2023-2024

TP 4

1°) Créer « à la main » une page Web comportant plusieurs lignes.
Créer des scripts Python qui permettent de lire (avec test) et d'afficher le contenu de cette page suivant les contraintes suivantes :

a) en tant que fichier texte, d'un seul tenant
b) en tant que fichier texte, ligne par ligne
c) en tant que fichier texte, toutes les lignes en une seule instruction de lecture
d) en tant que fichier binaire

2°) Dans cet exercice, nous allons modifier la précédente page Web. D'abord, « à la main », créer une copie de sauvegarde de cette page. Par programmation, ajouter une ligne supplémentaire dans la page Web juste avant </body>.
Contrainte : vous êtes libre du mode d'ouverture du fichier mais vous devez minimiser le nombre d'ouverture / fermeture du fichier.

Note : la méthode s = e.replace(ancienne_valeur, nouvelle_valeur, nb)
copie « e » dans « s » en échangeant « ancienne_valeur » par « nouvelle_valeur » « nb » fois (opt).
« e » et « s » peuvent être des chaînes de caractère ou des séquences binaires (mais « e », « s », « ancienne_valeur » et « nouvelle_valeur » doivent être du même type).

3°) Créer « à la main » un fichier texte comportant plusieurs lignes, d'un total d'au moins 30 caractères (sans caractères spéciaux car en UTF-8 obligatoirement) et incluant le mot « supprimer ».
Créer un script qui ouvre le fichier en mode binaire et :

a) lit et affiche tout sauf les 5 premiers et 5 derniers octets
b) lit et affiche tout sauf le mot « supprimer »

le tout en n'ouvrant qu'une seule fois le fichier et en n'utilisant que seek(), read() et index() (cf ci-dessous)
Rappel 1 : seek() retourne une position. C'est aussi la taille (en octets) du « morceau » de fichier depuis le début du fichier jusqu'à la position donnée par seek()
Rappel 2 : un saut de ligne sous Windows = « \r\n », soit 2 octets

Note : la méthode pos = e.index(valeur, d, f)
retourne la 1ère occurrence (la position) de « valeur » dans la chaîne de caractère ou séquence binaire « e ». « d » (opt) et « f » (opt) limitent la recherche entre le « d » ième et le « f » ième caractère ou octet de « e ».

4°) Soient S1 et S2, 2 scripts Python indépendants.

- ✔ S1 demande à l'utilisateur de saisir une chaîne puis l'insère (de force, cf. note ci-dessous) dans le fichier « test.txt » ainsi : 1 saisie = 1 ligne. Ceci en boucle, tant que « fin » n'est pas saisi. « test.txt » est initialement inexistant.
  Attention : input() n'inclut pas le(s) caractère(s) de fin de chaîne.
- ✔ S2 lit et affiche le contenu de « test.txt », ligne par ligne (donc en boucle), à intervalle régulier (grâce à une temporisation de 2s) jusqu'à ce que la lecture donne « fin ».

Lancer un 1er processus S1 (= P1_1). Lancer un processus S2 (= P2). Utiliser P1_1 et observer le résultat au niveau de P2. Sans arrêter P1_1 et P2, lancer et utiliser un 2nd processus S1 (= P1_2). Est-ce que vos saisies dans P1_2 sont prises en compte dans P2 ? Insister. Une explication ?

Note : par défaut, .write() n'écrit pas instantanément dans le fichier. Au contraire, les données sont placées dans un tampon d'E/S associé au fichier et au processus, tampon géré par l'O.S. Les données ne sont écrites dans le fichier que lorsque le tampon est plein (grande valeur) ou lorsqu'on ferme le fichier. Il existe cependant une méthode, .flush(), qui force le système à vider ce tampon immédiatement et donc à écrire « réellement » dans le fichier. Attention toutefois : aucune garantie car les supports de stockage ont un cache recevant les données du tampon avant la réelle écriture sur le dit support. Les données peuvent être perdues à ce moment là.

5°) Reprendre 4°) en ajoutant un « verrou » sous la forme d'un fichier « .lock ».
Plus précisément, lorsque S1 doit écrire dans « test.txt », il doit vérifier l'absence du fichier « .test.txt.lock ». S'il est présent, S1 doit retenter plus tard (toutes les 0,5 s, et en avertissant l'utilisateur). S'il est bien absent, S1 :

1. crée « .test.txt.lock »
2. écrit dans « test.txt » et avertit l'utilisateur
3. attend 1 s (pour bloquer un peu S2)
4. supprime « .test.txt.lock »

Pour S2, la démarche est semblable, à savoir : avant de lire dans « test.txt », S2 doit tester la présence de « .test.txt.lock ». S'il est présent, S2 doit retenter plus tard (toutes les 0,5 s, et en avertissant l'utilisateur). Sinon, S2 crée « .test.txt.lock », lit dans « test.txt » et affiche, attend 1 s (pour bloquer un peu S1), puis supprime « .test.txt.lock ».

Observer le fonctionnement avec un processus S1 et un processus S2, puis lancer en plus un 2nd processus S1. Est-ce que le verrou supprime le problème de 4°) ?