Traitement des données

Activité I : Les fichiers textes(extension .txt)

A. Position d'un fichier : chemin relatif et chemin absolu

Pour décrire l'arborescence d'un système, on a deux possibilités : le chemin absolu et le chemin relatif.

- Quand on décrit une cible (un fichier ou un répertoire) sous la forme d'un chemin absolu, on décrit la suite des répertoires menant au fichier. Par exemple, sous Windows, si notre fichier monFichier.txt est contenu dans un dossier test, lui-même présent sur le disque C:, alors le chemin absolu menant `a notre fichier sera

  C: \test\monFichier.txt.

- Quand on décrit la position d'un fichier grâce `a un chemin relatif, cela veut dire que l'on tient compte du dossier dans lequel on setrouve actuellement. Ainsi, si l'on se trouve dans le dossier **C:\test** et que l'on souhaite accéder au fichier monFichier.txt contenu dans ce même dossier, le chemin relatif menant à ce fichier sera tout simplement **monFichier.txt.** Maintenant, si on se trouve dans C:, notre chemin relatif sera **test\monFichier.txt.**

Les chemins absolus et relatifs sont donc deux moyens de décrire le chemin menant à des fichiers ou répertoires. Mais, si le résultat est le même, le moyen utilisé n'est pas identique.

- Quand on utilise un chemin absolu, on décrit l'intégralité du chemin menant au fichier, peu importe l'endroit où on se trouve. Un chemin absolu permet donc d'accéder à un endroit dans le disque quel que soit le répertoire de travail courant. L'inconvénient de cette méthode, c'est qu'on doit préalablement savoir où se trouvent, sur le disque, les fichiers dont on a besoin.

- Le chemin relatif décrit quant à lui la succession de répertoires à parcourir en prenant comme point d'origine non pas la racine, ou le périphérique sur lequel est stockée la cible, mais le répertoire dans lequel on se trouve. Cela présente certains avantages quand on code un projet. En effet, on n'est pas obligé de savoir où le projet est stocké pour construire plusieurs répertoires. Mais ce n'est pas forcément la meilleure solution en toutes circonstances.

Lorsqu'on lance un interpréteur Python, on dispose d'un répertoire de travail courant (que l'on peut afficher grâce à la fonction **os.getcwd()** (CWD = Current Working Directory). Ainsi, pour simplifier, on supposera dans la suite de ce paragraphe que le fichier ***monFichier.tx***t est toujours placé dans le répertoire courant, ce qui permettra d'y accéder à l'aide d'un chemin relatif.

B. Gestion d'un fichier

1. Ouverture d'un fichier

Pour ouvrir un fichier avec Python, on utilise la fonction open. Elle prend en paramètre : le chemin (absolu ou relatif) menant au fichier à ouvrir, le mode d'ouverture et le type d'encodage du fichier. Le mode d'ouverture est donné sous la forme d'une chaine de caractères dont les principaux sont :

'r' : ouverture en lecture (read). Si le fichier n'existe pas, on obtient une erreur,

'w' : ouverture en écriture (write) ; le contenu du fichier est écrasé. Si le fichier n'existe pas, il est créé,

'a' : ouverture en écriture en mode ajout (append) ; on écrit a la fin du fichier sans écraser l'ancien contenu du fichier. Si le fichier n'existe pas, il est créé.

La spécification du type d'encodage pour le fichier n'est pas nécessaire mais fortement recommandé pour éviter certains désagréments.

Dans la suite, on supposera que celui-ci est encodé en UTF-8.

Une fois le fichier ouvert, il est nécessaire de stocker l'adresse de son emplacement dans la mémoire dans une variable. Les syntaxes d'ouverture dans les trois modes précédents sont les suivantes :

```
F = open('monFichier.txt', 'r', encoding= 'utf-8') # mode lecture
F = open('monFichier.txt', 'w', encoding= 'utf-8') # mode écriture
F = open('monFichier.txt', 'a', encoding= 'utf-8') # mode ajout
```

La variable F ainsi définie est un objet de la classe **TextIoWrapper (Enveloppe d'entrée/sortie texte)**. Ainsi, pour interagir avec le fichier monFichier.txt, on doit manipuler des méthodes de cette classe.

2. Lecture d'un fichier : deux façons de procéder

- **F.read()** renvoie une chaîne de caractères contenant l'intégralité du fichier F.
- **F.readline()** renvoie une chaîne de caractères contenant uniquement la ligne du fichier F correspondant à la position du curseur de lecture.

3. Ecriture dans un fichier

F.write(texte) écrit dans le fichier F la chaîne de caractères texte passée en paramètre.

4. Fermeture d'un fichier

Pour fermer le fichier texte, on applique à l'objet F la méthode close(). Pour se faire, on utilise la syntaxe suivante :

```
F.close()
```

Remarque : *Une fois que l'on a fini de travailler sur un fichier, il ne faut surtout pas oublier de le fermer proprement avec la méthode close(), sans quoi il pourrait se produire certains problèmes, comme des pertes de données par exemple.*

Pour éviter des erreurs dues à l'oubli de fermeture, on pourra utiliser la syntaxe suivante pour lire un fichier :

```
with open('monFichier.txt', 'r', encoding= 'utf-8') as fichier:
    texte = fichier.read()
```

Explication de cette syntaxe :

```
with open("monfichier.txt", "w", encoding="utf-8") as myFile:
myFile.write("Ceci est une ligne de texte.\n")
myFile.write("Et une autre ligne !\n")
```

✔ **`open()`** ouvre le fichier avec un mode (`"w"`, `"a"`, `"r"`, etc.)
✔ **`encoding="utf-8"`** assure une bonne gestion des caractères spéciaux
✔ **`with ... as`** garantit que le fichier sera fermé proprement
✔ **`myFile`** est l'objet permettant d'écrire ou lire dans le fichier