

Chapitre 6

Manipulation de Fichiers

6.1 Accès à un fichier

6.1.1 Création d'un objet-fichier

• En `python`, création et manipulation d'un fichier se font par l'intermédiaire d'un objet particulier, appelé objet-fichier, généré par la fonction :

```
objet_fichier = open(nom du fichier, mode d'accès).
```

Les paramètres sont des chaînes de caractère :

– `nom du fichier` est le nom du fichier, avec son extension.

– `mode d'accès` peut être :

'w' : (write) ouverture pour écriture seule. Lorsque le fichier n'existe pas il est créé dans le répertoire courant ; lorsque le fichier existe il est écrasé.

'a' : (append) ouverture pour écriture seule. Lorsque le fichier n'existe pas il est créé dans le répertoire courant ; lorsque le fichier existe les données écrites le seront à la suite.

'r' : (read) ouverture pour lecture seule. Le fichier doit exister dans le répertoire courant.

'+' : ouverture pour lecture et écriture. Le fichier doit exister.

6.1.2 Contraintes

• Pour créer un objet-fichier :

– Le fichier doit être présent dans le dossier courant (*current working directory* ou *répertoire de travail*) : en général c'est le dossier utilisateur (celui de même nom que le compte utilisateur).

– Sinon on peut précéder le nom du fichier par le chemin d'accès :

```
obj_fich = open('/Users/JPP/Documents/MonFichier.ext',  
'r')
```

• Pour obtenir le répertoire de travail, utiliser la fonction `getcwd` du module `os` :

```
>>> import os  
>>> os.getcwd()  
'/Users/JPP'
```

• Pour modifier le répertoire de travail, utiliser la fonction `chdir` du module `os` :

```
>>> os.chdir('/Users/JPP/Documents')  
>>> os.getcwd()  
'/Users/JPP/Documents'
```

• **Remarque** : – Pour pouvoir être lu, un fichier texte doit être écrit en utilisant une table de caractères compatible ; `python` se charge de la conversion vers la table ASCII/Unicode.

– Un fichier écrit avec un encodage différent provoquera une erreur.

– Pour éviter cela, l'option `errors="surrogateescape"` permet d'ignorer les caractères non-compatibles pour éviter les messages d'erreur.

```
f = open('fichier', 'r', errors="surrogateescape")
```

6.1.3 Fermeture

• Une fois la lecture et l'écriture dans le fichier terminés il faut refermer l'objet-fichier avec l'instruction : `objet_fichier.close()`

– Les modifications apportées au fichier ne seront prises en compte qu'après la fermeture.

– Tant qu'un fichier n'a pas été fermé, son ouverture provoquera une erreur.



Ne jamais oublier de finir par refermer un objet-fichier par `objet_fichier.close()`

6.1.4 Méthodes des objets-fichiers

Un objet-fichier admet les méthodes :

Lorsque ouvert en écriture :

- `write()` écrit une chaîne de caractère en fin de fichier ouvert en écriture.
- `writelines()` écrit une liste de chaînes de caractères en les concaténant.

lorsque **ouvert en lecture** :

- `read()` lit l'intégralité du fichier. `read(n)` lit `n` caractères.
- `readline()` lit la ligne suivante.
- `readlines()` retourne une liste contenant toutes les lignes du fichier.

La position avance au fur et à mesure de la lecture/écriture :

- `tell()` renvoie la position (exprimée en caractères).
- `seek(n)` déplace la position au caractère `n`.

On peut aussi parcourir le fichier ligne après ligne :

```
for ligne in objet.fichier
```

6.1.5 Exemples



```
>>> objet = open('monfichier.txt','r')
>>> lect = objet.read()
>>> print(lect)
Premiere ligne
Deuxieme ligne...Suite deuxieme ligne
Troisieme ligne
>>> objet.close()
```

```
>>> objet = open('monfichier.txt','r')
>>> liste = objet.readlines()
>>> print(liste)
['Premiere ligne\n', 'Deuxieme ligne...Suite deuxieme ligne\n',
'Troisieme ligne\n']
>>> objet.close()
```

- Exemple : compter le nombre de ligne d'un fichier texte :

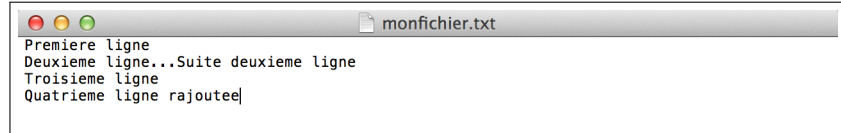
```
def nbreLignes(fichier):
    objet = open(fichier,'r')
    compteur = 0
    for ligne in objet:    # Parcours des lignes
        compteur += 1
    return compteur
```

```
>>> nbreLignes('monfichier.txt')
3
```

- Ajouter une ligne en fin d'un fichier texte :

```
def ajouteLigne(fichier,ligne):
    objet = open(fichier,'a')
    objet.write(ligne + '\n')
    objet.close()
```

```
>>> ajouteLigne('monfichier.txt','Quatrieme ligne rajoutee')
```



- Exemple : supprimer une ligne d'un fichier texte :

```
def effaceLigne(fichier,i):
    objet = open(fichier,'r')
    liste = objet.readlines()
    liste.pop(i-1)
    objet.close()
    objet = open(fichier,'w')
    objet.writelines(liste)
    objet.close()
```



6.2 Stockage de données numériques

6.2.1 Première méthode

- Et pour stocker des données numériques ?

Première méthode : On peut le faire en les stockant sous forme de chaînes de caractères séparées par un séparateur ; le retour à la ligne `'\n'` par exemple :

- Exemple : une fonction écrivant sous forme texte des données numériques issues d'une liste :

```
def wdata(fichier,liste):
    f = open(fichier,'w')    # Ouverture en écriture
    for x in liste:          # Parcours de la liste
        f.write(str(x)+'\n')
    objet.close()
```

- Lecture des données à partir du fichier et renvoi dans une liste :

```
def rdata(fichier):
    f = open(fichier,'r')
    liste = f.readlines()
    f.close()
    return [float(x) for x in liste]
```

- Création d'un fichier `datafile` contenant les nombres entiers de 1 à 10.

```
>>> wdata('datafile',[1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```



- Lecture de ses données :

```
>>> print(rdata('datafile'))
[1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0]
```

- Ajout d'une donnée en fin de fichier :

```
def adata(fichier,x):    # Ajout d'un nombre en fin de fichier
    f = open(fichier,'a'); f.write(str(x)+'\n'); f.close()
```

```
>>> adata('datafile', 11)
```

6.2.2 Deuxième méthode : avec la fonction `split`

On peut le faire en les stockant sur une ligne sous forme de chaînes de caractères séparées par un séparateur : un point-virgule `' ; '` par exemple :

- Fonction écrivant des données numériques issues d'une liste :

```
def wdata(fichier,liste):
    f = open(fichier,'w')    # Ouverture en écriture
    for x in liste[:-1]:    # Parcours de la liste
        # Ecriture de la donnée convertie en str + ';'
        f.write(str(x) + ';')
    f.write(str(liste[-1]))
    objet.close()
```

- Lecture des données à partir du fichier et renvoi dans une liste :

```
def rdata(fichier):
    f = open(fichier,'r')
    ligne = f.readline()
    liste = ligne.split(';')    # crée la liste des mots
    f.close()
    result = []
    for x in liste:
        result.append(float(x))
```

- On peut programmer une fonction faisant la même chose que la méthode `split` ainsi :

```
def split(s, c=' '): # découpe la chaîne s le long de c
    n = len(s)    # n lgr de la chaîne s
    result = []    # initialement result est la liste vide
    mot = ''    # initialement mot est la chaîne vide
    flag = False    # flag est vrai quand on remplit un mot
    for i in range(n):    # on parcourt la chaîne s
        if s[i] not in c:    # si n'est pas un séparateur de c
            mot += s[i]    # on l'ajoute à la fin de mot
            flag = True    # et on en garde le souvenir
        else:    # sinon
            if flag: # si ne suit pas un séparateur c
                result.append(mot)    # ajouter mot
                mot = ''    # réinitialisation de mot
                flag = False    # et de flag
    if flag:
        result.append(mot)    # ajouter le dernier mot
    return result
```

6.2.3 Troisième méthode : avec la fonction `eval`

On stocke la liste des données numériques (convertie en chaîne de caractère).

- Fonction écrivant des données numériques issues d'une liste :

```
def wdata(fichier,liste):
    f = open(fichier,'w')    # Ouverture en écriture
    f.write(str(liste))
    objet.close()
```

- Pour la lecture nous allons utiliser la fonction `eval()` : elle prend en paramètre une chaîne de caractère représentant une donnée, ou une instruction, et la convertit en ce qu'elle représente :

```
>>> eval('[1, 2]')
[1,2]
>>> eval("print('bonjour')")    # provoque l'exécution
bonjour
```

```
def rdata(fichier):  
    f = open(fichier,'r')  
    ligne = f.readline()  
    liste = eval(ligne)      # eval() reconvertit  
    return liste
```