IPT : Cours 6

Les itérables en Python (1h)

MPSI : Prytanée National Militaire

Pascal Delahaye

16 janvier 2016

1 Les types composés en Python

Cette section est consacrée à la présentation des *types composés* suivants :

1. Les listes
2. Les n-uplets
3. Les Chaînes de caractères

Ces types composés sont appelés des "itérables" en Python.
Comme l'indique le tableau suivant, ils partagent entre eux un grand nombre de fonctions communes :

Itérables	**Fonctions communes**	**Spécificités**
Les listes	• `len(L)` : longueur • `L1 + L2` : concaténation • `[]` : liste vide • `L[i]` : élément d'indice "i" • `L[a:b:p]` : liste partielle d'éléments • `4 in [1,4,6,2]` : test d'appartenance • `L1 == L2` : test d'égalité	• Mutable : `L[i]=a` • Possibilité de créer des tableaux de plusieurs dimensions • Composantes de types éventuellement différents • `L.append(x)` : ajout d'un élément en queue de liste
Les tuples	• `len(T)` : longueur • `T1 + T2` : concaténation • `(,)` : tuple vide • `T[i]` : élément d'indice "i" • `T[a:b:p]` : liste partielle d'éléments • `4 in (1,4,6,2)` : test d'appartenance • `T1 == T2` : test d'égalité	• Non mutable • `a,b,c = 1,4,2` : affectation multiple • `a,b = b,a` : échange de valeurs • Composantes de types éventuellement différents
Les chaînes de caractères	• `len(C)` : longueur • `C1 + C2` : concaténation • `""` : chaîne vide • `C[i]` : caractère d'indice 'i' • `C[a:b:p]` : liste partielle d'éléments • `"toto" in "vive toto"` : test d'appartenance • `C1 == C2` : test d'égalité	• Non mutable

2 Compléments sur le type "liste"

DÉFINITION 1 : **Les Listes**
Les listes se présentent sous la forme d'une suite ordonnée de données séparées par des virgules et encadrées par des crochets.
Elles vérifient les caractéristiques et fonctionnalités suivantes :

1. Les données ne sont pas nécessairement du même type
2. Les listes sont *dynamiques* : la fonction `append()` permet d'ajouter une donnée en queue de liste
3. Les listes sont *indexées* : on a accès à tout moment à n'importe quelle donnée de la liste.
4. Les listes sont *mutables* : il est possible de modifier les différentes composantes d'une liste.

`Exemple 1.` `[1,5.67,"toto",True]` est une liste de 4 éléments de 4 types différents.

Remarque 1. Le type `array` de `numpy` :
La structure de *tableau*, proposée par la bibliothèque NUMPY (appelée `array`) est une variante du type liste.

1. Contrairement aux listes, les éléments d'un tableau sont tous du même type
2. Numpy propose de nombreuses fonctions permettant la manipulation des tableaux :
   (a) l'insertion d'éléments
   (b) le tri
   (c) la réinitialisation
   (d) l'utilisation comme pile
   (e) le comptage d'éléments
   (f) l'élimination d'éléments

Il est possible de transformer une liste en "tableau numpy" à l'aide de la fonction `array()`.

Remarque 2. Même si en apparence, les listes sont des structures linéaires, elles peuvent servir à la représentation d'un tableau de dimension $n \geq 2$. Pour cela, il suffit de construire une liste dont les données sont des listes correspondant dans l'ordre aux lignes du tableau.

`Exemple 2.` Utiliser une structure de liste pour stocker le tableau suivant : $T = \begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4&5&6\\\hline 7&8&9\\\hline\end{array}$

On récupère alors les données du tableau avec l'instruction : T[i][j]
Sauriez-vous construire une procédure permettant de construire un tableau du même type contenant tous les entiers de $[\![1, n^2]\!]$?

Remarque 3. Récupération des éléments d'une liste L :

- Nous avons vu que nous pouvions utiliser la syntaxe `L[k]` pour récupérer la composante L_k.
- Nous pouvons aussi utiliser la commande `a1,a2,..., an = L` lorsque L admet n composantes.

3 Le type itérable

DÉFINITION 2 : **Le type itérable**
En fait, les 3 types précédents (listes, uplets et chaînes de caractères) sont des cas particuliers d'un type plus général qu'on appelle un *itérable*.

Remarque 4. Par défaut, un itérable n'a ni la structure de liste, ni de uplet, ni de chaîne de caractères. On peut cependant facilement convertir n'importe quel itérable en une liste ou un tuple grâce aux fonctions `list()` et `tuple()`.

DÉFINITION 3 : **La fonction range()**

`range(a,b,p)` renvoie un itérable contenant les entiers de a à b (non compris) par pas de p.

Cet objet peut alors être transformé :
- soit en une liste : avec la commande `list()`,
- soit en un tuplet : avec la commande `tuple()`

```
>>> list(range(1,11))       # renvoie la liste contenant les entiers de 1 à 10
>>> tuple(range(1,11))      # renvoie le n-uplet contenant les entiers de 1 à 10
```

Remarque 5. Python contient aussi la fonction `enumerate()` qui elle-aussi renvoie un itérable.

Remarque 6. Une particularité intéressante de Python :
Dans les boucles "FOR", l'indice de boucle prend ses valeurs dans un itérable.

Comme le montre les exemples suivants, cela permet de faire varier cet indice sur des données de nature très différentes...

Python

```
L = []                          L = ""                              C1 = []
for i in range(1,11):           for i in "toto est malade":         for L in [[1,2],[3,4]]
    L = L.append(2*i)               L = L + i + i                       C1 = C1.append([L[1]])
print(L)                        print(L)                            print(C1)
```