TP : lecture/écriture de fichiers et le tri par sélection

Un mémo : lecture/Ecriture de fichiers

En pratique, on utilise assez peu les fonctions print et input depuis un clavier ou vers l'écran : si on veut traiter une grande quantité de données (par exemple pour faire une étude statistique pour un TIPE...), ces données sont en général stockées dans des fichiers ad-hoc, qu'on veut pourvoir lire pour ensuite en manipuler le contenu. Il est assez maladroit de copier le contenu du fichier dans un script Python, il vaut mieux séparer les données du script qui les transforme. En particulier, même si les données changent (à la suite d'une nouvelle série de mesures, par exemple), le script Python reste identique. Après manipulation, on peut ensuite vouloir réécrire nos données transformées vers un autre fichier. Cette sous-section décrit la manipulation des entrées/sorties vers des fichiers, qui existent en dehors de notre programme Python.

Du point de vue du programmeur, un fichier ouvert en lecture doit être vu comme un tube (pipe en anglais) par lequel arrivent des données extérieures chaque fois que le programme les demande, de même qu'il faut voir un fichier ouvert en écriture comme un tube par lequel s'en vont les données que le programme envoie. Remarquez qu'ainsi, le clavier ou l'écran ne sont que des tubes particuliers.

Pour les fonctions Python, un fichier est une séquence de caractères : une fois qu'un fichier a été ouvert par un programme, celui-ci maintient un marqueur (fictif) à la position courante, qui indique à tout moment où sera lu/écrit le prochain octet. Toute opération de lecture ou d'écriture fait bouger ce pointeur vers l'avant.

Voici les principales fonctions pour le traitement des fichiers. Dans ce qui suit, f désigne un tube, qu'on pourra considérer comme étant un fichier ouvert en lecture ou en écriture.

fonctions	description
f=open(nom_du_fichier,'r')	Ouvre le fichier nom_de_fichier (donné sous la forme d'une chaîne de caractères indiquant son emplacement) en lecture (r comme read). Le fichier doit exister et seule la lecture est autorisée.
f=open(nom_du_fichier,'w')	Ouvre le fichier nom_de_fichier en écriture (w comme write). Si le fichier n'existe pas, il est créé, sinon il est écrasé (vidé avant utilisation).
f=open(nom_du_fichier,'a')	Ouvre le fichier nom_de_fichier en ajout (a comme append). Identique au mode 'w', sauf que si le fichier existe, il n'est pas écrasé et ce qu'on écrit est ajouté à partir de la fin du fichier.
f.close()	Sur un fichier ouvert comme précédemment, le ferme. Cette ligne est impérative pour les fichiers ouverts en écriture, puisque le fichier n'est réellement écrit complètement qu'à la fermeture.
f.read()	Lit tout le fichier d'un coup et le renvoie sous forme de chaîne de caractères (à ne réserver qu'aux fichiers de taille raisonnable).
f.readlines()	Pareil que précédemment, mais le résultat est une liste de chaînes de caractères, avec un élément de la liste par ligne. Attention, le saut de ligne $\backslash n$ est présent à la fin de la ligne.
f.readline()	Lit une unique ligne du fichier et la renvoie sous forme de chaîne (avec $\backslash n$ au bout). Le curseur de lecture (virtuel !) est positionné au début de la ligne suivante.
f.write(s)	Écrit la chaîne s à la suite du fichier.
f.writelines(T)	Écrit l'ensemble des éléments de T dans le fichier f comme des lignes successives. T est une liste, une séquence, un tuple... bref, un itérable
rstrip('$\backslash n$ $\backslash r$')	pour éviter les sauts de ligne. ex : f1.readline().rstrip('$\backslash n$ $\backslash r$)

Un fichier ouvert en lecture peut-être vu comme un itérable : on peut parcourir ses lignes avec une boucle for sans passer par readlines :

```
f=open('nom_du_fichier','r')
for ligne in f :
        [instructions]
```

Téléchargez le fichier classe.txt sur le site de la classe. Si ce n'est pas déja fait, créez un répertoire dédié au TP, positionnez l'archive dedans et dézippez là, et changez le répertoire de travail avec os dont on rappelle l'utilisation :

```
import os
os.chdir('chemin\vers\repertoire')
```

Changez les backslashs (comme ci-dessus) en slashs (/) si vous avez des soucis avec les caractères spéciaux comme $\backslash t$

ou $\backslash n$.

Premiers tests

Le fichier classe.txt les "nom, prénom " des MPSI2.

1. Tester les scripts suivants :

```
fichier = open ("classe.txt", "r")
print (fichier.readline ())
print (fichier.readline ())
fichier.close ()
print (fichier.readline ())
```

2. Afficher la liste des "noms, prénom" de la classe.
3. Lorsqu'on évalue une chaîne de caractères sous la forme d'un booléen, cette chaîne est évaluée comme True si elle est non vide et False si elle est vide. Il est donc possible de détecter si on est en fin de fichier.
   Tester le script (n'hésitez pas à rajouter .rstrip('$\backslash n$ $\backslash r$'))

```
fichier = open ("classe.txt", "r")
a = fichier.readline ()
while a :
        print a,
        a = fichier.readline ()
fichier.close ()
```

   En déduire le nombre d'élèves de la MPSI2.
4. Tester le script suivant :

```
fichier = open ("classe.txt", "r")
a = fichier.read (1)
while a :
        print a
        a = fichier.read (1)
fichier.close ()
```

   adapter le script pour afficher le nom du n-ième élève.

Petit exercice

---

1. Génération des notes. Nous allons choisir les notes au hasard. La fonction randint du module random prend en entrée deux entiers a et b et retourne un entier aléatoire de l'intervalle [a, b].
   (a) Écrire une fonction genere_note() sans argument, retournant une note entre 0 et 20. Celle-ci sera entière.
   (b) Écrire une fonction genere_notes(f1,f2) prenant en entrée un fichier ouvert en lecture et un en écriture, tel que si les lignes de f1 sont de la forme nom,prenom, la fonction écrive dans le fichier f2 des lignes de la forme nom,prenom,note1 avec note1 une note générée aléatoirement.
   Testez votre script avec le fichier classe.txt pour produire un fichier notes.txt.
2. Classement par ordre alphabétique.
   (a) Le tri par sélection :
      i. Ecrire une fonction donnant l'indice du minimum d'une liste.
      ii. Ecrire une fonction permettant d'échanger les indices i et j d'une liste.
      iii. Proposer une fonction tri permettant de trier une liste à l'aide de la méthode du tri par sélection.
   (b) Tester le script :

```
fichier = open ("notes.txt", "r")
print(fichier.readlines())
fichier.close ()
```

   Afficher les résultats de la classe par ordre alphabétique.