CPGE BCPST1
Année Académique: 2023 - 2024

TP N°4 d'Informatique

Gestion de fichiers

Anicet E. T. Ebou, ediman.ebou@inphb.ci

---

1. Lecture de fichiers

En programmation, la lecture d'un fichier se fait en trois temps :

1. l'ouverture du fichier;
2. la lecture proprement dite;
3. la fermeture du fichier.

Traitons un exemple avant de détailler le fonctionnement précis de chaque étape. Copiez la séquence d'instructions suivantes dans une fenêtre de fichier.

```
# ouverture
f = open("test.txt", "r")
# récupération du texte dans une variable
texte = f.read()
# affichage du texte récupéré
print(texte)
# fermeture
f.close()
```

Testez l'exécution du fichier.

Ouverture du fichier

L'ouverture d'un fichier se réalise grâce à la fonction `open` qui a deux paramètres:

1. Le premier paramètre est une chaîne de caractères contenant le nom (avec suffixe éventuel) du fichier.
2. Le deuxième paramètre est aussi une chaîne de caractères contenant le mode de traitement du fichier. La valeur de ce paramètre est `"r"` pour la lecture d'un fichier.

La fonction `open` retourne un descripteur de fichiers qui est un objet particulier. Ce descripteur doit être récupéré lors de l'appel dans une variable pour pouvoir être manipulé par la suite.

Fermeture d'un fichier

Rappelons que le traitement d'un fichier se fait en trois temps: ouverture, lecture (ou écriture un peu plus tard), fermeture. Comme la lecture peut être optionnelle, nous traitons la fermeture. Nous verrons ensuite la lecture.

La fermeture d'un fichier s'effectue grâce à la méthode `close()`. Elle peut s'effectuer même si le fichier n'a pas été lu.

Lecture de fichier

Methode 1

L'intégralité d'un fichier peut être récupérée dans une unique chaîne de caractères grâce à la méthode `read()`.

```
#(peut-etre faut-il ouvrir de nouveau le fichier ?)
texte = f.read()
texte
print(texte)
```

Méthode 2

La lecture en une fois du fichier (méthode 1) est peu pratique. Une seconde possibilité consiste à indiquer en paramètre de la fonction `read` le nombre de caractères qu'on désire lire.

(a) Ouvrez de nouveau le fichier texte.

(b) Testez, en ligne de commande, plusieurs fois de suite la suite d'instructions :

```
ch = f.read(5)
print(ch)
```

(c) Fermez le fichier.

Méthode 3

La lecture par paquets (méthode 2) peut s'avérer intéressante. Néanmoins dans la suite nous utiliserons une autre méthode consistant à lire un fichier texte ligne par ligne à l'aide de la méthode `readline()`.

(a) Ouvrez le fichier texte.

(b) Appliquez la séance d'instructions suivante autant de fois que nécessaire pour lire tout le fichier :

```
ch = f.readline()
print(ch)
```

(c) Fermez le fichier.

2. Exercices d'application

1. Ecrivez une fonction `lecture` qui lit le fichier `test.txt` ligne par ligne et affiche son contenu à l'écran.
2. Modifiez la fonction `lecture` pour afficher le contenu d'un fichier dont le nom est donné en paramètre.
3. Validez cette nouvelle fonction avec le fichier `test.txt`.
4. Essayez cette fonction avec un autre fichier texte.
5. Essayez cette fonction avec un nom de fichier qui n'existe pas.

3. Test d'existence d'un fichier

Rappelons le fonctionnement de la structure de contrôle `try` dont la syntaxe d'utilisation est:

```
try:
    {instructions}
except:
    {instruction en cas d'erreur}
```

L'ordinateur exécute les instructions. Si une erreur se produit pendant ce traitement, les instructions en cas d'erreur sont alors exécutées, et cela dès que l'erreur se produit.

1. Essayez la séquence d'instructions suivante avec des noms de fichiers existant ou non :

```
try:
    open("un_nom_de_fichier", "r")
    print("le fichier existe")
except:
    print("le fichier n'existe pas")
```

2. Modifiez la fonction `lecture` de manière à afficher "le fichier n'existe pas" si il n'existe pas.

4. Gestion des passages à la ligne

Vous n'êtes pas sans avoir remarqué les caractères `\n` qui apparaissent à la fin de chaque ligne affichée sauf peut-être la dernière, ou des passages à la ligne intempestifs.

Cette séquence `\n` de caractères s'appelle une séquence d'échappement et désigne ici le caractère de contrôle stocké dans les fichiers texte pour noter les passages à la ligne.

Attention ! Ce TP a été préparé sous un environnement Linux. Les passages à la ligne peuvent être notés différement sous Windows. À vous d'adapter le TP. Pour la suite du TP, il est conseillé de s'assurer que les fichiers textes manipulés terminent par un passage à la ligne.

Pour la suite du TP, il est conseillé de s'assurer que les fichiers textes manipulés terminent par un passage à la ligne.

Adapter la fonction lecture pour ne pas afficher ces séquences.

5. Ecriture de fichier

Comme pour la lecture, l'écriture d'un fichier se fait en trois temps :

(a) l'ouverture du fichier

(b) l'écriture proprement dite

(c) la fermeture du fichier

Pour l'ouverture, la valeur du paramètre "mode d'ouverture" est `"w"` (qui provient de "write") au lieu de `"r"` (qui provient de "read"). Le fichier peut ne pas exister. L'écriture se fait via la méthode `write()` en mettant en paramètre la chaîne de caractères à écrire dans le fichier. Il n'y a pas de changement pour la fermeture.

Tests

(a) Testez la séquence d'instructions suivante :

```
f = open("newTest.txt", "w")
f.write("Bonjour")
f.write("Paul\n")
f.write("4 croissants s'il te plait!")
```

Remarque: le fichier `newTest` n'existait pas. Il a été créé (regardez le contenu du fichier). Dans l'explorateur de votre système d'exploitation, visionnez le contenu du fichier. Fermez le fichier, retournez dans votre Jupyter notebook via l'instruction`close` comme précédemment. Retournez dans l'explorateur et rouvrez le fichier.

Conclusion: l'écriture du fichier n'est pas faite au moment de l'usage de la fonction `write()` mais au moment de la fermeture (en fait, s'il y a beaucoup d'écritures, cela peut être réalisé différemment selon les systèmes et selon le nombre d'informations à stocker). Pensez donc à fermer vos fichiers.

(b) Dans l'interpréteur Python, ouvrez le fichier en mode d'écriture. Visionnez son contenu. Conclusion ?

Exercices

1. Écrire une fonction `ecriture` qui écrit dans un fichier ce qui est saisi au clavier jusqu'à ce qu'une ligne vide soit saisie.
2. Testez votre fonction et observez en particulier que toutes les lignes saisies sont mises bout à bout dans le fichier.
3. Améliorer votre fonction `ecriture` de manière à ce que chaque ligne saisie soit sur une ligne dans le fichier écrit.

6. Ajout d'informations dans un fichier

Nous avons vu dans la partie précédente qu'à chaque fois qu'on ouvrait un fichier en écriture, le contenu de ce fichier était écrasé. Pour éviter cela, il existe un autre mode d'écriture, le mode `"a"` pour "append", qui permet d'ajouter des informations dans un fichier.

Tests

Ajoutez quelques lignes dans un fichier (et vérifiez que cela est bien réalisé).

Exercices

Modifié la fonction `ajouter` qui ajoute dans un fichier ce qui est saisi au clavier jusqu'à ce qu'une ligne vide soit saisie. Si le fichier n'existe pas il faudra le créer dans le cas contraire juste ajouter le contenu saisi au fichier existant.

7. Suppresion et modification de fichiers

Il n'existe pas de mode de suppression ou de modification de fichiers (et on ne peut pas ouvrir avec deux modes en même temps un fichier). Au moins deux techniques peuvent pallier ce problème. La première consiste à lire le fichier, stocker son contenu en mémoire vive de l'ordinateur, faire les modifications dans la mémoire et réécrire le contenu. Nous utiliserons cette méthode, mais il faut

noter que cela nécessite un contenu pas trop gros. La deuxième méthode consiste à lire le fichier et à effectuer les modifications au fur et à mesure dans un autre fichier.

8. Compléments: gestion de chemins d'accès

Un fichier n'est pas nécessairement dans le dossier de travail. Il peut s'avérer nécessaire de préciser sa localisation. Ci dessous suivent quelques informations à ce sujet.

Première solution: indiquer le chemin absolu

Une première solution consiste à spécifier le chemin précis du fichier qu'on désire manipuler.

Exemple: L'instruction `open("/Documents/Invite/Document/Python/test.txt", "r")` permettra de lire le fichier text.txt placé dans le dossier Python placé dans le dossier Document placé dans le dossier Invite placé dans le dossier Documents placé à la racine du disque (c'est un chemin absolu !). Notez que même sous Windows la syntaxe d'écriture des chemins se fait selon la syntaxe Unix avec des / et non des `/` entre autres.

Seconde solution

Il est également possible de savoir dans quel dossier travaille l'interpréteur grâce à la séquence d'instructions :

```
import os
print(os.getcwd())
```

Cela peut permettre d'utiliser des chemins relatifs (relatifs à l'endroit où travaille Python). Par exemple, si vous savez que le dossier ou travaille l'interpréteur contient un sous-dossier Exemples contenant un fichier test2.txt, alors il sera possible d'accéder à ce fichier grâce à l'instruction `open("Exemples/test2.txt","r")`.

Mais aussi il est également possible de changer de répertoire de travail de l'interpréteur. Par exemple, `os.chdir("/Documents/Invite/Document/Python/")`.

Pour aller plus loin, il existe de multiples possibilités de parcourir l'arborescence de fichiers (exemple : lister le contenu du dossier). A vous de chercher de l'information dans les livres, ou dans des ressources sur internet (par exemple python.org).

Troisième solution

Mettez dans un fichier (en utilisant une fenêtre spécifique de l'interpréteur) les instructions python à exécuter, et placer le fichier python au même endroit que le fichier à lire.