TD 14 : Lecture d'un fichier

Accéder à un fichier
Pour manipuler un fichier, `python` propose `f = open(filepath, mode='r', encoding=None)` (cf https://docs.python.org/fr/3/library/functions.html?highlight=open#open) où
- `filepath` désigne un chemin (relatif (au répertoire du script) ou absolu (depuis la racine du disque dur) vers le fichier considéré
- `mode` permet de sélectionner si on ouvre le fichier en lecture (`r` pour read) ou en écriture (`w` pour write). Il existe d'autres modes, notamment le mode `a` (pour append).
- `encoding` est l'encodage à utiliser pour décoder le fichier. Depuis quelques années, l'encodage `utf-8` est devenu la norme.

Manipuler un objet-fichier
On obtient un objet-fichier `f` qui permet d'accéder au texte avec les méthodes suivantes :
- `f.read(n)` : lit `n` caractères ou le texte en entier si `n` est omis
- `f.readline()` : lit la ligne suivante
- `f.readlines()` : lit toutes les lignes suivantes
- `f.write(st)` : écrit la chaîne `st` dans le fichier (dépend du mode d'accès choisi)
- `f.close()` : supprimer l'accès au fichier (étape **obligatoire**)

Le caractère de fin de ligne est noté `\n` et il faut le gérer.

Manipuler une chaîne de caractère (type `str`, variable `st`)
(cf https://docs.python.org/fr/3/library/stdtypes.html#text-sequence-type-str)
- `st.upper()` ou `st.lower()` : majuscule / minuscule
- `sf.strip()` permet d'enlever des caractères en début et fin de chaîne (`help(st.strip)` si besoin)
- `st.split(m)` renvoie un tableau de caractère, découpage de `st` au niveau du motif `m`
- `st.replace(c,r)` remplace dans la chaîne `st` la sous-chaîne `c` par `r`
- `len(st)` renvoie le nombre de caractère dans la chaîne

Manipuler un dictionnaire (type `dict`)
Un dictionnaire python est une structure de donnée constituée de couple 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 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` crée**, on peut modifier sa valeur associée (ex : `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.

Les dernières versions de python semblent garantir un certain ordre dans les clés (l'ordre de création des clés), ce qui n'était pas le cas avant. Et rien ne garantie que ce choix perdurera. Il semble recommandé de considérer cette structure de donnée comme non-ordonnée.

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_Anglais.txt` et une traduction française dans le fichier `Hamlet_fr.txt`. Ces deux fichiers sont encodés en `utf-8`.

1. Ouvrir le fichier `Hamlet_fr.txt` (que l'on placera dans le même répertoire que les fichiers `.py` créés) dans `pyzo` pour comprendre la suite.
2. Créer un objet-fichier lié au fichier `Hamlet_fr.txt`.
3. Afficher dans la console les 20 premières lignes non-blanches de ce fichier.
4. Afficher dans la console le nombre d'interventions des acteurs 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 `mise_en_forme(mot)` qui renvoie `mot` après avoir réalisé les opérations suivantes
   - Remplacement des caractères suivants : tiret court, tiret long, apostrophe par une espace.
   - Mise en minuscule de tout le texte.
   - Suppression en fin ou début de chaque mot des caractères : point-virgule, deux-points, virgule, point d'interrogation, point d'exclamation, parenthèses, chiffres de 0 à 9, point.
   - s'il y a une apostrophe comme dans `mot="d'artillerie.)"`, la fonction doit renvoyée `artillerie`.
6. Créer un tableau `mots` (type `list`) 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_Anglais.txt`.