I Objectifs

- Lire des données contenues dans un fichier-texte et les enregistrer dans une variable.
- Enregistrer dans un fichier-texte des données calculées à l'aide d'un programme.
- Modifier un fichier-texte.

II L'arborescence des données informatiques

- Un **lecteur** (*drive*) désigne un support physique ou virtuel où les données sont enregistrées. On parle d'un lecteur comme la **racine** (*root*) d'un chemin. Exemples : `C:\` , `D:\` ...
- Un **répertoire** (*directory*), ou *dossier*, désigne un conteneur virtuel à l'intérieur duquel se trouvent des sous-répertoires ou des fichiers.
- Les **fichiers** (*files*) sont enregistrés dans des (sous-)répertoires. Ils possèdent ou non une extension.
- Une **extension** est un complément au nom d'un fichier, qui peut renseigner l'utilisateur sur le type de fichier, ou plus généralement qui sert à pointer vers un programme de lecture adapté lorsque l'utilisateur actionne ce fichier (en double-cliquant par exemple).
- Le **chemin** (*path*) est la chaîne de caractères donnant l'adresse (complète ou partielle) d'un fichier ou d'un répertoire.

Exemple de chemin complet :

`C:\Mes_documents\Travail\Python\script_2024 _02_05.py`

III Répertoire de travail

Le répertoire de travail est celui dans lequel les fichiers manipulés seront enregistrés, et où les fichiers appelés seront recherchés.
Avec *Pyzo*, la fenêtre *File browser* permet de définir le répertoire de travail.
Sélectionner dans la barre d'adresse le répertoire désiré, cliquer sur l'étoile et utiliser la commande :

Go to this directory in the current shell

Le compilateur renvoie alors un message indiquant le répertoire de travail selectionné :

`cd C:\Mes_documents\Travail\Python`

IV Créer, lire ou modifier des fichier-textes

Un fichier-texte est un fichier d'extension **.txt**, constitué d'une chaîne de caractères.
Les fichier-textes sont principalement destinés à être universellement lisibles.
On rappelle que le caractère `\n` correspond à un retour à la ligne.

1 Ouvrir ou créer un fichier-texte

`T = open(`*`chemin, mode`*`)`

crée une variable `T` utilisée par la suite pour créer, lire ou modifier un fichier-texte.

Ici, *`chemin`* est le chemin d'un fichier-texte existant, ou dont on veut qu'il corresponde à un nouveau fichier-texte ; et *`mode`* désigne le mode d'ouverture :

- `'r'` en mode *lecture* (*read*),
- `'w'` en mode *écriture* (*write*),
- `'a'` en mode *ajout* (*add*).

Exemple : `T = open('document.txt', 'w')`
En mode 'lecture' ou 'ajout', cette variable contient les données du fichier-texte spécifié par le chemin.
En mode 'écriture', cette variable est vide, et permettra de créer un fichier-texte de chemin spécifié.

Attention ! Si un fichier-texte de même chemin existe déjà, celui-ci sera effacé.

2 Écrire dans un fichier-texte

En mode 'écriture' ou 'ajout' uniquement :

`T.write(`*`string`*`)`

concatène la chaîne de caractères *`string`* au contenu pré-existant de la variable `T`.

Exemple (suite) : `T.write('Bonjour tout le monde !\n')`

3 Fermeture et enregistrement

Pour enregistrer toute modification du fichier-texte, on utilise la commande :

```
T.close()
```

Exemple (suite) :

```
T.write('Au revoir ...')
T.close()                              # Un fichier-texte a été créé dans le répertoire de travail.
T = open('document.txt', 'a')          # On ouvre ce même fichier-texte en mode 'ajout'.
T.write('\nPS : et à bientôt.')
T.close()                              # Le fichier-texte a été modifié, une ligne est ajoutée.
```

4 Lire un fichier-texte

On peut extraire tout ou partie d'un fichier-texte :

```
T = open('document.txt', 'r')
chaine = T.read()
T.close()
```

crée une chaîne de caractères qui contient tout le contenu du fichier `'document.txt'`.

```
T = open('document.txt', 'r')
chaine = T.read(n)
T.close()
```

crée une chaîne de caractères qui contient les n premiers caractères du fichier `'document.txt'`.

```
T = open('document.txt', 'r')
T.readline()
T.close()
```

renvoie la première ligne du fichier-texte (jusqu'au premier caractère de nouvelle ligne `\n`), et supprime cette ligne de `T`. L'utilisation répétée de cette commande renvoie les lignes du fichier-texte, une à une.

```
T = open('document.txt', 'r')
T.readlines()
T.close()
```

renvoie une liste contenant une chaîne de caractères pour chaque ligne du fichier-texte.

Exemple :

```
T = open('document.txt', 'r')
ligne1 = T.readline()
ligne2 = T.readline()
T.close()
print(ligne2)
      >>> Au revoir ...
```

V Travail à réaliser

Exercice 1 :
Créer un fichier-texte qui contient la liste des entiers de 1 à 1000 séparés par un espace.

Pour cela :

1. Créer une variable en mode 'écriture' qui correspondra à un nouveau fichier-texte.
2. À l'aide d'une boucle, créer une chaîne de caractères de contenu désiré.
3. Utiliser la commande `.write` pour enregistrer cette chaîne de caractères dans la variable.
4. Utiliser la commande `.close()` pour fermer la variable et créer le fichier-texte correspondant.

Exercice 2 :
Étant donné un entier $p \geqslant 1$, on souhaite créer un script créant un fichier-texte "Syracuse-p" dans lequel sont enregistrées toutes les valeurs de la suite de Syracuse de premier terme p, jusqu'à la première occurence de la valeur 1.

On rappelle qu'une suite de Syracuse S est définie par la relation de récurrence :

$S_{n+1} = 3 \times S_n + 1$ si S_n est un entier impair,
$S_{n+1} = \frac{S_n}{2}$ sinon. *(voir TD 01 & 06)*

1. Créer en mode 'écriture' une variable `Syracuse` pointant vers un fichier-texte `Syracuse-`p`.txt`.
   On pourra concaténer les mots `Syracuse-`, p et `.txt`

2. Enregistrer la valeur p dans la variable `Syracuse`, suivie d'un espace.
3. À l'aide d'une boucle `while`, déterminer la valeur suivante de la suite de Syracuse, et l'enregistrer dans la variable Syracuse, la faire suivre d'un espace.
4. La boucle `while` doit s'arrêter lorsque la valeur 1 est obtenue.

Exercice 3 :
Écrire un script permettant de lire un fichier-texte contenant des valeurs numériques séparées par le caractère & et créant un fichier-texte contenant ces mêmes valeurs classées par ordre croissant, et précisant à la fin la moyenne et la médiane de ces valeurs.

On pourra tester ce script en utilisant le fichier `Liste_de_valeurs.txt` enregistré dans le répertoire classe, dont on créera au préalable une copie dans le répertoire de travail, ouverte en mode 'lecture'.

1. Créer une variable `T` en lisant le fichier-texte.
2. Créer une variable `Chaine` en utilisant la commande `.read()`
3. Initialiser une liste vide.
4. À l'aide d'une boucle dont l'indice parcourt `Chaine`, compléter la liste en adjoignant les valeurs obtenues en convertissant en entiers les chaînes de caractères formées jusqu'à la lecture de chaque caractère `&` (ou utiliser une fonction du TD 02).
5. Trier la liste précédente à l'aide d'une fonction de tri, déterminer médiane et moyenne.
6. Enregistrer un fichier-texte `Liste_triee.txt` contenant les valeurs classées et précisant médiane et moyenne.

Exercice 4 :
On veut créer un fichier-texte contenant la liste des nombres premiers inférieurs à une borne N donnée.
On veut que le nom du fichier évoque la borne choisie.
Compléter le script ci-dessous pour résoudre le problème :

```
def listePremiers (N) :
    nom = "Premiers_inferieurs_a_" + str(N) + ".txt"
    Liste = open(nom, '...')          # Quel est le mode d'ouverture ?
    L, k = [2], 3                     # L sera une liste de nombres premiers
    while k < ... :                   # k est l'entier qu'on teste pour savoir s'il est premier
        premier = True
        for p in L :                  # p est un nombre premier déjà trouvé
            if p**2 > k : break
            if k%p == ... :
                premier = False
                break
        if ... : L.append(...)
        k += ...                      # Quel sera le prochain nombre k testé ?
    for p in L :
        texte = str(p) + ","          # On sépare les nombres premiers par une virgule
        Liste. ...(texte)             # On veut ajouter le nombre p au fichier-texte
    Liste. ...()                      # On enregistre le fichier-texte
    message = "La liste des nombres premiers inférieurs à " + str(N)
    message += " est enregistrée dans un fichier-texte du répertoire de travail. \n"
    message += "Merci, et à bientôt."
    return message                    # On note que cette fonction produit un effet,
                                      # le return est ici pour la forme.
```