Enregistrer et lire des fichiers

I – Création d'un fichier texte

Pour créer un fichier-texte, il faut utiliser la fonction native **open**, lui fournir un argument **"w"** comme write et l'encodage qu'il faudra utiliser pour stocker le texte sous forme d'octets.

On crée alors un objet-fichier noté **f** ici qui permettra d'interagir avec les octets enregistrés sur le support physique.

La méthode write permettra alors de placer des octets dans ce fichier.

Il est vitale de le refermer avec la méthode **close**.

```
# 1 - Création de l'objet-fichier
f = open('aaa.txt', 'w', encoding="utf-8")

# 2 - Remplissage progressif du fichier
f.write("Premier contenu\n")
f.write("Deuxième ajout\n")

# 3 - Fermeture de l'objet-fichier
f.close()
```

ATTENTION AUX PASSAGES A LA LIGNE – LINE FEED \n : ils ne sont pas rajoutés automatiquement. Il s'agit bien d'un caractère que vous devez rajouter vous même.

II – Modification d'un fichier-texte

L'utilisation de open avec **"w"** crée un nouveau fichier ou écrase l'ancien si il existe déjà !

Si on veut rajouter du contenu à la suite du contenu qui existe déjà, il faut utiliser l'argument **"a" comme append**.

Si le fichier n'existe pas, il sera créé.

```
# 1 - Ouverture de l'objet-fichier
f = open('aaa.txt', 'a', encoding="utf-8")

# 2 – RAJOUT dans le fichier
f.write("Troisième contenu\n")
f.write("Quatrième ajout\n")

# 3 - Fermeture de l'objet-fichier
f.close()
```

III – Lecture d'un fichier-texte

Il faudra cette fois utiliser l'argument **"r" comme read**.

3.1 METHODE DE LECTURE DIRECTE

On peut lire manuellement ligne par ligne avec la méthode **readline**. Qu'est-ce qu'une ligne ? Une suite de caractère dont le dernier caractère est **\n**.

```
f = open('aaa.txt', 'r', encoding="utf-8")

print(f.readline(), end="")
print(f.readline(), end="")
print(f.readline(), end="")

f.close()
```

Remarque : si on stocke la ligne plutôt que de juste l'afficher, le string contient bien \n à la fois.

```
f = open('aaa.txt', 'r', encoding="utf-8")
ligne = f.readline()
f.close()
```

–> ligne contient **"Premier contenu\n".**

3.2 METHODE DE LECTURE TANT QUE

On peut utiliser un TANT QUE la ligne qu'on vient de lire n'est pas vide :

```
f = open('aaa.txt', 'r', encoding="utf-8")

ligne = f.readline()
while ligne != "":
      print(ligne, end="")
      ligne = f.readline()

f.close()
```

3.3 METHODE DE LECTURE FOR

On peut lire l'intégralité d'une fichier ligne par ligne en utilisant un FOR de façon itérative.

```
f = open('aaa.txt', 'r', encoding="utf-8")

for ligne in f:
      print(ligne, end="")

f.close()
```

3.4 Lecture un caractère à la fois

Dans ce cas, il faut utiliser la méthode **read** à qui on peut demander de lire 1, 2 ou 5 caractères…

```
f = open('aaa.txt', 'r', encoding="utf-8")

lecture = "o"
while lecture:
	lecture = f.read(1)
	if lecture == 'a' or lecture == 'A':
		lecture = '@'
	print(lecture, end="*")

f.close()
```

3.5 Taille et encodage

Le choix de l'encodage influence la taille du fichier-texte :

→ **Encodage Table 1 octet** : un caractère pèse toujours un seul octet. La plus connue est **latin-1** (de son petit nom **iso8859_1**).

→ **Encodage UTF-8** : utilise le système UNICODE en encodant les caractères ASCII sur un octet, et les autres sur 2, 3 ou 4 octets.

→ **Encodage UTF-16** : utilise le système UNICODE en encodant les caractères assez courants sur 2 octets et les autres sur 3 ou 4 octets.

→ **Encodage UTF-32** : utilise le système UNICODE en encodant les caractères sur 4 octets.

IV – Gestion des strings (suite de celle de l'activité précedente)

Python intègre beaucoup de contrôle et de méthodes qui peuvent s'appliquer aux strings. Voici quelques connaissances à avoir en tête.

Méthode replace

Cette méthode renvoie un nouveau string en remplaçant les caractères désignés par les seconds. Elle sert notamment à supprimer certains caractères.

```
>>> "Bonjour à tous\n".replace("\n", "")
'Bonjour à tous'
```

Test d'appartenance

```
>>> "B" in "bonjour")
False

>>> "u" in "Bonjour")
True
```

Méthode endswith

Cette méthode booléenne qui teste si la fin est bonne.

```
>>> 'perceval.txt'.endswith('.txt')
True

>>> 'perceval.txt'[-4:] == '.txt'
True
```

Méthode startswith

Cette méthode booléenne qui teste si la fin est bonne.

```
>>> 'perceval.txt'.startswith('per')
True

>>> 'perceval.txt'[0:3] == 'per'
True
```

Remarques finales

→ On peut également lire directement les octets (bytes). Exemple en lecture d'octets :

```
open('lecture.txt', 'br')
```

→ On peut retourner en début de fichier avec **f.seek(0)**

→ On peut récupérer avec la méthode **readlines()** (avec un s) l'intégralité du fichier-texte dans un tableau où chaque élément est une des lignes (mais attention, si le fichier est grand, ça va faire beaucoup de choses en mémoire vive…)

```
t = f.readlines()
print(t[0])   # Affiche la première ligne
```