Les listes et chaînes de caractères

1. Conversion entre listes et chaînes

La méthode pour sauvegarder des informations dans un fichier utilisant les fonctions `write()` et `readline()` ne vous permet pas de sauvegarder des structures plus complexes comme les listes. Nous aimerions pouvoir écrire : f.write([4,5,6]), mais ce n'est pas possible. Déjà, la fonction `write()` n'accepte pas les nombres. Certes, on peut trouver une solution en convertissant chaque nombre en chaîne de texte avec la fonction `str()` avant l'écriture et on peut utiliser la fonction `int()` pour convertir la chaîne correspondante en nombre lors de la lecture.

Cependant, il y a beaucoup à écrire dans un programme et lorsque les données commencent à devenir complexes et nombreuses, on a envie que les choses se passent simplement et rapidement. Ainsi, lorsqu'on a une liste par exemple, on aimerait avoir une fonction qui convertisse son contenu, quel qu'il soit, c'est-à-dire des chaînes comme des chiffres, en une chaîne de caractères. De plus, il faudrait que cette fonction gère automatiquement la quantité d'éléments présents dans la liste ainsi que le cas particulier des listes vides. Et comme bien souvent en Python, cela existe déjà ! Le package json nous fournit des méthodes permettant de compacter des données quelconques dans une chaîne (les pros disent "sérialiser"). Pour utiliser ces fonctions, il faut importer le package json en début de programme. Un package représente un groupe de fonctions accessibles à travers la syntaxe `json.mafonction()`. Voici un exemple :

```
import json
L = [ 4, 6.0, "TOTO", ["SOLEIL", 99,-3.14]]
print(L)
var = json.dumps(L)
print ("JSON : ",var)
print(type(var))
newL = json.loads(var)
print(newL)
print(type(newL))
```

Ce qui produit comme résultat :

```
[4, 6.0, 'TOTO', ['SOLEIL', 99, -3.14]]
JSON : [4, 6.0, "TOTO", ["SOLEIL", 99, -3.14]]
<class 'str'>
[4, 6.0, 'TOTO', ['SOLEIL', 99, -3.14]]
<class 'list'>
```

Pour ce test, nous avons fait tout ce qui était interdit, car nous avons mis tout et n'importe quoi dans une même liste : des entiers, des flottants et des chaînes. Et nous avons ajouté une liste à l'intérieur de la liste ! Avec cela, on va savoir ce que le package json a dans le ventre ! La fonction `dumps()` permet de convertir une liste en une chaîne de caractères. Même si l'affichage ne trahit aucune conversion, le test `print(type(var))` nous indique bien que c'est un type string. En fait, json a fait une conversion liste → chaîne en créant une chaîne de caractères qui correspondrait à celle que l'on taperait au clavier en mode interactif pour créer cette liste. Ceci explique donc la similarité d'affichage entre les deux premiers `print()`. Une fois la chaîne de texte json construite, nous la décodons dans l'autre sens pour construire la variable `newL`. Cette variable est bien une liste correspondant exactement à la liste originale `L`. Victoire !

La sérialisation permet d'attaquer le problème de sauvegarde d'une liste dans un fichier. Dès que vous allez vouloir sauvegarder des informations ou les recharger, vous allez arriver dans ces problématiques de traitement et le

package json vous fournira des solutions simples et efficaces.

Le package json fournit des fonctions permettant la lecture/écriture directement depuis un fichier, contournant ainsi l'usage des fonctions de lecture/écriture `read()` et `write()` traditionnelles. Par contre, vous devrez toujours ouvrir et fermer votre fichier avec les fonctions `open()` et `close()`. L'objet fichier généré sera transmis aux fonctions : `json.dump(L,f)` et `json.load(f)` pour savoir où lire/écrire les données.

2. Fractionner une chaîne de caractères

Il existe une fonction très pratique qui s'applique à travers une chaîne de caractères. Son objectif consiste à découper la chaîne originale en sous-chaînes chaque fois qu'elle rencontre un caractère donné. Ainsi, cette fonction retourne une liste de chaînes composées de différentes parties de la chaîne originale. Attention, bien que la description soit ambiguë, la fonction `split()` ne modifie pas la chaîne originale.

Prenons un exemple :

```
Phrase = "Il fera beau demain"
Lmots = Phrase.split(" ")
for mot in Lmots :
  print(mot)
print(Phrase)
```

La fonction `split()` fractionne la phrase donnée en coupant la chaîne chaque fois qu'elle rencontre un caractère espace. Dans cet exemple, le résultat récupéré correspond à la liste des mots :

```
Il
fera
beau
demain
Il fera beau demain
```