Fichiers et utilitaires

Complément - niveau basique

Nous présentons dans ce complément quelques utilitaires autour des fichiers. Il ne s'agit pas d'outils liés au type `file` à proprement parler, mais qui sont utiles malgré tout dans le contexte de la manipulation de fichiers.

Le module `os.path`

Ce module propose un certain nombre d'utilitaires à large spectre sur les fichiers, et entre autres les choses suivantes.

Manipulations de noms de fichier

Pour commencer, nous vous recommandons d'utiliser le module `os.path` pour manipuler les noms de fichier, surtout si vous écrivez du code multi-plateforme. Voyez en particulier dans cette catégorie

- `os.path.join` ajoute '/' ou '\' entre deux morceaux de chemin, selon l'OS
- `os.path.basename` trouve le nom de fichier dans un chemin
- `os.path.dirname` trouve le nom du directory dans un chemin
- `os.path.abspath` calcule un chemin absolu, c'est-à-dire à partir de la racine du filesystem

Existence de fichier / métadonnées

Les fonctions suivantes retournent des informations obtenues auprès de l'OS dans les métadonnées associées aux fichiers (concrètement, sans ouvrir le fichier).

- `os.path.exists` pour savoir si un chemin existe ou pas (fichier ou répertoire)
- `os.path.isfile` (et `isdir`) pour savoir si un chemin est un fichier (et un répertoire)
- `os.path.getsize` pour obtenir la taille du fichier
- `os.path.getatime` et aussi `getmtime` et `getctime` pour obtenir les dates de création/modification d'un fichier

Et autres

Reportez-vous à la documentation complète de `os.path` (https://docs.python.org/2/library/os.path.html) pour plus de détails sur ce module.

Le module `os`

Signalons aussi, mais cette fois du module `os`, les fonctions

- `os.remove` (ou son ancien nom `os.unlink`), qui permet de supprimer un fichier
- `os.rmdir` pour supprimer un répertoire (mais qui doit être vide)
- `os.removedirs` pour supprimer tout un répertoire avec son contenu, récursivement si nécessaire
- `os.rename` pour renommer un fichier

Et autres

Le module `os` comporte un grand nombre d'autres fonctionnalités que celles relatives aux fichiers, vous pouvez consulter la documentation complète de `os` (https://docs.python.org/2/library/os.html) pour une liste exhaustive.

Un petit exemple

Voici un exemple simple qui utilise les deux modules `os` et `os.path` pour s'assurer qu'un fichier n'existe pas, le créer, accéder sa taille et sa date de modification, puis l'effacer à nouveau.

```
# dans cet exemple, la seule fonction que nous utilisons
# du module 'os' est 'unlink'
import os

import os.path

nom_fichier = 'fichier-temoin'

# au départ le fichier n'existe pas
print 'Début: existence de', nom_fichier, os.path.exists(nom_fichier)
# on le crée
with open(nom_fichier,'w') as output:
    output.write('0123456789\n')
# il doit exister maintenant
print 'Milieu: existence de', nom_fichier, os.path.exists(nom_fichier)
# regardons sa taille -- 11 caratères (avec la fin de ligne)
print 'taille', os.path.getsize(nom_fichier)
# et sa date de modification
mtime_timestamp = os.path.getmtime(nom_fichier)
print 'date de dernière modification (1)', mtime_timestamp
# pour l'afficher d'une manière un peu plus lisible
from datetime import datetime
mtime_datetime = datetime.fromtimestamp(mtime_timestamp)
print 'date de dernière modification (2)', mtime_datetime

# on le détruit
os.remove(nom_fichier)
# vérifions qu'il n'est plus là
print 'Fin: existence de', nom_fichier, os.path.exists(nom_fichier)
```

Le module `glob` - recherche de fichiers

On a parfois le besoin de chercher, par exemple, "tous les fichiers se terminant par `.txt`".

Le module `glob` fournit des fonctions pour ce genre de besoins, comme illustré ici

```
import glob

# tous les fichiers .json dans le répertoire data/
for json in glob.glob("data/*.json"):
    print json
```

Comme toujours, voyez la documentation complète de `glob` (https://docs.python.org/2/library/glob.html) pour plus de précisions.