Math93.com

Cours et TD 1 - NSI
Gestion de fichier

Objectifs

- Manipulation de fichiers sous Python.
  Ouverture, fermeture de fichiers .txt.
- Lecture et écriture via Python dans un fichier .txt.
  Mode lecture (r), écriture (w), ajout (a) ....

I Création de fichiers et de répertoires

Pour faire le travail qui suit, il faut le faire en local donc avec Thonny, Notepad++, Edupython ou Spyder. Il est difficile à faire sous repl.it.

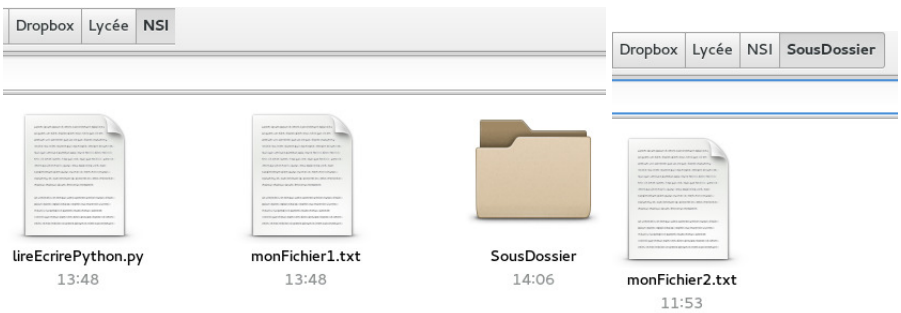
I.1 Création de fichiers dans le répertoire courant et sous-répertoire

Pour comprendre comment lire un fichier en python, nous allons créer un fichier dans le répertoire de travail.

Exercice 1

Toutes les étapes suivantes peuvent se faire sur votre poste.

- Vérifier d'abord que les extensions de fichiers s'affichent. Dans l'explorateur de fichier : Sélectionnez **Afficher** > **Afficher**, puis **extensions de nom de fichier** pour afficher les extensions de nom de fichier.
- Créer un dossier **MonNom-NSi** dans **Document**.
- Créez un fichier python *lireEcrirePython.py* dans le dossier de travail **MonNom-NSi**.
- Ouvrir le bloc-note (Wordpad sous windows et TextEdit sous mac).
- Créer un fichier *monFichier1.txt* dans votre répertoire courant (celui où vous enregistrez votre travail comme **MonNom-NSi**) dans lequel vous écrivez "Hello world!"
- Puis créer un nouveau fichier *monFichier2.txt* que vous enregistrez dans un nouveau dossier nommé **SousDossier**.
  Vous écrivez le texte suivant : "Ceci est le texte du fichier 2 du sous-dossier!" .



I.2 Chemin relatif / chemin absolu

Un chemin relatif en informatique est un chemin qui prend en compte l'emplacement de lecture. Si on exécute le programme créé par *lireEcrirePython.py* à partir du dossier courant,

— le chemin vers *monFichier1.txt* est simplement le nom du fichier *monFichier1.txt*.

— le chemin vers *monFichier2.txt* est *SousDossier/monFichier2.txt*.

Un chemin absolu est un chemin complet qui peut être lu quelque soit l'emplacement de lecture. Si on exécute le programme créer par *lireEcrirePython.py* à partir d'un dossier quelconque, Il faut connaître le nom du répertoire racine (Souvent C : sous windows) et indiquer le chemin vers le fichier. Par exemple :

— le chemin vers *monFichier1.txt* est *C :/courtois/Dropbox/Lycée/NSI/monFichier1.txt*

— le chemin vers *monFichier2.txt* est *C :/courtois/Dropbox/Lycée/NSI/SousDossier/monFichier2.txt*.

Dans la pratique on utilisera très peu le chemin absolu.

II Ouverture et Fermeture de fichier

II.1 Ouverture de fichier

Ouvrir un environnement de travail python comme Thonny et enregistrer un fichier *gestionDefichier.py* dans **MonNom-NSi** .

```
# A mettre dans l'éditeur et executer
fichier1 = open("monFichier1.txt", "r")
print (fichier1)
```

Vous obtenez le résultat :

```
#En python 2.7
<open file 'monFichier1.txt', mode 'r' at 0x7ff6cf3fe4b0>
```

OU

```
#En python 3.
<_io.TextIOWrapper name='monFichier1.txt' mode='r' encoding='cp1252'>
```

Le résultat n'est pas le contenu du fichier mais des informations sur la variable fichier1. C'est ce qu'on appelle un pointeur qui va pointer sur les caractères contenus dans le fichier *monFichier1.txt*.

* le mode d'ouverture qui peut être
  — **r**, pour une ouverture en lecture (READ).
  — **w**, pour une ouverture en écriture (WRITE), à chaque ouverture le contenu du fichier est écrasé. Si le fichier n'existe pas python le crée.
  — **a**, pour une ouverture en mode ajout à la fin du fichier (APPEND). Si le fichier n'existe pas python le crée.
  — **b**, pour une ouverture en mode binaire.
  — **x**, crée un nouveau fichier et l'ouvre pour écriture et génération d'une erreur si le fichier existe.
* Ce qui se trouve aprés "at" est l'adressse en hexadecimal du fichier.

Exercice 2

Ecrire la ligne de commande pour ouvrir le fichier *monFichier2.txt*. Lisez bien la partie sur chemin abolu et chemin relatif.

```
# CODE PYTHON
>>> fichier2 =...
```

II.2 Fermeture de fichier

Comme tout élément ouvert, il faut le refermer une fois les instructions terminées. Pour cela on utilise la méthode close() .

```
# CODE PYTHON
>>> fichier1.close()
>>> fichier2.close()
```

ATTENTION

Fermer toujours un fichier avec la méthode **.close()**, après son utilisation
sinon les données risquent de ne plus être accessibles pour un autre logiciel.

III Lecture et écriture

III.1 Lire le contenu d'un fichier

Un fichier texte est un fichier qui contient des caractères imprimables et des espaces organisés en lignes successives, ces lignes étant séparées les unes des autres par un caractère spécial non-imprimable appelé « marqueur de fin de ligne ».
Les fichiers texte sont donc des fichiers que nous pouvons lire et comprendre à l'aide d'un simple éditeur de texte, par opposition aux fichiers binaires dont le contenu est - au moins en partie - inintelligible pour un lecteur humain, et qui ne prend son sens que lorsqu'il est décodé par un logiciel spécifique. Par exemple, les fichiers contenant des images, des sons, des vidéos, etc. sont presque toujours des fichiers binaires.
Pour afficher tout le contenu d'un fichier, vous pouvez utiliser la méthode *read()* sur l'objet-fichier.

```
# coding: utf-8

fichier1 = open("monFichier1.txt", "r")
print (fichier1.read())
fichier1.close()
```

III.2 Écrire dans un fichier : mode ajout 'a' ou mode écriture 'w' mais le contenu est écrasé

Il est très facile de traiter des fichiers texte avec Python. Par exemple, les instructions suivantes suffisent pour créer un fichier texte de quatre lignes :

III.2.1 Mode écriture w (write) : tout l'ancien contenu est écrasé.

```
f = open("monFichier1.txt", "w") # mode w donc l'ancien contenu est effacé
f.write("Ceci est la ligne un\n Voici la ligne deux\n")
f.write("Voici la ligne trois\n Voici la ligne quatre\n")
f.close()
```

Notez bien le marqueur de fin de ligne \n inséré dans les chaînes de caractères, aux endroits où l'on souhaite séparer les lignes de texte dans l'enregistrement. Sans ce marqueur, les caractères seraient enregistrés les uns à la suite des autres, comme dans les exemples précédents.

III.2.2 Mode ajout a (append) : on ajoute à l'ancien contenu.

Pour ajouter du texte à ce qui précède :

```
f= open("monFichier1.txt", "a") # mode append a donc ajout au contenu
f.write("Bonjour monde")
f.close()
```

Regarder maintenant le contenu de *monFichier1.txt*.

```
f = open("monFichier1.txt", "r") # ouverture en mode lecture
print(f.read())
f.close()
```

Exercice 3

ATTENTION

Fermer toujours un fichier après son utilisation. Voyons ce qu'il peut se passer à cause de cet oubli.

- Copier le code ci-dessous, lancer le puis essayer d'ouvrir **toto.txt** avec le Bloc-Note ou Notepad++. Que constatez vous ?
- Fermer NotePad++, rajouter **f.close()** à la fin du code, lancer le puis réouvrer **toto.txt** avec BlocNote ou NotePad++.

```
f=open("toto.txt","w")
f.write("Que se passe-t-il si on ne ferme pas le fichier ?")
```

IV Quelques autres commandes

IV.1 Le mot clé with

Il existe une autre syntaxe plus courte qui permet de s'émanciper du problème de fermeture du fichier : le mot clé *with* .
Voici la syntaxe (notez l'indentation qui permet de savoir quand fermer le fichier) :

```
with open("monFichier1.txt", "r") as fichier1:
        print (fichier1.read())

# Cela équivaut en fait à l'écriture :

fichier1=open("monFichier1.txt", "r")
print (fichier1.read())
fichier1.close()
```

IV.2 Lecture ligne par ligne

IV.2.1 La méthode readline() : ligne par ligne

Lors des opérations de lecture, les lignes d'un fichier texte peuvent être extraites séparément les unes des autres. La méthode **readline()** , par exemple, ne lit qu'une seule ligne à la fois (en incluant le caractère de fin de ligne) :

```
>>> f = open('monFichier1.txt','r')
>>> t = f.readline()
>>> print(t)
Ceci est la ligne un
>>> print(f.readline())
Voici la ligne deux
```

A chaque appel, on va lire une ligne du fichier. Pour afficher les lignes une par une il faut donc effectuer une boucle et tester si la ligne extraite est vide.

```
f = open('monFichier1.txt','r')
while True: # c'est une boucle sans fin, on utilise un break pour en sortir
    txt = f.readline()
    if txt =='':
        break
    print(txt)
f.close()
```

Exercice 4

Modifier le programme précédent afin de ne pas utiliser d'instruction **break**.

IV.2.2 La méthode readlines() : toutes les lignes restantes. Notez le s de readlineS()

La méthode readlines() transfère toutes les lignes restantes dans une liste de chaînes :

```
>>> t = f.readlines()
>>> print(t)
['Voici la ligne trois\n', 'Voici la ligne quatre\n', 'Bonjour le monde ! ']
>>> f.close()
```

Remarque

- La liste apparaît ci-dessus en format brut, avec des apostrophes pour délimiter les chaînes, et les caractères spéciaux sous leur forme conventionnelle. Vous pourrez bien évidemment parcourir cette liste (à l'aide d'une boucle while, par exemple) pour en extraire les chaînes individuelles.
- **La méthode readlines()** permet donc de lire l'intégralité d'un fichier en une instruction seulement. Cela n'est possible toutefois que si le fichier à lire n'est pas trop gros : puisqu'il est copié intégralement dans une variable, c'est-à-dire dans la mémoire vive de l'ordinateur, il faut que la taille de celle-ci soit suffisante.
- Si vous devez traiter de gros fichiers, utilisez plutôt la **méthode readline()** dans une boucle.
- Notez bien que readline() est une méthode qui renvoie une chaîne de caractères, alors que la méthode readlines() renvoie une liste. À la fin du fichier, readline() renvoie une chaîne vide, tandis que readlines() renvoie une liste vide.

IV.3 Créer un nouveau répertoire

Pour créer un nouveau répertoire dans le répertoire courant :

```
import os
os.mkdir('myDirectory')
```

Remarque

- **mkdir** sert à créer un dossier et surtout pas un fichier.

.

IV.4 Créer un nouveau fichier, Rappels

Remarque

- Pour créer un nouveau fichier il faut utiliser la commande **open** en mode écrite. Par exemple :

.

- **Option "w".** Effacement du contenu de "nouveauFichier.txt" si celui-ci existe sinon le crée.

```
f=open("nouveauFichier.txt","w")
```

- **Option "a".** Ajoute du contenu de "nouveauFichier.txt" si celui-ci existe sinon le crée.

```
f=open("nouveauFichier.txt","a")
```

— **Option "x".** Génération d'une erreur si "nouveauFichier.txt" existe.

```
f=open("nouveauFichier.txt","x")
```

IV.5 Appel multiple de mkdir

Remarque

- Si on rappelle plusieurs fois **mkdir** pour créer un dossier dèjà existant, cela génére une erreur. Pour éviter ce problème on peut utiliser le code suivant :

.

```
if not os.path.exists('moDossier'):#si le chemin vers 'moDossier' n'existe pas
    os.mkdir('moDossier')#Création du dossier 'moDossier'
```

V Exercices

A chaque fois que l'on demande d'ouvrir ou d'écrire, il faut le faire en python.

Exercice 5

Faites l'algorithme suivant en python :

— Ouvrir le fichier *monFichier2.txt*

— Afficher le contenu de *monFichier2.txt*.

— Créer un sous-répertoire *monDossier*.

— Écrire le contenu de *monFichier2.txt* dans un fichier *monFichier3.txt* et le mettre dans le dossier *monDossier*.

Exercice 6

Cet exercice est un simple exercice de manipulation de fichiers pour comprendre la différence entre les modes append(ajouter) 'a', **write**(ecrire) 'w' et **read**(lire) 'r' d'ouverture de fichier ainsi que **readlines**().

- Créez le fichier *exerciceF5.txt* en l'ouvrant en mode écriture, écrivez un texte dedans, puis refermez-le.
- Ouvrez-le en mode lecture, affichez ce que vous lisez dedans, puis refermez-le.
- Ouvrez-le en mode écriture, écrivez "Hello " dedans, puis refermez-le.
- Ouvrez-le en mode lecture, affichez ce que vous lisez dedans, puis refermez-le. Que remarquez-vous ?
- Ouvrez-le en mode ajout, écrivez "World!" dedans, puis refermez-le.
- Ouvrez le en mode lecture, affichez ce que vous lisez dedans, puis refermez-le. Que remarquez-vous ?
- Ouvrez-le en mode ajout, écrivez les lignes suivantes une par une dedans, puis refermez-le.

  La rue assourdissante autour de moi hurlait.

  Longue, mince, en grand deuil, douleur majestueuse,

  Une femme passa, d'une main fastueuse

  Soulevant, balançant le feston et l'ourlet .

  Charles Baudelaire, "A une passante"
- Ouvrez-le en mode ajout, mettez le contenu dans une variable nommée **montexte** le contenu du fichier obtenu avec la méthode **readlines()** (avec un "s") puis refermez-le.
- Afficher la variable **montexte**.
- Utiliser la fonction **len** pour afficher le nombre de ligne.
- Utiliser la méthode **" ".join()** pour compter le nombre de mots.

Exercice 7

Créer en python une fonction *NbrLigne* qui a pour paramètre le nom d'un fichier (texte) et qui renvoie le nombre de lignes de ce fichier. Vous pouvez créer un fichier avec plusieurs lignes pour tester le programme. La question est de savoir quand le fichier est terminé

Exercice 8

1. Avant de commencer l'exercice, veuillez enregistrer le fichier suivant dans le répertoire où vous travaillez :

   Pour télécharger sous windows faites un clic droit puis **enregistrer la cible du lien sous** :

   Demi.txt

   Voici une chaîne de caractères :

   " La mesure de l'homme. \n \n Ce n est pas celui qui critique qui est important, ni celui qui montre du doigt comment l'homme fort trébuche ou comment l'homme d action aurait pu faire mieux. \n L'hommage est dû à celui ou à celle qui se bat dans l arène, dont le visage est couvert de poussière et de sueur, qui va de l'avant vaillamment, qui commet des erreurs et en commettra encore, car il n'y a pas d efforts humains sans erreurs et imperfections. C'est à lui ou à elle qu'appartient l'hommage, à celui ou à celle dont l'enthousiasme et la dévotion sont grands, à celui ou à celle qui se consume pour une cause importante, à celui ou à celle qui, au mieux, connaîtra le triomphe du succès, et au pis, s'il échoue, saura qu'il a échoué alors qu'il risquait courageusement."

   L'exercice consiste à :

2. Créez en python, un autre fichier texte nommé *La_mesure_de_lhomme.txt* et écrire la chaîne de caractères dedans .(Copier la chaîne dans l'éditeur python puis écrire le programme pour l'insérer dans *La_mesure_de_lhomme.txt*)
3. Écrivez le contenu du fichier *Demi.txt* à la suite du fichier *La_mesure_de_lhomme.txt*.
4. Affichez le contenu du fichier *La_mesure_de_lhomme.txt*.

Exercice 9

Nombre d'occurrences

Écrire une fonction **nombre_occurrence(lettre,texte)** qui compte le nombre d'occurrences de la variable *lettre* dans le *texte* qui est une chaîne de caractères.
Exemples :

```
# CODE PYTHON
>>>print(nombre_occurrence("a","aabhkzgygzaa"))
4
>>>f=open("Demi.txt")
>>>montexte=f.read()
>>>print(nombre_occurrence("f",montexte))
2
>>>f.close()
```

```
# CODE PYTHON
def nombre_occurrence(lettre:str,texte:str)->int:
    '''le nombre d'occurrence de la lettre dan le texte'''
```

Exercice 10

La disparition ...

1. Effectuer des tests de votre fonction **nombre_occurrence(lettre,texte)** avec une lettre et des chaînes de caractères importées via vos fichiers.
2. Importer maintenant le fichier texte *ladisparition.txt* qui propose un extrait du roman « La Disparition » écrit en 1968 par l'écrivain français Georges Perec (1936-1982) et publié en 1969.
3. Essayer votre fonction de recherche sur ce texte avec quelques lettres comme 'j' , 'l', 'a', 'i', 'o', 'u', 'y'. Puis avec la lettre 'e', Que remarquez-vous?
4. Écrire une fonction qui renvoie le nombre de lignes et le nombre de caractères de votre texte .

Remarque : : Pour télécharger sous windows faites un clic droit puis **enregistrer la cible du lien sous** :
lien vers le fichier ladisparition.txt