TP2 LECTURE ET ECRITURE DANS UN FICHIER

1 LE FICHIER TEXTE

1.1 Intérêt

L'intérêt d'un fichier texte est simple : il permet de stocker des données, qui seront conservées une fois que le programme python ait été déroulé. On peut imaginer par exemple un dispositif quelconque de mesure, qui enregistre les données recueillies dans un fichier texte pour une analyse ultérieure.

Le fichier texte n'est pas le seul moyen de stocker des données, bien au contraire, avec par exemple les tableurs et les bases de données, mais il présente l'intérêt d'être extrêmement simple à générer et à lire.

1.2 Chemin relatif ou absolu

Pour pouvoir lire un fichier texte ou le remplir, il est important de spécifier dans le script l'emplacement du fichier dans la mémoire, en lui fournissant un chemin d'accès. Ce chemin peut être :

— **absolu** : il est alors écrit dans sa totalité, en partant de la racine du disque dans lequel il est stocké. Par exemple : *"D :\MP1 2023\Cours favori\Info\Cours\Lecture et écriture\Test.txt"*.

Attention à ne pas oublier l'extension .txt du fichier texte !

— **relatif** : Si le fichier texte est stocké dans le même répertoire que le fichier python, il suffira d'indiquer seulement le nom du fichier avec son extension .txt pour qu'il soit reconnu et pour qu'il soit utilisé par le programme.

2 LECTURE ET ECRITURE

2.1 Ouvrir un fichier texte

Manipuler un fichier texte en python est simple :

— On commence par l'ouvrir. Si le fichier n'existe pas, il est alors créé. Le code est le suivant :

```
fichier=open("chemin_du_fichier", mode)
```

Le mode est à choisir parmi trois caractères : "w" (write), "r" (read), "a" (append). Nous détaillerons ces modes par la suite.

— On peut ensuite lire ou écrire dans le fichier.

— Enfin, il faut fermer le fichier :

```
fichier.close()
```

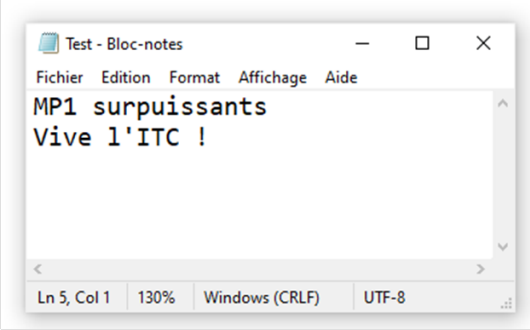
2.2 Écrire dans un fichier

Pour écrire dans un fichier, nous avons le choix entre les modes suivants :

— "w" (write) : dans ce mode, les données du fichier sont écrasées, ce qui signifie qu'elles sont effacées pour faire place à de nouvelles données. La syntaxe est la suivante :

```
fichier=open("Test.txt","w")
fichier.write("MP1 surpuissants \nVive l'ITC !!! ")
fichier.close()
```

On obtient alors, en ouvrant le fichier :



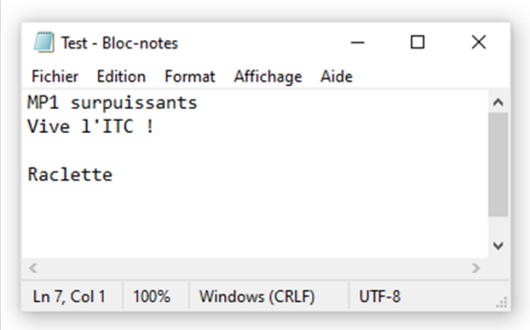
On remarque que le caractère " \n " est utilisé pour effectuer un passage à la ligne. Chaque passage à la ligne dans le fichier correspond donc également à un caractère codé sur un octet.

— "a" (append) : dans ce mode, les données déjà présentes dans le fichier sont conservées, et les nouvelles données sont écrites directement à la suite des précédentes. La syntaxe est la même qu'en mode "w".

Par exemple si l'on fait tourner le code suivant après le précédent :

```
fichier=open("Test.txt","a")
fichier.write("\n\nRaclette ")
fichier.close()
```

On obtient le fichier suivant :



2.3 Lire le contenu d'un fichier

La lecture d'un fichier, qui utilise le mode " r ", peut se faire de deux façons :

— **lecture de la totalité du fichier :**

```
fichier=open("Test.txt","r")
C=fichier.read()
fichier.close()
```

La fonction "read" renvoie une chaîne de caractère, contenant l'ensemble des caractères du fichier, y compris les passages à la ligne, qu'il faut alors stocker dans une variable pour pouvoir l'analyser :

```
>>> C
"MP1 surpuissants \nVive l'ITC !\n\nRaclette"
```

Ce n'est pas très pratique et on préfèrera généralement lire le fichier de la manière suivante :

— **lecture ligne par ligne :**

```
L=[ ]
fichier=open("Test.txt","r")
for k in fichier :
    L.append(k)
fichier.close()
```

Ce mode est très pratique lorsque les informations du fichier sont structurées par ligne. On obtient en effet une liste comportant un élément par ligne du fichier texte parcouru.

```
>>> L
['MP1 surpuissants \n', "Vive l'ITC !\n", '\n', 'Raclette']
```

On utilera la fonction "split" pour analyser les fichiers avec ce mode d'ouverture. Cette fonction permet de couper une chaîne de caractère suivant un caractère que nous lui indiquerons. Voici comment l'utiliser :

```
>>> donnees = "M.P.1"
>>> donnees
'M.P.1'
>>> donnees_coupees=donnees.split(".")
>>> donnees_coupees
['M', 'P', '1']
```

3 Exercice 1 : Météo

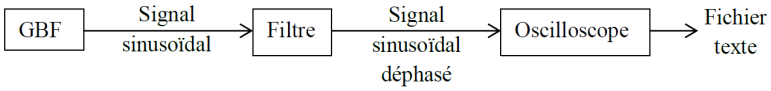
Un météorologue amateur enregistre la température à l'aide d'un capteur thermique relié à un raspberry pi. II enregistre la température maximale et minimale de chaque jour dans un fichier texte dont la structure est la suivante :

— Sur la première ligne est écrit « Date, Haut, Bas »
— Sur chaque autre ligne on peut lire, séparés par des virgules : la date sous le format AAAA-MM-JJ, la température maximale, la température minimale

1. Écrire la fonction **lecture**, prenant en argument un fichier tel que celui-ci, et renvoyant une liste de listes de trois éléments :
   — le premier élément est la liste [JJ,MM,AAAA], dont chaque élément doit être un entier,
   — le second élément est un décimal représentant la température minimale du jour,
   — le troisième élément est un décimal représentant la température maximale du jour
2. Représenter graphiquement l'évolution des températures en fonction des jours qui s'écoulent, en lisant le fichier meteo.txt présent sur cahier de prépa.

4 Exercice 2 : Diagramme de Bode

Au labo de physique on dispose d'un filtre électronique dont les caractéristiques sont inconnues. Afin de les déterminer on a réalisé une étude fréquentielle de ce filtre à l'aide d'un oscilloscope pour visualiser les signaux et d'un GBF pour générer un signal d'entrée appliqué au filtre suivant le schéma ci-dessous :



Une série de 60 mesures ont été réalisées. Elles sont disponibles sous forme de fichiers textes, 1 fichier par mesure. Afin de s'éviter la tâche pénible de traiter toutes ces mesures à la main, on souhaite écrire un programme Python dont l'objectif et de tracer le diagramme de Bode du filtre et d'identifier ses caractéristiques.

4.1 Lecture et visualisation d'un mesure

L'objectif de cette partie est de lire un fichier de mesure et tracer les courbes correspondantes.

1. Ouvrir un fichier de mesure avec un lecteur de fichier texte tel que le Bloc-notes de Windows. Analyser le contenu de ce fichier.
2. Ecrire une fonction **lecture** qui prend en argument une chaîne de caractères **nomFichier** qui contient le nom du fichier à lire et son chemin d'accès si besoin. Cette fonction doit renvoyer le contenu du fichier sous la forme de 3 listes de flottants, **Lest**, **LesE** et **LesS**, qui contiennent respectivement les instants t, les valeurs du signal d'entrée et les valeurs du signal de sortie.
3. Ecrire un script qui permet de tracer sur un même graphe les deux courbes de l'entrée et de la sortie en fonction du temps. Ce graphique devra comporter un titre, et ses axes devront être légendés correctement. Vous pouvez alors visualiser le contenu de plusieurs fichiers afin de vous faire une idée des différentes courbes.

4.2 Analyse d'un mesure

L'objectif de cette partie est d'analyser numériquement une mesure afin d'en déduire la pulsation ω, le gain réel et le déphasage correspondant à la mesure.

4. A partir de l'observation des courbes d'une mesure, expliquer comment déterminer la pulsation ω et proposer un algorithme permettant de le faire automatiquement. Afin d'obtenir une bonne précision de l'algorithme il sera judicieux de repérer les passages par zéro du signal d'entrée ou de sortie.

5. Ecrire une fonction **pulsation** qui prend en argument la liste **Lest** et la liste **LesE** ou **LesS** et qui renvoie la pulsation **omega** pour laquelle la mesure a été réalisée.
6. Ecrire une fonction **maximum** qui prend en argument une liste **L** et qui renvoie le maximum **M** de cette liste.
7. Expliquer comment, à partir des valeurs maximales des signaux d'entrée et de sortie du filtre, on peut déterminer le gain réel du filtre pour la pulsation ω de la mesure étudiée.
8. Ecrire une fonction **gain** qui prend en argument les deux listes **LesE** et **LesS** et qui renvoie, à l'aide de la fonction **maximum**, le gain réel **G** entre le signal d'entrée et le signal de sortie.
9. A l'aide du tracé graphique de la question 3 et en vous inspirant de la méthode utilisée pour déterminer ω, proposer une méthode pour déterminer le **déphasage** φ entre les signaux d'entrée et de sortie.
10. Ecrire une fonction **dephasage** qui prend en argument les listes **Lest**, **LesE** et **LesS** et qui renvoie la valeur phi en degré du déphasage (Attention au signe).
11. A partir des fonctions précédentes, écrire un script qui permet de lire tous les fichiers texte de chaque mesure et qui génère les 3 listes suivantes :
    - **LesW** : la liste des pulsations ω correspondant à chaque mesure rangées dans le même ordre que la numérotation des fichiers de mesure,
    - **LesG** : la liste des gains réels correspondant à chaque mesure rangés dans le même ordre que la numérotation des fichiers de mesure,
    - **LesPhi** : la liste des déphasages en degrés correspondant à chaque mesure rangés dans le même ordre que la numérotation des fichiers de mesure.
12. Ecrire un script qui permet de tracer le diagramme de Bode du filtre. Le graphique obtenu devra être correctement légendé.
13. Compléter votre script pour qu'il renvoie les paramètres caractéristiques du filtre.

5 Annexe représentation graphique

Ci-dessous sont listées des commandes qui peuvent être utiles pour configurer et personnaliser vos graphiques :

- plot $(X, Y, 'b - o')$, permet de tracer une courbe avec :
    - X la liste des abscisses,
    - Y est la liste des ordonnées, elle doit avoir la même longueur que X,
    - b désigne la couleur de la courbe (rouge = r, bleu=b, vert=g, noir = k, jaune = y, cyan = c et magenta = m)
    - - désigne le type de trait (-=trait plein, –=pointillés, -.=alterné...)
    - o désigne la marque (0 = point, + = plus, * = =étoile...)

— **show()** permet d'afficher la courbe
— **title("titre")** permet de donner un titre au graphique
— **xlabel("titre")** permet de donner un titre à l'axe x, de même pour y
— **xlim(xmin,xmax)** permet de fixer les bornes de la fenêtre graphique selon x, de même poury.
— **axis("equal")** permet d'avoir une fenêtre graphique orthonormée
— **yscale("log")** permet d'avoir une échelle logarithmique