La lecture et l'écriture de fichiers en Python

Planche de TD pour un cours dispensé à l'université de Rennes 2

Romain Tavenard

1 Organisation de votre code et de vos fichiers

Pour ce TD, vous écrirez toutes vos fonctions dans un même script Python nommé `TD_fichiers.py`. Celui-ci sera organisé comme indiqué lors des séances précédentes. De plus, vous aurez besoin des données disponibles sur CURSUS dans un fichier `data.zip`.

Le projet que vous ouvrez dans Pycharm doit être organisé comme suit :

```
L2_Python (ou tout autre nom de répertoire parent)
|- data/
   |- a.txt
   |- b.txt
   |- ...
|- TD_fichiers.py
|- ...
```

2 Chemins de fichiers

Pour ces questions, pas besoin de coder en Python, munissez-vous uniquement d'un crayon et d'un papier.

1. En vous référant à la structure de dossiers représentée ci-dessus, et en supposant que l'on est en train de travailler dans le script `TD_fichiers.py`, quel est le chemin relatif permettant d'ouvrir le fichier `a.txt` ?
2. Pour être sûrs que ce chemin relatif soit bien valide quel que soit le système d'exploitation (Windows, Linux, MacOS) en train d'exécuter `TD_fichiers.py`, comment devez-vous faire pour définir ce chemin relatif dans votre code Python ?
3. Supposons maintenant que l'on ait la structure de fichiers suivante :

```
L2_Python (ou tout autre nom de répertoire parent)
|- data/
   |- truc/
```

```
      |- machin/
         |- chose.txt
   |- a.txt
   |- b.txt
   |- ...
|- TD_fichiers.py
|- ...
```

quel est le chemin relatif permettant d'ouvrir le fichier `chose.txt` ?

4. Pourquoi préfère-t-on indiquer le chemin relatif, plutôt que le chemin absolu du fichier de données dans le fichier programme ?

3 Lecture de fichier textuel

5. 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).
6. Utilisez la fonction codée précédemment pour afficher, fichier par fichier, le nombre de mots des fichiers de votre répertoire `"data"` dont l'extension est `".txt"`.

4 Écriture de fichier textuel

7. Écrivez une fonction qui prend en entrée un nom de fichier et une chaîne de caractères et écrit la chaîne dans le fichier indiqué.
8. 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.

5 Fichiers texte structurés

5.1 Fichiers CSV

9. 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. On supposera que le séparateur à utiliser pour les fichiers CSV est `";"` et on n'utilisera pas de `Sniffer`.
10. Appliquez cette fonction pour déterminer les nombres de lignes et colonnes de chacun des fichiers de votre répertoire `"data"` dont l'extension est `".csv"`. Que remarquez-vous pour le fichier `test.csv` ?

5.2 Fichiers JSON

11. Écrivez une fonction qui lit le fichier `"rando_gps.json"` contenu dans le répertoire `"data"` (ce nom de fichier ne devra pas être un paramètre de la fonction, il devra être défini “en dur” dans celle-ci) et retourne le nombre de randonnées listées dans ce fichier.
12. Écrivez une fonction qui lit le fichier `"rando_gps.json"` contenu dans le répertoire `"data"` (ce nom de fichier ne devra pas être un paramètre de la fonction, il devra être défini “en dur” dans celle-ci) et retourne une liste contenant les intitulés des randonnées listées dans ce fichier.