TP (S1) 6 Lecture et écriture de fichiers

Objectifs
- Savoir écrire un fichier.
- Savoir lire un fichier.

Fichier externe ?

OUI `entier.txt`, `lettre.py`, `notes.py` *(présent(s) dans le répertoire partagé de la classe)*

1. MANIPULATIONS DE FICHIERS

Les structures de données que nous avons vues jusque là (chaînes, listes, tuples, dictionnaires) peuvent être utilisées au sein d'un programme, mais ne permettent pas de stocker durablement des données. Elles n'ont d'existence que dans la mémoire vive (ou RAM) de l'ordinateur, et à la fermeture de `python`, les données contenues dans ces variables sont perdues. Pour stocker ces données de manière durable (c'est-à-dire sur un disque dur, une clé USB ...), il faut utiliser une structure largement utilisée en informatique, dite de **fichier**. Dans ce chapitre nous nous limiterons aux fichiers **textes**. Un traitement sur un fichier se déroule toujours en trois temps :

1. l'**ouverture** du fichier ;
2. le **traitement** proprement dit ;
3. la **fermeture** du fichier.

Il existe trois types d'ouverture de fichiers texte, chacun se faisant par l'intermédiaire de l'instruction suivante :

```
open('NomDeFichier.extension','option')
```

Le choix de l'**option** permet :

- l'ouverture en **lecture** (suppose que le fichier existe déjà), avec l'option `'r'` (pour read) ;
- l'ouverture en **écriture** (pour créer un nouveau fichier ou écraser un fichier existant), avec l'option `'w'` (pour write) ;
- l'ouverture en **ajout** (permet d'ajouter du texte à la fin d'un fichier existant), avec l'option `'a'` (pour append) ;

Dans tous les cas, `open('NomDeFichier.extension','option')` renvoie un objet de type fichier, qu'il convient d'affecter dans une variable du programme.

1.1. Écriture d'un fichier

Commençons par créer un fichier texte en utilisant un script python. Pour que le fichier apparaisse dans **le même dossier** que celui dans lequel vous aurez sauvegardé le script, il faudra exécuter celui en choisissant dans le menu `pyzo` « Exécuter » puis « Démarrer le script » (raccourcis « Ctrl+Maj+E »). Pour exécuter le fichier en pressant simplement la touche « F5 » , il serait nécessaire d'indiquer le **chemin absolu** permettant d'accéder au fichier à partir de la racine (de la forme `C:\dossier \sousDossier \…\script.py`), ce qui est assez fastidieux. En fait, il n'est nécessaire d'exécuter la commande « Ctrl+Maj+E » qu'une seule fois, on peut se contenter de « F5 » par la suite (tant que l'on ne doit pas relancer `pyzo`).

Écrivons dans un fichier python le code suivant, puis exécutons-le par « Ctrl+Maj+E ».

```
Contenu = "Ceci est un exemple de fichier texte."
f = open("essai.txt", "w")
f.write(Contenu)
f.close()
```

La première ligne définit une chaîne de caractères qui formera le contenu du fichier. La deuxième ligne crée et ouvre en écriture un fichier nommé `essai.txt`, référencé par la variable `f`, dans le même répertoire que le script python (si un fichier portant le nom `essai.txt` existe déjà dans ce répertoire, celui-ci sera **détruit** et remplacé par le nouveau : **attention !** aux fausses manipulations qui pourraient vous amener

à détruire certains fichiers importants). La troisième ligne recopie dans le fichier référencé par `f` la chaîne de caractères contenue dans la variable `Contenu`. La dernière ligne referme le fichier référencé par `f`. Ce fichier `essai.txt` apparaît sur le disque dur de l'ordinateur, et en l'ouvrant par l'éditeur on observe sans surprise le contenu suivant (notez que les guillemets encadrant le texte n'apparaissent pas) :

```
Ceci est un exemple de fichier texte.
```

Le contenu du fichier apparaît ici sur une seule ligne. Pour séparer le texte sur plusieurs lignes, on insère le caractère spécial `\n` (bien que formé de deux signes, l'ensemble est considéré comme **un seul caractère** par Python) indiquant à l'ordinateur de revenir à la ligne à cet endroit. Par exemple :

```
Contenu = "Ceci est \n un exemple \nde fichier texte."
f = open("essai.txt","w")
f.write(Contenu)
f.close()
```

produit le nouveau fichier :

```
Ceci est
 un exemple
de fichier texte.
```

Notez que la nouvelle ligne commence **juste après** le caractère « \n » , éventuellement par un caractère d'espace s'il y en a un. Il est possible d'effectuer plusieurs opérations d'écritures successives dans le fichier (avant sa fermeture), chacune venant s'insérer à la suite de la précédente. Par exemple :

```
Contenu1 = "Ceci est \n un exemple \nde fichier texte."
Contenu2 = "Et la suite."
f = open("essai.txt","w")
f.write(Contenu1)
f.write(Contenu2)
f.close()
```

produit :

```
Ceci est
 un exemple
de fichier texte.Et la suite.
```

Notez que le texte rajouté n'a pas été mis automatiquement à la ligne (il aurait pour cela fallu ajouter un caractère `\n`). Il existe une autre syntaxe permettant d'effectuer des écritures successives dans un fichier, consistant à utiliser la commande `writelines`, qui permet l'écriture successive des éléments d'une liste dont les éléments sont des chaînes de caractères (si ces éléments se terminent par le caractère `\n` , ce qui est généralement l'usage, ils formeront alors les lignes successives du fichiers). Par exemple :

```
Contenu = ["Ceci est \n","un exemple \n","de fichier texte."]
f = open("essai.txt","w")
f.writelines(Contenu)
f.close()
```

produit :

```
Ceci est
un exemple
de fichier texte.
```

Enfin, il arrive régulièrement que l'on souhaite stocker des valeurs **numériques** (entiers ou flottants) dans un fichier. Pour cela ces valeurs doivent être converties en les chaînes de caractères correspondantes, ce qui peut se faire par l'intermédiaire de l'instruction `str(valeur)`. Par exemple, le fichier

```
Toto
1.23
Tata
45.6
Tutu
789.0
```

peut s'obtenir par le script :

```
Noms = ["Toto", "Tata", "Tutu"]
Valeurs = [1.23, 45.6, 789.0]
f = open("essai.txt", "w")
for i in range(len(Noms)) :
    f.write(Noms[i] + "\n")
    f.write(str(Valeurs[i]) + "\n")
f.close()
```

Noter comment le caractère `\n` a été ajouté « manuellement » à la fin de chaque ligne. Le même résultat peut être produit en utilisant la commande `writelines` de la manière suivante :

```
Noms = ["Toto", "Tata", "Tutu"]
Valeurs = [1.23, 45.6, 789.0]
L = []
for i in range(len(Noms)) :
    L.append(Noms[i]+"\n")
    L.append(str(Valeurs[i])+"\n")
f = open("essai.txt","w")
f.writelines(L)
f.close()
```

Noter que les lignes #3 à #6 ont pour effet de construire la liste
`L = ["Toto\n", "1.23\n", "Tata\n", "45.6\n", "Tutu\n", "789.0\n"]`
Enfin, l'ouverture du fichier en mode **ajout**, avec l'option `'a'`, fait en sorte que si le fichier existe déjà il ne soit pas écrasé mais que les instructions d'écritures s'ajoutent **à la fin** du fichier. Par exemple si le fichier `essai.txt` contient :

```
Toto
1.23
Tata
45.6
Tutu
789.0
```

alors le script suivant :

```
f = open("essai.txt","a")
f.write("Titi\n")
f.write("13.5\n")
f.close()
```

a pour effet de modifier ce fichier (qui reste de même nom) en :

```
Toto
1.23
Tata
45.6
Tutu
789.0
Titi
13.5
```

1.2. Lecture d'un fichier

On suppose maintenant qu'un fichier existe (pour simplifier placé dans le même répertoire que le script python que nous allons utiliser, sans quoi il faudrait indiquer son chemin absolu) et nous souhaitons extraire son contenu. La démarche est similaire à celle d'écriture, mais l'option est `'r'` pour un accès en **lecture** (une tentative d'écriture dans un fichier ouvert en lecture provoque une erreur à l'exécution), et l'instruction est `read` pour une lecture globale. Par exemple, si le fichier `essai.txt` contient :

```
Ceci est
un exemple
de fichier texte.
```

alors le script suivant :

```
f = open("essai.txt","r")
Contenu = f.read()
f.close()
```

a pour effet de placer dans la variable `Contenu` la chaîne de caractères :
`"Ceci est\nun exemple\nde fichier texte.\n"`
Sur le même fichier, la commande `readlines` permet de récupérer une liste dont chaque élément est une ligne du fichier :

```
f = open("essai.txt","r")
L = f.readlines()
f.close()
```

a pour effet de placer dans la variable `L` la liste de chaînes de caractères :
`"Ceci est\n","un exemple\n","de fichier texte.\n"`

Dans le même esprit, il est possible d'utiliser une boucle **`for`** pour parcourir successivement toutes les lignes du fichier :

```
f = open("essai.txt","r")
for ligne in f :
    instructions
f.close()
```

Dans cet exemple, la variable `ligne` recevra successivement les chaînes de caractères `"Ceci est\n"` puis `"un exemple\n"` et enfin `"de fichier texte.\n"`, sur lesquelles un traitement pourra être opéré. Par exemple, si le fichier `essai.txt` contient :

```
Toto
1.23
Tata
45.6
Tutu
789.0
```

et que l'on veut reconstruire les listes `Noms=["Toto","Tata","Tutu"]` et `Valeurs=[1.23,45.6,789.0]`, on peut procéder ainsi :

```
f = open("essai.txt","r")
Noms = []
Valeurs = []
i = 1
for ligne in f :
    if i%2 == 1:
        Noms.append(ligne[:-1])
    else :
        Valeurs.append(float(ligne))
    i += 1
f.close()
```

Notons plusieurs points. D'abord, le traitement doit être différent suivant les lignes. Celles contenant un nom doivent être ajoutées à la liste `Noms` après avoir enlevé le caractère `\n` final (ce qui peut se faire par l'opération de « slicing » sur les indices `[:-1]`, puisque l'indice −1 désigne de dernier caractère). Par contre celles qui contiennent un nombre doivent être ajoutées à la liste `Valeurs` après avoir converti la chaîne de caractères en nombre, ce qui se fait par l'instruction `float(chaîne)` puisque la chaîne correspond à une valeur flottante ici ; si c'était un entier on utiliserait bien sûr `int(chaîne)` (il n'est pas nécessaire ici d'enlever le caractère `\n` final, qui ne perturbe pas la conversion : par exemple `float("1.23\n")` renvoie bien le flottant 1.23). La distinction entre les deux types de lignes se fait à l'aide d'un compteur dont on teste la parité (si impaire c'est une ligne contenant un nom, si paire c'est une ligne contenant une valeur).

1.3. Manipulations autour des fichiers

Lorsqu'on travaille avec des fichiers, on peut avoir besoin de faire certaines manipulations « systèmes » du type : renommer un fichier, le changer de répertoire, créer un répertoire, …. On peut effectuer ces opérations directement en langage Python en utilisant des commandes regrouper dans le module `os`. En voici par exemple quelques unes (les informations de cette partie sont donnée à titre de complément et ne sont pas à connaître) :

- `mkdir(rep)` : crée un répertoire ;
- `listdir(rep)` : renvoie la liste des fichiers contenus dans le répertoire ;
- `remove(fichier)` : efface un fichier (opération non réversible !) ;
- `rename(source,dest)` : renomme le fichier *source* en *dest*.

2. EXERCICES

2.1. Écriture

Exercice 1 [Sol 1]

Écrire une fonction `fichAlea(n : int, nomFichier : str) -> None` où n est un entier naturel non nul, et qui va créer un fichier (s'il existe déjà il sera écrasé), dont le nom sera la chaîne contenue dans le paramètre `nomFichier`, et contenant n lignes correspondant chacune à un nombre flottant tiré au sort entre 0 et 1.

Par exemple, l'appel `fichAlea(4,"essai.txt")` pourra produire le fichier `essai.txt` suivant :

```
0.17051593186373637
0.0895356653826912
0.6449238466556816
0.40568348964764167
```

L'obtention d'un nombre flottant tiré au sort entre 0 *et* 1 *se fait par la commande* `random()`*, après l'avoir importée du module* `random` *par la commande* `from random import random`.

Exercice 2 **Création d'un fichier** [Sol 2] On désire écrire une fonction `listeClasse(N:int)->None` dont le but est de créer un fichier `liste.txt` contenant les prénoms de N élèves de la classe. On se limitera à un nombre d'élèves faible (`N = 4` par exemple).

1. Recopier, éxécuter et commenter le code suivant. On remarquera, notamment, l'utilisation de `input(message:str)->str` qui permet de stocker dans une variable la saisie au clavier de l'utilisateur après l'affichage d'un message

```
N = 4
liste = []
for i in range(N):
    liste.append(input('Quel est le prénom de l élève \
    ↪ '+str(i+1)+' '))
print(liste)
```

2. Modifier le code précédent pour que le retour à la ligne `\n` soit accolé à chacun des prénoms.
3. Proposer une fonction `listeClasse(N:int)->None` qui crée le fichier demandé et vérifie le résultat à l'aide d'un éditeur de texte.
4. En ouvrant le fichier avec l'attribut `'a'`, ajouter un nouveau nom dans le fichier à la suite des noms déjà enregistrés.

Exercice 3 **Table de multplication** [Sol 3]

1. Créer une fonction `tableMulti(x:int)->None` qui crée un fichier `tablex.txt` (où x est remplacé par la valeur rentrée en argument de la fonction, par exemple `table18.txt`) et dont chaque ligne est de la forme `x*1 = ...`, `x*2 = ...`, en remplaçant x par sa valeur et en indiquant le résultat jusqu'à `x*100`.
2. Exécuter la fonction pour `x = 5`, vérifier le fichier obtenu à l'aide d'un éditeur de texte, puis, avec Python, rouvrir le fichier en mode append `'a'` et ajouter une ligne supplémentaire pour `101*x`.

2.2. Lecture

Exercice 4 [Sol 4] Écrire une fonction `moyFich(nomFichier : str) -> float` où `nomFichier` contient le nom d'un fichier dont chacune des lignes correspond à un nombre, et renvoie la moyenne de ces nombres.

Par exemple, l'appel `moyFich("essai.txt")`, sur le fichier `essai.txt` produit à l'exercice précédent, va renvoyer la moyenne des quatre nombres

0.17051593186373637 ; 0.0895356653826912 0.6449238466556816 ; 0.40568348964764167

qui vaut 0.32766473338743773.

Exercice 5 **Lecture de fichiers** [Sol 5]

1. Le fichier `entier.txt` contient une liste d'entiers. Déterminer la somme de ces entiers. *Réponse : 377958*
2. Le fichier `lettre.txt` contient une lettre de recommandation pour un étudiant. Afficher le contenu de la lettre puis recommencer en n'affichant que les lignes impaires. *Il est possible que vous rencontriez des problèmes d'affichage des accents, dans ce cas, il faut ouvrir le fichier avec l'instruction* `open('lettre.txt', 'r', encoding='utf-8')`

Exercice 6 **Traitement d'un fichier de notes** [Sol 6] Le fichier `notes.txt` contient les notes à 3 devoirs surveillés d'une classe. Chacune des lignes concerne un étudiant différent et est organisée de la façon suivante : `prénom note` au devoir 1 note au devoir 2 note au devoir 3. Par exemple : `Thomas 12.5 17.3 8`.

1. Pour éviter la correction fastidieuse d'un quatrième devoir, l'enseignant choisit d'attribuer à chaque étudiant une note supplémentaire tirée au hasard entre 0 et 20 (et arrondie au dixième de point). On pourra pour cela utiliser `round(np.random.random()*20,1)` après avoir importé le module `numpy`.
   Écrire une fonction `ajoutNote(nomDeFichier:str)->None` qui crée un nouveau fichier `notes2.txt`, construit sur la même base que le fichier notes.txt mais contenant le prénom et les notes aux quatre devoirs.
2. La fonction `tableNotes(nomDeFichier:str)->list` du fichier `TP_fichiers.py` renvoie une liste dont chaque élément est une liste contenant : le prénom de l'étudiant (de type `str`) et ses notes à chaque devoir (de types `float`).
   Pour information, cette fonction utilise la méthode `s.split(c)` qui permet de séparer une chaîne `s` autour d'un caractère particulier `c` (par défaut le caractère d'espacement `''`) :

```
>>> s = "abc 13 15 16"
>>> s.split()
['abc', '13', '15', '16']
```

   Après avoir testé la fonction `tableNotes`, écrire une fonction `calculmoyennes(tableau:list)->list` qui, à partir du tableau de notes généré par la fonction `tableNotes`, renvoie une liste contenant uniquement la moyenne (arrondie au dixième) de chaque étudiant (de la forme `[moyenne1, moyenne2, moyenne3...]`).

3. Écrire une fonction `stats(nomDeFichier:str)->(float,float)` qui renvoie la moyenne générale et l'écart type de la classe et qui trace l'histogramme des moyennes des étudiants. Appliquez cette fonction au fichier `notes2.txt`.
Vous pourrez, pour cela, vous appuyer sur l'exemple suivant :

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.normal(12, 3, 48)  #génère 48 nombres aléatoires \
↪ autour de 12
plt.hist(x, 20)
plt.show()
```

SOLUTIONS DES EXERCICES

Solution 1

```
from random import random

def fichAlea(n : int, nomFichier : str) -> None :
    """
    Crée un fichier de nom nomFichier
    et formé de n lignes contenant
    chacune un flottant tiré au sort
    entre 0 et 1.
    """
    f = open(NomFichier,"w")
    for _ in range(n) :
        f.write(str(random())+"\n")
    f.close()
```

Solution 2

1. Liste des prénoms avec input

```
N = 4
liste = []
for i in range(N):
    liste.append(input('Quel est le prénom de l élève \
    ↪ '+str(i+1)+' '))
print(liste)
```

2. Il suffit de concaténer \n avec le résultat du `input`

```
N = 4
liste = []
for i in range(N):
    liste.append(input('Quel est le prénom de l élève \
    ↪ '+str(i+1)+' ')+'\n')
print(liste)
```

3. Définition de la fonction

```
def listeClasse(N:int)->None:
    f = open('liste.txt','w')
    liste = []
    for i in range(N):
        liste.append(input('Quel est le prénom de l élève \
        ↪ '+str(i+1)+' ')+'\n')
    f.writelines(liste)
    f.close()
```

4. ouverture du fichier en mode append

```
f = open('liste.txt','a')
f.write('Thomas\n')
f.close()
```

Solution 3

1.
```
def tableMulti(x:int)->None:
    nomFichier = 'table'+str(x)+'.txt'
    f = open(nomFichier,'w')
    for i in range(1,101):
        ligne = str(x)+'x'+str(i)+'='+str(x*i)+'\n'
        f.write(ligne)
    f.close()
    return()
```

2.
```
f = open('table12.txt','a')
f.write('12*101='+str(101*12)+'\n')
f.close
```

Solution 4

```
def moyFich(nomFichier : str) -> None :
    """
    Renvoie la moyenne des nombres figurant
    sur chacune des lignes du fichier de nom
    nomFichier
    """
    f = open(nomFichier,"r")
    n, S = 0, 0
    for ligne in f :
        S += float(ligne)
        n += 1
    f.close()
    return S/n
```

Solution 5

1. Sommes des entiers du fichiers

```
f = open("entiers.txt",'r')
listeNbre = f.readlines()
f.close()
S = 0
for nbre in listeNbre:
   S += int(nbre[:-1])
print(S)
```

2. Lecture de la totalité de la lettre puis d'une ligne sur deux

```
f = open("lettre.txt",'r')
lettreComplete = f.read()
f.close()
print(lettreComplete)
```

```
f = open("lettre.txt",'r')
lignes = f.readlines()
f.close()
lettrepartielle = ''
for i in range(0,len(lignes),2):
    lettrepartielle += lignes[i][:-1]
print(lettrepartielle)
```

Solution 6

1.
```
def ajoutNote(nomDeFichier:str)->None:
    f = open(nomDeFichier,'r')
    f2 = open("notes2.txt",'w')
    for ligne in f:
        nouvellenote = round(np.random.random()*20,1)
        f2.write(ligne[:-1] + ' '+ str(nouvellenote) +'\n')
    f2.close()
    return()
```

2.
```
 def MoyenneElts(L:list)->float:
    #Fonction intérmédiaire calculant la moyenne des éléments \
    ↪ d'une liste
    S = 0
    for element in L:
        S += element
    moyenne = S/len(L)
    return moyenne

def calcMoyenne(tableauNotes:list)->list:
    tmoyenne = []
    for elt in tableauNotes:
        tmoyenne.append(round(MoyenneElts(elt[1:]), 1))
    return(tmoyenne)
```

3.
```
def ecartType(listenote:list)->float:
    moyenne = MoyenneElts(listenote)
    E = 0
    for elt in listenote:
        E += (elt-moyenne)**2
    return(round((E/len(listenote))**(1/2),1))

def stats(nomDeFichier:str)->list:
    tableau = tableaunotes(nomDeFichier)
    moyennes = calcMoyenne(tableau)
    print(moyennes)
    Moyenne = MoyenneElts(moyennes)
    Ect = ecartType(moyennes)
    plt.hist(moyennes,20)
    plt.show()
    return([Moyenne, Ect])
```