Activités : Structures de données et Fichiers

Activité 1 : Structure de données

On souhaite stocker des informations sur les aliments disponibles pour le petit déjeuner d'une association.

- 4 bouteilles de lait ;
- 10 paquets de biscottes ;
- 8 paquets de céréales ;
- 5 bouteilles de jus d'orange ;
- 3 mottes de beurre.

1. Créer une structure de données en Python pour enregistrer ces données.
2. Créer une fonction **afficher_produits()** qui retourne la liste des produits disponibles pour le petit déjeuner.
3. Créer une fonction **quantite_produit()** qui pour un produit donné retourne la quantité disponible pour le petit déjeuner.

   Quelle est la quantité de beurre disponible ?
4. Créer une fonction **modifier_stock()** qui permet de modifier les quantités de produit disponibles pour le petit déjeuner.

   On a acheté 2 bouteilles supplémentaires de jus d'orange ; modifier le stock en utilisant la fonction précédente.
5. Créer une fonction **ajouter_produit()** qui permet d'ajouter un produit à la liste de ceux disponibles pour le petit déjeuner.

   Ajouter 10 yaourts pour le petit déjeuner.

Activité 2 : Utilisation de fichiers

On peut écrire et lire des données dans un fichier externe, comme du texte, une image, du son...

Création de Fichiers Texte

Pour travailler avec un fichier, on procède en trois étapes :

- l'ouverture du fichier ;
  exemple : **mon_fichier = open('nom du fichier', 'mode d'ouverture')**
- le traitement du fichier ;
- la fermeture du fichier.
  exemple : **mon_fichier.close()**

Remarque : Lorsque l'on ouvre le fichier, on choisit un mode d'ouverture parmi les trois suivants :

r : on ouvre le fichier en mode lecture ;

w : on l'ouvre en mode écriture ; on écrase ce qui était déjà écrit ou bien on crée le fichier s'il n'existe pas ;

a : on l'ouvre en mode ajout ; on écrit dans le fichier, à la suite de son contenu.

Partie A]

1. Recopiez le programme ci-dessous et exécutez-le :

```
#Création d'un nouveau fichier
mon_fichier = open('fichier1.txt','w')
mon_fichier.write("Hello !")
mon_fichier.close()
```

2. Regardez dans le dossier où vous avez enregistré le programme, un fichier texte (.txt) est apparu. Ouvrez-le (avec Bloc-note), pour vérifier son contenu.
3. Complétez le programme précédent pour ajouter du texte au fichier précédent.

Remarques :

- Le fichier doit se trouver dans le même dossier que le programme.
- Il est important de bien fermer le fichier après utilisation car cela permet de finaliser l'écriture dans le fichier et de le libérer pour les autres utilisateurs.
- Le symbole **'\n'** permet d'écrire dans un fichier en passant à la ligne.
- La méthode **write** ne permet d'écrire que des chaînes de caractères dans le fichier.

Ouverture de Fichiers

Il existe en Python le mot-clé **with** qui permet d'ouvrir et de fermer un fichier de manière efficace. Si pour une raison ou une autre l'ouverture ou la lecture du fichier conduit à une erreur, l'utilisation de with garantit la bonne fermeture du fichier, ce qui n'est pas le cas dans le code précédent.

4. Testez le code suivant, équivalent au précédent :

```
#Création d'un nouveau fichier
with open('fichier1.txt','w') as mon_fichier:
    mon_fichier.write("Hello version 2 !")  #on indente le traitement du fichier
```

Lecture de Fichiers

Pour lire le contenu d'un fichier :

- on peut utiliser la méthode **read()**, qui va lire l'intégralité du fichier et le retourner sous la forme d'une chaine de caractères.

  exemple : **mon_fichier.read()**

- La méthode **readline()** (au singulier) permet de lire une ligne du fichier à la fois :

  exemple : **ligne = mon_fichier.readline()**

- On peut placer les lignes du fichier dans une liste avec la méthode readlines() :

  exemple : **liste = mon_fichier.readlines()**

- Un fichier est itérable, on peut utiliser une boucle **for in** :

  exemple : **for ligne in mon_fichier :**
  print(ligne)

5. Testez les différentes façons de lire un fichier.

Exercice :

Récupérez le fichier *mots* sur Pearltree, dans le dossier NSI/Langage Python/Fiches d'exos.

1. Écrivez un programme qui compte le nombre de mots du fichier *mots*, puis affiche un des mot du fichier au hasard.
2. Créez une fonction qui retourne le mot le plus long du fichier *mots* et sa taille.