Chaînes de caractères ; Slicing ; Fichiers externes

1 Chaînes de caractères

1.1 Nature des chaînes de caractères

Les chaînes de caractères (string) sont des tuples spécialisés, qui ne contiennent que des caractères (char).

Comme tous les tuples ils ne sont pas modifiables, ce qui constitue leur principale différence avec les listes. Certains opérations spécifiques existent néanmoins.

Les chaînes de caractères s'écrivent entre guillemets ou entre apostrophes. Il est possible de mettre des guillemets à l'intérieur d'apostrophes et réciproquement.

Ainsi, a='Il dit : "Bonjour !"' est une chaîne de caractères valide en Python.

1.2 Opérations sur les chaînes

Concaténer deux chaînes de caractères consiste à les mettre l'une après l'autre. Ceci est possible à l'aide de l'opérateur + en Python.

Par exemple :

```
S="Bonjour"
T=" a tous !"
U=S+T
print(U) # affiche Bonjour a tous
```

Attention au caractère fortement typé de Python qui ne permet pas de concaténer une chaîne de caractères et un entier. Ainsi "Tu as"+3+"points" renvoie une erreur ; "Tu as"+str(3)+"points" est correct en revanche.

1.3 Fonctions et méthodes générales sur les chaînes de caractères

Si l'accès à un caractère fonctionne de la même manière sur les chaînes de caractères que sur les listes, la plupart des méthodes sont différentes, voire n'auraient pas de sens pour les listes.

Retenons les méthodes suivantes :

1. `len(S)` : fonctionne de la même manière que pour les listes
2. `S.upper()` : transforme une chaîne de caractères en la même chaîne composée de majuscules
3. `S.lower()` : transforme une chaîne de caractères en la même chaîne composée de minuscules

2 Recherche d'un mot dans une chaîne de caractères

Nous avons vu au cours précédent des algorithmes de recherche d'un élément dans une liste. Nous allons désormais étudier la recherche d'un mot dans une chaîne de caractères (problème qui est équivalent à la recherche de plusieurs éléments consécutifs dans un tableau). Il existe des algorithmes performants résolvant ce problème et qui s'appuient sur la théorie des automates finis et des expressions régulières ; nous nous contenterons d'un algorithme simple et intuitif.

Notre problème consiste donc à chercher un *mot* dans une *Chaîne de caractères*. Nous allons parcourir la chaîne de caractères. Pour chaque caractère donné nous allons comparer si les len(Mot) caractères qui suivent coïncident dans la chaîne et le mot.

Concrètement nous allons utiliser une boucle imbriquée. La boucle extérieure sera parcourue jusqu'à ce qu'il ne soit plus possible pour des questions de longueur que "mot" soit dans chaîne :

Chaine[0]	Chaine[1]	...	Chaine[F]	Chaine[F+1]	...	Chaine[len(Chaine)-1]
			mot[0]	mot[1]	...	mot[len(Mot)-1]

Nous parcourons donc Chaine jusqu'à F tel que :

$$\text{len(Chaine)} - 1 - F = \text{len(Mot)} - 1 - 0$$

donc

$$F = \text{len(Chaine)-len(Mot)}$$

La boucle intérieure sera une boucle conditionnelle qui sera parcourue tant que les caractères de mot ne sont pas épuisés et que les caractères de mot et ceux testés de Chaine coïncident. Si tout mot est parcouru on renverra la position du caractère où on a comparé Chaine et mot ; si ceci n'arrive jamais on renverra None. En Python cela donne :

```
def recherche_mot(Chaine,Mot):
   for i in range(1+len(Chaine)-len(Mot)):
      j=0
      while j<len(Mot) and Mot[j]==Chaine[i+j]:
         j=j+1
      if j==len(Mot):
         return i
   return None
```

Exercice 1. Quel est la complexité de cet algorithme au pire ?

3 Slicing

La technique de "slicing", ou découpage en français, fonctionne sur les listes et les chaînes de caractères. Elle permet de créer une nouvelle liste, ou une nouvelle chaîne de caractères, extraite de la première, entre deux indices à préciser.

La syntaxe générale est :

`L[a:b]`

où :

- `L` est une chaîne de caractères ou une liste
- `a` a est l'indice de la borne inférieure, inclue. `L[a]` appartient à `L[a:b]`
- `a` b est l'indice de la borne inférieure, exclue. `L[b]` n'appartient pas à `L[a:b]`

Trois variantes de cette construction existent :
- `L[a:]` crée la liste ou chaîne de caractères construite avec les éléments à partir de a et jusqu'au dernier élément.
- `L[:b]` crée la liste ou chaîne de caractères construite avec les éléments à partir du premier élément jusqu'à b exclu.
- `L[:]` crée une copie de la liste ou de la chaîne de caractères.

Exercice 2. On a `s="abcdefghijklmonopqrstuvwxyz"`.
Que valent `s[4:7]` ? `s[:5]` ? `s[24:]` ? `s[26:]` ?

Exercice 3. Écrire une fonction `divise(L)` qui prend en argument une liste `L` et qui renvoie deux listes constituée de la première moitié des éléments de `L` pour la première et des autres éléments de `L` (dans l'ordre de L) pour la seconde. Si L est de longueur impaire, la seconde liste sera plus longue d'un élément.

4 Fichiers textes externes

4.1 Présentation générale

Un fichier externe est un fichier qui a un certain formatage et qui peut contenir des données éventuellement issues depuis des moyens d'acquisitions divers et qui peuvent être réutilisées d'une session à l'autre et par d'autres programmes.

De manière générale en Python, on travaillera sur les fichiers externes de la manière suivante :

1. On ouvre le fichier dans un premier temps. On précise le nom du fichier avec **son chemin d'accès** et le **mode d'accès**
2. On lit ou on écrit des données dans ce fichier
3. On **ferme** le fichier une fois qu'on a terminé de travailler avec.

4.2 Ouverture et fermeture de fichiers ; Chemin d'accès

Les fichiers s'ouvrent en associant un objet de type *file* à un fichier :

mon_fichier=open(Chemin,par)

où Chemin est le chemin d'accès du fichier et par le paramètre d'ouverture du fichier.

Le chemin d'accès peut être relatif au répertoire dans lequel on travaille (c'est à proscrire surtout sur un réseau) ou absolu : on indique alors le chemin à partir de la racine en Windows, entre guillemets, et en utilisant le simple slash(/) pour l'arborescence.

Par exemple, sur un ordinateur personnel, le chemin d'accès d'un fichier stocké dans "Mes Documents" serait typiquement :

`'C:/Users/Toto/Mes documents/fichier.txt'` (l'extension .txt est un exemple..)

Sur le réseau du lycée cela pourra être :

`"U:/Python/Projet1/fichier.txt'`

Il est souvent plus simple, lorsque c'est possible de mettre le script Python et le fichier externe dans le même répertoire. Le chemin d'accès est alors constitué du seul nom de fichier : par exemple si on souhaite travailler sur le fichier "poeme.txt", enregistré dans le même répertoire de travail que le script, on écrira :

```
f=open("poeme.txt",'r')
```

Le paramètre indique ce que l'on fait sur le fichier :
– 'r' pour lecture du fichier ('r' pour **read**)
– 'w' pour une écriture depuis un fichier vide ('w' comme **write**)
– 'a' pour une écriture à la fin d'un fichier non vide ('a' comme **append**)

Après la lecture du fichier, on ferme le fichier avec la méthode **close**.

Un exemple de lecture du fichier "adn.txt" stocké dans le répertoire courant avec stockage de la chaîne lue dans la variable `s` :

```
fichier = file.open('adn.txt','r')
s=fichier.read()
fichier.close()
```

4.3 Lecture de fichier texte

Après avoir un fichier en mode lecture, on va lire les différentes lignes du fichier soit avec la méthode `read` sur les objets fichiers, soit à l'aide de méthodes plus complexes comme les méthodes `readline` ou l'itérable `readlines`

Concrètement, on peut lire, en une seule ligne, le contenu d'un fichier texte à l'aide de l'instruction :

```
s=f.read()
```

`s` est alors une chaîne de caractères, que l'on peut traiter, notamment avec les méthodes `split` sur les chaînes de caractères.

Un exemple de lecture du fichier "adn.txt" stocké dans le répertoire courant avec stockage de la chaîne lue dans la variable `s` :

```
fichier = file.open('adn.txt','r')
s=fichier.read()
fichier.close()
```

4.4 Écriture dans un fichier externe

Le principe est le même. Si on a ouvert le fichier avec le paramètre 'w' celui-ci sera écrasé si il existait déjà et l'écriture se fera à partir d'un fichier vide. Si on a ouvert le fichier avec le paramètre 'a' l'écriture se fera à la fin du fichier.

L'écriture se fait avec la méthode **write**. De même que pour la lecture, il faudra fermer le fichier avec la méthode **close**. Son oubli serait catastrophique, l'écriture effective ne s'effectuant que lors de la fermeture du fichier.

Un exemple d'écriture d'une chaîne de caractères s dans un fichier :

```
fichier = file.open('masortie.txt','w')
fichier.write(s)
fichier.close()
```

5 Méthode split sur les chaînes de caractères

Les fichiers textes que nous seront amenés à exploiter seront souvent formatés d'une certaine manière lors de leur création, par des logiciels externes qui les créeront, dédiés à des sciences expérimentales.

Le problème de récupérer les données sous une forme exploitable par Python, qui permettra de traiter ensuite ses données se posera naturellement. La méthode `split` sur les chaînes de caractères nous permettra d'effectuer la première partie du travail, à savoir se débarrasser des délimiteurs apparaissant dans le fichier externe.

`s.split(c)` prend en argument une chaîne de caractères `s` et renvoie une liste de chaînes de caractères, où les éléments sont les sous-éléments de `s`, créées par séparation selon les caractères `c` successifs apparaissant.

Par exemple si `s="rouge,bleu,vert,jaune"`
`s.split(",")` renvoie la liste : `["rouge","bleu","vert",jaune"]`

Si `s` est la chaîne de caractères codées sur plusieurs lignes suivante :

```
s=0
1.26
2.57
3.89
```

Comme \n est la caractère qui représente le passage de la ligne,
`s.split("\n")` renvoie la liste : `["0","1.26","2.57","3.89"]`

Si l'on souhaite traiter cette liste comme une liste de nombres, on convertira (à l'aide d'un parcours de la liste créé par une boucle) tous les éléments de cette liste en flottants.

Exercice 4. Dans le cadre d'une expérience en électricité, on relève la tension aux bornes d'un dipôle toutes les secondes. Celle-ci est stockée dans un fichier `"tension1.elc"` sur une seule ligne, où les tensions mesurées sont séparées par un point-virgule. Par exemple le fichier pourrait être : "0 ;0.5 ;0.63"

Écrire une séquence d'instructions :
- qui ouvre le fichier et récupère son contenu dans une chaîne de caractères ;
- crée une liste des différentes tensions mesurées ;
- calcule la moyenne de la tension aux bornes du dipôle durant l'expérience.

Exercice 5. Un nom de fichier externe est toujours de la forme nom.extension où nom est une chaine de caractères ne contenant pas de "." et extension est une chaîne de caractères.

Ainsi un nom de fichier valide est "vacances2678.jpg"

Écrire une fonction `transforme(s)` qui prend en argument une chaîne de caractères `s` représentant un nom de fichier valide et qui renvoie la chaîne de caractères représentant un nom de fichier légal, avec la même extension, et de sorte à ajouter la chaîne de caractères "_test" à la fin du nom du fichier.

6 Cas particulier des images

Les images se traitent légèrement différemment des fichiers quelconques. On dispose en effet d'un module **Image** qui fournit un ensemble de fonctions spécialisées aux images.

Comme pour les autres fichiers l'ouverture se fait à l'aide d'une fonction **open** :

```
monimage = Image.open(chemin.png)
```

On peut afficher une image à l'aide de la méthode `show`

```
monimage.show()
```

Ses dimensions sont données par l'attribut **size** (sans parenthèses, il ne s'agit pas d'une méthode) ; L'attribut **mode** renvoie 'RGB' pour une image en couleurs et 'L' pour une image en niveaux de gris.

```
monimage.size monimage.mode
```

Pour effectuer un traitement de l'image il faut la décomposer en pixels. Pour cela avec le module numpy on transforme directement une image en tableau :

`tableau = numpy.array(monimage)` Chaque pixel est représenté par un entier entre 0 et 255 pour une image en niveaux de gris, ou par un tableau de trois entiers entre 0 et 255 pour une image en couleurs.

On dispose d'une fonction qui permet de créer une image à partir du tableau de ses pixels.

```
imagenew = Image.fromarray(tableau)
```

Enfin, pour enregistrer une image, on utilise la méthode save. L'extension donnée au nom du fichier détermine le format d'enregistrement, qui n'est pas forcément le même que celui de l'image d'origine.

```
imagenew.save('toto.jpg')
```