Fichiers de données et tableurs

Mémento

Syntaxe	Rôle
	importation
import **pandas** as pd	module pour gérer les données
	lire un fichier csv
df = pd.read_csv(“nomFichier.csv”, sep = “;”)	la première ligne sert d’entête
df = pd.read_csv(“nomDuFichier.csv”, sep = “^”, header = None)	pas d’entête
	recupérer les données dans un tableau numpy
A = df.values	stocke les données numériques dans la matrice A
A[:,0]	1ère colonne
A[:,1]	2ème colonne
A[:,k]	k+1 ième colonne

1 Saisie des données en Python Numpy

1.1 Problématique

Numpy permet de manipuler facilement des données numériques (calculer moyennes, écart-type, faire des régressions linéaires, etc. . .).

Afin de pouvoir manipuler ces données, il faut au préalable les **charger dans la mémoire de l’ordinateur** . Pour cela deux possibilités.

1.2 Saisie directement dans un code Python

1/ On peut taper directement les valeurs dans un code Python à l’aide de la commande `np.array()` qui renvoie une variable au format *tableau Numpy*, donc directement utilisable. Exemple ci-dessous:

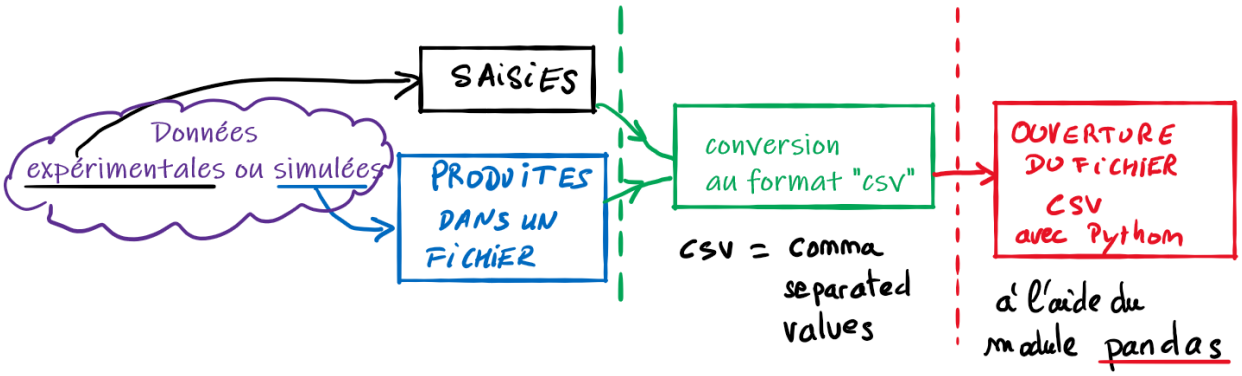
[5]:
```
import numpy as np # import du module numpy avec l'alias np
## Saisie des données expérimentales
xvalues = np.array([1.02, 2.23, 5.29, 11.23, 32.2, 48.7, 63.4]) # 7␣
  ↪valeurs numériques
yvalues = np.array([3.43e-3, 7.61e-3, 18.4e-3, 37.8e-3 , 109.8e-3,␣
  ↪214.8e-3 ])
```

Dans l’exemple ci-dessus, les données sont stockées dans les variables `xvalues` et `yvalues` et sont alors utilisables directement par Python.

1.3 Saisie des données avec un tableur

2/ On peut prévilégier l'utilisation d'un logiciel davantage adapté à la saisie de données (par exemple un tableur). Dans ce cas les données vont être stockées dans un fichier sur le disque dur de l'ordinateur avant d'être récupérées par Python. Nous nous limiterons au cas d'un fichier au format *csv = comma separated values*.

Ci-dessous le schéma qui illustre ces différentes étapes :



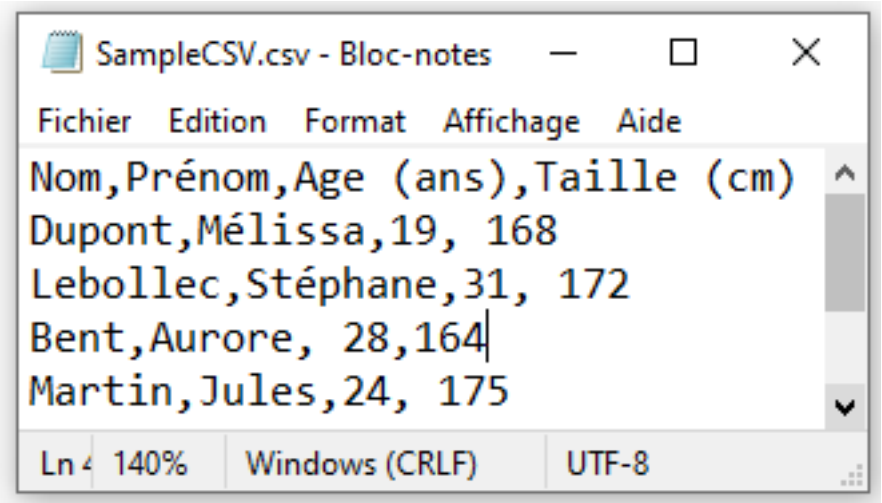
2 Le format de fichier *csv*

C'est un format dans lequel les données sont lisibles directement avec un éditeur de texte (ex: *bloc-note* sous Windows).

Les données se présentent sous formes de **colonnes**.

Les valeurs entre chaque colonnes sont généralement séparées par le caractère 'virgule' (*comma*, en anglais).

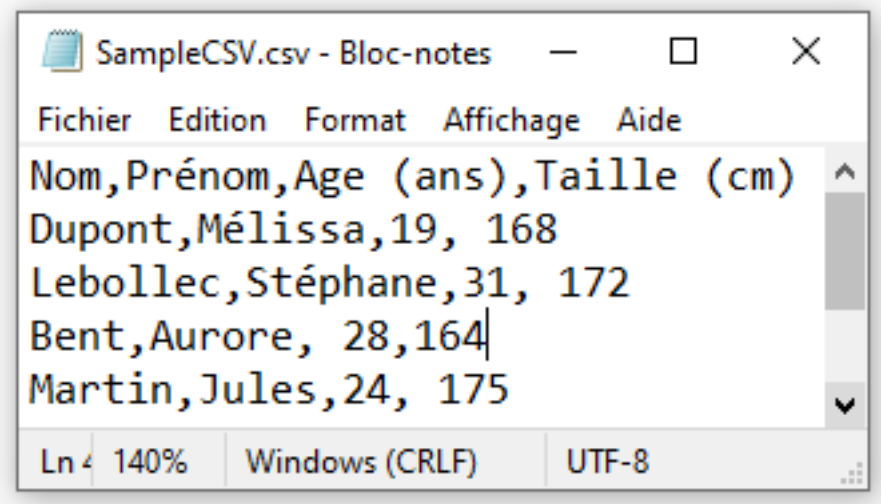
Dans l'exemple ci-dessous, la première ligne (appelée *header*) contient les noms des **champs de l'entête** ("Nom", "Prénom", "Age (ans)","Taille (cm)").



Notes :

- la première ligne peut éventuellement être omise.

- les valeurs peuvent être de **type** *numérique* (Age et Taille) ou *chaîne de caractères* (Nom et Prénom).



2.1 Ouvrir un fichier csv avec Python

On utilise pour cela la fonction *read_csv()* du module *pandas*.

[24]:
```
import pandas as pd # importation du module pandas avec l'alias 'pd'
```

La syntaxe de la cette fonction est la suivante

```
df = pd.read_csv("nomDuFichier.csv", sep = ";") # La première ligne sert d'
df = pd.read_csv("nomDuFichier.csv", sep = "\t", header = None) # Pas d'en-
```

2.1.1 Exemple 1: cas où les données sont séparées par des virgules

C'est l'option par défaut.

[69]:
```
df1 = pd.read_csv("SampleCSV.csv") # l'argument "sep" est par défaut␣
↪des virgules.
df1 # affiche les données sous forme de table.
```

[69]:

	Nom	Prénom	Age (ans)	Taille (cm)
0	Dupont	Mélissa	19	168
1	Lebollec	Stéphane	31	172
2	Bent	Aurore	28	164
3	Martin	Jules	24	175

2.1.2 Exemple 2: cas où les données sont séparées par des tabulations

Dans ce cas il faut renseigner l'argument `sep` en indiquant que le séparateur des colonnes est le caractère de tabulation qui s'écrit ‘‘.

[70]:
```
```

```
df2 = pd.read_csv("SampleCSVTab.csv",sep='\t') # séparateur =␣
  ↪tabulation
df2
```

[70]:

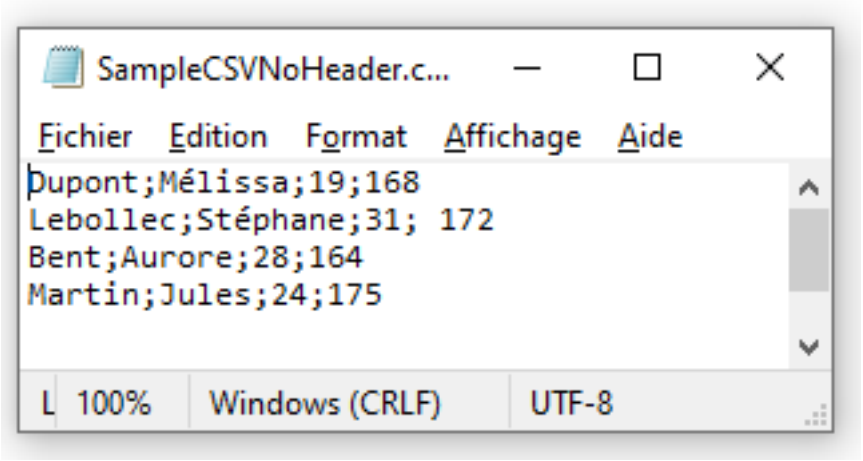
	Nom	Prénom	Age (ans)	Taille (cm)
0	Dupont	Mélissa	19	168
1	Lebollec	Stéphane	31	172
2	Bent	Aurore	28	164
3	Martin	Jules	24	175

Note : selon les cas les séparateurs peuvent être des caractères point virgule ";".

C'est pourquoi on recommande de toujours visualiser brièvement le contenu d'un fichier csv.

2.1.3 Exemple 3: cas où le fichier csv ne possède pas d'entête

Dans ce cas, il faut ajouter l'argument `header= None` afin de préciser que les données "utiles" débutent dès la première ligne.



Considérons le fichier ci-dessous:

- l'entête n'est pas présente,
- le séparateur est le caractère point virgule ';'.

Voici la syntaxe pour charger en mémoire les données contenues dans ce fichier *csv*.

[71]:
```
df3 = pd.read_csv("SampleCSVNoHeader.csv", sep=';', header = None) #␣
  ↪séparateur = tabulation
df3
```

[71]:

	0	1	2	3
0	Dupont	Mélissa	19	168
1	Lebollec	Stéphane	31	172
2	Bent	Aurore	28	164
3	Martin	Jules	24	175

Noter : 1) l'utilisation de **header = None** et 2) l'absence d'étiquettes pour les champs de colonnes.

3 Travailler avec les données numériques d'un fichier csv

La fonction `pd.read_csv("NomDeFichier.csv", sep='..', header = ...)` renvoie un **objet** de type "DataFrame".

[2]:
```
import pandas as pd
df = pd.read_csv("SampleCSV.csv", sep=',', header= 'infer') #␣
  ↪paramètres par défaut
print("Type de l'objet data =  ", type(df))
df
```

```
Type de l'objet data =   <class 'pandas.core.frame.DataFrame'>
```

[2]:

	Nom	Prénom	Age (ans)	Taille (cm)
0	Dupont	Mélissa	19	168
1	Lebollec	Stéphane	31	172
2	Bent	Aurore	28	164
3	Martin	Jules	24	175

La méthode `.info()` de l'objet DataFrame donne des informations sur les données.

Le champ `.values` permet de récupérer les données du DataFrame sous la forme d'array numpy 2d.

[97]:
```
A = df.values # convertie les données en une matrice 2D-Numpy
print(A) # est une matrice
```

```
[['Dupont' 'Mélissa' 19 168]
 ['Lebollec' 'Stéphane' 31 172]
 ['Bent' 'Aurore' 28 164]
 ['Martin' 'Jules' 24 175]]
```

La variable *A* désigne une matrice numpy.

[99]:
```
A.shape # renvoie (nb lignes x nb colonnes)
```

[99]:
```
(4, 4)
```

[100]:
```
NOMS = A[:,0] # extraction de la 1ère colonne de la matrice => listes␣
  ↪des NOMS
print(NOMS)
```

```
['Dupont' 'Lebollec' 'Bent' 'Martin']
```

[103]:
```
ages = A[:,2] # extraction de la 3ème colonne, celle des ages en␣
  ↪années
```

```
tailles = A[:,3] # extraction de la 4ème colonne, celle des tailles␣
  ↪en cm
print("ages \t= ",ages)
print("tailles\t= ",tailles)
```

```
ages    =  [19 31 28 24]
tailles =  [168 172 164 175]
```

4 Création pratique d'un fichier csv avec un tableur

Quel que soit le tableur utilisé (LibreOffice.Calc, MicroSoft Excel, ou Google.spreadsheets, la démarche est toujours la même:

1. Saisir les données dans le tableur en laissant la **1ère ligne comme étiquette** des noms de données.
2. Exporter la feuille de calcul au format 'csv' (pour *spreadsheets*, choisisser Fichier> Télécharger> csv).

5 Complément : création d'un objet de type DataFrame

[89]:
```
import numpy as np
import pandas as pd
ar = np.array([[1.1, 2, 3.3, 4], [2.7, 10, 5.4, 7], [5.3, 9, 1.5,␣
  ↪15]])
df0 = pd.DataFrame(ar, index = ['a1', 'a2', 'a3'], columns = ['A',␣
  ↪'B', 'C', 'D'])
```

[90]:
```
df0.shape # donne la taille des dimensions du dataFrame
```

[90]:
```
(3, 4)
```

[91]:
```
N = 2
df0.head(N) # n'affiche que les N premières lignes de données
```

[91]:

	A	B	C	D
a1	1.1	2.0	3.3	4.0
a2	2.7	10.0	5.4	7.0

[93]:
```
N = 1
df0.tail(N) # n'affiche que les N dernières lignes de données
```

[93]:

	A	B	C	D
a3	5.3	9.0	1.5	15.0