Table de données

Les tables de données sont le moyen le plus repandu de stocker et traiter de l'information structurée.

Cours

Une table de données est un ensemble de valeurs organisées sous forme de table :

- où chaque **ligne** correspond à un **élément**, appelé aussi une **entité** ou une **entrée** de la table,
- où chaque **colonne** correspond à une catégorie d'information de même type, appelé un **champs** ou **attribut**.

Exemple : une liste d'information sur les pays utilisée au chapitre 'Types construits':

Pays	Capitale	Population (ml)
France	Paris	68
Espagne	Madrid	48
Italie	Rome	60

Cours

Les noms des colonnes, ici Pays, Capitale et Population (ml), sont appelés les **descripteurs** de la table.

Les bases de données permettent de manipuler d'importantes quantités de données organisées en tables.

Le format csv

Cours

Le format **csv**, pour ***Comma-Separated Values***, est un **format texte** représentant des **données en table séparées par des virgules** (comme son nom l'indique) ou d'**autres séparateurs**, par exemple point-virgule (« `;` »), tabulation (« `\t` »), etc.

Chaque ligne du texte correspond à une ligne du tableau et les virgules correspondent aux séparations entre les colonnes. Un fichier csv est un fichier texte mais on utilise l'extension .csv pour indiquer la nature de ses données.

Ecrire une table de données dans un fichier csv

Un fichier csv est un simple fichier texte, on peut enregister des données structurées dans un fichier csv comme on l'a fait avec un fichier texte en séparant les données par une virgule ou un autre séparateur et les lignes par `\n`. Il suffit juste de changer l'extension du fichier et d'ajouter une ligne de descipteurs :

f = open(...)

```
f = open('pays.csv, 'w')
f.write('Pays;Capitale;Population (ml)\n')
f.write('France;Paris;68\n')
f.write('Espagne;Madrid;48\n')
f.write('Italie,Rome;60\n')
f.close()
```

with open(...) as f:

```
with open('pays.csv', 'w') as f:
    f.write('Pays;Capitale;Population (ml)\n')
    f.write('France;Paris;68\n')
    f.write('Espagne;Madrid;48\n')
    f.write('Italie,Rome;60\n')
```

Le format csv est un format de fichier texte, ouvert qui peut être créé, lu et modifié par de nombreux logiciels, en particulier des éditeurs de texte (bloc note), mais aussi des tableurs comme Microsoft Excel[1] ou OpenOffice Calc:

```
Pays;Capitale;Population (ml)
France;Paris;68
Espagne;Madrid;48
Italie;Rome;60
```

Il existe d'autres formats de données structurées, par exemple JSON ou XML.

Lire une table de données depuis un fichier csv

On peut lire les données depuis un fichier csv et enregistrer ses données dans un tableau en le lisant ligne par ligne :

f = open()

```
f = open("pays.csv", "r")
pays = []
for li in f:
    pays.append(li)
f.close()
```

with open as f:

```
with open("pays.csv", "r") as f:
    pays = []
    for li in f:
        pays.append(li)
```

On obtient un tableau de chaînes de caractères, un longue chaîne par ligne :

```
>>> pays
['Pays;Capitale;Population (ml)\n',
'France;Paris;68\n',
'Espagne;Madrid;48\n',
'Italie;Rome;60\n']
```

On peut maintenant remplacer `li` par `li[:-1]` pour supprimer les caractères de retour à la ligne « `\n` » à la fin de chaque chaîne et ajouter une instruction `f.readline()` après l'ouverture du fichier pour supprimer la première ligne contenant les descripteurs :

f = open()

```
f = open("pays.csv", "r")
f.readline()
pays = []
for li in f:
    pays.append(li[:-1])
f.close()
```

with open as f:

```
with open("pays.csv", "r") as f:
    f.readline()
    pays = []
    for li in f:
        pays.append(li[:-1])
```

On obtient un tableau de chaînes de caractères contenant seulement les données qui nous intéressent :

```
>>> pays
['France;Paris;68',
'Espagne;Madrid;48',
'Italie;Rome;60']
```

Les données de chaque pays dans une seule chaîne de caractères ne seront pas faciles à manipuler, une dernière modification consiste à les « découper » en tableau avec `.split()`, en indiquant le séparateur utilisé, ici « ";" » :

f = open()

```
f = open("pays.csv", "r")
f.readline()
pays = []
for li in f:
    pays.append(li[:-1].split(";"))
f.close()
```

with open as f:

```
with open("pays.csv", "r") as f:
    f.readline()
    pays = []
    for li in f:
        pays.append(li[:-1].split(";"))
```

On obtient un tableau de tableaux qui permettra de manipuler les données efficacement (en prenant soin de convertir les nombres en `int` ou `float` si besoin) :

```
>>> pays
[['France', 'Paris', '68'],
['Espagne', 'Madrid', '48'],
['Italie', 'Rome', '60']]
```

On pouvait aussi écrire la même chose directement par compréhension :

f = open()

```
f = open("pays.csv", "r")
f.readline()
pays = [li[:-1].split(";") for li in f]
f.close()
```

with open as f:

```
with open("pays.csv", "r") as f:
    f.readline()
    pays = [li[:-1].split(";") for li in f]
```

Exercice corrigé

Importer dans un tableau de tableaux les données du fichier des codes postaux depuis https://public.opendatasoft.com/explore/dataset/laposte_hexasmal/export

Réponse

De la même façon qu'on a importé des données en table dans un tableau de tableaux, on peut très bien les importer dans un tableau de p-uplets (pour avoir des données de types différents)[2] ou un tableau de dictionnaires (pour utiliser les descripteurs)[3].

Lire un fichier csv avec le module csv

La fonction `reader()` du module `csv` permet de lire les données contenues dans un fichier csv.

```
import csv

with open('pays.csv', 'r') as f:
    pays = csv.reader(f, delimiter=';')
```

La variable `pays` ne peut pas être affichée directement :

```
>>> pays
<_csv.reader object at 0x00000224603A0BE0>
```

Mais c'est un objet « itérable », on peut donc la parcourir :

```
    for ligne in pays:
        print(ligne)
```

ou la convertir directement en tableau avec la fonction `list()` :

```
import csv

with open('pays.csv', 'r') as f:
    pays = list(csv.reader(f, delimiter=';'))
```

On obtient un tableau de tableaux :

```
>>> pays
[['Pays', 'Capitale', 'Population (millions)'],
['France', 'Paris', '68'],
['Italie', 'Rome', '60']]
```

⚠️ À noter que toutes les valeurs sont au format `str`, y compris les nombres, il faudra en tenir compte dans l'utilisation de ces données par la suite programme.

Le tableau de tableaux, n'est pas toujours idéal, en particulier la première ligne de descripteurs n'est pas séparée du reste des données. La méthode `DictReader` permet de garder les descripteurs en créant un tableau de dictionnaires. Comme avec `reader`, on utilise `list()` pour convertir le résultat en tableau.

```
import csv

with open('pays.csv', 'r') as f:
    pays = list(csv.DictReader(f, delimiter=','))
```

On obtient un tableau de dictionnaires :

```
>>> pays
[{'Capitale': 'Paris', 'Pays': 'France', 'Population (millions)': '68'}, {'Capitale': 'Madrid',
'Pays': 'Espagne', 'Population (millions)': '48'}, {'Capitale': 'Rome', 'Pays': 'Italie',
'Population (millions)': '60'}]
```

On trouvera de nombreuses autres fonctionnalités du module `csv` dans la documentation Python.

Exercice corrigé

Importer dans un tableau de dictionnaires les données du fichier des codes postaux depuis https://www.data.gouv.fr/fr/datasets/base-officielle-des-codes-postaux/.

Réponse

---

1. Avec le menu Fichier/Ouvrir puis utiliser la fenêtre « Assistant d'importation du texte » pour choisir le séparateur utilisé ou Données/Récupérer des données externes/fichier texte. ↩
2. Par exemple avec le code suivant :

```
f = open("pays.csv", "r")
f.readline()         # ligne de descripteurs ignorée
pays = []
for li in f:
    tab = li[:-1].split(';')
    puplet = (tab[0], tab[1], int(tab[2]))     # la population est convertie en int
    pays.append(puplet)
f.close()
```

↵

3. Par exemple avec le code suivant :

```
f = open("pays.csv", "r")
descripteurs = f.readline()[:-1].split( ',')   # tableau des descripteurs
pays = []
for li in f:
    tab = li[:-1].split(';')
    dico = {descripteurs[0]: tab[0] ,
            descripteurs[1]: tab[1],
            descripteurs[2]: int(tab[2])
        }
    pays.append(dico)
f.close()
```

↵