P2 – LECTURE DE FICHIER, TRACE DE COURBE, DERIVATION

Préambule :

1. Ouvrir une session sur l'ordinateur en entrant Identifiant / Mot de passe ;
2. Lancer Spyder en cherchant Spyder dans la barre de recherche ;

Objectifs de ce TP :

- *Manipuler des fichiers ;*
- *Rechercher des valeurs dans un fichier ;*
- *Manipuler les données dans un fichier ;*
- *Exploiter des résultats d'une mesure dans le cadre d'un TP de Si à l'oral.*

Exercice 1 : recherche de mot dans un dictionnaire

Le fichier `dictionnaire.txt` est un fichier contenant 336 531 mots de la langue française. Il est disponible dans le dossier `Espaces d'échanges` de votre classe de TSI1.

1. Copier le fichier `dictionnaire.txt` dans votre dossier personnel.
2. Ouvrir le fichier `dictionnaire.txt` et observez le contenu.
3. Ouverture d'un fichier.

Pour ouvrir un fichier, nous utiliserons la fonction **`open()`** :

```
>>> dico=open('dictionnaire.txt','r')
```
avec 'r' : en lecture seulement

Attention : le fichier doit se trouver dans le même répertoire que l'exécutable python, sinon il faudra spécifier le chemin d'accès. Vous pouvez pour cela naviguer dans l'explorateur Windows jusqu'au dossier où se trouve le fichier dictionnaire.txt puis cliquer dans la barre horizontale indiquant le chemin et copier le chemin avec un clic droit.

Exemple de chemin : I:\Public\IPT_CPGE\TP5\dictionnaire.txt

```
>>>dico=open(' I:\Public\IPT_CPGE\TP5\dictionnaire.txt ','r')
```

Remarque : la variable `'dico'` désigne ici le fichier ouvert par python, mais le fichier n'est pas copié dans `'dico'`, `'dico'` ne contient pas le fichier ouvert, il permet seulement d'y accéder.

4. Lecture d'une ligne entière du fichier.

On peut lire le fichier **ligne après ligne** avec la fonction **`readline().`** On lit une seule ligne à la fois ou alors un morceau de ligne.

Essayez par exemple :

```
>>>ligne1= dico.readline()
```
on stocke les données de la ligne dans ligne 1 et on passe à la ligne suivante
```
>>>print(ligne1)

>>>ligne2= dico.readline()
```
on stocke les données de la ligne suivante dans ligne 2 et on passe à la ligne suivante
```
>>>print(ligne2)
>>>ligne2= list(ligne2)
```
on convertit la ligne 2 en liste
```
>>>print(ligne2)
```
on remarque que le dernier caractère de la ligne est toujours `/n`

5. Lecture d'un fragment de ligne.

On peut lire un fragment de ligne, il suffit d'ajouter le nombre de caractère à lire comme argument à la fonction readline().

Essayez par exemple :

```
>>>fragment= dico.readline(2)
```
on stocke les données de 2 caractères de la ligne dans fragment
```
>>>print(fragment)

>>>fragment2= dico.readline(1)
```
on stocke les données du caractère suivant dans fragment2

```
>>>print(fragment2)

>>>fin= dico.readline()        on stocke les données de la fin de la ligne dans fin et on passe à la ligne suivante
>>>print(fin)
```

6. Lecture du fichier entier.

La commande **`readlines()`** écrit toutes les ligne du fichier dans une liste :

```
>>>liste_mots=dico.readline()
```

On peut alors accéder à la ligne i=25 :

```
>>>ligne25=liste_mots[25]        on stocke la ligne 25 dans ligne25
>>>print(ligne25)

>>>ligne20=liste_mots[20]        on stocke la ligne 20 dans ligne20
>>>print(ligne20)

>>>caractere=liste_mots[20][3]  on stocke le 4ème caractère de la ligne 20 dans caractère
>>>print(caractere)
```

Cela signifie que le fichier ouvert est copié intégralement dans le tableau `liste_mots`, il faut donc éviter cette fonction avec les fichiers de taille très importante.

Attention : une fois que l'on a lu le fichier, il vaut mieux le fermer pour toujours repartir de la première ligne :

```
>>>dico.close()
```

7. Utilisation des données.

Les données issues du fichier sont des chaînes de caractères. On peut accéder aux éléments de ces chaines de caractères de la même façon que les listes.

```
>>>dico=open('dictionnaire.txt','r')        on ouvre le fichier
>>>dico.readline()                          on saute une ligne
>>>dico.readline()                          on saute une ligne

>>>ligne3=dico.readline()                   on stocke la ligne 3 dans ligne3
>>>print(ligne[0 :3 :1])                    on affiche les 3 premiers caractères
>>>print(ligne[:-1])                        on enlève le dernier caractère
```

Travail à réaliser :

8. Ecrire une fonction `motligne(i)`, qui renvoie le mot de la i-ème ligne du dictionnaire. Tester votre fonction !

9. Ecrire une fonction `affiche(i)`, qui affiche tous les mots commençant par la lettre i. Tester votre fonction !

10. Écrire une fonction `cherche(mot)` qui cherche le mot demandé. Le dictionnaire ne contenant pas de définition, le programme se contentera de renvoyer la position du mot dans le dictionnaire (n^{ième} mot). Tester votre fonction !

Exercice 2 : exploitation des résultats d'une mesure

Dans cette partie, nous allons nous appuyer sur un support de TP qui est utilisé lors aux oraux de concours de SI : le système CoMax présenté sur la figure 1. Ce système est une adaptation pédagogique de la solution industrielle ZE de SAPELEM. Le principe de fonctionnement de ces deux systèmes repose sur l'utilisation d'un système de levage motorisé, associé à une poignée, équipée d'un capteur d'effort (voir figure 1).

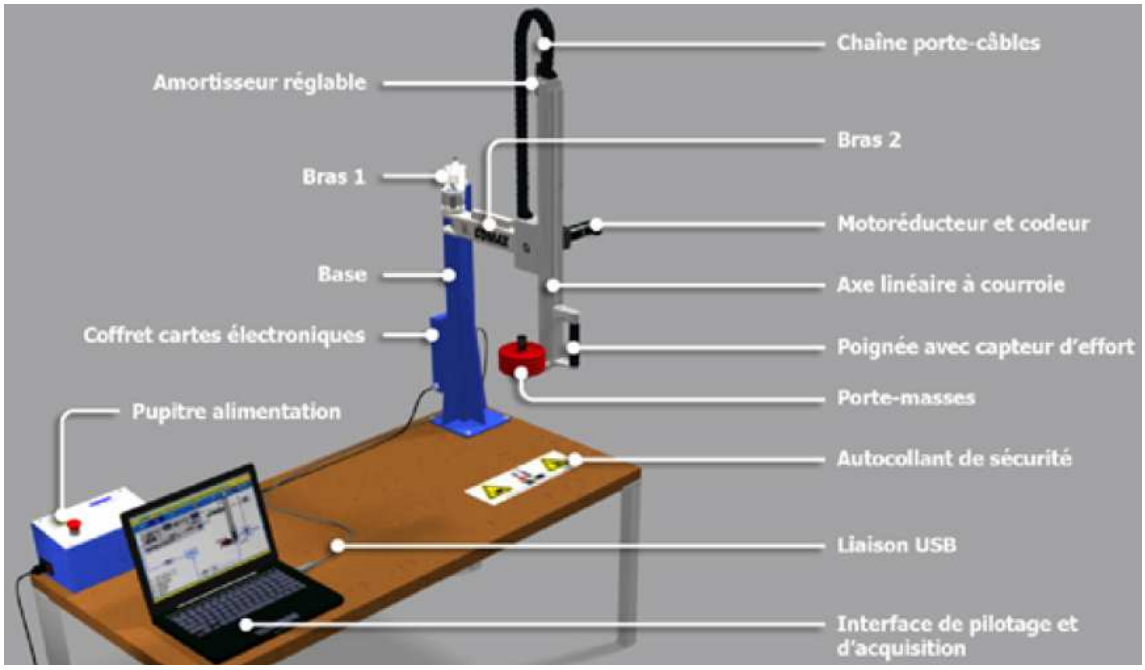

Figure 1 : système pédagogique comax

Nous avons réalisée une mesure lors du déplacement vertical de la poignée. La mesure issue du capteur de position qui équipe le système a été stockée dans le fichier `comax.txt` disponible sur l'Espace d'échanges de votre classe.

1. Copier le ficher `comax.txt` dans votre dossier personnel.
2. Ouvrir le ficher `comax.txt` et observez les données présentes à l'intérieur.

La troisième colonne du fichier indique la position z (mm) de l'axe linéaire.

Travail à réaliser :

3. Ecrire une fonction `positionz(i)`, qui renvoie la position $z(mm)$ et le temps $t\ (s)$ pour le i-ème point de mesure. Tester votre fonction !

4. Ecrire une fonction `temps()`, qui renvoie une liste contenant toutes les valeurs de temps disponibles dans le fichier. Tester votre fonction !

5. Écrire une fonction `positions()`, qui renvoie une liste contenant toutes les valeurs de positions disponibles dans le fichier. Tester votre fonction !

La vitesse de déplacement à une position de la ligne i dans le fichier peut-être calculée à partir d'un taux d'accroissement :

$$v(i) = \frac{z_i - z_{i-1}}{t_i - t_{i-1}} \text{ en mm/s}$$

Travail à réaliser :

6. Ecrire une fonction `vitesse()`, qui renvoie une liste contenant les vitesses de déplacement pour toutes les mesures du fichier (sauf la première). Tester votre fonction.

7. Conclure en trouvant la vitesse maximale atteinte par le comax lors du déplacement.