Lecture et écriture dans un fichier [1]

On considère un fichier existant 'nom_fichier' (où nom_fichier peut être remplacé par le chemin complet pour accéder au fichier).

Pour accéder au fichier:

```
>>> f=open('nom_fichier' ,'w')
```

Le premier argument est une chaîne contenant le nom du fichier. Le deuxième argument est une autre chaîne contenant quelques caractères décrivant le mode d'ouverture:

- 'r' quand l'accès au fichier n'est permis qu'en mode lecture.
- 'w' quand l'accès au fichier n'est permis qu'en écriture seulement (un fichier existant portant le même nom sera alors écrasé).
- 'a' ouvre le fichier en mode ajout :toute donnée écrite dans le fichier est automatiquement ajoutée à la fin
- 'r+' ouverture du fichier en lecture/écriture.
- 'b' collé à la fin du mode indique que le fichier doit être ouvert en mode binaire c'est-à-dire que les données sont lues et écrites sous formes d'octets (type bytes). Ce mode est à utiliser pour les fichiers contenant autre chose que du texte.

Méthodes de l'objet fichier
On considère le fichier `tortue.txt contenant les lignes` :

```
Rien ne sert de courir ; il faut partir à point.
Le Lièvre et la Tortue en sont un témoignage.
Gageons, dit celle-ci, que vous n'atteindrez point
Si tôt que moi ce but. Si tôt ? Êtes-vous sage ?
Repartit l'Animal léger.
Ma Commère, il vous faut purger
Avec quatre grains d'ellébore.
Sage ou non, je parie encore.
Ainsi fut fait : et de tous deux
On mit près du but les enjeux.
```

Pour coller du texte dans l'éditeur `Vim` : Ctrl+Maj+V

- Pour lire le contenu d'un fichier, on appelle la méthode f.**read**(*taille*) : elle lit une certaine quantité de données et les donne sous la forme d'une chaîne (en mode texte).Plus précisément, La lecture se fait sur au maximum un nombre de caractères égal à l'argument *taille*(en mode texte) ou sur au maximum *taille* octets (en mode binaire) .

  par exemple

  ```
  >>>f=open("tortue.txt",'r')
  >>>x=f.read(9)
  >>>print(x)
  Le Lièvre
  ```

  Lorsque taille est omis ou négatif, la totalité du contenu du fichier sera lue et renvoyée .

[1] https://docs.python.org/fr/3/tutorial/inputoutput.html#reading-and-writing-files

- La méthode **readline()** lit le fichier ligne par ligne .

```
>>> f.readline()
Rien ne sert de courir ; il faut partir à point.
```

Ainsi le premier appel à readline() retourne la première ligne du fichier sous forme de chaîne de caractères qui se termine par le caractère **n** de fin de ligne.
si f.readline() renvoie une chaîne vide, c'est que la fin du fichier a été atteinte, alors qu'une ligne vide est représentée par **n**

Exercice : Ecrire un programme en `Python` qui affiche toutes les lignes de "tortue.txt". (utiliser `readline()`

- Pour lire ligne à ligne, vous pouvez aussi boucler sur l'objet fichier. C'est plus efficace en termes de gestion mémoire, plus rapide et donne un code plus simple :

```
for ligne in f:
     print(ligne, end='')
```

- La méthode **readlines**()

```
f=open("tortue.txt", 'r')
lecture=f.readlines()
for ligne in lecture:
    print(ligne)
f.close()
```

Ici la méthode *readlines*() agit sur l'objet fichier déplaçant le curseur de lecture du début à la fin du fichier, puis elle renvoie une liste contenant toutes les lignes du fichier .
Chaque ligne est une chaîne de caractères qui se termine par \n

- `>>> fichier.close()`

on applique la méthode **close**() sur l'objet fichier, ce qui ferme le fichier . La méthode close() ne renvoie rien mais modifie l'état de l'objet fichier fermé. Toute tentative de lecture des lignes du fichier, est signalée par une erreur.

- Une bonne pratique est d'utiliser le mot-clé ***with*** ,qui ferme correctement le fichier, même si une exception est levée.

```
>>>with open('nom fichier', mode )
```

Le premier argument est une chaîne contenant le nom du fichier.
Le deuxième argument est une autre chaîne contenant quelques caractères décrivant la façon dont le fichier est utilisé. mode peut être 'r','w','a' ou 'r+' (voir plus haut) .
L'argument mode est optionnel, sa valeur par défaut est 'r'.

Ainsi, pour lire dans un fichier :

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

Si vous n'utilisez pas le mot clé with, vous devez appeler f.close() pour fermer le fichier, et ainsi immédiatement libérer les ressources qu'il utilise.

- Pour écrire dans un fichier :

```
>>> with open('nom_fichier','w') as f:
...f.write(chaine)
```

f.**write**(chaine) écrit le contenu de chaine dans le fichier et renvoie le nombre de caractères écrits.

Remarque importante:Appeler f.write() sans utiliser le mot clé with ni appeler f.close() pourrait mener à une situation où les arguments de f.write() ne seraient pas complètement écrits sur le disque, même si le programme se termine avec succès.

Exercice 1

1. Ecrire une fonction qui copie les données d'un dictionnaire dans un fichier texte.

```
d={'a':'xx','b':'yy','c':'zz}
```

le fichier contient alors les lignes :

```
a:xx
b:yy
c:zz
```

2. Ecrire une fonction qui réalise l'opération inverse: elle lit dans un fichier où les données sont écrites comme précédemment et renvoie un dictionnaire.

Exercice 2

1. On considère le dictionnaire suivant contenant deux éléments du tableau de Mendeliev.

```
tab_mend={'h':['hydrogene',1],'li':['lithium',3]}
```

Ecrire un programme qui copie les données du dictionnaire dans un fichier nommé *"Mendeliev.txt"* sous la forme:

```
symbole,nom,numéro
h,hydrogène,1
li,lithium,3
```

2. Recopier les éléments de la dernière colonne du tableau de Mendeliev dans un fichier "derniere_colonne.txt" sous la forme de la question précédente.
3. Ecrire un programme en Python qui lit les données du fichier "derniere_colonne.txt"ligne par ligne et qui retourne le dictionnaire

```
{'He':['Helium',2],'Ne':['Neon',10],...}
```

Exercice 3

1. Créer un fichier `mesadresses.txt` à l'aide de l'éditeur `Vim` , dans lequel vous écrirez 3 ou 4 lignes , chaque ligne comportant une adresse IP en binaire.

2. Dans un fichier Python , Vous écrirez les fonctions suivantes:

   - Une fonction `binToDec`(Nbinaire) qui prend comme paramètre une chaîne contenant un binaire et qui renvoie l'entier décimal correspondant.

     On pourra utiliser l'algorithme de Horner vu en début d'année dont on rappelle ici la démarche :
     si $n = (x_3x_2x_1x_0)_2$ alors la valeur décimale $ndec$:

     $ndec = x_3 \times 2^3 + x_2 \times 2^2 + x_1 \times 2^1 + x_0 = ((x_3 \times 2 + x_2) \times 2 + x_1) \times 2 + x_0$

     Si on applique l'algorithme suivant:
     $r_3 = x_3$
     $r_2 = r_3 \times 2 + x_2$
     $r_1 = r_2 \times 2 + x_1$
     $r_0 = r_1 \times 2 + x_0$

     alors on obtient $r_0 = x_3 \times 2^3 + x_2 \times 2^2 + x_1 \times 2^1 + x_0$

   - Une fonction `traduireAdresses`(nom_fichier) qui prend pour paramètre un fichier du type `mesadresses.txt` .
     La fonction ne renvoie rien , mais affiche les adresses IP contenues dans `mesadresses.txt` en adresses écrites sous format décimal.

   Par exemple , si le fichier `mesadresses.txt` contient les lignes :

```
10001000.11001100.10011001.00001100
10001000.11111111.10101010.11110001
11000000.10101000.10110011.11010000
11001111.11101100.10000000.11100001
```

   alors l'affichage fourni par la fonction `traduireAdresses` sera :

```
136.204.153.12.
136.255.170.241.
192.168.179.208.
207.236.128.225.
```