TRAVAUX PRATIQUES n° 11
Manipulation de fichiers

Dans ce TP, on va apprendre à :

- ⋆ lire un fichier enregistré sur l'ordinateur et traiter les données qu'il contient à l'aide d'un programme Python.
- ⋆ écrire nos propres fichiers à l'aide d'un programme Python.

Nous utiliserons uniquement des fichiers de type texte (.txt) pour l'instant car des fichiers de type différents nécessitent d'importer des bibliothèques adaptées.

I. Le répertoire courant

Pour commencer, il faut dire à Python dans quel répertoire il doit lire ou écrire les fichiers. Pour cela, vous devez :

- ⋆ Créer un nouveau répertoire et copier son chemin d'accès (ce chemin est souvent dans la barre de navigation).
- ⋆ Créer un nouveau fichier Python et écrire ces lignes **dans l'éditeur et non dans le shell** en début de fichier :

```
from os import chdir
chdir("chemin d'accès")
```

Remarques :

- `"chemin d'accès"` est à remplacer par le chemin de votre répertoire.
- `chdir` qui signifie « change directory » permet de modifier le répertoire courant.
- `os` qui signifie « operating system » est une bibliothèque de fonctions qui permettent de manipuler les fichiers.

- ⋆ Si vous travaillez sous windows, votre chemin d'accès contiendra des backslashs `\` qui sont des caractères spéciaux pour Python. Pour que Python comprenne qu'il ne doit pas « interpréter » ces caractères, il faut les « échapper » c'est-à-dire qu'il faut remplacer chaque `\` par un double `\\` (et donc remplacer chaque `\\` par `\\\\`).
- ⋆ Valider avec CTRL + E et vérifier que tout se passe bien !

II. Lecture d'un fichier texte

1) Ouverture du fichier et lecture du contenu

Commencez par créer un fichier texte (à l'aide du bloc-notes) dans lequel vous écrirez ce que vous voulez sur plusieurs lignes (au moins une dizaine). Enregistrez votre fichier sous le nom `"mon texte.txt"` dans le dossier que avez précédemment créé.

- ⋆ Pour lire un fichier, il faut d'abord l'ouvrir avec la fonction `open` qui prend en arguments le nom du fichier et le mode d'ouverture (`"r"` pour lecture, `"w"` pour écriture et `"a"` pour ajout) et qui renvoie un fichier que l'on mettra toujours dans une variable : `fichier = open("mon texte.txt", "r", encoding="utf8")`
- ⋆ On peut ensuite récupérer le contenu du fichier :
  - en entier à l'aide de l'instruction `fichier.read()`
  - caractère par caractère à l'aide de l'instruction `fichier.read(1)`
  - ligne par ligne à l'aide de l'instruction `fichier.readline()`
  - toutes les lignes d'un coup à l'aide de l'instruction `fichier.readlines()`
- ⋆ Enfin, il ne faut pas oublier de refermer le fichier car sinon on risque d'avoir des erreurs dans la suite éventuelle de nos programmes si on essaye d'y accéder de nouveau. Fermer le fichier et le réouvrir permet également de remettre le curseur en début de fichier. Pour cela, on utilise l'instruction `fichier.close()`

Exercice 1 : Essayer les instructions suivantes **dans le shell** jusqu'à avoir compris le rôle des fonctions `read`, `readline` et `readlines`. N'hésitez pas à essayer d'autres instructions du même type tant que vous n'avez pas compris leur fonctionnement !

```
fichier = open("mon texte.txt", "r", encoding="utf8")
fichier.read(1)      # À taper plusieurs fois
fichier.read(3)      # À taper plusieurs fois (en changeant la valeur en paramètre)
fichier.readline()   # À taper plusieurs fois
fichier.close()
fichier = open("mon texte.txt", "r", encoding="utf8")
fichier.readline()   # À taper plusieurs fois
lignes = fichier.readlines()
print(lignes[5])
fichier.close()
```

Important : lorsque vous écrivez une fonction qui manipule un fichier, elle devra être construite sur le schéma suivant :

```
def nomFonction(nom_fichier):
    fichier = open(nom_fichier, "r", encoding="utf8")
    Traitement du fichier
    fichier.close()
    return resultat
```

Exercice 2 :

1) Écrire une fonction qui calcule le nombre de caractères dans un fichier texte.
2) Écrire une fonction qui calcule le nombre de lignes d'un fichier texte.
3) Écrire une fonction qui affiche le contenu d'un fichier texte.

2) Traitement du contenu

Pour travailler sur un fichier texte et analyser son contenu, nous aurons besoin de manipuler des chaînes de caractères. Il existe deux fonctions très utiles pour cela :

- ⋆ `rstrip` : qui permet de supprimer d'éventuels caractères de fin de chaîne comme le retour à la ligne `"\n"`.
- ⋆ `lstrip` : qui permet de supprimer d'éventuels caractères en début de chaîne.
- ⋆ `replace` : qui permet de remplacer toutes les occurrences d'une sous-chaîne par une autre.

⋆ `split` : qui permet de couper une chaîne de caractères en plusieurs parties en fonction d'un séparateur.

Attention : ces fonctions ne modifient pas le texte mais renvoient un texte modifié.

Exercice 3 : Essayer les instructions suivantes **dans le shell** jusqu'à avoir compris le rôle des fonctions `rstrip` et `split` :

```
texte = "Bonjour ! Je suis un Python.\n"
texte.rstrip("\n")
texte.rstrip("Python.\n")
texte.lstrip("Bonjour ! ")
texte.replace("o", "a")
texte.replace("Bonjour", "Au revoir")
texte.split(" ")
texte.split("o")
texte
```

Exercice 4 :

1) Écrire une fonction qui calcule le nombre de mots d'un fichier texte.
2) Télécharger le fichier `"Les misérables.txt"` qui contient le premier tome du roman de Victor Hugo. Combien de lignes contient-il ? Combien de mots contient-il (environ) ?
3) Remplacer dans ce texte toutes les occurrences du prénom `"Fantine"` par votre propre prénom et enregistrer le résultat dans un nouveau fichier à l'aide des instructions :

```
fichier = open("Les misérables 2.txt", "w", encoding="utf8")
fichier.write(contenu)
fichier.close()
```

Exercice 5 : Télécharger le fichier `"Admissibles.txt"` et enregistrez-le dans le dossier précédemment créé. Ce fichier contient les résultats de l'écrit du concours des Mines 2013 dans la filière MP. Il est structuré de la façon suivante :

⋆ chaque ligne donne le résultat d'un candidat sous la forme :

`Numéro ; NOM Prénom ; Résultat ; Série d'oral\n`

⋆ `Numéro` est le numéro du candidat (composé uniquement de chiffres)

⋆ `Résultat` est soit `"Admissible"`, soit `"Elimine"`

⋆ `Série d'oral` est un numéro entre 1 et 4 lorsque le candidat est admissible et vaut 0 sinon

1) Combien de candidat ont passé l'écrit du concours ?
2) Pour pouvoir travailler plus facilement sur ce fichier, écrire une fonction `conversion` qui prend en argument une chaine de caractères sous la forme

`"Numéro ; NOM Prénom ; Résultat ; Série d'oral\n"`

et qui renvoie une liste sous la forme

`[Numéro, "NOM Prénom", "Résultat", Série d'oral]`

Par exemple, on doit avoir :

```
>>> conversion("48906 ; AABOUCHE Yacine ; Elimine ; 0\n")
[48906, "AABOUCHE Yacine", "Elimine", 0]
```

Remarque : cette fonction ne doit pas ouvrir le fichier `"Admissibles.txt"`*. Lisez bien la consigne !*

3) Écrire une fonction `listeCandidats` qui ne prend aucun argument et qui renvoie une liste de liste de la forme suivante :

```
[[48906, "AABOUCHE Yacine", "Elimine", 0], [23264, "AAHDI Hajar", "Elimine", 0],
                                                    ...]
```

Remarque : cette fonction se chargera d'ouvrir et de fermer le fichier `"Admissibles.txt"`.

4) Combien y a-t-il d'admissibles ? Combien d'étudiants sont dans la série d'oral numéro 1 ?

5) Écrire une fonction `resultat` qui prend en argument un numéro de candidat et qui affiche une phrase du type

```
Bravo ! Tu es admissible. Tu passeras en série d'oral numéro 1.
```

ou bien

```
Désolé mais tu n'es pas admissible. Retente ta chance l'année prochaine !
```

III. Écriture d'un fichier texte

1) Création et modification d'un fichier texte

⋆ Pour écrire dans un fichier texte, on a le choix entre deux mode d'ouverture du fichier :

- si le fichier n'existe pas ou s'il existe mais qu'on veut l'effacer entièrement, on utilise le mode `"w"` :

```
fichier = open("mon texte.txt", "w", encoding="utf8")
```

- si le fichier existe et qu'on veut conserver son contenu et ajouter du nouveau contenu à la fin, on utilise le mode `"a"` :

```
fichier = open("mon texte.txt", "a", encoding="utf8")
```

⋆ Une fois le fichier ouvert, on dispose des fonctions :

- `write` qui prend en argument une chaîne de caractères et qui écrit son contenu dans le fichier :

```
fichier.write("-- Bonjour. Comment ca va ?\n-- Très bien, et vous ?")
```

- `writelines` qui prend en argument une liste de chaîne de caractères et qui écrit chacune de ces chaînes dans le fichier les unes à la suite des autres (il faut prévoir les retours à la ligne) :

```
fichier.writelines(["-- Bonjour. Comment ca va ?\n", "-- Très bien, et vous
                                                    ?"])
```

⋆ Après avoir écrit ce qu'on voulait dans le fichier, il ne faut pas oublier de fermer le fichier avec l'instruction `fichier.close()`

Pour modifier un fichier de façon plus complexe (pas seulement en ajoutant quelque chose à la fin), on utilisera la procédure suivante :

⋆ ouvrir le fichier en mode lecture `"r"`

⋆ récupèrer son contenu à l'aide de la fonction `read` ou de la fonction `readlines` (en général cette dernière est plus pratique)

⋆ refermer le fichier

⋆ modifier le contenu comme on le souhaite

⋆ ouvrir le fichier en mode écriture `"w"`

⋆ écrire le contenu à l'aide de la fonction `write` ou de la fonction `writelines`

⋆ refermer le fichier

2) Exercices

Exercice 6 : Écrire une fonction `table` qui prend en argument un entier `N` et qui crée un fichier contenant la table de multiplication de `N`.

Par exemple l'instruction `table(2)` doit créer un fichier appelé `"table de 2.txt"` dont le contenu sera :

```
2 x 1 = 1
2 x 2 = 4
2 x 3 = 6
.
.
2 x 10 = 20
```

Exercice 7 : Écrire une fonction `copie` qui prend en argument un nom de fichier `"fichier.txt"` et qui crée un nouveau fichier `"fichier-copie.txt"` ayant le même contenu.

Exercice 8 : Écrire un script qui permet de créer et de relire aisément un fichier texte.

Ce script doit d'abord demander à l'utilisateur d'entrer le nom du fichier. Ensuite, il doit lui proposer le choix soit d'enregistrer de nouvelles lignes de texte, soit d'afficher le contenu du fichier.

L'utilisateur devra pouvoir entre ses lignes de texte successives en utilisant simplement la touche **Entrée** pour les séparer les unes des autres. Pour terminer l'écriture, il lui suffira d'entrer une ligne vide.

Pour cette dernière partie, on utilisera une boucle `while` *et la ligne de commande* `ligne = input("")`.