Python – Fichiers texte

Objectif ✔ : extraire et écrire des données dans un fichier texte (extensions .txt, .csv).

Structure des fichiers texte

Un fichier texte contient des **caractères invisibles**, par exemple :
- les ***sauts de lignes*** codés par `'\n'` en python ;
- les ***tabulations*** (sauts de colonnes, touche tab du clavier) codés par `'\t'` en python.

Exemple : deux fichiers texte contenant les mêmes informations (1 ligne d'en-tête, 1 ligne vide et 3 lignes de données)

Séparateur colonnes = tabulation

```
t	y1	y2	y3

0	0.54030230586814	1	-1
0.009765625	0.485310422984771	0.995129089050063	-0.995129089050063
0.01953125	0.429023740491728	0.990281903873608	-0.990281903873608
```

Séparateur colonnes = point-virgule

```
t;y1;y2;y3

0;0,54030230586814;1;-1
0,009765625;0,485310422984771;0,995129089050063;-0,995129089050063
0,01953125;0,429023740491728;0,990281903873608;-0,990281903873608
```

Fichiers ouverts dans Notepad++ avec affichage des caractères « invisibles »

```
t⟶y1→y2→y3CR LF
CR LF
0⟶0.54030230586814⟶1⟶-1CR LF
0.009765625→0.485310422984771⟶0.995129089050063⟶-0.995129089050063CR LF
0.01953125→0.429023740491728⟶0.990281903873608⟶-0.990281903873608CR LF
```

```
t;y1;y2;y3CR LF
CR LF
0;0,54030230586814;1;-1CR LF
0,009765625;0,485310422984771;0,995129089050063;-0,995129089050063CR LF
0,01953125;0,429023740491728;0,990281903873608;-0,990281903873608CR LF
```

⚠ ***L'inspection visuelle*** du contenu du fichier (Notepad++) est indispensable pour déterminer :
- le nombre de lignes à ignorer (titre, unités des colonnes, commentaires...) ;
- la nature du séparateur de colonnes (tabulation, point-virgule, virgule, ...) ;
- la nature du séparateur décimal (point ou virgule).

Un fichier est constitué de **lignes** séparées par un saut de ligne (symboles CR LF dans Notepad++ sous Windows, ci-dessus).
La méthode **`strip()`** permet de supprimer les sauts de lignes.

Exemple de ligne lue dans un fichier texte :

```
s = '4.51;0.7\n'        → présence d'un saut de ligne '\n'
s.strip()               → suppression
```

Renvoie la ***chaîne*** : `'4.51;0.7'`

Chaque ligne est une **chaîne** de caractères qui contient des données séparées par un caractère particulier appelé ***séparateur de colonnes*** (point-virgule, virgule, tabulation… ***l'inspection visuelle*** du fichier texte permet de savoir quel est le caractère utilisé).
La découpe d'une ligne (i.e. de la chaîne associée) en "colonnes" est effectuée par la méthode **`split()`** qui renvoie une liste contenant autant de chaînes qu'il y a de colonnes dans le fichier (i.e. de valeurs sur une ligne).

Exemple de ligne lue dans un fichier texte :

```
s = '4.51;0.7'          → deux valeurs séparées par un point-virgule
s.split(';')            → « découpage » de la chaîne s sur le critère « point-virgule »
```

Renvoie la liste de chaînes : `['4.51', '0.7']`

L'objectif après lecture et traitement est de reconstituer un tableau de mesures (tableau numpy de flottants).

Principe de l'extraction de données dans un fichier texte

1. ***Ouverture*** du fichier en lecture (mode 'read', `'r'` en abrégé).
2. ***Répéter*** en boucle :
   a. ***lire*** une ligne ;
   b. ***traiter*** la ligne → suppression des caractères « invisibles » grâce à la méthode **`strip()`** ;
   c. ***extraire*** les données d'une ligne → découpage de la ligne en « colonnes » grâce à la méthode **`split()`** ;
   d. ***remplacer*** si nécessaire le séparateur décimal ;
   e. ***stocker*** la ligne traitée dans une variable.
3. ***Fermer le fichier***. ⚠ **Ne pas oublier cette instruction.**

Ouverture d'un fichier : modes lecture, écriture ou ajout

Deux syntaxes possibles :

Syntaxe 1

```
1 | f = open(chemin_fichier, 'r')
2 | # Instructions
3 | f.close()
```

Syntaxe 2

```
1 | with open(chemin_fichier, 'r') as f:
2 | # Instructions
3 |
```

Dans les deux cas, une variable *f* contenant toutes les informations relatives au fichier est créée.
Cependant, *f* n'est pas le "contenu" du fichier à proprement parler, c'est un objet qui possède des méthodes permettant de lire le contenu du fichier.

La syntaxe 2 est préférable (fermeture automatique du fichier).

Paramètres de la fonction `open()` :

- `chemin_fichier` : chaîne (chemin absolu ou chemin relatif vers le fichier)
  Exemple de chemin absolu (pyzo sous Windows) : `Chemin_fichier = "D:\\Downloads\\mon_fichier.txt"`
- `'r'` (pour 'read') : ouverture en mode lecture
  Ouverture en mode écriture `'w'` (pour 'write') : efface et remplace le contenu du fichier
  Ouverture en mode ajout `'a'` (pour 'append') : ajoute du contenu à la fin du fichier

Une fois le fichier ouvert, il faut lire le contenu.

Lecture d'un fichier

Trois syntaxes possibles pour la ligne 2 des codes ci-dessus :

1.	Lecture de la totalité du fichier	`texte = f.read()`	`texte` est de type chaîne
2.	Lecture de la totalité des lignes du fichier	`lignes = f.readlines()`	`lignes` est une liste de chaînes
3.	Lecture ligne par ligne à l'aide d'une boucle	`ligne = f.readline()`	`ligne` est une chaîne

La 3ème syntaxe permet le traitement et l'extraction de données en cours de lecture, ce sera la méthode utilisée.

Une boucle « infinie » lit le fichier ligne par ligne, la sortie de boucle est provoquée par une ligne vide (sans saut de ligne) :

```
1 | with open(chemin_fichier, 'r') as f:     # Ouverture du fichier en lecture
2 |     while 1:                            # Boucle « infinie » (1 = True)
3 |         ligne = f.readline()            # Lecture d'une ligne
4 |         if ligne == "":                 # Si la fin du fichier est atteinte…
5 |             break                       # … on sort de la boucle
6 |         else:                           # Sinon…
7 |             # Instructions              # … on procède aux traitements, on stocke…
```

Il faut ensuite insérer les instructions nécessaires au traitement des lignes, à l'extraction et au stockage des données.

Exemple de fichier (inspection visuelle) :

```
t;y1;y2;y3

0;0,54030230586814;1;-1
0,009765625;0,485310422984771;0,995129089050063;-0,995129089050063
0,01953125;0,429023740491728;0,990281903873608;-0,990281903873608
0,029296875;0,371666995280259;0,985458328904506;-0,985458328904506
0,0390625;0,313466773471951;0,980658249139539;-0,980658249139539
```

- ✓ On constate que les deux premières lignes ne comportent pas de données (en-têtes de colonnes et ligne vide), elles devront être ignorées au moment du traitement.
- ✓ Le séparateur de colonnes est le point-virgule.
- ✓ Le séparateur décimal est la virgule et non le point.

Code complet

```
1  import numpy as np
2
3  # ! Configuration à ADAPTER au fichier utilisé !
4  lgn_ign = 2          # ADAPTER - Nombre de lignes à ignorer (en-tête…)
5  sep_dec = ','        # ADAPTER - Séparateur décimal : ',' ou '.'
6  sep_col = ';'        # ADAPTER - Séparateur de colonnes : '\t' si tabulation
7  chemin_fichier = 'C:\\TP\\regressi.csv'  # ADAPTER - Chemin vers le fichier
8
9
10 # Stockage des lignes lues dans le fichier dans une liste
11 data = [] # Liste de chaînes (cf. ci-dessous)
12
13 with open(chemin_fichier, 'r') as f:     # Ouverture fichier (fermeture auto)
14     while 1 :                            # Boucle "infinie" (jusqu'au break)
15         ligne = f.readline()             # Lecture d'une ligne = une chaîne
16         ligne = ligne.strip()            # Suppression du saut de ligne ('\n')
17         if ligne == "" :                 # Si ligne vide...
18             break                        # ...on quitte la boucle
19         else:                            # sinon...(traitement de la ligne)
20             lg = ligne.split(sep_col)    # 1 chaîne -> n chaînes (n = nbre colonnes)
21             if sep_dec == ',':           # Traitement séparateur décimal
22                 data.append([s.replace(',','.') for s in lg]) # Ajout à data
23             else:
24                 data.append(lg)
25
26 # Transformation en tableau numpy de flottants sauf lignes à ignorer
27 dataN = np.array(data[lgn_ign:], dtype=float)
```

Si le séparateur décimal est le point dans le fichier, les lignes 5, 21 à 23 sont inutiles.

La conversion en tableau numpy permet d'effectuer des traitements mathématiques sur les colonnes du tableau obtenu.
Exemple : `x = np.sin(dataN[:,0])*3` permet de définir un nouveau tableau de valeurs à partir de la colonne d'index 0.
Il est ensuite possible de tracer des courbes.