Travaux Pratiques n°5
Manipulation de fichiers

Le but de ce TP est d'apprendre à lire, créer ou modifier des fichiers.

I. Le module os

▷ Pour commencer ouvrir le répertoire Python. Ne pas créer de répertoire TP05 !

Laisser simplement la fenêtre ouverte sur le bureau Windows.

▷ Ouvrir Pyzo (ou autre), importer le module os (operating system) :

```
>>> import os
>>> os.getcwd()
```

L'instruction getcwd indique le répertoire courant de travail (Current Working Directory).

On peut accéder à l'adresse complète du répertoire Python via le bureau Windows : cliquer sur la barre d'adresse et copier celle-ci. À la maison comme sur vos PCs, on devrait obtenir quelque chose comme C:\User\Alphonse\Mes documents\Python.

▷ Changer le répertoire de travail grâce à l'instruction chdir (Change directory).

```
>>> os.chdir("C:\\...
   MonRépertoirePython")
>>> os.getcwd()
```

▷ Créer le répertoire TP05 à l'aide de l'instruction mkdir (make directory) et le choisir comme répertoire courant.

```
>>> os.mkdir("TP05")
>>> os.chdir("TP05")
>>> os.getcwd()    # vérification
```

▷ Constater dans l'explorateur de Windows que le répertoire est bien créé.

▷ A titre d'info, si l'on veut dans le sous-répertoire dir du répertoire courant, on le note "./dir/". Si l'on doit remonter en arrière, c'est "../autr_dir/dir/". Si l'on doit remonter deux fois en arrière, "../../autre_dir2/autre_dir/dir/".

Tous les fichiers de ce TP seront créés et sauvegardés dans ce répertoire, donc on ne modifiera plus le répertoire courant de travail.

Les fonctions principales du module os :

getcwd()	Renvoie le répertoire courant de travail
chdir(chemin)	change de répertoire courant
mkdir(rep)	crée le répertoire rep
rmdir(rep)	supprime le répertoire rep
listdir(chemin)	affiche la liste des fichiers contenus dans le répertoire chemin (par défaut le répertoire courant)
remove(fichier)	supprime fichier
rename(source,dest)	renomme source en dest

II. Fichiers texte

Les fichiers `txt` sont des fichiers ne contenant que du texte, sans mise en forme, mais avec les retours à la ligne et éventuellement des tabulations. Sous Windows ils sont ouverts par l'application Bloc-notes (Notepad en anglais).

Pour accéder à un fichier `txt` en Python on l'ouvre grâce à la fonction **open**. En fin d'utilisation on le ferme grâce à la méthode `close`.

Un fichier peut être ouvert en lecture (**open** avec l'option `'r'`) ou en écriture (**open** avec l'option `'w'`). Cette option permet également de créer un fichier. Il existe d'autres options comme `'a'` (append) pour l'ajout à la fin du fichier.

Ouverture d'un fichier en écriture :

▷ Taper les instructions ci-dessous (sans les commentaires) dans le Shell.

```
>>> montxt=open('Test.txt','w')
```

Comme le fichier n'existait pas il est créé. Pour l'instant l'objet `montxt` désigne le contenu qui sera sauvegardé, on peut le nommer comme on veut. On replace ce pointeur en début de fichier si l'on ferme puis réouvre le fichier.

```
>>> montxt.write("J'écris un texte,\n et il va être sauvegardé.\n")
    # Python renvoie le nombre de caractères écrits)
>>> montxt.write("C'est super !")
>>> montxt.close()
```

Ce n'est qu'à l'exécution de la fonction `close` que le fichier est physiquement créé (ou modifié).

▷ Retourner dans la fenêtre de l'explorateur de Windows contenant votre répertoire de travail Python, double-cliquer sur `Test.txt` pour vérifier son contenu. Le bloc-notes doit s'ouvrir.

Ne pas oublier la chaîne de caractère `\n` pour le retour à la ligne, ni `\t` pour la tabulation. Ces deux chaînes ne comptent que pour un seul caractère.

Ouverture en lecture :

▷ Taper les instructions ci-dessous, toujours dans le Shell :

```
>>> montxt=open('Test.txt','r')
>>> S=montxt.read()
>>> print(S)
>>> montxt.close()
```

Penser à fermer le fichier pour éviter les conflits.

Un autre moyen courant d'utiliser un fichier est de le traiter ligne par ligne, donc de parcourir les lignes.

▷ Taper l'exemple ci-dessous dans *l'éditeur* :

```
montxt=open('Test.txt')    # L'option 'r' est sélectionnée par défaut
k=0
for L in montxt:
    k=k+1
    print("Ligne",k,": ",L)
montxt.close()
```

À retenir

Écriture dans un fichier

```
montxt=open('fichier.txt','w')
...
montxt.write('texte...')
...
montxt.close()
```

Lecture d'un fichier

```
montxt=open('fichier.txt','r')
...
S=montxt.read() # ou
L=montxt.readline()
...
montxt.close()
```

Méthodes principales d'utilisation d'un fichier :

`read()`	lit le fichier en entier	renvoie une chaîne de caractères
`read(n)`	lit n caractères	renvoie une chaîne de longueur au plus n, vide si tout le fichier a été lu.
`readline()`	lit une ligne	renvoie une chaîne de caractère, vide si le fichier est fini
`readlines()`	lit toutes les lignes	renvoie la liste de toutes les lignes sous forme de chaînes de caractères.
`write(S)`	écrit la chaîne `S`	
`writelines(L)`	écrit les lignes de la liste `L`	
`close()`	enregistre le fichier	

Toutes ces méthodes s'appliquent à un objet de type fichier comme `montxt` : `L=montxt.readline()`, `montxt.write("Hello")`...

Remarque sur les méthodes :

Une *méthode* est une syntaxe particulière aux langages orientés objets. Par exemple :

```
>>> L=[6,8,3,5,6]
>>> sorted(L) # fonction de tri
[3,5,6,6,8]
```

```
>>> L=[6,8,3,5,6]
>>> L.sort()  # méthode de tri
              # L est triée
```

L'instruction `sorted` est une *fonction* qui a pour paramètre la variable L.

L'instruction `sort()` est une *méthode* appliquée à l'objet L.

Les instructions **open()**, `read()`,**open()**,`close()`,`write()`,`readline()`, `readlines()` sont à connaître.

À retenir

Utilisation d'une méthode :

```
objet.methode()
```

Utilisation d'une fonction :

```
fonction(objet)
```

Avec arguments :

```
objet.methode(arg)
```

Avec arguments :

```
fonction(objet,arg)
```

III. Chaînes de caractères

Quelques fonctions et méthodes s'appliquant à une chaîne de caractères `S` :

`len(S)`	longueur de `S`	
`S[i]`	élément d'indice `i` de `S`	`S[-1]` pour le dernier élément
`S[i:j]`	sous-chaîne de `S[i]` à `S[j-1]`	`S[:j]`, `S[i:]` pour le début, la fin
`c in S`	teste si `c` est dans `S`	
`S.split()`	renvoie la liste des mots de `S`	
`S.upper()`	transforme les minuscules en majuscules	`S.lower()`, `S.capitalize()` existent aussi

Pour la liste complète : **`help(str)`**. Les méthodes `count`, `index`, `replace`, `join` peuvent être utiles également. **`len()`** et `split()` sont à connaître.

▷ Tester les méthodes données ci-dessus, en tapant les lignes suivantes dans le Shell (sans les commentaires) :

```
>>> montxt=open("Test.txt")
>>> Texte=montxt.read()
>>> montxt.close()

>>> print(Texte.upper())
>>> "super" in Texte
>>> Texte.index("super")   # indice du premier caractère de "super"
>>> Texte.split()
```

La méthode `split` sépare une chaîne de caractères en une liste de chaînes de caractères selon un caractère particulier. Par défaut le délimiteur est l'espace (`S.split()`) mais il est possible d'en spécifier un autre.

▷ Tester (sans les commentaires) :

```
>>> Ligne="1;5;6.5;200"
>>> L1=Ligne.split(";")
```

```
>>> L1
>>> type(L1)      # Quel est le type de L1 ?
>>> type(L1[0])   # Quel est le type d'un élément de L1 ?
>>> Liste=[float(x) for x in L1]
>>> Liste
```

On peut aussi essayer `Ligne.split("5")`.

IV. Exercices

▷ Télécharger depuis le site `moodle-deodat.fr` le fichier compressé `TP05-Fichiers.zip`, en utilisant la fonction « enregistrer la cible du lien sous ». Le décompresser et placer dans votre répertoire de travail tous les fichiers qu'il contient.

▷ Si vous utilisez linux ou macOS, l'encodage par défaut des fichiers texte est différent de celui utilisé par Windows (les caractères ne sont pas codés de la même manière en mémoire). Windows utilise l'encodage latin1, alors que macOS et linux utilisent l'encodage utf8. Les fichiers que vous venez de télécharger ont été créés sous Windows. Pour les ouvrir, il faut utiliser la commande :

```
montxt=open('fichier.txt', encoding='latin1')
```

▷ **Exercice 1.**

a. Écrire une fonction `Occurrences(a,S)` comptant le nombre d'occurrences d'un caractère `a` dans une chaîne `S`.

b. Ouvrir le ficher `LaFontaine.txt` à l'aide de Python. Stocker son contenu, par exemple dans la variable `Texte`.

Convertir toutes ses majuscules en minuscules.

c. Compter le nombre d'occurrences de chaque lettre de l'alphabet dans ce texte.

Pour ceci il peut être utile de noter `Alphabet="azert..."` la chaîne de toutes les lettres de l'alphabet, non obligatoirement classées. Il suffira alors d'écrire :

```
for x in Alphabet:
    print(x,"apparaît",Occurrences(x,Texte),"fois.")
```

d. Créer un fichier `Compte.csv` contenant 26 lignes, toutes de la forme `"x,k"` où k est le nombre d'occurrences de la lettre `x`.

Attention ! Ces lignes doivent être des chaînes de caractères (et il faut leur ajouter le retour à la ligne `'\n'`). Pour concaténer des chaînes de caractères, on utilise +.

Rappel : la fonction convertissant un nombre (5) en chaîne de caractères ("5") est :

Réponse :

On crée un fichier `csv` *pour pouvoir le manipuler avec LibreOffice ou Excel. Les colonnes d'un tel fichier sont généralement séparées par des virgules.*

e. Sous Windows double-cliquer sur le ficher `Compte.csv` pour voir le nombre d'occurrences de chaque lettre. Trier les lignes pour voir les lettres les plus fréquentes et les plus rares.

Pour trier des lignes selon les éléments d'une colonne sous LibreOffice ou Excel, sélectionner les lignes en questions (toutes en l'occurrence), menu Données cliquer sur Tri et sélectionner la colonne selon laquelle le tri est souhaité. On peut aussi utiliser Bloc-Notes.

Quelles sont les deux lettres qui n'apparaissent pas dans cette fable ?

Réponse :

Combien de `"e"` apparaissent ?

Réponse :

▷ **Exercice 2 : Tracé de la carte de France.**

On utilise le fichier texte `Coordonnees_France.txt`. Comme on peut le constater en double-cliquant dessus sous Windows, ses lignes sont de la forme :

`Coordonnees_France.txt`

```
849460.0  6524534.0
848984.9  6525112.0
849022.0  6527350.0
...
```

Il s'agit des coordonnées des points frontières de la France métropolitaine. Ces coordonnées sont au format Lambert 93, système géodésique officiel depuis 2001.

a. Ouvrir depuis `Pyzo` (ou autre) le fichier `Coordonnees_France.txt`. Afficher toutes ses lignes.

C'est très long ! (CTRL-I pour stopper, ou la croix sur fond rouge au-dessus du Shell).

b. Compter le nombre de lignes de ce fichier.

Nombre de points :

c. Stocker les données contenues dans le fichier `Coordonnees_France.txt` dans deux listes `X` et `Y` de flottants, une pour la première colonne et l'autre pour la seconde.

Utiliser la méthode `split()` pour découper et la fonction `float` pour convertir en flottant.

```
>>> A="9.6"
>>> float(A)
```

d. Afficher le graphique reliant les points dont les coordonnées ont été stockées dans les listes `X` et `Y`. Tracer ce graphique en noir, dans un repère orthonormé.

Compléments

Les exercices suivants peuvent être traités dans le désordre.

▷ **Exercice 3 : îles.**

En réalité la France métropolitaine n'est pas un seul bloc, elle possède aussi des îles.

Les fichiers `Coordonnees_France_n.txt` où n va de 1 à 28 contiennent toutes ces parties. Le premier contient la partie principale, les autres contiennent les îles.

Tracer à l'aide de ces fichiers la carte complète de France métropolitaine.

▷ **Exercice 4 : départements.**

Ouvrir sous Windows le fichier `Coordonnees_France_Dept.txt`.

Il contient la liste des points frontière des départements, chacun étant initialisé par une ligne `"Département n"`. Il contient aussi des lignes `"cut"` pour des coupures, permettant de tracer les îles.

Tracer la carte des départements métropolitains.