I) Cours

Un programme Python peut accéder au contenu d'un fichier sur le disque. On s'intéresse ici au cas des fichiers contenant du texte.

1) Fichier sur le disque $\longleftrightarrow$ objet Python

On ouvre un fichier avec la commande `open`.

a) Lecture

```
fichier_python=open('fichier_disque.txt','r')
```

Les arguments de la fonction `open` sont de type `str`.
`fichier_python` est un flux, qui accède au contenu du fichier sur le disque (`fichier_disque.txt`) en lecture (`'r'` comme *read*). Il ne contient pas le contenu du fichier : python se contente de créer un lien vers le fichier `fichier_disque.txt`, en prévenant le système d'exploitation : « attention, je suis entrain de lire ce fichier ».

b) Écriture

```
nouveau_python=open(nouveau_disque.txt','w')
```

`nouveau_python` est un flux, qui va écrire dans le fichier (`nouveau_disque.txt`) sur le disque (`'w'` comme *write*). S'il y a déjà quelque chose dans `nouveau_disque.txt`, tout est effacé.

c) Autres option de `open`

Il existe des options `'a'` (comme *append*) qui écrit en fin de fichier, et `'r+'` qui ouvre le fichier en lecture et écriture.

d) Clôture

Quand un fichier est ouvert (en lecture ou en écriture), le système d'exploitation bloque un certain nombre d'action. Par exemple, si ce fichier est sur une clé USB, vous ne pouvez pas retirer la clé (c'est entre autre pour cette raison qu'il faut toujours retirer « proprement » les clé USB).
Donc, lorsque vous avez fini d'utiliser un fichier, il faut toujours fermer le flux qui accède au fichier sur le disque :

```
fichier_python.close()
nouveau_python.close()
```

Le lien est fermé : vous ne pouvez plus accéder à `fichier_disque.txt` via `fichier_python.txt`. Idem pour écrire dans `nouveau_disque.txt` via `nouveau_python`.

e) `with`

Une alternative aux instructions `open` / `close` consiste à utiliser `with`.

```
fichier_python=open('fichier_disque.txt','r')
[bloc d'instruction]
fichier_python.close()
[suite du programme]
```

est équivalent à

```
with open('fichier_disque.txt','r') as fichier_python:
   [bloc d'instruction]
[suite du programme]
```

Cette rédaction évite tout oubli de fermeture du fichier : dès la sortie du bloc indenté, python ferme `fichier_python`.

2) Manipulation de l'objet Python

a) Lecture

`fichier_python.readline()`
Lit la première ligne disponible, et la retourne sous forme de `str` (techniquement : lit le fichier jusqu'à la première occurrence de `'\n'`, symbole de fin de ligne).
La ligne est « consommée » : désormais `fichier_python` pointe sur la ligne suivante.
`fichier_python.readlines()`
Lit l'ensemble des lignes et retourne une liste de toutes les lignes du fichier.
Lorsque `fichier_python` a atteint la fin du fichier texte, `fichier_python.readline()` retourne une `str` vide : `''`.
Pour d'autres commandes (`fichier_python.seek()` par exemple), cf. la documentation en ligne.

b) Écriture

`nouveau_python.write('Test\n')`
Retourne le nombre de caractères écrits dans le fichier.

II) Exercices

Exercice 1

1) Ouvrir un fichier texte que vous aurez préalablement créé avec un éditeur de texte (par exemple notepad, le fichier doit contenir au moins 2 lignes). Lire les deux premières lignes et les stocker dans `ligne1` et `ligne2`

2) Sauver ces deux lignes dans un nouveau fichier.

3) Effacer l'ensemble des espaces dans le fichier texte de départ, sur le disque.

Exercice 2

1) Ouvrir le fichier `colles_ptsi_2013_2.csv`, et en extraire la liste des groupes de colles.

2) Extraire toutes les informations du fichier et les stocker dans une liste (à deux dimensions : `[[1,2],[3,4]]`).

3) Écrire un programme qui trouve l'ensemble des colles (nom du colleurs, horaire et jour de la semaine) correspondant à un groupe de colle.

4) Écrire un programme qui calcule la date de la colle à partir de la case où est affichée la colle.

5) Écrire un programme qui donne l'ensemble des colles à venir, à partir d'un nom et d'une date.