Projet NSI – pour le 11/03/2025

A rendre sur Pronote en un seul fichier Python, si nécessaire commenté avec vos remarques. Vous pouvez le faire à 2 ou 3. N'oubliez pas de mettre votre (vos) nom(s) dans le nom de fichier.
Préliminaire : bien comprendre le fonctionnement du programme lecture_ecriture_fichier.py, que vous avez déjà utilisé pour le DM sur l'ASCII art.
Le but de ce devoir à la maison est de réaliser l'analyse du vocabulaire d'un roman.
Remarque : l'essentiel de la notation (15 points sur 20) repose sur les questions non facultatives, il y a tout de même 5 points sur les questions dites « facultatives ».
Autre remarque : posez-vous la question pour le 3 et 4 : pensez-vous que le programme met longtemps à s'exécuter ?

1. Écrire une fonction `occurrences(chaine)` qui, étant donnée un chaîne de caractères, renvoie un dictionnaire où les clés sont les mots de la chaîne, et les valeurs sont le nombre de fois où apparaît le mot dans la chaîne.
   On utilisera la méthode `chaine.split()` qui découpe une chaîne en mots (les séparateurs sont les espaces et les retour à la ligne).
   Exemple : `occurrences("coucou coucou c est le coucou qui dit coucou ne dit pas bonjour ne dit pas au revoir même pour se revoir")`
   renverra :
   `{'coucou': 4, 'c': 1, 'est': 1, 'le': 1, 'qui': 1, 'dit': 3, 'ne': 2, 'pas': 2, 'bonjour': 1, 'au': 1, 'revoir': 2, 'même': 1, 'pour': 1, 'se': 1}`

2. Écrire une fonction `occ_max(dico,longueur)` qui, dans un dictionnaire comme celui renvoyé au 1, donne le mot de longueur donnée en paramètre ayant le nombre d'occurrences maximal, ainsi que ce nombre d'occurrences. On ne renverra qu'un seul des mots, au cas où il y en aurait plusieurs de même longueur
   Exemple : `occ_max(dico,6)` renvoie `'coucou'`et `4,` (et pas `'revoir'` et `2` , le mot est de même longueur mais a moins d'occurrences)

3. Facultatif, difficulté moyenne.
   Écrire une fonction `nettoyage(nom_texte)` qui, étant donné un fichier texte :
   - remplace tous les symboles de ponctuation par des espaces ;
     avec la liste
     `ponctuation = ["'", '"', "(", ")", ",", ";", ":", ".", "?", "!", "‘", "«", "»", "_", "-"]`
   - attention aux différents types de guillemets et d'apostrophes. La liste est donnée sur pronote, vous pouvez la recopier.
   - écrit tout le texte en minuscules ;
   - Le fichier texte sera donné en paramètre par son nom sous forme de chaîne de caractères ;
   - Il sera ouvert (et fermé) dans la fonction ;
   - La fonction créera un nouveau fichier texte, correspondant au texte d'origine mais sans la ponctuation ;
   - Elle renverra une chaîne de caractères donnant le nom de ce nouveau fichier.

   Aide :
   - pour passer en minuscules, on utilisera la méthode `chaine.lower()` : `'CoUCOu'.lower()` renvoie `'coucou'`, `'U'.lower()` renvoie `'u'`
   - On utilisera les méthodes de l a question préliminaire bien sûr.

4. Analyse du vocabulaire
   - Version sans la question 3 : utiliser le fichier « germinal_nettoye.txt »
   - Version avec la question 3 :
     Utiliser le fichier « germinal.txt » ou bien télécharger un des romans suivants (ou un autre, si c'est le cas choisissez-en un de taille conséquente).
     Il y a souvent des problèmes de connexion sur le site, ne pas hésiter à réessayer (erreur 503).

http://www.gutenberg.org/ebooks/41211.txt.utf-8 (la Comédie Humaine, Balzac)
http://www.gutenberg.org/ebooks/41211.txt.utf-8 (Proust, Du Côté de chez Swann)
http://www.gutenberg.org/ebooks/18921.txt.utf-8 (Eugène Sue, Les Mystères de Paris)
http://www.gutenberg.org/ebooks/798.txt.utf-8 (Stendhal, le Rouge et le Noir)
http://www.gutenberg.org/ebooks/17949.txt.utf-8 (Tolstoï, Guerre et paix tome I)
http://www.gutenberg.org/ebooks/14287.txt.utf-8 (Jules Verne, L'Île Mystérieuse)
http://www.gutenberg.org/ebooks/5154.txt.utf-8 (Zola, La Bête Humaine)
Nettoyer à la main le texte, en enlevant à la fin les notes et la licence, et au début un éventuel avant-propos, introduction, etc… Puis supprimer ponctuation et minuscules dans le texte avec la fonction de la question 3.

Travail à faire avec ou sans la question 3 : chercher les mots de longueur inférieure ou égale à 10 les plus fréquents dans le texte.
S'il y a égalité :

- Version facile : on choisit arbitrairement
- Version plus compliquée : on les citera tous.

5. Facultatif et pas compliqué. Compléter en cherchant les mots le(s) moins fréquent(s). S'il y a égalité, on les sélectionnera tous. Plus compliqué : afficher juste le(s) plus court(s) et le(s) plu(s) long(s)

6. Partie complémentaire, facultative, pas très compliquée et assez fascinante.
Cette partie donne une approche de l'analyse sémantique des textes par un programme (traitement automatisé du langage). Il s'agît de faire « comprendre » par la machine de quoi parle le texte. Pour le tout début de cette analyse, on procède en trois étapes (on n'ira pas plus loin que cette initiation !)

- 1ère étape : vous l'avez déjà faite, c'est la question 4 : le dictionnaire des mots et de leur fréquence
- 2ème étape, interchangeable avec la première : nettoyer le texte (si fait avant la 1ère étape) ou bien le dictionnaire de tous les « mots vides », c'est-à-dire des mots qui n'apportent pas de sens au texte, comme les articles « le », « la », les verbes « est », « a » etc…
- 3ème étape : rechercher les correspondances du texte avec des champs lexicaux. Les champs lexicaux sont des champs de vocabulaire se rapportant à une même notion. Par exemple, vous avez à disposition des champs lexicaux autour de la révolte, de la grève, de l'amour, du désir, du charbon et de la mer. Remarque : le champ lexical du charbon a été choisi de préférence à « mine », qui renvoie également à face ou grimace, et à « mineur », qui renvoie à « majeur », « enfant » etc…

Pour réaliser cette analyse, vous compléterez le programme « analyse_semantique.py ». Pour que ce programme fonctionne, il est nécessaire de télécharger tous les fichiers texte (en « .txt ») utilisés dedans. Vous les mettrez dans le même dossier que le programme.
Ce programme crée sept listes :

- une liste de mots vides : `mots_vides`
- six listes de champs lexicaux : `cl_revolte` `cl_greve` `cl_amour` `cl_desir` `cl_charbon` `cl_mer`

Utiliser ces listes pour réaliser un programme qui facilite l'analyse sémantique pour un utilisateur humain.
Remarque : suivant votre IDE, vous pouvez écrire un nouveau programme où, aux deux premières lignes, il y aura : `from analyse_semantique import *` et `from programme_question_5 import *`.
On importe alors le contenu des fichiers en question dans le programme. Cela évite une trop grande lourdeur lorsque l'on tape le programme. Attention à l'emplacement de ces fichiers dans l'arborescence suivant l'IDE, et attention à ne pas mettre l'extension .py. Il est aussi possible qu'il faille exécuter ces fichiers avant de les utiliser par ailleurs.
Puis, après cette analyse réalisée, lisez le roman et concluez sur les aptitudes des IA à comprendre un roman !
Vous pouvez trouver d'autres champs lexicaux sur https://www.rimessolides.com/motscles.aspx .
Il faut les mettre en forme dans un fichier texte, par exemple sous notepad++ en remplaçant tous les « \t » par des « \n » (mode de remplacement étendu), ou bien sinon à la main, c'est rapide.