Représentation des données : Types Construits

Sylvain Gibaud

29 Novembre 2017

Plan

Definition

Une liste en Python est un objet **MUTABLE** dans laquelle on peut mettre plusieurs variables.

Pour créer une liste vide on fait :

FIGURE – Liste Vide

```
liste = []
```

Pour créer une liste non vide on fait :

FIGURE – Liste Non vide

```
liste = [1,2,3]
```

Pour avoir et/ou modifier un élément d'une liste on fait :

FIGURE – Élement d'une liste

```
>>> L = ["A","B","C"]

>>> L[1]
'B'

>>> L[1] = "D"

>>> L
['A', 'D', 'C']
```

Pour convertir un objet (compatible en liste) on fait :

FIGURE – Convertir en Liste

```
>>> x = 1,2,3

>>> x
(1, 2, 3)

>>> y = list(x)

>>> y
[1, 2, 3]
```

Propriété

Comme une liste est une collection de variables une bonne question est de savoir si un élément est dans une liste.
Pour savoir si 3 est dans liste on fait :

FIGURE – *Tester si un item est dans une liste*

```
3 in liste
```

Le résultat est un booléen.

Propriété

On peut aussi "additionner" des listes. En fait on les concatène (i.e. on les met les l'une après l'autre). Par exemple :

FIGURE – *Addition de Listes*

```
>>> L1 = [1,2,3]

>>> L2 = [4,5,6]

>>> L1 + L2
[1, 2, 3, 4, 5, 6]
```

Comme une multiplication par un entier est une addition répétée, on peut multiplier des listes.

Propriété (Boucles sur les listes)

On peut réaliser des boucles sur des listes. On fait alors pour liste :

FIGURE – *Boucle sur Liste*

```
>>> L = [1,3.14,"Babar","papa","MAMAN"]

>>> for item in L:
...     StrItem = str(item)
...     STRITEM = StrItem.upper()
...     print("l'item est : " + StrItem)
...     print("L'ITEM EST : " + STRITEM)
...
...
l'item est : 1
L'ITEM EST : 1
l'item est : 3.14
L'ITEM EST : 3.14
l'item est : Babar
L'ITEM EST : BABAR
l'item est : papa
L'ITEM EST : PAPA
l'item est : MAMAN
L'ITEM EST : MAMAN
```

Inspiration

Cette partie est très amplement inspirée de deux sites web : Python Doctor (pour avoir les bases claires) et Sam & Max (pour avoir plus d'intuition et des conseils plus durables)

Supprimer une entrée avec un index

Il est parfois nécessaire de supprimer une entrée de la liste. Pour cela vous pouvez utiliser la fonction `del` .

```
>>> liste = ["a", "b", "c"]
>>> del liste[1]
>>> liste
['a', 'c']
```

Supprimer une entrée avec sa valeur

Il est possible de supprimer une entrée d'une liste avec sa valeur avec la méthode `remove` .

```
>>> liste = ["a", "b", "c"]
>>> liste.remove("a")
>>> liste
['b', 'c']
```

Compter le nombre d'items d'une liste

Il est possible de compter le nombre d'items d'une liste avec la fonction `len` .

```
>>> liste = [1,2,3,5,10]
>>> len(liste)
5
```

Compter le nombre d'occurences d'une valeur

Pour connaitre le nombre d'occurences d'une valeur dans une liste, vous pouvez utiliser la méthode `count` .

```
>>> liste = ["a","a","a","b","c","c"]
>>> liste.count("a")
3
>>> liste.count("c")
2
```

Trouver l'index d'une valeur

La méthode `index` vous permet de connaitre la position de l'item cherché.

```
>>> liste = ["a","a","a","b","c","c"]
>>> liste.index("b")
3
```

Manipuler une liste

Voici quelques astuces pour manipuler des listes:

```
>>> liste = [1, 10, 100, 250, 500]
>>> liste[0]
1
>>> liste[-1] # Cherche la dernière occurence
500
>>> liste[-4:] # Affiche les 4 dernières occurrences
[500, 250, 100, 10]
>>> liste[:] # Affiche toutes les occurences
[1, 10, 100, 250, 500]
>>> liste[2:4] = [69, 70]
[1, 10, 69, 70, 500]
>>> liste[:] = [] # vide la liste
[]
```

Copie de Liste

Propriété

La copie de liste n'est pas chose aisée. Il faut être vigilant à la dépendance entre les listes. Regardez l'exemple suivant :

FIGURE – *Dépendance Liste*

```
>>> x = [1,2,3]

>>> x
[1, 2, 3]

>>> y = x

>>> y
[1, 2, 3]

>>> y[0] = 4

>>> y
[4, 2, 3]

>>> x
[4, 2, 3]
```

Proposition

Pour créer une copie indépendante d'une liste donnée on utilise la fonction deepcopy du package copy.

FIGURE – Utilisation de deepcopy

```
>>> from copy import *

>>> x = [1,2,3]

>>> y = deepcopy(x)

>>> y
[1, 2, 3]

>>> y[0] = 4

>>> y
[4, 2, 3]

>>> x
[1, 2, 3]
```

Plan

Definition

Un dictionnaire est un ensemble de couples { clé : définition }. Une clé est un string,un nombre ou un t-uplet permettant de retrouver la définition. La définition peut être n'importe quel objet (Liste, str, int, objet que vous avez défini). Les couples sont séparés par des virgules.

Exemple

Dico1 = { "un" :1,2 :"deux","trois" :[3,3,3] }
Dico1["trois"] $\rightarrow$ [3,3,3]

Comment récupérer les clés d'un dictionnaire python par une boucle?

Pour récupérer les clés on utilise la méthode *keys* .

```
>>> fiche = {"nom":"Wayne", "prenom":"Bruce"}
>>> for cle in fiche.keys():
...     print cle
...
nom
prenom
```

Comment récupérer les valeurs d'un dictionnaire python par une boucle?

Pour cela on utilise la méthode *values* .

```
>>> fiche = {"nom":"Wayne", "prenom":"Bruce"}
>>> for valeur in fiche.values():
...     print valeur
...
Wayne
Bruce
```

Comment récupérer les clés et les valeurs d'un dictionnaire python par une boucle?

Pour récupérer les clés et les valeurs en même temps, on utilise la méthode *items* qui retourne un `tuple` .

```
>>> fiche = {"nom":"Wayne","prenom":"Bruce"}
>>> for cle,valeur in fiche.items():
...     print cle, valeur
...
nom Wayne
```

Comment vérifier la présence d'une clé dans un dictionnaire python?

Vous pouvez utiliser la méthode *haskey* pour vérifier la présence d'une clé que vous cherchez:

```
>>> a.has_key("nom")
True
```

Comment ajouter des valeurs dans un dictionnaire python?

Pour ajouter des valeurs à un dictionnaire il faut indiquer une clé ainsi qu'une valeur:

```
>>> a = {}
>>> a["nom"] = "Wayne"
>>> a["prenom"] = "Bruce"
>>> a
{'nom': 'Wayne', 'prenom': 'Bruce'}
```

Vous pouvez utiliser des clés numériques comme dans la logique des listes .

Comment supprimer une entrée dans un dictionnaire python?

Il est possible de supprimer une entrée en indiquant sa clé, comme pour les listes:

```
>>> del a["nom"]
>>> a
{'prenom': 'Bruce'}
```

Comment créer une copie indépendante d'un dictionnaire python?

Comme pour toute variable, vous ne pouvez pas copier un dictionnaire en faisant *dic1 = dic2* :

```
>>> d = {"k1":"Bruce", "k2":"Wayne"}
>>> e = d
>>> d["k1"] = "XXX"
>>> e
{'k2': 'Wayne', 'k1': 'XXX'}
```

Pour créer une copie indépendante vous pouvez utiliser la méthode copy :

```
>>> d = {"k1":"Bruce", "k2":"Wayne"}
>>> e = d.copy()
>>> d["k1"] = "XXX"
>>> e
{'k2': 'Wayne', 'k1': 'Bruce'}
```

Comment fusionner des dictionnaires python?

La méthode update permet de fusionner deux dictionaires .

```
>>> a = {'nom': 'Wayne'}
>>> b = {'prenom': 'bruce'}
>>> a.update(b)
>>> print(a)
{'nom': 'Wayne', 'prenom': 'Bruce'}
```

Comment utiliser des tuples comme clé dans un dictionnaire python?

Une des forces de python est la combinaison tuple/dictionnaire qui fait des merveilles dans certains cas comme lors de l'utilisation de coordonnées.

```
>>> b = {}
>>> b[(32)]=12
>>> b[(45)]=13
>>> b
{(4, 5): 13, (3, 2): 12}
```

Plan

Definition

Un t-uplet est une liste qui ne peut plus être modifié.

A quoi sert un tuple alors?

Le tuple permet une affectation multiple:

```
>>> v1, v2 = 11, 22
>>> v1
11
>>> v2
22
```

Il permet également de renvoyer plusieurs valeurs lors d'un appel d'une fonction:

```
>>> def donne_moi_ton_nom():
...     return "olivier", "engel"
...
>>> donne_moi_ton_nom()
('olivier', 'engel')
```

On utilisera un tuple pour définir des sortes de constantes qui n'ont donc pas vocation à changer.

Afficher une valeur d'un tuple

Le tuple est une sorte de liste, on peut donc utiliser la même syntaxe pour lire les données du tuple.

```
>>> mon_tuple[0]
1
```

Plan

Plan