Manipulation de fichiers

Un fichier peut-être de deux types : binaire ou texte. S'il est de type binaire, il faut une librairie particulière pour le décoder (cas des images, des sons, des vidéos ou des données compressées). On en verra un exemple dans le cas des images. Dans le cas des fichiers textes, on peut lire et écrire les données directement.

1 — Ouverture, lecture et fermeture

On ouvre un fichier avec la commande `open()`, avec en paramètre le nom du fichier : par exemple `f = open("catilinaires.txt")`. La variable `f` est un objet qui permet d'accéder au contenu du fichier.
Pour lire une seule ligne du fichier, on utilise la méthode `readline()` : `f.readline()` lit une ligne du fichier, déplace un curseur virtuel au début de la ligne suivante et enfin renvoie la ligne qui a été lue.
Pour lire le fichier ligne par ligne, on utilise une boucle. Par exemple, pour afficher le contenu du fichier précédent :

```
f = open("catilinaires.txt")
for ligne in f:
    print(ligne)
```

Notez la forme particulière de la boucle, typique de la gestion de fichiers. La commande `for` s'occupe de gérer le curseur, de s'arrêter à la fin du fichier, etc. Au début de chaque itération, la variable de boucle `ligne` contient la ligne lue.
À la fin d'un programme, il est important de fermer le fichier avec la méthode `close()` : `f.close()`. `Python` écrit alors les dernières informations utiles sur le disque dur et « libère » le fichier pour le système d'exploitation.

2 — Manipulation des données

La plupart du temps dans un fichier de données chaque enregistrement occupe une ligne. Les données elle-mêmes sont séparées par des virgules, des tabulations ou des espaces. Il faut donc traiter chaque ligne du fichier pour en extraire les données. Par exemple les premières lignes du fichier `cathedral.txt` ressemblent à

```
nom style   haut    long
Bath    goth    75  225
Bristol goth    52  300
Canterb rom 80  522
```

On comprend qu'il faut ignorer la première ligne et que les données sont séparés par des tabulations. Chaque ligne est une chaîne de caractère qui doit être découpée en morceaux, et chaque morceau doit être converti suivant le type désiré.
Le découpage se fait avec la méthode `split("\t")` avec en paramètre le caractère de séparation (ici `"\t"` pour une tabulation, mais ça peut aussi être une virgule ou un point). La méthode `split()` renvoie une liste contenant les différents morceaux.

```
f = open("cathedral.txt")
l1 = f.readline() # lecture de la ligne d'entête
l2 = f.readline()
l2
'Bath\tgoth\t75\t225\n'
info = l2.split("\t")
info
['Bath', 'goth', '75', '225\n']
(nom, type, hauteur, longueur) = [info[0], info[1], int(info[2]),
nom
'Bath'
longueur
225
```

L'unité de longueur n'est pas précisée : il s'agit du pied (1 pied $\simeq$ 0,30 m).
Il est alors facile de traiter les données lues. Établissons par exemple la liste des cathédrales gothiques et celle des cathédrales romanes.

```
f = open("cathedral.txt")
LRoman = []
LGothic = []
f.readline()
for ligne in f:
    info = ligne.split("\t")
    (nom, style, hauteur, longueur) = [info[0], info[1],
                                       int(info[2]), int(info[3])]
    if style == 'goth':
        LGothic.append(nom)
    else:
        LRoman.append(nom)

print("Il y a", len(LGothic),"cathédrales gothiques : ",
LGothic)
print("Il y a", len(LRoman),"cathédrales romanes : ", LRoman)
```

Ex. 1 — Calculer la hauteur et la largeur moyenne des cathédrales gothiques et romanes enregistrées dans `cathedral.txt`

3 — Écriture d'un fichier

Pour écrire dans un fichier, il faut d'abord l'ouvrir avec l'une des options `'a'` ou `'w'`. L'option `'a'` (`'a'` comme `'append'`) ouvre le fichier et place le curseur à la fin : les nouvelles données seront donc ajoutées à la fin du fichier. Avec l'option `'w'` (comme `'write'`) , le contenu du fichier est effacé et le curseur est placé au début : les nouvelles données remplacent les anciennes.
On écrit ensuite des données dans un fichier avec la méthode `write`.

```
f = open('tmp.txt', 'w')
f.write("Bonjour tout le monde !\n")
f.write("Comment allez-vous ? \n")
f.close()
f = open('tmp.txt')              # Attention : pas d'options !
f.readlines()                    # Renvoie le contenu de f
                                 # sous forme d'une liste

f.close()
```

Il est ici crucial de bien fermer le fichier avec `close()`. En effet, pour limiter les accès au disque dur, `Python` n'écrit pas les données au fur et à mesure de l'exécution du programme. Il attend d'avoir suffisamment de données à écrire pour le faire. Ainsi, à la fin de l'exécution d'un script, il est possible que quelques données doivent encore être écrites : la méthode `close()` se charge en particulier de ce travail.

Ex. 2 — En utilisant le fichier `cathedral.txt`, écrire un fichier `roman.txt` qui contient la liste des cathédrales romaines, ainsi que leur hauteur et leur longueur. Les données seront séparées par des virgules.

4 — Traitement de données

Ex. 3 — 1. Écrire une fonction `Moyenne(L)` et une fonction `EcartType(L)` qui calculent la moyenne et l'écart-type des réels de la liste `L`. Pour rappel

$$\overline{x} = \frac{1}{n}\sum_{k=1}^{n} x_k \qquad \text{et} \qquad \overline{\sigma_x}^2 = \frac{1}{n}\sum_{k=1}^{n}(x_k - \overline{x})^2$$

2. Dans une étude publié en 1962, une équipe de chercheur a relevé l'angle et la largeur de l'édéage de trois espèces d'altise (un petit coléoptère ravageur de culture). Les trois espèces considérées sont *concinna* (Con), *heikertingeri* (Hei), et *heptapotamica* (Hep). Ces mesures figurent dans le fichier `FleaBeetles.txt`.
   a) Ouvrir le fichier et observer le format des données.
   b) Calculer, pour chaque espèce, la moyenne et l'écart-type des angles et de la largeur de l'édéage.
   c) Afficher les bornes des intervalles de confiances à 5%. Ils sont définis comme $[\,\overline{x} - 1{,}96\,\overline{\sigma_x}\,;\,\overline{x} + 1{,}96\,\overline{\sigma_x}\,]$. Se chevauchent-ils ? Peut-on conclure ?

Ex. 4 — Régression linéaire

1. Écrire une fonction **`Covariance(X,Y)`** qui renvoie la covariance pour deux séries de données X et Y. Pour rappel
$$\mathrm{cov}_{x,y} = \frac{1}{n}\sum(x_i - \overline{x})(y_i - \overline{y})$$
2. Écrire une fonction **`RegLineaire(X,Y)`** qui renvoie la pente, l'ordonnée à l'origine et le coefficient de corrélation linéaire pour deux séries de données. Pour rappel, dans un modèle d'approximation $Y = aX + b$ avec la méthode des moindres carrés, on trouve
$$a = \frac{\mathrm{cov}_{x,y}}{\overline{\sigma_x}^2} \qquad b = \overline{y} - a\overline{x} \qquad r^2 = \frac{\mathrm{cov}_{x,y}}{\overline{\sigma_x}\,\overline{\sigma_y}}$$
3. Dans l'objectif d'étalonner un chromatographe, on a mesuré la réponse à 4 échantillons contenant différentes quantité d'un même produit (le tout en unité arbitraires). Les résultats figurent dans le fichier **`chromatograph.txt`**.
   a) Ouvrir le fichier et observer le format des données.
   b) On veut tester si une hypothèse linéaire peut ici interpréter la réponse du chromatographe.
   Représenter sur un même graphe les données relevées (sous forme de point) et la droite de régression linéaire.
   Pensez-vous que l'hypothèse linéaire soit ici pertinente ?

Ex. 5 — Transformation des données Si une simple régression linéaire ne peut permettre de prédire un lien entre deux séries de données X et Y, il est fréquent de transformer chaque série et de tester ensuite une régression linéaire.
Par exemple, on peut transformer étudier la relation entre $\ln(Y)$ et $\ln(X)$. Si l'hypothèse linéaire semble fiable, alors de $\ln(Y) = a\ln(X) + b$ on tire $Y = CX^a$ (dépendance algébrique).
Le fichier **`misc.txt`** contient 3 séries de données présentées en 6 colonnes. Les couples de variables étudiées sont
- **`NBR`** nombre de bateaux construits dans un chantier naval et **`HRS`** milliers d'heures de travail par bateau ;
- **`DENS`** densité du trafic sur une autoroute et **`KMH`** vitesse moyenne observée sur le même autoroute ;
- **`DIST`** longueur d'une course d'athlétisme et **`SEC`** temps du record du monde en 1969.

Pour chaque couple de variables, tester une hypothèse de régression linéaire.
Si elle est insatisfaisante, transformer les données afin de trouver le modèle minimisant le coefficient de régression linéaire. Les transformations possibles sont $\ln(X)$, $\ln(Y)$, $\ln(X)$ *et* $\ln(Y)$, $1/X$, $1/Y$.