Informatique XII
Lecture & écriture dans un fichier

Laurent Kaczmarek

PCSI^2 2013-2014
Lycée Louis Le Grand

Lundi 3 février 2013

Utilisation du système de fichiers

- Le système d'exploitation gère l'ensemble des fichiers enregistrés dans la mémoire.
- Tout environnement de développement (cf Pyzo pour Python par exemple) permet de sauvegarder des programmes dans un fichier.
- Sous Python, comme dans tout langage de programmation, l'utilisateur peut accèder à des fichiers qui ne contiennent pas des programmes afin de les ouvrir et d'en traiter les données. Par exemple, des fichiers contenant des images, de la video, du son, des données numériques brutes, du texte, etc.
- Il peut également écrire dans un fichier existant ou créer des fichiers afin d'enregistrer le résultat de ses programmes.

Structure d'un fichier

Principe et exemple

- Les données d'un fichier sont traitées par Python comme des chaînes de caractères regroupées en lignes.
- Dans la suite de ce cours, nous manipulerons le fichier `ADN.txt` qui s'affiche comme suit dans un éditeur de texte quelconque :
- ```
  AGAATGCGTAGTCTGCCA
  ATCGGTACGTGTCGCAT
  ACGTACTGA
  GTCAAACCTTG
  GTCAGATAT
  ```
  ```

Lecture dans un fichier

Principe

- Indiquer le chemin d'accès au fichier.
- Ouvrir le fichier en mode écriture.
- Lire les lignes successivement ou en une seule fois.
- Fermer le fichier.

Spécification du chemin d'accès

- Si le fichier n'est pas dans le répertoire courant de l'environnement de développement, il faudra indiquer à quel endroit le trouver au moyen de la fonction `chdir` (*change directory*) de la bibliothèque `os` (*operating system*) :
- ```
  from os import chdir
  chdir("/home/toto/tppython")
  ```
  ```

Lecture d'un fichier

Ouverture du fichier

- On utilise la fonction `open` pour associer un objet de type `file` à un fichier existant.
- L'option `'r'` (pour *read*) permet de spécifier une ouverture en mode lecture.
- Exemple : `fichier=open('donnees.txt','r')`

Lecture des lignes du fichier

- Python lit un fichier ligne par ligne.
- Attention, la fonction `open` ne crée pas une liste : on ne peut accéder directement à une position connue du fichier sans traitement supplémentaire.

Lecture d'un fichier

Lecture des lignes du fichier

- La lecture peut se faire pas à pas par la méthode `readline()`, qui lit la ligne courante du fichier puis qui passe à la suivante : `fichier.readline()`.
- La lecture peut se faire globalement par la méthode `readlines()`, qui permet des itérations au moyen d'une boucle for : `fichier.readlines()`.

Exemple 1

- Considérons le programme suivant :
- ```
  fichier=open('ADN.txt','r')
  l1=fichier.readline()
  t=fichier.readlines()
  print(t[0])
  ```
- Le résultat renvoyé est : `ATCGGTACGTGTCGCAT`
  ```

Exemple 1

- Attention : les méthodes `readline()` et `readlines()` agissent comme des pointeurs vers la partie du fichier qui n'a pas été encore lue.
- Il est donc plus prudent d'enregistrer *initialement* tout le fichier dans une liste au moyen de `readlines()` avant de traiter les données : on encombre certes la mémoire mais on manipule alors une liste ce qui est recommandé aux programmeurs débutants.
- On retiendra que l'appel suivant `t=fichier.readlines()` crée la liste `t` des lignes de `fichier` considérées comme des chaînes de caractères.
- La balise `\n` indique la fin d'une ligne.

Exemple 2

Considérons le programme suivant :

- ```
  fichier=open('ADN.txt','r')
  t=fichier.readlines()
  print(t)
  ```
- Il a pour résultat :
  ```
  ['AGAATGCGTAGTCTGCCA\n',
  'ATCGGTACGTGTCGCAT\n',
  'ACGTACTGA\n', 'GTCAAACCTTG\n',
  'GTCAGATAT\n']
  ```
- Devinez ce que renvoie le code suivant :
  ```
  n=len(t[2])
  print(t[2],n,t[2][n-2])
  ```
- Réponse : `'ACGTACTGA\n',10,'A'`
  ```

Découpage d'une ligne

- La méthode `split` appliquée à une chaîne de caractère permet de la *couper* à chaque occurrence d'un caractère choisi par l'utilisateur :
- Le programme suivant...
```
t1=t[2].split('A')
print(t[2],t1)
```
- ...a pour résultat :
```
'ACGTACTGA\n',['','CGT','CTG','\n']
```
- On utilise souvent cette méthode lorsque les données sont écrites en lignes et séparées par des virgules :
- Devinez ce que renvoie :
```
'34,4,5'.split(',')
```
- `['34','4','5']`

Fermeture du fichier

- Il est recommandé de fermer le fichier tout de suite après sa lecture par les méthodes `readline()` ou `readlines()`.
- En effet, certains fichiers particulièrement volumineux pourraient encombrer la mémoire inutilement.
- On utilise pour cela la méthode `close()`.
- La syntaxe est la suivante :

```
fichier.close()
```

Écriture dans un fichier

Principe

- Indiquer le chemin d'accès au fichier.
- Ouvrir le fichier en mode écriture.
- Ajouter des lignes au fichier.
- Fermer le fichier.

Ouverture en mode écriture

- On utilise la fonction `open` avec l'option `'a'` (*append*) pour ajouter à un fichier existant ou avec l'option `'w'` (*write*) pour créer un nouveau fichier.
- Attention : l'ouverture en écriture avec l'otion `'w'` d'un fichier existant aura pour effet d'écraser le fichier.
- Exemple : `fichier=open('donnees.txt','w')`

Écriture dans un fichier

Ajout d'une ligne

- On utilise la méthode `write` sans oublier la balise `\n` si l'on souhaite passer à la ligne au prochain ajout.
- Exemple : `fichier.write('blabla\n')`

Exemple

- On ajoute la ligne `GTTACCAGC` au fichier `ADN.txt` :
- Programme :

```
fichier=open('ADN.txt','a')
fichier.write('GTTACCAGC')
fichier.close()
fichier=open('ADN.txt','r')
t=fichier.readlines()
print(t[len(t)-1])
```

- Résultat : `GTTACCAGC`