TP 2| Les Fichiers : lecture et écriture

1. Entrées/Sorties fichiers

1.1. Rappels

On s'intéresse dans cet exercice aux Entrées/Sorties fichier, autrement dit à la lecture ou l'écriture d'un fichier. Ces entrées/Sorties Fichier concernent 2 types de fichiers :

- Fichiers ACSII (fichier texte), où chaque représente un caractère (code ASCII).
  - ACSII délimités : données séparées par un délimiteur : **espace** **\t** **;** **,**
  - ASCII tabulé : données séparées par tabulation (ASCII \t) (export tableur)
  - ASCII CSV : Comma Separated Value (export tableur)
  - ASCII libre : format spécifique...
- Fichier binaire, contenant des données au format binaire :
  - Nombres : entiers, flottants 32 ou 64 bites (codage IEEE754),
  - Objets applicatifs : mélange de nombre et de données binaires spécifiques

L'ouverture d'un fichier se fait grâce à la fonction : **open(nom, mode).**
nom : Chaîne de caractère, nom du fichier
mode : Chaîne de caractère, mode accès au fichier (**'r'** : read, **'w'** : write, **'a'** : append)

La fonction open retourne un **objet** de type **file**.
Les principales méthodes de la classe file sont les suivantes :

.read()	retourne tout le fichier comme un **str**
.read(n)	retourne un **str** d'au moins n octets
.readline()	retourne une ligne
.readlines()	retourne la liste de toutes les lignes du fichier
.write(s)	écrit s (objet **str**) dans le fichier
.writeslines(lines)	écrit **lines** (liste de **str**) dans le fichier
.close()	ferme le fichier
.next()	retourne la ligne suivante

1.2. Lecture d'un fichier ASCII délimité

Copier-coller le fichier « data2.txt » dans un dossier à votre nom, libre d'accès en écriture.

1) Ouvrir avec l'éditeur de Spyder ce fichier et observer la structure de ce fichier. On souhaite ensuite ouvrir le fichier en tant qu'objet de type « file » en mode écriture sous python. Faire quelques manipulations.

2) Dans un fichier script, commencer un programme dans lequel on va ouvrir le fichier en lecture dans un objet file nommé « myFile ». Créer ensuite une première liste « data » dans laquelle chaque

élément stocké sera une ligne du fichier ouvert (sans les lignes « commentaires »). Observer la forme de la liste « data ».

Afin de séparer les termes de chaque élément de la liste précédemment obtenue, nous allons utiliser la méthode **.split** associée aux chaînes de caractères. Afin de tester cette fonction, saisir les lignes de codes suivantes :

```
>>> car="Première année PTSI\nDeuxième année PT"
>>> car.split()
```

3) Construire deux listes « t » et « temperature » dans lesquelles seront stockées les valeurs numériques du temps et de la température du fichier « data2.txt ». Les afficher.

Dans la programme précédent, la connaissance de la structure du fichier avec la mise en commentaires de certaines lignes (#) nous permettait de savoir quelles données devaient être extraites. Que se passe-t-il si la structure du fichier est un peu plus complexe ? Nous allons alors tester si la méthode de construction de la liste effectuée précédemment est possible ligne par ligne. Cela se fera grâce à deux nouvelles instructions de test : **try** et **except**. Saisir les lignes de codes suivantes :

```
try:
    float("2.1")
except:
    print("erreur")
```

```
try:
    float("2.1a")
except:
    print("erreur")
```

4) A l'aide de ces deux fonctions proposer une amélioration du programme précédent, en testant chaque ligne du fichier « myFile » et afficher « t » et « temperature ».

1.3. Ecriture dans un fichier ASCII

5) Avec la fonction **open**, créer un nouveau fichier (« nouveau.txt ») dans le répertoire courant de travail. On nommera l'objet file correspondant **fo**.

Faire le test suivant :

```
fo.write("Ecriture dans un fichier\n");
for i in range(1,5):
    data="%.3E\t%.3E\n" %(2*i*1e-5, i*i)
    fo.write(data)
fo.close()
```

On souhaite enregistrer dans un nouveau fichier « corresp.txt » le tableau de correspondance entre écriture décimale, binaire et hexadécimale (voir ci-après).

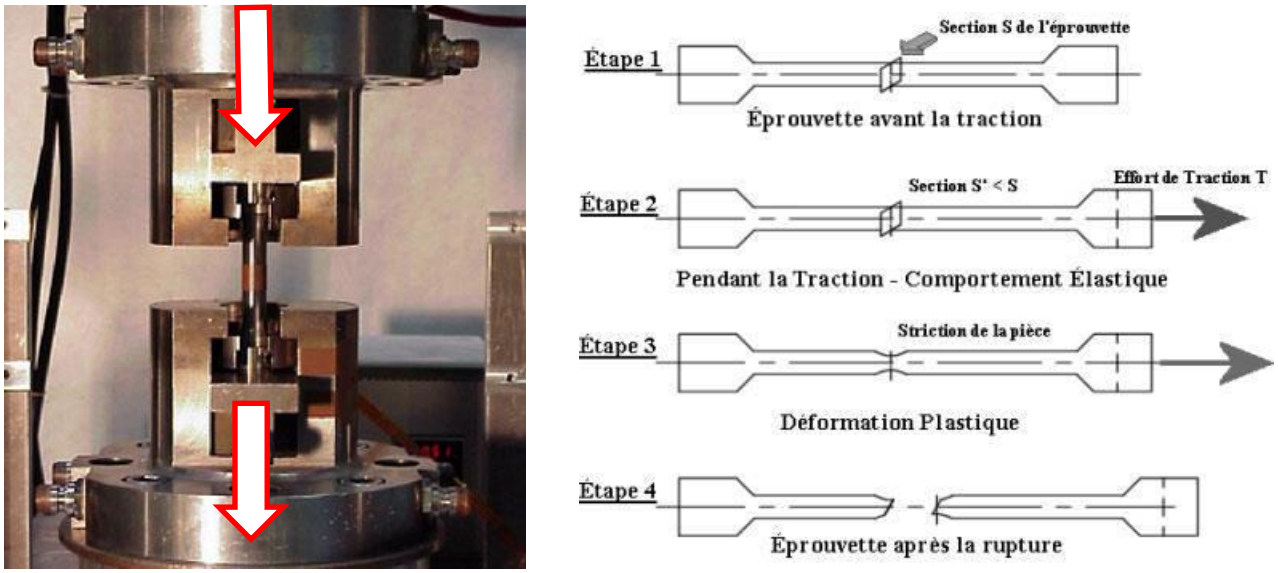
Decimal	Binaire	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
…	…	…
14	1110	E
15	1111	F

Dans le fichier « corresp.txt » les éléments entre les colonnes du tableau ci-dessus seront séparés par des tabulations.

6) Construire ce fichier texte « corresp.txt ». (On écrira au préalable une fonction n2b(n) qui permet de convertir un décimal en binaire sous forme d'une chaîne de caractère, puis on créera les listes Dec, Bin et Hex via une boucle).

2. Essai de traction

On se propose dans cette activité d'étudier les résultats issus d'un enregistrement effectué lors d'un essai très communément utilisé en science des matériaux : l'essai de traction. Cet essai consiste à fixer entre deux mors dans une machine dédiée une éprouvette, généralement de géométrie simple (cylindrique ou parallélépipédique), puis à tirer sur cette dernière jusqu'à sa rupture. Les efforts de traction, ainsi que l'allongement de l'éprouvette sont mesurés pendant l'essai. La forme de la courbe et les efforts mis en jeu pendant l'essai permettent alors de déterminer des caractéristiques mécaniques du matériau, tels que la limite élastique, la résistance mécanique à la rupture par traction, le module d'Young, etc…



Déroulement d'un essai de traction

Pour la suite, on travaillera depuis le fichier « data1.txt » que vous copierez et collerez sur un espace disque libre d'accès en écriture.

1) Observer en l'ouvrant la structure du contenu de ce fichier.

2) Mettre en œuvre dans un nouveau fichier script (« essai_traction.py ») un programme permettant d'extraire les données et de représenter graphiquement :

- l'évolution de la force en fonction du temps, ainsi que l'évolution du déplacement en fonction du temps ;
- l'évolution de l'effort en fonction du déplacement.

On peut sauvegarder les courbes au format .png à l'aide de la fonction **plt.savefig('nom_fichier.png')**.