W4-S1-C3-format-json-et-autres

December 15, 2014

1 Formats de fichiers : JSON et autres

1.1 Compléments - niveau basique

Voici quelques mots sur des outils python fournis dans la librairie standard, et qui permettent de lire ou écrire des données dans des fichiers.

1.1.1 Le problème

Les données dans un programme python sont stockés en mémoire (la RAM), sous une forme propice aux calculs. Par exemple un petit entier est fréquemment stocké en binaire dans un mot de 64 bits, qui est prêt à être soumis au processeur pour faire une opération arithmétique.

Ce format ne se prête pas forcément toujours à être transposé tel quel lorsqu'on doit écrire des données sur un support plus pérenne, comme un disque dur, ou encore sur un réseau pour transmission distante - ces deux supports étant à ce point de vue très voisins.

Ainsi par exemple il pourra être plus commode d'écrire notre entier sur disque, ou de le transmettre à un programme distant, sous une forme décimale qui sera plus lisible, sachant que par ailleurs toutes les machines ne codent pas un entier de la même façon.

Il convient donc de faire de la traduction dans les deux sens entre représentations d'une part en mémoire, et d'autre part sur disque ou sur réseau (à nouveau, on utilise en général les mêmes formats).

1.1.2 Le format JSON

Le format sans aucun doute le plus populaire à l'heure actuelle est le format JSON pour JavaScript Object Notation.

Sans trop nous attarder nous dirons que JSON est un encodage - en anglais marshalling - qui se prête bien à la plupart des types de base qu'on trouve dans les langages modernes comme python, ruby ou, donc, JavaScript.

La libraririe standard python contient le module json, que nous avons déjà utilisé dans les exercices de la semaine passée, et que nous illustrons très rapidement ici:

```
In []: import json

       # En partant d'une donnée construite à partir de types de base
       data = [
           # des types qui ne posent pas de problème
           [1, 2, 'a', [3.23, 4.32], {'eric': 32, 'jean': 43}],
           # un tuple
           (1, 2, 3),
       ]

       # sauver ceci dans un fichier
       with open("s1.json","w") as json_output:
           json.dump(data, json_output)
```

```
# et relire le résultat
with open("s1.json") as json_input:
    data2 = json.load(json_input)
```

Limitations de json Certains types de base ne sont pas supportés par le format JSON (car ils ne sont pas natifs en JavaScript), c'est le cas notamment de * `tuple`, qui se fait encoder comme une liste; * `complex`, `set` et `frozenset`, qu'on ne peut pas encoder du tout (sans étendre la librairie).

```
In []: # le premier élément de data est intact,
       # comme si on avait fait une *deep copy* en fait
       print "première partie de data", data[0] == data2[0]
       # par contre le ‘tuple‘ se fait concoder comme une ‘list‘
       print "deuxième partie", "entrée", type(data[1]), "sortie", type(data2[1])
```

Malgré ces petites limitations, ce format est de plus en plus populaire, notamment parce qu'on peut l'utiliser pour communiquer avec des applications web écrites en JavaScript, et aussi parce qu'il est très léger, et supporté par de nombreux langages.

1.2 Compléments - niveau intermédiaire

1.2.1 Le format `csv`

Le format `csv` pour Comma Separated Values, originaire du monde des tableurs, peut rendre service à l'occasion, il est proposé dans le module `csv`.

1.2.2 Le format pickle

Le format `pickle` remplit une fonctionnalité très voisine de `JSON`, mais est spécifique à python. C'est pourquoi, malgré des limites un peu moins sévères, son usage tend à rester assez marginal. Il est implémenté dans le module `pickle`.

1.2.3 Le format XML

Vous avez aussi très probablement entendu parler de XML, qui est un format assez populaire également.

Cela dit, la puissance, et donc le coût, de XML et JSON ne sont pas du tout comparables, XML étant beaucoup plus flexible mais au prix d'une complexité de mise en œuvre très supérieure.

Il existe plusieurs souches différentes de librairies prenant en charge le format XML, qui sont introduites ici.

1.2.4 Pour en savoir plus

Voyez la page sur les formats de fichiers dans la documentation python.