Exercice 1 : Lecture dans un fichier

(1) Ouvrir un fichier avec **gedit**, écrire un texte de plusieurs lignes, le sauvegarder comme **texte1.txt** sur votre clé USB dans le répertoire du TP du jour.

(2) Ouvrir un fichier **Python** avec **Pyzo** que l'on nommera **lecture_script.py**. Enregistrer ce fichier dans le même repertoire que le fichier texte.

(3) Écrire dans votre fichier les lignes suivantes.

```
with open("texte1.txt", "r", encoding="utf-8") as fichier_texte:
    texte = fichier_texte.read()
```

a. Afficher le type de la variable **texte**

b. Afficher la variable **texte**

c. Proposer une fonction qui compte le nombre de *e* dans votre texte.

d. Comment peut-on vérifier simplement que le mot **Bonjour** est dans votre texte ?

(4) a. Modifier **read** par **readline**

b. Afficher toutes les lignes du texte.

c. Afficher la première lettre de chaque ligne.

(5) a. Modifier **readline** par **readlines**.

b. Afficher la première lettre de chaque ligne.

Conclusion : Il existe 3 façons pour lire dans un fichier, il faut savoir les utiliser et bien choisir la façon en fonction du contexte

Exercice 2 : Écriture dans un fichier

(1) Dans le même répertoire, créer le fichier **ecriture_script.py** et recopier les lignes suivantes :

```
with open("texte2.txt", "w", encoding="utf-8") as fichierTexte:
    fichierTexte.write("test d'ecriture")
```

(2) Vérifier que le résultat attendu est obtenu.

(3) Proposer un script permettant d'écrire 100 nombres aléatoires avec un nombre par ligne (Le caractère `"\n"` permet d'aller à la ligne).

(4) Proposer un script permettant d'écrire votre texte (dans texte1) à l'envers.

(5) Proposer un script permettant d'écrire votre texte (dans texte1) en remplaçant les *e* par des *o*. (on pourra voir la fonction replace)

(6) Lors de l'ouverture du fichier, modifier `"w"` par `"a"`. Que se passe-t-il ?

On se donne maintenant le fichier **lacnoir.csv** qui donne la quantité de pollens de quelques espèces en fonction de la profondeur dans une tourbe.

On va manipuler automatiquement ce fichier.

Exercice 3 : Lecture d'un fichier CSV

(1) Télécharger le fichier **lacnoir.csv**.

Les fichiers CSV ont une structure bien définie que l'on peut observer en ouvrant le fichier avec **gedit** puis à l'aide de **OpenOffice**.

En Python, il est facile d'utiliser cette structure pour travailler le fichier. On peut importer la bibliothèque **CSV**

(2) On se donne la fonction :

```
def dict_fichier_csv(nomFichier):
    with open(nomFichier, 'r') as fichierTexte:
        listeCSV = list(csv.DictReader(fichierTexte))
    return listeCSV
```

On obtient alors une liste de dictionnaire, on va analyser cette fonction.

a. Analyser cette fonction et tester là avec **lacnoir.csv**.

b. Comment obtenir le nombre de grains de pollen d'un noisetier à une profondeur de 100 mètre.

(3) Proposer une fonction **taux** qui prend en arguments une profondeur et une espèce et retourne le nombre de pollens.

(4) Proposer une fonction **taux_max** qui prend en argument une profondeur et renvoie l'espèce ayant le plus grand nombre de pollens à cette profondeur.

Exercice 4 : Écriture d'un fichier CSV

On peut aussi écrire facilement un fichier CSV à partir d'un dictionnaire en utilisant la fonction suivante :

```
def ecrire_fichier_csv(dictionnaire, nomFichierSortie):
    # L'entete du fichier est composee des cles du dictionnaire de chaque el
    entete = dictionnaire[0].keys()
    # On ecrit ensuite cette liste dans le fichier csv
    with open(nomFichierSortie, 'w') as ListeAEcrire:
        # fichierListe est le curseur dans notre fichier.
        fichierListe = csv.DictWriter(ListeAEcrire, entete)
        # On ecrit l'entete :
        fichierListe.writeheader()
        # Puis on ecrit l'ensemble des lignes
        fichierListe.writerows(dictionnaire)
```

(1) Faire fonctionner cette fonction pour recopier à l'identique le fichier **lacnoir.csv** dans **lacnoir2.csv**

(2) Proposer une fonction **profondeur_espece_superieur** qui prend en arguments un nom de fichier à lire, un nom de fichier à écrire, une espèce et un nombre et n'écrit dans le fichier que les profondeurs supérieurs au nombre entré en argument.

Attention, toutes les entrées lues dans un fichier sont du type string. On devra utiliser des conversions vers des entiers avant de comparer des nombres.

Exercice 5 : Rajouter une ligne

Pour lire et écrire à la fois dans un fichier, on utilise la commande :

```
with open(nom_fichier_entre, "r+")
```

Toute ligne écrite se rajoute à la fin.

(1) Écrire une fonction **moyenne** qui prend un fichier à lire et rajoute, à la dernière ligne dans le même format que le fichier, la moyenne du nombre de pollens par espèce.

(2) Écrire une fonction **mediane** qui prend un fichier à lire et rajoute, à la dernière ligne dans le même format que le fichier, la médiane du nombre de pollens par espèce.

Exercice 6 : Illustrer graphiquement les données

En utilisant le dernier TP, illustrer les données à l'aide d'un graphique. On pourra prendre comme modèle le fichier **lacnoir.pdf**

Exercice 7 : Construire un protocole

On se donne le fichier **prefer_climat_coul.png** qui donne les espèces en fonction des climats.

Proposer un protocole qui permet de dater les changements climatiques (en terme d'hygrométrie et de températures) et de déterminer le type de climat à chaque fois.