour prendre en entrée non plus une chaîne de caractères à découper en lignes et colonnes mais un nom de fichier à lire (vous pourrez tester cette fonction avec le fichier `"data/etudiants.csv"`).
10. Écrivez une fonction en Python qui prend en entrée un nom de fichier au format CSV et retourne le contenu de ce fichier sous forme de liste de dictionnaires : les clés étant les en-têtes de colonnes (écrites en première ligne du fichier) et les valeurs étant la valeur associée à la ligne courante. Si la valeur associée à une colonne est vide, vous la remplacerez par `None`.
11. Écrivez une fonction qui prend en entrée un nom de fichier et un dictionnaire (qui stocke, pour chaque clé, une liste d'entiers) et écrit le contenu du dictionnaire au format suivant :

— chaque ligne correspond à une clé du dictionnaire ;
— la première colonne indique la clé considérée ;
— les autres colonnes sont constituées des données stockées pour cette clé.

3.2 Fichiers JSON

12. Le fichier `blabla.json` fournit des séries de positions GPS correspondant à des traces GPS de sorties randonnée de M. Toulemonde. Écrivez un programme calculant les dénivelés cumulés positif et négatif de chacune de ces randonnées. Pour cela, vous utiliserez l'API `Google Maps Elevation` pour laquelle vous aurez au préalable demandé une clé.