Lire et écrire des fichiers de données csv

> Dans de nombreuses situations, il est préférable d'ouvrir les fichiers de type .csv via la librairie pandas !

Les fichiers csv sont des fichiers de données séparées par des virgules (ou point-virgules), pour “comma separated values”. Comme ceci :

```
1;0.1;3
2;0.3;5
3;0.5;7
4;0.6;11
5;0.9;21
6;1.5;39
```

Ils peuvent être facilement importés ou exportés de tableurs ou logiciels de graphiques scientifiques.

La librairie csv facilite la lecture et l'écriture de fichiers au format csv. Voici un programme effectuant une lecture d'un fichier csv, suivie d'une écriture (souvent bien sûr on n'utilisera qu'une seule de ces possibilités) :

<sxh python; title : exemple_simple-csv-02.py> #!/usr/bin/env python # -*- coding: utf-8 -*- “”“ pour la documentation sur le module csv (comma separated variable) de python, voir à partir de http://docs.python.org/lib/csv-examples.html Autres refs : http://www.linuxjournal.com/content/handling-csv-files-python http://code.activestate.com/recipes/577423-convert-csv-to-xml/ ”“”

import csv # module nécessaire “comma separated values”

fichier d'entrée with open(“Classeur1.csv”, “rb”) as ifile:

```
  reader = csv.reader(ifile, delimiter=';')
  # Classeur1.csv est un simple fichier texte, comme celui qui suit par
exemple :
  # 1;0.1;3
  # 2;0.3;5
  # 3;0.5;7
  # 4;0.6;11
  # 5;0.9;21
  # 6;1.5;39
  rows=[]  # initialisation d'une liste qui contiendra les lignes à écrire
  for row in reader:  # on parcourt les lignes successives du fichier
d'entrée
      print row   # facultatif : pour visualiser ces lignes
      rows.append(row)   # ajout à la liste de sortie
      for chaine in row:   # traitement des lignes
          nombre=float(chaine)   # on peut effectuer ici un calcul sur
nombre...
```

```
          print float(nombre)     # ...ou simplement le visualiser
  print rows   # visualisation de la liste complète avant création du
fichier de sortie
```

fichier de sortie with open(“Classeur-out.csv”,“wb”) as ofile:

```
  writer = csv.writer(ofile, delimiter=';')
  writer.writerows(rows)
```

</sxh>

Pour satisfaire les contraintes d'un tableur, il est important de vérifier le séparateur utilisé , ainsi que l'usage ou non de “guillemets” :

- delimiter = ' '
- quotechar = '“'

Cf. la page http://docs.python.org/2/library/csv.html#dialects-and-formatting-parameters.

Références

- documentation officielle du module csv
- The most (time) efficient ways to import CSV data in Python
- csv - Lecture et écriture de fichiers CSV
- + autres formats : Importing Data in Python - Little summary on different ways to import different data