Chap 13. Python 3 : CSV

Livre p 203 – Chap 15 Indexation de tables

1. Gestion des fichiers texte

a. Ouverture et fermeture

- fichier = *open*(file, mode) : ouvre le fichier « file » dans un des modes suivant :
  - ″r″ read : ouverture en lecture seule pour lire le contenue d'un fichier texte existant.
  - ″w″ write : ouverture en écriture pour créer et écrire dans un nouveau fichier texte.
  - ″a″ append : ouverture en ajout pour écrire des données à la fin d'un fichier existant.

Remarques : Commencer par sauvegarder le programme en cours dans un dossier, les fichiers seront créés et lus dans ce dossier. (Il existe des modules nommés os et os.path pour gérer l'emplacement des fichiers)
Le format de codage est souvent un problème sous Windows, on peut le préciser avec le paramètre optionnel *encoding*="utf-8".
Attention : Si le fichier « file » n'existe pas dans le répertoire courant, le mode ″r″ va provoquer une erreur, alors que les modes ″w″ et ″a″ vont créer un fichier à ce nom.
Si le fichier « file » existe déjà dans le répertoire courant, le mode ″w″ va l'effacer et en recréer un !

- fichier.*close*() : ferme le fichier cité en préfixe.

Attention : Si on oublie de fermer le fichier en court d'écriture, les modifications ne seront pas sauvées !

b. Lecture et écriture

- fichier.*read*() : renvoie tout le contenu du fichier dans une chaîne de caractères.
- fichier.*read*(n) : renvoie uniquement les n prochains caractères du fichier.
- fichier.*readline*() : renvoie le contenu de la ligne suivante du fichier.
- fichier.*readlines*() : renvoie le contenu de toutes les lignes restantes du fichier dans une liste.
- fichier.*write*() : écrit une chaîne de caractère dans le fichier à la suite de la précédente.

Remarque : La lecture et l'écriture d'un fichier s'effectue à partir du 1er caractère jusqu'au dernier, impossible de revenir en arrière.

Exemples :

Création du fichier texte :
```
fichier = open("test.txt", "w")
fichier.write("Ceci est un test :\n")
fichier.write("Hello world !\n")
fichier.close()
```

Lecture du fichier, méthode 1 :
```
fichier = open("test.txt", "r")
lecture = fichier.read()
fichier.close()

lecture contient alors :
'Ceci est un test :\nHello world !\n'
```

Lecture du fichier, méthode 2 :
```
fichier = open("test.txt", "r")
lecture = fichier.readlines()
fichier.close()

lecture contient alors :
['Ceci est un test :\n', 'Hello world !\n']
```

Attention : Il faut gérer les retours à la ligne.
En écriture, il faut l'ajouter car il n'est pas créé automatiquement contrairement à l'instruction *print*().
En lecture, on peut les supprimer avec la méthode « .*strip*(″\n″) »

Lecture du fichier, méthode 3 :
```
fichier = open("test.txt", "r")
ligne_1 = fichier.readline().strip("\n")
ligne_2 = fichier.readline().strip("\n")
fichier.close()

ligne_1 contient alors : 'Ceci est un test :'
ligne_2 contient alors : 'Hello world !'
```

Remarque : Il existe un autre moyen pour ouvrir un fichier :
Pas besoin de le refermer dans ce cas mais il faut écrire les instructions de lecture ou écriture dans un bloc indenté.

Lecture du fichier, méthode 4 :
```
with open("test.txt", "r") as fichier :
    ligne_1 = fichier.readline().strip("\n")
    ligne_2 = fichier.readline().strip("\n")
```

2. Format CSV

a. Comma-Separated Values : CSV

Le format CSV est un format de fichier texte pour représenter des données en tableau.
La première ligne comporte les titres des colonnes, les lignes suivantes correspondent aux données.
Malheureusement il n'y a pas de standard pour le séparateur de liste et le symbole décimal !

b. Format classique anglo-saxon

À l'étranger on utilise habituellement la virgule comme séparateur de liste et le point comme symbole décimal.

Exemple :

```
Nom,Prenom,Moyenne
MACHIN,Pierre,14.5
TRUC,Paul,12.6
```

c. Format classique français

En France on utilise habituellement le point-virgule comme séparateur de liste et la virgule comme symbole décimal.

Exemple

```
Nom;Prenom;Moyenne
MACHIN;Pierre;14,5
TRUC;Paul;12,6
```

d. Autres formats rencontrés

On trouve aussi parfois la tabulation comme séparateur de liste.
De plus les valeurs sont parfois entourées de guillemets.

Exemple :

```
"Nom"    "Prenom"    "Moyenne"
"MACHIN"    "Pierre"    "14.5"
"TRUC"    "Paul"    "12.6"
```

3. Tableur Excel

a. Ouvrir un fichier CSV avec Excel

Le tableur Excel de Microsoft peut ouvrir directement les fichiers texte au format CSV mais il utilise alors un format par défaut qui n'est pas forcément le bon !
Pour pouvoir choisir les options de format il faut passer par le menu « **Données** » puis choisir le sous-menu : « **À partir d'un fichier texte/CSV** »
On peut alors choisir :
- L'origine du fichier (Unicode UTF-8, …)
- Le délimiteur (virgule, point-virgule, tabulation…)

b. Sauvegarder au format CSV avec Excel

Pour obtenir un fichier texte au format CSV à partir d'un tableau Excel, il faut passer par le menu « **Fichier** » puis choisir le sous-menu : « **Enregistrer sous** », on peut alors choisir le format « **CSV (séparateur : point-virgule)** »
Le format du fichier texte obtenu est géré par Windows !
Pour changer les paramètres d'enregistrement du fichier CSV, il faut passer par l'application « **Paramètres** » de Windows puis: « **paramètres régionaux, de date et d'heure supplémentaires** », dans l'option « **Région** » aller alors dans les « **Paramètres supplémentaires…** ». On y trouve alors le « **Symbole décimal** » et le « **Séparateur de listes** ». Mais il est déconseillé de modifier ces paramètres !

4. Le module csv de Python 3

a. Conversion en liste de listes

- *csv.reader*(nom)

Exemple :

```
import csv
exemple = []
with open("Exemple.csv", encoding="utf-8") as fichier :
    fichier_csv = csv.reader(fichier, delimiter=";")
    for ligne in fichier_csv:
        exemple.append(ligne)
```

b. Conversion en liste de dictionnaires

- *csv.DictReader*(nom)

Exemple :

```
import csv
exemple = []
with open("Exemple.csv", encoding="utf-8") as fichier:
    fichier_csv = csv.DictReader(fichier, delimiter=";")
    for ligne in fichier_csv:
        exemple.append(dict(ligne))
```