



MANIPULATIONS DE FICHIERS

Ce chapitre a pour but une présentation succincte de différentes manipulations de fichiers à l'aide de programmation Python. Cette manipulation permet de traiter des quantités importantes de données potentiellement utilisables dans un autre programme ou logiciel qu'un programme Python. Nous verrons quelques traitements sont possibles mais nous ne serons (et ne pourrons...) être exhaustifs : pour des manipulations plus évoluées ou pour l'utilisation d'options ou de méthodes non présentées ici, il convient d'utiliser l'aide en ligne sur Python voire les différentes pages internet servant de formation ou de tutoriels....

I) Chemin d'accès

Pour lire ou écrire sur un fichier, il faut utiliser son "chemin d'accès". Ce chemin d'accès depuis le gestionnaire de fichiers désigne "sa place" dans le répertoire (et les différents sous-dossiers). Par exemple un document "essai.txt" peut se trouver dans le répertoire : C:\Documents\Dossiers1\DossierA\ à l'adresse
C:\Documents\Dossiers1\DossierA\essai.txt

Ce document n'est accessible par un programme Python que si le fichier d'édition de ce programme Python est enregistré dans l'un des sous-dossiers parents de "essai.txt"

Par exemple si le fichier d'édition du programme Python est dans le dossier C:\Documents\, le chemin d'accès au fichier "essai.txt" est Dossiers1\DossierA\essai.txt

En pratique, on essaiera de travailler dans un dossier explicitement destiné aux traitements de programme Python (style "TP_Python" ou "TIPE" dans "Documents") et on mettra les fichiers à traiter dans un de ses sous-dossiers.

II) Lire les lignes d'un fichier texte

Principe de lecture

Le principe est le suivant :

- On ouvre un fichier
- On lit les lignes successivement
- On ferme le fichier

L'ouverture se fait avec la fonction **open**. On associe le fichier ouvert à un objet de type *file*.

Pour la lecture, on lit les lignes de l'objet *file* créé. Chaque ligne est considérée comme une chaîne de caractères (si on a affaire à des nombres, il faudra donc utiliser les fonctions **int** ou **float** selon que ces nombres sont des entiers ou des flottants). Remarquons



cependant qu'une ligne du fichier ne se termine pas au bord droit de la page mais au renvoi à la ligne ('\n').

La fermeture se fait avec la fonction **close**. Remarquons que si on ne ferme pas le fichier, on ne pourra pas l'utiliser tant que le programme Python l'ayant ouvert n'a pas été refermé...

Exemple 1 : On suppose que le texte est le fichier du cours sur les fonctions continues enregistré dans le bon répertoire au format txt, sous le nom cours07fonctionscontinues. Le script suivant va traiter les lignes successivement, compter le nombre de leurs caractères, calculer le cumul de ces caractères et afficher les lignes, et ce jusqu'à ce que le cumul dépasse 300.

En voici le script :

```

1 from os import getcwd, chdir
2
3 chdir("D:\\calio\\mpsi\\MPSI20152016\\")
4
5 mon_fichier = open('cours07fonctionscontinues.txt', 'r')
6
7 s , cpt = 0,0
8 for L in mon_fichier:
9     print(L)
10    s +=len(L)
11    cpt+=1
12    if s>300:
13        break
14 print('Dans les ',cpt," premières lignes du fichier, il y a ", s,"
    caractères")
15 mon_fichier.close()

```

Le 'r' dans la première ligne signifie que l'on va ouvrir le fichier en mode "lecture"

```

In [1]: (executing lines 1 to 15 of "coursfonction.py")
FONCTIONS CONTINUES DE ( DANS (

A ETUDE LOCALE D'UNE FONCTION

I) Limite d'une fonction en un point a
1) Limite

I est un intervalle de ( , f une fonction définie sur I et a?( est un élément de I ou (extrémité de I). Soit b?(.

Définition : On dit que f admet b pour limite en a ssi :

| ??? (+, ? ?? (+ | ?x?I, |x - a| ? ? ==> | f(x) - b | ? ??

Dans les 9 premières lignes du fichier, il y a 365 caractères

```

On constate que Python compte 1 ligne par passage à la ligne : un après le titre (FONCTIONS ...) et un autre après (A ETUDE LOCALE...). (Accessoirement on voit que les fichiers.txt n'aiment pas les symboles mathématiques)

Remarque : la méthode readlines permet de générer la liste de toutes les lignes du fichier. Ainsi le script :

```

| mon_fichier = open('cours07fonctionscontinues.txt', 'r')

```



```
Liste = mon_fichier.readlines()
mon_fichier.close()
```

permet de garder en mémoire toutes les lignes du fichier même après fermeture de celui-ci.

Cas des fichiers contenant des données sous forme de tableau

Lorsque le fichier est constitué de données d'un tableau, à l'intérieur de chaque ligne, les données sont séparées soit par une tabulation ('\t') soit par des séparateurs (';', ',', ':') par exemple). Pour l'extraction de ces données, on peut utiliser la méthode **split** qui renvoie la liste des différents éléments de la chaîne de caractères.

```
>>> "Nous sommes le 25".split(' ') renvoie ['Nous', 'sommes', 'le', '25']
```

Exemple 2 : On suppose que le texte est le fichier du planning des DS enregistré dans le bon répertoire au format txt, sous le nom planningDS.txt . On suppose également que dans chaque ligne, les éléments de chaque cellule sont séparés par une tabulation '\t'. Le script suivant va compter le nombre de DS de Maths et celui de Physique pour la MPSI.

Pour ce faire, on regarde d'abord le format du fichier :

Les deux premières colonnes contiennent des dates ou des noms de vacances

La troisième colonne est vide

La 2^{ème} colonne contient les informations que nous souhaitons

```
PLANNING DES DS des MPSI et PCSI: 2015 / 2016
```

```
Les DS ont lieu de 13h00 à 17h00 en amphithéâtre le lundi.
```

```
PREMIER SEMESTRE
DATE Classe MPSI      Classe PCSI
21 Septembre  Mathématiques  Mathématiques
28 Septembre  Sciences Physiques  Sciences Physiques
05 Octobre  Mathématiques  Mathématiques
12 Octobre  Sciences Physiques  Sciences Physiques
```

Script Python :

```
mon_fichier = open(planningDS.txt', 'r')
cptLigne = 0
devM, devP = 0,0
for L in mon_fichier:
    cptLigne +=1
    Liste = L.split('\t')          # On crée la liste associée à la ligne
    if '0' in Liste[0] or '1' in Liste[0] or '2' in Liste[0]: # On ne veut traiter que les lignes où le premier
                                                                    élément est une date

        if cptLigne > 6 and 'M' in Liste[6]:
            devM += 1
        if cptLigne > 6 and 'P' in Liste[6]:
            devP += 1

print(devM, devP)
mon_fichier.close()
```



III) Ecrire des données dans un fichier texte

Principe de l'écriture

Le principe est le suivant :

- On ouvre un fichier en mode écriture
- On y écrit des chaînes de caractères à l'aide de la méthode **write**
- On ferme le fichier avec la méthode **close**

L'ouverture se fait encore avec la fonction **open**. Par contre l'option à ajouter n'est plus **'r'** (pour read) mais **'w'** (pour write) ou **'a'** (pour append). L'option **'w'** efface les données préexistantes dans le fichier à ouvrir puis écrira celles que l'on va créer, l'option **'a'** écrira les nouvelles données à la suite des données préexistantes.

Pour l'écriture, on utilise la méthode **write**

Exemple 3 : On reprend le fichier `cours07fonctionscontinues` vu précédemment. Le script suivant va créer un fichier `cours07fonctionscontinuesMAJ` qui reprend le fichier de départ en remplaçant toutes les lettres minuscules par les majuscules correspondantes.

```
mon_fichier_ecrit = open('cours07fonctionscontinuesMAJ.txt','w')
mon_fichier_lu = open('cours07fonctionscontinues.txt','r')

for L in mon_fichier_lu:
    Lmaj = L.upper()
    mon_fichier_ecrit.write(Lmaj)

mon_fichier_ecrit.close()
mon_fichier_lu.close()
```

IV) Autres types de fichiers

On peut également traiter avec des modules adaptés d'autres types de fichiers que les fichiers texte. Par exemple les images peuvent être traitées avec le module **Image**, et les fichiers csv (tableur avec le séparateur ;) avec le module **csv**.

Dans les deux cas on ouvre les fichiers de la même façon, on crée une variable contenant une copie au format csv ou Image du fichier ouvert, on travaille sur cette variable puis on ferme le fichier avec `close`.

Dernier exemple : On peut sauvegarder au format pdf ou d'autres formats image une figure dessinée avec `matplotlib.pyplot` soit directement à partir de la fenêtre ouverte du dessin soit de façon automatique à l'aide de la méthode **savefig**...