

Lecture et écriture dans un fichier

Les chemins de fichiers

Il y a deux façons de décrire l'emplacement d'un fichier : son chemin absolu ou son chemin relatif :

- Son **chemin absolu** décrit l'intégralité des dossiers (ou répertoires³) menant au fichier, peu importe le répertoire courant, depuis un répertoire dit « racine ». Sous Windows, la racine est le nom de volume (C:\, D:\...), sous les systèmes de la famille Unix, c'est « / ».
- Son **chemin relatif** décrit la succession de répertoires à parcourir en prenant comme point de départ le répertoire courant dans lequel on se trouve.

Exemple : Quel est le chemin du fichier `explorer.exe` dans l'arborescence suivante ?

```
graph TD
  A[C:] --> B;
  A --> C[Program Files];
  B[Windows] --> D[{explorer.exe}];
  B --> E[{notepad.exe}];
  A --> F["Program Files(x86)"];
  A --> G[Users];
```

- Son chemin absolu est `C:\Windows\explorer.exe`.
- Son chemin relatif dépend du répertoire courant⁴, par exemple :

Répertoire courant	Chemin relatif
Windows	<code>explorer.exe</code>
C:	<code>Windows\explorer.exe</code>
Program Files	<code>..\Windows\explorer.exe</code> ⁵

Python reconnaît indifféremment les chemins indiqués avec *backslash* (norme Windows) « \ » qu'avec *slash* « / » (norme Unix et protocoles Internet).

f = open() et f.close()

Pour lire ou écrire dans un fichier depuis un programme Python, il faut d'abord **ouvrir le fichier**. On utilise la fonction `open()` qui prend pour paramètre le chemin (absolu ou relatif) du fichier et le mode d'ouverture :

- `'r'` pour ouvrir un fichier existant en mode lecture (*read-only*) est l'option par défaut.
- `'w'` en mode écriture (*write*) pour écrire dans un nouveau fichier (ou écraser un fichier déjà existant).

- `'a'` en mode ajout (*append*) pour écrire à la fin d'un fichier déjà existant (ou le créer s'il n'existe pas).

On peut compléter le mode d'ouverture avec `b` pour un fichier binaire (image ou son par exemple), on obtient `rb`, `wb`, `ab`.

⚠ On ne peut pas ouvrir en mode lecture un fichier qui n'existe pas.

```
f = open('fichier.txt', 'r')

>>> Traceback (most recent call last):
      File "<interactive input>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or directory: 'fichier.txt'
```

Par contre, en mode écriture `w` ou ajout `a`, si le fichier n'existe pas quand il est ouvert, alors il est créé.

```
f = open('fichier.txt', 'w')
```

Si « fichier.txt » ne se trouve pas dans le répertoire du programme Python, il faut donner son chemin pour y accéder avec `/` ou `\\`.

```
f = open('C:/.../.../fichier.txt', 'r')
f = open('C:\\...\\...\\fichier.txt', 'r')
```

⚠ Attention, il faut **toujours fermer un fichier après l'avoir ouvert**¹. La méthode à utiliser est `close()`.

```
>>> f.close()
```

with open() as f:

Une autre façon d'ouvrir « fichier.txt » est d'utiliser la construction suivante :

```
with open('fichier.txt', 'w') as f:
    # bloc d'instructions
```

Dans ce cas-là, le fichier est automatiquement fermé à la fin du bloc d'instructions (attention à l'indentation), il n'y a pas besoin de le fermer, cela évite beaucoup d'erreurs.

Écrire dans un fichier

Pour écrire dans un fichier, on utilise la méthode `.write()` en lui passant en paramètre une **chaîne de caractères** à écrire². Pour écrire des nombres il faut les convertir en `str` avant.

Créons un fichier qui contient des noms de pays avec leur capitale et leur population :

```
f = open(...)
```

```
f = open('pays.txt', 'w')
f.write('France;Paris;68\n')
f.close()
```

with open(...) as f:

```
with open('pays.txt', 'w') as f:
    f.write('France;Paris;68\n')
```

Noter le caractère « \n » pour indiquer un retour à la ligne. On peut ouvrir `pays.txt` par exemple avec le blocnote et vérifier que le texte a bien été écrit.

On peut ensuite écrire deux autres lignes à la suite en mode `'a'`.

f = open(...)

```
f = open('pays.txt', 'a')
f.write('Espagne;Madrid;48\n')
f.write('Italie;Rome;60\n')

f.close()
```

with open(...) as f:

```
with open('pays.txt', 'a') as f:
    f.write('Espagne;Madrid;48\n')
    f.write('Italie;Rome;60\n')
```

On peut maintenant ouvrir le fichier texte dans un éditeur de texte quelconque :

```
France;Paris;68
Espagne;Madrid;48
Italie;Rome;60
```

⚠ Noter qu'en utilisant l'instruction `f = open('pays.txt', 'a')`, rien n'est écrit dans le fichier si on oublie de fermer le fichier avec `f.close()` !

🔍 Exercice corrigé

Ecrire un programme qui crée un fichier 'parite.txt' contenant tous les nombres entre 0 et 100 suivis de pair ou impair :

```
0;pair
1;impair
2;pair
etc.
```

✓ Réponse



Lire un fichier

Il existe plusieurs approches pour lire les données dans un fichier.

La méthode `.read()`

La méthode `.read()` renvoie l'intégralité du fichier dans une chaîne de caractères :

f = open(...)

```
f = open('pays.txt', 'r')
data = f.read()
f.close()
```

with open(...) as f:

```
with open('pays.txt', 'r') as f:
    data = f.read()
>>>
```

⚠ À noter, le fichier n'est lu qu'**une seule fois** avant d'être refermé. Par exemple, suite au programme :

```
f = open('pays.txt', 'r')
data = f.read()
data2 = f.read()
f.close()
```

la variable `data2` sera une chaîne de caractères vide. Après le premier `read()`, l'interpréteur Python est arrivé au bout du fichier. Il ne recommence pas à le lire depuis le début et le second `read()` ne lit plus rien. Pour recommencer à lire au début du fichier, il faut le fermer et le rouvrir.

La méthode `.readline()`

La méthode `.readline()` permet de lire une seule ligne d'un fichier

f = open(...)

```
f = open('pays.txt', 'r')
ligne1 = f.readline()
f.close()
```

with open(...) as f:

```
with open('pays.txt', 'r') as f:
    ligne1 = f.readline()
>>>
```

⚠ À noter, une fois la première ligne lue, l'instruction `.readline()` suivante lit la seconde ligne et ainsi de suite jusqu'à la fin du fichier. Pour recommencer au début du fichier il faut fermer et rouvrir le fichier :

```
f = open('pays.txt', 'r')
ligne1 = f.readline() # première ligne du fichier
ligne2 = f.readline() # deuxième ligne du fichier
f.close()
```

⚠ Attention donc à ne pas confondre `readline()` qui renvoie une seule ligne dans une chaîne de caractères, avec `readlines()` qui renvoie un tableau de toutes les lignes.

La boucle `for... in ...`

Une boucle `for... in ...` permet d'itérer sur toutes les lignes d'un fichier.

`f = open(...)`

```
f = open('pays.txt', 'r')
for ligne in f:
    print(ligne)
f.close()
```

`with open(...) as f:`

```
with open('pays.txt', 'r') as f:
    for ligne in f:
        print(ligne)
```

Ici, la variable `ligne` est une chaîne de caractère qui prend la valeur de chaque ligne de `pays.txt`.

Exercice corrigé




« Green Eggs and Ham is one of Seuss's "Beginner Books", written with very simple vocabulary for beginning readers. The vocabulary of the text consists of just 50 words and was the result of a bet between Seuss and Bennett Cerf, Dr. Seuss's publisher » Source : Wikipedia.

Ecrire un programme pour vérifier si Dr. Seuss a gagné son pari d'écrire un livre en utilisant moins de 50 mots dans son livre : <https://www.clear.rice.edu/comp200/resources/texts/Green%20Eggs%20and%20Ham.txt>

Aide : Utiliser les méthodes `.lower()` pour convertir une chaîne de caractères en minuscule, `.replace()` pour remplacer les les signes de ponctuation (., - ! ?) par des espaces, et `.split()` pour séparer les mots dans une chaîne de caractères.

Réponse



1. On peut essayer de supprimer un fichier ouvert depuis Windows explorer pour s'en convaincre. 
2. `write()` renvoie le nombre de caractères qui ont été écrits dans le fichier, utile par exemple pour vérifier que le fichier contient bien le texte qu'on y a écrit. 
3. Depuis Windows 7, le terme « dossier » remplace « répertoire » 
4. En Python, on peut déterminer le répertoire courant avec l'instruction `getcwd()` du module `os` (noter le double `\\` dans la chaîne de caractère pour 'échapper' le caractère `\\`)

```
>>> import os
>>> os.getcwd()
'C:\\Program Files\\PyScripter'
```



5. « `..` » désigne le répertoire parent. 