MPSI - ITC - TD 19 - 2 février 2026

Manipuler des fichiers depuis Python

Accéder à un fichier

Pour manipuler un fichier en python, il faut procéder en 3 étapes :

1. Ouvrir le fichier
2. Effectuer les opérations souhaitées dans le fichier.
3. Fermer le fichier

Étape 1 : Ouvrir un fichier

On utilise la fonction python `open`, qui prend deux arguments :

- Une chaîne de caractères contenant le nom du fichier (dans ce TP on ne manipulera que des fichiers qui se trouvent dans le même dossier que le code python exécuté, pour simplifier).
- Une chaîne de caractères qui vaut soit `'r'` (comme *read*) si on veut ouvrir le fichier en mode lecture, ou `'w'` (comme *write*) si on veut ouvrir le fichier en mode écriture. D'autres modes existent, qu'on n'étudiera pas.

La fonction `open` renvoie un *objet-fichier*, qui a plusieurs méthodes permettant de manipuler le contenu du fichier.

Par exemple, si on veut ouvrir un fichier qui s'appelle `exemple.txt` en mode lecture et appeler `f` l'objet-fichier correspondant, la ligne python correspondante est

```
f = open("exemple.txt", "r")
```

Étape 2 : manipuler un fichier en mode lecture

Si `f` contient un objet-fichier d'un fichier ouvert en mode lecture, alors

- La méthode `f.readline()` renvoie une chaîne de caractères correspondant à la ligne suivante dans le fichier.
- La méthode `f.readlines()` renvoie une liste de chaînes de caractères. Chaque élément de la liste correspond à une ligne du fichier, jusqu'à la fin du fichier.
- La méthode `f.read(n)` renvoie une chaîne de caractères contenant les `n` caractères suivants dans le fichier.

Étape 2 : manipuler un fichier en mode écriture

Si `f` contient un objet-fichier d'un fichier ouvert en mode écriture, alors on dispose de la méthode `f.write(s)` qui écrit dans le fichier la chaîne de caractères passée en argument.

Si on veut écrire un saut de ligne à un moment donné, il suffit de placer une barre oblique inversée suivie de la lettre `n` (comme *new line*) dans la chaîne de caractères (par exemple `f.write("\n")` pour n'écrire que un saut de ligne).

Étape 3 : Fermer un fichier

Si `f` contient un objet-fichier, l'instruction python

```
f.close()
```

permet de fermer le fichier.
N'oubliez pas cette étape faute de quoi vous pourriez avoir des problèmes plus tard.

Exemple complet en lecture

Le code suivant affiche dans la console python toutes les lignes d'un fichier `exemple.txt` :

```
f = open("exemple.txt", "r")
for line in f.readlines():
    print(line)
f.close()
```

Exemple complet en écriture

Le code suivant écrit dans un fichier `compter.txt` un comptage de 1 à 10 avec un nombre par ligne :

```
f = open("compter.txt", "w")
for i in range(10):
    f.write(str(i + 1) + "\n")
f.close()
```

Manipuler une chaîne de caractères (type `str`) (cf https://docs.python.org/fr/3/library/stdtypes.html#text-sequence-type-str) Si la variable `st` contient une chaîne de caractères, on dispose des méthodes suivantes :

- `st.upper()` renvoie une copie de la même chaîne de caractères où les lettres ont toutes été converties en majuscules. De même, `st.lower()` renvoie une copie toute en majuscules.
- `st.strip()` permet d'enlever l'espace blanc en début et fin de chaîne (`help("".strip)` si besoin)
- Si `m` est aussi une chaîne de caractères, la méthode `st.split(m)` renvoie une liste de chaînes de caractères correspondant à un découpage de `st` autour des occurrences de `m`. Par exemple, l'expression `"Ceci est une phrase. Incroyable, non?".split(" ")` renvoie la liste `["Ceci", "est", "une", "phrase.", "Incroyable,", "non?"]`.
- `st.replace(c,r)` remplace dans la chaîne `st` la sous-chaîne `c` par `r`
- `len(st)` renvoie le nombre de caractères dans la chaîne `st`.

Manipuler un dictionnaire (type `dict`)
Un dictionnaire python est une structure de données constituée de couples clé/valeur. Contrairement à un tableau où on accède à une valeur par son indice, un dictionnaire permet d'accéder aux valeurs par des clés (qui peuvent être presque tout ce qu'on veut, on prendra dans ce TD des chaînes de caractères).

- `d = {}` crée un dictionnaire `d` vide
- `d[cle]=valeur` crée la clé `cle` associée à la valeur `valeur`
- **une fois** `cle` **créée**, on peut modifier sa valeur associée (ex : `d[cle] = d[cle] + 1`)
- `len(d)` renvoie le nombre de clés du dictionnaire
- on peut tester l'existence d'une clé dans un dictionnaire : `if cle in d`
- on peut parcourir un dictionnaire par clé (`for cle in d` ou `for cle in d.keys()`)
- on peut parcourir un dictionnaire par couple cle/valeur (`for cle,valeur in d.items()`)
- `d.values()` renvoie un tableau des valeurs contenues dans `d`
- `d = {'un':1, 'deux':2, 3:3, 'quatre':'quatre'}` : exemple de création d'un dictionnaire : l'association clé/valeur se fait avec `:`, les couples clé/valeur sont séparés par une virgule.

Exercice 1 (The Tragedy of Hamlet, Prince of Denmark)

Plus couramment désigné sous le titre abrégé *Hamlet*, la plus longue et la plus célèbre des pièces de William Shakespeare fut présentée entre 1598 et 1601. Le texte anglais est présenté dans le fichier `hamlet_eng.txt` et une traduction française dans le fichier `hamlet_fr.txt`.

1. Enregistrer les fichiers `hamlet_eng.txt` et `hamlet_fr.txt` dans le même dossier où vous créerez votre code.
2. Visualisez par ailleurs (en les ouvrant dans Pyzo, par exemple) le contenu des fichiers `hamlet_eng.txt` et `hamlet_fr.txt`, pour mieux comprendre les questions suivantes.
3. Afficher dans la console python les 20 premières lignes non-blanches de `hamlet_fr.txt`.
4. Afficher dans la console le nombre d'interventions des personnages de la liste suivante : `['FRANCISCO.', 'BERNARDO.', 'HORATIO.', 'HAMLET.', 'MARCELLUS.', 'LE ROI.', 'LA REINE.', 'OPHÉLIA.', 'LAERTES.', 'POLONIUS.', 'REYNALDO.', 'ROSENCRANTZ.', 'GUILDENSTERN.', 'VOLTIMAND.', 'OSRICK.', 'FORTINBRAS.']`
5. Créer une fonction `simplifier(ligne)` qui prend en argument une chaîne de caractères représentant une ligne du texte et qui
   — Remplace tous les caractères qui ne sont pas des lettres (les tirets courts, les tirets longs, les apostrophes, les points, les deux-points, les virgules les point-virgules, points d'exclamation, les points d'interrogation, les parenthèses, les chiffres de 0 à 9) par des espaces.
   — Met toutes les lettres en minuscule.
6. Créer une liste `mots` contenant tous les mots du texte à l'aide de la fonction précédente.
7. Créer un dictionnaire des mots du texte avec comme clé le mot et comme valeur le nombre d'occurrences du mot (ou fréquence du mot). Combien de mots différents comporte le texte ?
8. Quel est le mot le plus fréquent ? Combien de fois apparaît-il ?
9. Créer la liste des couples (mots, fréquence) des mots du texte qui apparaissent plus de 200 fois. Afficher le contenu de cette liste dans la console.
10. Créer avec python le fichier `hamlet_fr_frequences.txt` et y écrire par fréquence décroissante les 50 mots les plus fréquents.
11. Reprendre en les adaptant les questions précédentes pour le fichier `hamlet_eng.txt`.