



Math93.com

# Traitement données en tables

## Fichiers CSV TD 2 - NSI

### Objectifs

- Indexation de tables.  
*Importer une table depuis un fichier texte tabulé ou un fichier CSV.*
- Manipulation de fichiers CSV.  
*Lire un fichier CSV ou écrire dans un fichier CSV.*

## I Fichiers CSV

### I.1 Définition

L'organisation des données en tableau est très courante et très ancienne. Le bulletin d'un élève par exemple est organisé en table.

En 1970 l'informaticien britannique Edgar Frank « Ted » Codd (1923 - 2003) alors employé chez IBM, introduit le *modèle relationnel* qui propose des tables pour stocker des bases de données et un langage pour les traiter.

On peut par exemple proposer la table suivante dans un fichier *exemple1.txt* (fichier texte).

```
Nom,Prenom,classe
Codd,Edgar,premiere 2
Gates,Bill,premiere 1
Proces,marc,premiere 2
```

#### Définition 1 (Fichier CSV)

- Le format **CSV** de *Comma-Separated Values*, est un format texte ouvert représentant des **données tabulaires** sous forme de valeurs séparées par des **virgules** (ou des points virgules).
- Un fichier CSV est un **fichier texte**, par opposition aux formats dits « binaires » (comme des images par exemple).
- Chaque ligne du texte correspond à une ligne du tableau et les virgules correspondent aux séparations entre les colonnes. Les portions de texte séparées par une virgule correspondent ainsi aux contenus des cellules du tableau.
- Toutes les lignes du fichiers ont le **même nombre de champs** (colonnes)
- La première ligne peut représenter des **noms d'attributs** ou commencer directement par des données.

**Remarque**

La syntaxe des fichiers CSV n'est pas complètement standardisée, aussi des variations peuvent exister :

- Les **chaînes de caractères** peuvent être protégées par des guillemets (les guillemets s'expriment alors avec un **double guillemet "** ).
- Le **caractère de séparation** des nombres décimaux peut être le point ou la virgule (si c'est la virgule, le caractère de séparation doit être différent).
- **La virgule** est un standard pour les données anglo-saxonnes, mais pas pour les données aux normes françaises. En effet, en français, la virgule est le séparateur des chiffres décimaux. Il serait impossible de différencier les virgules des décimaux et les virgules de séparation des informations. C'est pourquoi on utilise un autre séparateur : **le point-virgule (;)**. Dans certains cas cela peut engendrer quelques problèmes, vous devrez donc rester vigilants sur le type de séparateur utilisé.
- Un des problème les plus importants reste l'encodage des caractères qui n'est pas spécifié dans le fichier et peut donc être source de problèmes, lors de changement d'OS typiquement.

**I.2 Lecture d'un CSV avec un tableur**

Les tableurs, tels que "Calc" (Libre Office) ou Excel sont normalement capables de lire les fichiers au format CSV. Certains tableurs gèrent mal le séparateur CSV "point-virgule" et le séparateur des chiffres décimaux "virgule", il faut parfois préciser le séparateur utilisé.

**Exercice 1**

1. Créer un fichier avec le bloc-note (Bloc-Note ou NotePad sous windows et TextEdit sous mac) et recopier les quatres lignes de l'exemple1 (Nom, Prénom, classe).
2. Cliquez droit sur votre fichier et ouvrez-le avec *OpenOffice Calc* ou/et *Excel*. Vous allez normalement avoir des options d'importation. Vous pourrez définir vos séparateurs.
3. Faire de même en renommant votre fichier en .csv.

**Exercice 2**

1. Créer un fichier nommé **NotesElevés.csv** avec le bloc-note (Bloc-Note ou NotePad sous windows et TextEdit sous mac) contenant les données du tableau ci-dessous.

| Nom    | Maths | NSI | Anglais |
|--------|-------|-----|---------|
| Galois | 17    | 14  | 17      |
| Gauss  | 20    | 12  | 8       |
| Euler  | 19    | 15  | 13      |

2. Vérifier en ouvrant votre fichier texte avec *OpenOffice Calc* ou/et *Excel*.

## II Lecture d'un CSV avec Python

La bibliothèque Python propose un module csv. A partir de maintenant allez sur un environnement en ligne pour conserver votre travail. On va aller sur [Capitale](#) au lien [Manipulation des Fichiers CSV Chapitre II et III](#)

### II.1 Avec csv.reader(fichier) : liste de listes.

Le code Python suivant permet de charger notre fichier **NotesElevés.csv** dans une variable nommée **Table**. Notez qu'il faut convertir l'importation en liste pour pouvoir l'utiliser.

```
# Dans l'éditeur Python
import csv
# ouverture du fichier et précision de l'encodage utf8
fichier=open("NotesElevés.csv", encoding='utf8')

# conversion en liste, on précise le caractère de séparation
table=list(csv.reader(fichier,delimiter=","))

fichier.close()
print(table)
```

```
# Cela doit vous donner dans la console
[['Nom', 'Maths', 'NSI', 'Anglais'], ['Galois', '17', '14', '17'],
 ['Gauss', '20', '12', '8'], ['Euler', '19', '15', '13']]
```



### Exercice 3

| Faites-ce qui précède avec le fichier "NotesElevés.csv" que vous avez fabriqué sur le bloc-note.

Voici une fonction d'affichage ligne par ligne qui permet de mieux visualiser vos tableaux :

```
def affiche(tableau) -> None:
    """Affichage d'un tableau"""
    if type(tableau) == list:
        car1, car2 = "[", "]"
    if type(tableau) == dict:
        car1, car2 = "{", "}"
    print(car1, tableau[0], ",")
    for i in tableau[1:-1]:
        print(" ", i, ",")
    print(" ", tableau[-1], car2)
affiche(table)
```



#### Remarque

Cette solution présente des inconvénients.

- **La première ligne**, celle des attributs a été chargée comme une ligne de données.
- Chaque ligne est représentée comme une **liste**.
- Toutes les données ont été converties en **chaînes de caractères**, même les notes.

## II.2 Avec csv.DictReader(fichier) : liste de dictionnaires.

Dans ce cas la variable contient un tableau de dictionnaires ordonnés (qui conserve l'ordre d'insertion des clés). Le format d'affichage peut déconcerter mais on obtient une **liste de dictionnaires**.

Notez le gros avantage de cette méthode.

```
# Dans l'éditeur Python
import csv
fichier=open("NotesEleves.csv")
table=list(csv.DictReader(fichier,delimiter=",")) # liste de dictionnaires
fichier.close()

print(table) #Ou affiche(table)
# Pour afficher Gauss, le 'Nom' du dictionnaire 2
print(table[1]['Nom'])
```

```
# Cela doit vous donner dans la console
[{'Nom': 'Galois', 'Maths': '17', 'BSI': '14', 'Anglais': '17'},
{'Nom': 'Gauss', 'Maths': '20', 'BSI': '12', 'Anglais': '8'},
{'Nom': 'Euler', 'Maths': '19', 'BSI': '15', 'Anglais': '13'}]

'Gauss'
```



### Exercice 4

1. Faites-ce qui précède avec le fichier "NotesEleves.csv" que vous avez fabriqué sur le bloc-note.
2. Pour accéder au nom 'Gauss', il faut utiliser le deuxième dictionnaire de la **table** avec la clé 'Nom' comme ci-dessus.  
Maintenant comment accéder au nom 'Euler' et à sa note en NSI?
3. Comment accéder au nom 'Galois' et à sa note en Maths?



### Exercice 5

Remarquer que dans la table précédente les notes sont des chaînes de caractères. On ne peut pas faire d'opérations arithmétiques.

1. Compléter la fonction **convert(dico)**, ci-dessous, qui va créer un nouveau dictionnaire avec les notes des dictionnaires de notre table précédente **converties en entiers**.
2. Afficher la nouvelle table : **table\_valide** .

```
import csv
fichier=open("NotesEleves.csv")
table=list(csv.DictReader(fichier,delimiter=","))
fichier.close()

def convert(dico):
    nsi=int(dico['NSI'])
    maths=int(dico['Maths'])
    anglais=int(dico['Anglais'])
    return {'Nom':dico['Nom'],'Maths':.....,'NSI':.....,'Anglais':.....}

# On définit la nouvelle table où les str sont remplacés par des int:
table_valide=[]
for ligne in table:
    table_valide.append(convert(ligne))

print(table_valide)
```

```
# Cela doit vous donner dans la console
>>> table_valide
[{'Nom': 'Galois', 'Maths': 17, 'NSI': 14, 'Anglais': 17},
{'Nom': 'Gauss', 'Maths': 20, 'NSI': 12, 'Anglais': 8},
{'Nom': 'Euler', 'Maths': 19, 'NSI': 15, 'Anglais': 13}]
```

### III Écriture (export) d'un CSV avec Python

On peut aussi écrire le contenu d'un dictionnaire dans un fichier csv.

**With open** permet d'éviter d'avoir à signifier la fermeture du fichier, elle se fera automatiquement à la fin de l'indentation.

On veut écrire la table suivante qui recense les notes des élèves, dans un fichier :

| Nom       | Grec | Maths | Philosophie |
|-----------|------|-------|-------------|
| Pythagore | 10   | 20    | 15          |
| Thalés    | 13   | 13    | 8           |
| Euclide   | 19   | 14    | 11          |



#### Exercice 6

| Copier et lancer le code suivant et vérifier le contenu du fichier "NouveauFichier.csv"

```
# Un nouveau fichier en ouvert en mode écriture
# Si il existe déjà il est écrasé
with open("NouveauFichier.csv","w") as sortie:
    objet=csv.DictWriter(sortie,['Nom','Grec','Maths','Philosophie'])

# Pour écrire la ligne d'en têtes dans le fichier
    objet.writeheader()

# On peut donner la table directement
    table2=[{'Nom': 'Pythagore', 'Grec': 10, 'Maths': 20, 'Philosophie': 15},
            {'Nom': 'Thalés', 'Grec': 13, 'Maths': 13, 'Philosophie': 8},
            {'Nom': 'Euclide', 'Grec': 19, 'Maths': 14, 'Philosophie': 11}]

#Pour Ecrire la table complete dans le fichier
    objet.writerows(table2)
```



#### Exercice 7

| Écrire une fonction **stats\_csv(monfichier)** où **monfichier** est le nom d'un fichier CSV. La fonction **stats\_csv(monfichier)** retourne le nombre de lignes et le nombre de colonnes de la table stockée dans le fichier **monfichier**.



#### Exercice 8

| On utilise le fichier CSV précédent, **NotesEleves.csv**.

1. Ajouter dans le fichier **NotesEleves.csv** une colonne avec la moyenne des élèves.
2. Ajouter dans le fichier **NotesEleves.csv** une ligne qui contient la moyenne par matière.

↩ Fin du devoir ↪