

# Introduction à la gestion de fichiers .txt avec Python

# De multiples possibilités



On peut utiliser le langage Python pour gérer de nombreux types de fichiers :

- fichiers images (.jpg, .png, ...)
- fichiers sons (.wav, ...)
- fichiers vidéos (.avi, .mpg, ...)
- ...

Il peut aussi être utilisé pour piloter des cartes d'acquisition (comme la carte Sysam) ou des microcontrôleur (Micro Python ...) ou encore générer des applications smartphones ou des jeux vidéos.

# Des microcontrôleurs commandés directement en Python

# Types de fichiers



On divise habituellement les fichiers en deux catégories :

- Les fichiers textes

Ceux-ci sont humainement lisibles. Autrement dit, ils contiennent du texte (lettres, ponctuations, nombres, ...). Leur contenu est souvent divisé en lignes. Ces fichiers peuvent être lus et manipulés avec les outils traditionnels. Les fichiers .txt (ou .csv) sont un exemple type de fichier texte que l'on peut ouvrir dans bloc note.

- Les fichiers binaires

Ceux-ci ne contiennent pas (exclusivement) du texte. Ils ne peuvent être convenablement traités que par des logiciels spécialisés. Un fichier PDF, une image JPEG ou un mp3 sont quelques exemples de fichiers binaires.

Ici, nous allons exclusivement nous intéresser aux fichiers textes.

# Instanciation d'un Objet fichier



On commence par attribuer à une variable (désignée ici par `ObjetFichier`) le contenu du fichier texte que l'on ouvre. La syntaxe de base est `ObjetFichier=open('chemin d'accès du fichier','mode d'ouverture')`. On dit qu'on instancie un objet fichier.

Le nom du fichier correspond au chemin d'accès exact (relatif ou absolu)

# Modes d'ouverture d'un fichier .txt



Il existe plusieurs modes d'ouverture :

- **'r'**, pour une ouverture en lecture (READ).
- **'w'**, pour une ouverture en écriture (WRITE), à chaque ouverture le contenu du fichier est écrasé. Si le fichier n'existe pas python le crée. a, pour une ouverture en mode ajout à la fin du fichier (APPEND). Si le fichier n'existe pas python le crée.
- **'b'**, pour une ouverture en mode binaire.
- **'t'**, pour une ouverture en mode texte.
- **'x'**, crée un nouveau fichier et l'ouvre pour écriture

# Fermer un fichier .txt



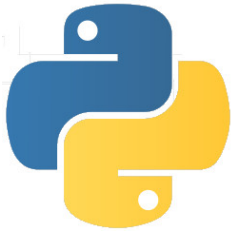
Comme tout élément ouvert, il faut le refermer une fois les instructions terminées. Pour cela on utilise la méthode **close()** :

```
ObjetFichier.close()
```

# Écrire dans un fichier .txt



L'écriture dans un fichier texte ne peut se faire que si le fichier est ouvert en mode écriture ou append.
On écrit alors en utilisant la méthode **write()**



# Lire un fichier .txt



La lecture d'un fichier .txt n'est possible que si le fichier est ouvert en mode lecture. De nombreuses syntaxes permettent de lire tout ou partie d'un fichier .txt avec Python. En voici quelques unes :

- La méthode **read()** permet de lire tout le fichier.
- La méthode **readlines()** lit toutes les lignes du fichier et les range dans une séquence.
- La méthode **readline()** (au singulier) qui lit une ligne dans le fichier. Indispensable pour lire des lignes au coup par coup.
- On peut aussi utiliser l'objet fichier comme un itérateur. Pratique si on souhaite traiter l'une après l'autre toutes les lignes d'un fichier. (`for lignes in ObjetFichier: ...`)

Le contenu de l'objet fichier est une chaîne de caractères.

# Quelques fonctionnalités supplémentaires



Les fonctionnalités suivantes sont des méthodes associées aux chaînes de caractères et peuvent être utilisées à profit pour extraire des informations d'un fichier .txt :

- La méthode **split** peut être très utile pour séparer des données contenues dans une ligne.
- La méthode **replace** remplace une chaine de caractères par une autre
- La méthode **rstrip** permet de supprimer un caractère donné. C'est une fonctionnalité particulièrement pratique pour supprimer les retours à la ligne (\n)

Pour des informations détaillées sur ces fonctionnalités, utiliser help(str) dans la console python