Manipulation de fichiers CSV

## En Bref

Le format CSV est couramment utilisé pour échanger des données habituellement traitées à l’aide de tableurs ou de logiciels de bases de données principalement. Nous allons apprendre à importer et exporter des données en utilisant ce format.

## Enregistrements

Un enregistrement est une structure de données de types éventuellement différents auxquelles on accède grâce à un nom.

Par exemple, on peut représenter les notes d’un élève dans différentes discipliines à l’aide d’un enregistrement:

{'Nom': 'Joe', 'Anglais': 17, 'Info': 18, 'Maths': 16}

Les *champs* (ou *clés* ou *attributs*) de ces enregistrements sont ici Nom, Anglais, Info et Maths. On leur associe des *valeurs*, ici Joe, 17, 18 et 16.

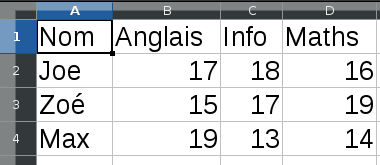
En Python, on utilisera dans cet ouvrage les *dictionnaires* pour représenter les enregistrements conformément au programme.

## Fichiers CSV

Le format CSV (*Comma Separated Value*) est couramment utilisé pour importer ou exporter des données d’une feuille de calcul d’un tableur. C’est un fichier texte dans lequel chaque ligne correspond à une ligne du tableau, les colonnes étant séparées par des virgules. Il permet donc de représenter une liste d’enregistrements ayant les mêmes champs.

Histoire Ce format est né bien avant les ordinateurs personnels et le format xls puisque c’est en 1972 qu’il a été introduit.

Pour éviter les problèmes dus à l’absence de standardisation du séparateur décimal (virgule en France, point ailleurs), mieux vaut paramétrer son tableur pour utiliser le point (français suisse par exemple). Voici un exemple de feuille de calcul:



Le fichier CSV correspondant étant:

'Nom','Anglais','Info','Maths'  
'Joe','17','18','16'  
'Zoé','15','17','19'  
'Max','19','13','14'

## Lecture de fichiers CSV

On peut choisir de représenter sur Python les fichiers CSV par des *listes* de *dictionnaires* dont les *clés* seront les noms des colonnes.

Pour reprendre l’exemple précédent, on obtient:

Table1 =   
[{'Nom': 'Joe', 'Anglais': 17, 'Info': 18, 'Maths': 16},  
 {'Nom': 'Zoé', 'Anglais': 15, 'Info': 17, 'Maths': 19},  
 {'Nom': 'Max', 'Anglais': 19, 'Info': 13, 'Maths': 14}]

Note En utilisant le vocabulaire habituel décrivant une feuille de calcul d’un tableur: \* une table est une liste de dictionnaires: Table1; \* une ligne est un dictionnaire: Table1[0] \* une cellule est une valeur associée à une clé d’un dictionnaire: Table1[0]['Info']

Voici des propositions d’import/expot de fichiers CSV:

import csv  
def depuis\_csv(nom\_fichier\_csv):  
 """  
 Crée une liste de dictionnaires, un par ligne.  
 La 1ère ligne du fichier csv est considérée comme la ligne des noms des champs  
 """  
 lecteur = csv.DictReader(open(nom\_fichier\_csv,'r'))   
 return [dict(ligne) for ligne in lecteur]

## Export vers un fichier CSV

def vers\_csv(nom\_table, ordre\_cols):  
 """  
 Exporte une liste de dictionnaires sous forme d'un  
 fichier csv. On rentre le nom de la table sous forme de chaîne.  
 On donne l'ordre des colonnes sous la forme d'une liste d'attributs.  
 >>> vers\_csv('Groupe1', ordre\_cols=['Nom','Anglais','Info','Maths'])  
 """  
 with open(nom\_table + '.csv', 'w') as fic\_csv:  
 ecrit = csv.DictWriter(fic\_csv, fieldnames=ordre\_cols)  
 table = eval(nom\_table)  
 ecrit.writeheader() # pour le 1ère ligne  
 for ligne in table:  
 ecrit.writerow(ligne) # lignes ajoutées 1 à 1  
 return None