

## Les conteneurs

Chaînes de caractères  
Listes

# Programmation Python Partie 1 : Les conteneurs: chaînes de caractères et listes

Alexandre Gramfort

Alexandre Gramfort

## Les conteneurs

### Chaînes de caractères Listes

## Chaines de caractères (Strings)

```
In [45]: s = 'Bonjour Telecom ParisTech!'
          # ou avec " "
          s = "Bonjour Telecom ParisTech!"
          print(s)
          print(type(s))
```

```
Bonjour Telecom ParisTech!
<type 'str'>
```

## Les conteneurs

### Chaînes de caractères Listes

## Accéder à un élément ou une sous-chaine

L'accès à un élément : *variable[indice]*

**Attention:** les indices commencent à 0!

```
In [46]: s[0] # premier élément
```

```
Out[46]: 'B'
```

```
In [47]: s[-1] # dernier élément
```

```
Out[47]: '!'
```

## Les conteneurs

### Chaînes de caractères Listes

On peut extraire une sous-chaine avec la syntaxe `[start:stop]`, qui extrait les caractères entre `start` et `stop` (**exclu**):

```
In [48]: s[0:7]
```

```
Out[48]: 'Bonjour'
```

```
In [49]: start, stop = 1, 7  
print(len(s[start:stop]))  
print(stop - start)
```

```
6
```

```
6
```

## Les conteneurs

### Chaînes de caractères Listes

On peut omettre `start` ou `stop`. Dans ce cas les valeurs par défaut sont respectivement 0 et la fin de la chaîne.

```
In [50]: s[:7]  # 7 premières valeurs
```

```
Out[50]: 'Bonjour'
```

```
In [51]: s[8:]  # de l'entrée d'indice 8 à la fin
```

```
Out[51]: 'Telecom ParisTech!'
```

```
In [52]: print(len(s[8:]))  
print(len(s) - 8)
```

```
10  
18
```

```
In [53]: s[-10:]
```

```
Out[53]: 'ParisTech!'
```

## Les conteneurs

### Chaînes de caractères Listes

Il est aussi possible de définir le `step` (pas d'avancement) avec la syntaxe `[start:stop:step]` (la valeur par défaut de `step` est 1):

```
In [54]: s = "ababababab"  
         s[0::2]
```

```
Out[54]: 'aaaaa'
```

```
In [55]: s[1::2]
```

```
Out[55]: 'bbbbbb'
```

Cette technique est appelée *slicing*. Pour en savoir plus:

<http://docs.python.org/release/2.7.3/library/functions.html?highlight=slice#slice>

et

<http://docs.python.org/2/library/string.html>

## Les conteneurs

### Chaînes de caractères Listes

# Mise en forme de chaînes de caractères

```
In [56]: print("str1", 1.0, False, -1j) # print convertit toutes les variables en chaînes
```

```
('str1', 1.0, False, -1j)
```

```
In [57]: print("str1" + "str2" + "str3") # pour concatener +
```

```
str1str2str3
```

```
In [58]: print("str1" * 3)
```

```
str1str1str1
```

## Les conteneurs

Chaînes de caractères

Listes

```
In [59]: a = 1.0
print("val = %e" % a)      # comme en C (cf. printf)
print("val = %f" % a)      # comme en C (cf. printf)
print("val = %s" % a)      # comme en C (cf. printf)
```

```
val = 1.000000e+00
val = 1.000000
val = 1.0
```

```
In [60]: # Plus avancé
s = "val1 = %.2f, val2 = %d" % (3.1415, 1.5)
print(s)
```

```
val1 = 3.14, val2 = 1
```



## Les conteneurs

Chaînes de caractères  
**Listes**

### Listes

Les listes sont très similaires aux chaînes de caractères sauf que les éléments peuvent être de n'importe quel type.

La syntaxe pour créer des listes est `[...]`:

```
In [61]: l = [1, 2, 3, 4]  # création de liste avec []  
         print(l)  
         print(type(l))  
  
[1, 2, 3, 4]  
<type 'list'>
```

## Les conteneurs

Chaînes de caractères  
Listes

# Accès à un élément et à une sous-liste:

**Attention:** On commence à indexer à 0!

```
In [62]: l[0]
```

```
Out[62]: 1
```

```
In [63]: l[-2]
```

```
Out[63]: 3
```

```
In [64]: print(l[1:3])  
         print(l[:2])
```

```
[2, 3]
```

```
[1, 3]
```

## Les conteneurs

Chaînes de caractères  
**Listes**

On peut mélanger les types:

```
In [65]: l = [1, 'a', 1.0, 1-1j]
          print(l)
```

```
[1, 'a', 1.0, (1-1j)]
```

On peut faire des listes de listes:

```
In [66]: liste_de_listes = [1, [2, [3, [4, [5]]]]]
          liste_de_listes
```

```
Out[66]: [1, [2, [3, [4, [5]]]]]
```

## Les conteneurs

### Chaînes de caractères Listes

La fonction `range` pour générer une liste d'entiers:

```
In [67]: start, stop, step = 10, 30, 2  
         print(range(start, stop, step))  
         print(list(range(start, stop, step))) # in Python 3
```

```
[10, 12, 14, 16, 18, 20, 22, 24, 26, 28]  
[10, 12, 14, 16, 18, 20, 22, 24, 26, 28]
```

```
In [68]: range(-10, 10)
```

```
Out[68]: [-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5,  
          6, 7, 8, 9]
```

Itération de  $n-1$  à 0

```
In [69]: n = 10  
         print(range(n - 1, -1, -1))
```

```
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]
```

## Les conteneurs

Chaînes de caractères  
Listes

Comment convertir une chaîne de caractères en liste

In [70]:

```
s = "zabcda"  
l = list(s)  
print(l)
```

```
['z', 'a', 'b', 'c', 'd', 'a']
```

## Les conteneurs

Chaînes de caractères

Listes

## Comment trier une liste

```
In [71]: l_sorted = sorted(l)  # renvoi une nouvelle liste
print(l)
print(l_sorted)
```

```
['z', 'a', 'b', 'c', 'd', 'a']
['a', 'a', 'b', 'c', 'd', 'z']
```

```
In [72]: l.sort()  # tri sans copie
print(l)
```

```
['a', 'a', 'b', 'c', 'd', 'z']
```

## Les conteneurs

Chaînes de caractères  
**Listes**

Attention `l.sort()` ne renvoie rien c'est-à-dire `None`

```
In [73]: out = l.sort()  
          print(out)
```

None

## Les conteneurs

Chaînes de caractères

Listes

### Ajouter, insérer, modifier et enlever des éléments d'une liste:

```
In [74]: # création d'une liste vide
l = [] # ou l = list()

# ajout d'éléments avec `append`
m = l.append("A")
l.append("d")
l.append("d")

print(m) # append ne renvoie rien
print(l)
```

```
None
['A', 'd', 'd']
```

```
In [75]: l.count('d')
```

```
Out[75]: 2
```



## Les conteneurs

Chaînes de caractères  
Listes

On peut modifier une liste par assignation:

```
In [76]: l[1] = "p"  
         l[2] = "p"  
         print(l)  
  
['A', 'p', 'p']
```

```
In [77]: l[1:3] = ["d", "d"]  
         print(l)  
  
['A', 'd', 'd']
```

## Les conteneurs

Chaînes de caractères  
Listes

### Insertion à un index donné avec `insert`

```
In [78]: l.insert(0, "i")
          l.insert(1, "n")
          l.insert(2, "s")
          l.insert(3, "e")
          l.insert(4, "r")
          l.insert(5, "t")

          print(l)

['i', 'n', 's', 'e', 'r', 't', 'A', 'd', 'd']
```

### Suppression d'un élément avec `remove`

```
In [79]: l.remove("A")
          l.remove("d")  # supprime uniquement le premier d
          print l

['i', 'n', 's', 'e', 'r', 't', 'd']
```

## Les conteneurs

Chaînes de caractères  
**Listes**

```
In [80]: 'n' in l  # tester la présence d'un élément dans une liste
```

```
Out[80]: True
```

```
In [81]: l.index('n')  # rechercher son indice dans une liste
```

```
Out[81]: 1
```

## Les conteneurs

Chaînes de caractères  
Listes

Suppression d'un élément à une position donnée avec `del`:

```
In [82]: del l[6]  
print l  
['i', 'n', 's', 'e', 'r', 't']
```

## Les conteneurs

Chaînes de caractères  
Listes

Concatenation de listes avec + et duplication avec \*

```
In [83]: l1 = [1, 2, 3]
          l2 = [4, 5, 6]
          print(l1 + l2)  # attention différent de l1.append(l2)
          print(l1 * 2)
```

```
[1, 2, 3, 4, 5, 6]
```

```
[1, 2, 3, 1, 2, 3]
```

help(list) pour en savoir plus.