Les fichiers

Les fichiers représentent un ensemble d'informations stockées sur un support magnétique. Leur utilisation permet de sauvegarder des données et de les exploiter, En effet, lorsque l'on ferme un programme, les données sont perdues. Or, les fichiers peuvent être, justement, un excellent moyen de garder les valeurs de certains objets pour pouvoir les récupérer quand vous rouvrirez votre programme.

Vous avez l'habitude d'utiliser des fichiers (fichiers texte, excel...), mais dans le cadre de notre formation nous allons utiliser des fichiers pour stocker des données.

Un fichier de données possède plusieurs lignes que l'on appelle **enregistrements**. Exemple, si je veux enregistrer un fichier des élèves qui suivent l'option ICN, ce fichier possèdera 8 enregistrements: un par élève.

Un enregistrement est divisé en **champs** ou rubriques ou propriétés ou **colonnes**. Ce sont ses caractéristiques. Par exemple dans le fichier présenté ci-dessus, un enregistrement peut être caractérisé par les champs suivants: nom de l'élève, prénom, date de naissance...

Nous allons nous intéresser particulièrement aux fichiers de **type csv,** c'est à dire un fichier de type tableur, dont les champs sont séparés par un caractère de ponctuation (virgule ou point-virgule par exemple)

Utilisation d'un fichier en Python

Ouverture du fichier

Pour utiliser un fichier, il faut toujours l'ouvrir. La commande est la suivante:

```
mon_fichier = open("test.csv", "a" ou "w" ou "r")
```

Le mode est donné sous la forme d'une chaîne de caractères. Voici les principaux modes :

- '**r**' : ouverture en lecture (Read).
- '**w**' : ouverture en écriture (Write). Le contenu du fichier est écrasé. Si le fichier n'existe pas, il est créé.
- '**a**' : ouverture en écriture en mode ajout (Append). On écrit à la fin du fichier sans écraser l'ancien contenu du fichier. Si le fichier n'existe pas, il est créé.

Fermeture du fichier

Après avoir utilisé un fichier, il faut toujours le fermer. Si vous oubliez et que vous voulez ouvrir un fichier déjà ouvert, eh bien... Vous aurez des problèmes!!!

```
mon_fichier.close()
```

Lecture d'un fichier

Lecture de tout le fichier

Dans l'exemple qui suit, considérons un fichier test.csv qui contient des enregistrements. Chaque enregistrement comporte un numéro, un nom et un prénom:
12013;FAHY;Yvelise
12101;BLUM;Nicolas
12211;RIGOUT;Amandine
1010;Pasqualini;claude

Le programme pour lire tout le fichier sera le suivant:

```
mon_fichier = open("test.csv", "r")
contenu = mon_fichier.read()
print(contenu)
mon_fichier.close()
```

On ouvre le fichier en lecture, puis la variable “contenu” contient le fichier dans sa globalité que l'on peut ensuite imprimer. Enfin, on ferme le fichier.

Autre écriture pour accéder au fichier :

```
with open("test.csv", "r") as mon_fichier:
   contenu = mon_fichier.read()
print(contenu)
```

le fichier est automatiquement fermé

Lecture d'une ligne du fichier

On peut lire un fichier ligne par ligne et de plus isoler chaque colonne de la ligne qui vient d'être lue.
La méthode readline permet de lire une seule ligne du fichier.

```
with open("test.csv", "r") as mon_fichier:
   ligne = mon_fichier.readline()
```

Ici, ligne contient tous les champs de l'enregistrement qui vient d'être lu, chaque champ séparé par un point-virgule. Pour isoler chaque champ, il faut utiliser la commande split vue lors du cours sur les chaînes de caractères.

```
mon_fichier = open("test.csv", "r")
ligne = mon_fichier.readline()
nom = ligne.split(';')[1]
prenom = ligne.split(';')[2]
```

ligne.split(' ; ') avec l'indice i permet d'isoler le ième champ de la ligne. *Il faut se rappeler que les champs sont numérotés à partir de 0.*

Comment détecter la fin d'un fichier du fichier

Il faut savoir que chaque ligne se termine par un caractère de contrôle invisible \n qui correspond à “passage à la ligne suivante”. Si l'on veut lire un fichier de la 1ère à la dernière ligne, on peut utiliser l'exemple suivant:

```
mon_fichier = open("test.csv", "r")
ligne = mon_fichier.readline()
while ligne !="\n":
    ligne = mon_fichier.readline()
    #exploitation de la ligne
mon_fichier.close
```

Solution plus élégante avec une boucle for :

```
mon_fichier = open("test.csv", "r")
for ligne in mon_fichier:
    #exploitation de la ligne
mon_fichier.close
```

Ecriture dans un fichier

L'écriture dans un fichier permet d'ajouter une ligne dans le fichier. Vous pouvez ouvrir le fichier avec le mode w ou le mode a. Le premier écrase le contenu éventuel du fichier, alors que le second ajoute ce que l'on écrit à la fin du fichier. À vous de voir en fonction de vos besoins. Dans tous les cas, ces deux modes créent le fichier s'il n'existe pas.

Pour écrire dans le fichier, il faut utiliser la méthode write en indiquant la chaîne de caractères que l'on veut écrire. **Il ne faut pas oublier le séparateur** (ici le point-virgule) entre les champs **ni le \n** pour indiquer la fin de la ligne.

```
mon_fichier = open("test.csv", "a")
mon_fichier.write("1010"+";"+"Pasqualini"+";"+"claude\n")
mon_fichier.close()
```

Autre écriture pour **ajouter** une information au fichier :

```
with open("test.csv", 'a') as mon_fichier:
   mon_fichier.write("1010"+";"+"Pasqualini"+";"+"claude\n")
```

le fichier est automatiquement fermé

Utiliser le module CSV de Python

Le module Python csv facilite la lecture et l'écriture des fichiers au format CSV :

- **csv.reader** permet de lire et décoder un flux CSV ;
- **csv.writer** permet d'encoder et d'écrire un flux CSV

Lire un fichier CSV

```
import csv
mon_fichier = open("eleves.csv", "r")
contenu = csv.reader(mon_fichier, delimiter=';')
for ligne in contenu:
    #exploitation de la ligne
    print(ligne[1], " ", ligne[2])
mon_fichier.close
```

Ecrire dans une fichier CSV

```
import csv
mon_fichier = open("eleves.csv", "a")
contenu = csv.writer(mon_fichier)
contenu.writerow( ("1515","Dupond","Charles") )
mon_fichier.close
```

Gérer des dictionnaires

Python permet d'enregistrer des données d'un **dictionnaire** en utilisant le **format JSON**. Voici les mêmes données sous forme d'un tableau contenant des dictionnaires :

```
contenu = [ {"numéro":"12013", "nom":"FAHY", "prénom":"Yvelise"} ,
            {"numéro":"12101", "nom":"BLUM", "prénom":"Nicolas"} ,
            {"numéro":"12211", "nom":"RIGOUT", "prénom":"Amandine"}
          ]
```

La démarche à suivre :

- le **module json** doit être importé dans le programme ;
- Ce dictionnaire doit être **transformé** au format JSON avec la fonction **json.dumps()** avant enregistrement dans un fichier ;
    - lors de la lecture du fichier, son contenu doit être **transformé** du format JSON en dictionnaire avec la fonction **json.load()** ;

Ecrire dans un fichier

```
# coding: utf8
import json
contenu = [ {u"numéro":"12013", "nom":"FAHY", "prénom":"Yvelise"} ,
            {u"numéro":"12101", "nom":"BLUM", "prénom":"Nicolas"} ,
            {u"numéro":"12211", "nom":"RIGOUT", "prénom":"Amandine"}
          ]
with open("test.csv", "w") as mon_fichier:
  mon_fichier.write(json.dumps(contenu, indent=4))
```

Lire dans un fichier

```
# coding: utf8
import json

with open("test.csv", "r") as mon_fichier:
  contenu = json.load(mon_fichier)
print(contenu)
```

Retour au cours : Les instructions du langage Python

- Cours : Les instructions du langage Python