# Python 32 - Enregistrer et lire des fichiers

## I – Création d'un fichier texte

**Encodage des caractères** : transformation de caractères en valeurs précises d'octets (0-255).

ASCII → Encodage historique de 127 caractères fondamentaux.

UTF-8 → Encodage standard aujourd'hui, permet d'encoder n'importe quel caractère.

### Créer un nouveau fichier-texte

Pour créer un fichier-texte, il faut utiliser la fonction native **open()** avec un argument **"w"** (write) et l'encodage à utiliser(**"utf-8"**).

La méthode **write()** permet de placer les octets du texte encodé dans ce fichier.

Il est vital de le refermer avec la méthode **close()**.

**Les passages à la ligne (line feed en anglais, \n en Python)** ne sont pas rajoutés automatiquement. C'est à vous de demander explicitement le passage à la ligne.

[DOC 1]
```
# 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()
```

## II – Modification d'un fichier-texte

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.

[DOC 2]
```
# 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 avec "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**.

[DOC 3]
```
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 un **\n** à la fin. C'est pour cela qu'on utilise ici le paramètre **end=""** qui permet de ne pas rajouter un passage à la ligne automatiquement à la fin de **print()**. Sinon, il y en aurait deux : celui du string **\n** et celui du print.

On notera qu'on peut lire au-delà de la « fin », on obtient juste le code ligne-vide **""**.

### 3.2 METHODE DE LECTURE TANT QUE

On lit la première ligne et on boucle TANT QUE la ligne qu'on vient de lire n'est pas vide :

[DOC 4]
```
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.

[DOC 5]

```
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

On peut utiliser la méthode **read()** à qui on peut demander de lire 1, 2 ou 5 caractères...

[DOC 6]

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

lecture = "string pas vide"
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. Un peu plus lourd mais on peut coder tous les caractères possibles.

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

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

## IV – [COMPLEMENT STRING] : Gestion des strings

### Méthode replace()

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

### Méthode endswith()

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

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

### Méthode split()

Permet de séparer un string en plusieurs morceaux et récupérer les parties dans un tableau.

```
>>> t = 'un deux trois'.split(' ')
>>> t
['un', 'deux', 'trois']
```

## Remarque finale

→ 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
```

# Python 32 – DOCUMENTS **Enregistrer et lire des fichiers**

[DOC 1]

```
# 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()
```

[DOC 2]

```
# 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()
```

[DOC 3]

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

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

f.close()
```

[DOC 4]

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

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

f.close()
```

[DOC 5]

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

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

f.close()
```

[DOC 6]

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

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

f.close()
```