

Les Listes Python

Chapitre 8



Python for Informatics: Exploring Information
www.pythonlearn.com



Une Liste est une sorte de Collection



- Une **collection** nous permet de mettre plusieurs valeurs dans une seule “**variable**”
- Une **collection** est pratique car elle nous permet de transporter **plein de valeurs** dans un unique paquet.

```
amis = [ 'Joseph', 'Glenn', 'Sally' ]
```

```
affaires = [ 'chaussettes', 'chemise', 'parfum' ]
```

Qu'est-ce qui n'est **pas** une "Collection"

La plupart de nos **variables** contiennent une valeur - lorsque nous ajoutons une autre valeur à notre **variable**, l'ancienne valeur est remplacée

```
$ python
Python 2.5.2 (r252:60911, Feb 22 2008, 07:57:53)
[GCC 4.0.1 (Apple Computer, Inc. build 5363)] on darwin
>>> x = 2
>>> x = 4
>>> print x
4
```

Les Constantes de Listes

- Les constantes de **listes** sont entourées par des crochets et les éléments des listes sont séparés par des virgules
- Une **liste** d'éléments peut être n'importe quel objet Python - même **une autre liste**
- Une **liste** peut être vide

```
>>> print [1, 24, 76]
[1, 24, 76]
>>> print ['red', 'yellow',
'blue']
['red', 'yellow', 'blue']
>>> print ['red', 24, 98.6]
['red', 24, 98.59999999999999994]
>>> print [ 1, [5, 6], 7]
[1, [5, 6], 7]
>>> print []
[]
```

Nous utilisons déjà des listes!

```
for i in [5, 4, 3, 2, 1] :  
    print i  
print 'Blastoff!'
```

5

4

3

2


1

Blastoff!

Les Listes et les boucles définies – les meilleures amies

```
friends = ['Joseph', 'Glenn', 'Sally']  
for friend in friends :  
    print 'Bonne année:', friend  
print 'Fini!'
```

Bonne année: Joseph
Bonne année: Glenn
Bonne année: Sally
Fini!





Effectuer des recherches dans les listes

Tout comme les chaînes, nous pouvons atteindre chaque élément d'une liste grâce à un index spécifié dans des **crochets**

Joseph	Glenn	Sally
0	1	2

```
>>> amis = [ 'Joseph', 'Glenn', 'Sally' ]
>>> print amis[1]
Glenn
>>>
```

Les Listes sont Mutables

- Les chaînes sont “**immuables**” – nous *ne pouvons pas* changer le contenu d’une chaîne– nous devons créer une **nouvelle chaîne** afin de pouvoir le modifier
- Les listes sont “**mutables**” – nous *pouvons changer* un élément d’une liste en utilisant l’opérateur **index**

```
>>> fruit = 'Banane'
>>> fruit[0] = 'b'
Traceback
TypeError: 'str' object does not
support item assignment
>>> x = fruit.lower()
>>> print x
banane
>>> lotto = [2, 14, 26, 41, 63]
>>> print lotto[2, 14, 26, 41, 63]
>>> lotto[2] = 28
>>> print lotto
[2, 14, 28, 41, 63]
```


Quelle est la Longueur d'une Liste?

- La fonction `len()` prend comme paramètre une **liste** et indique le nombre d'*éléments* présents dans la **liste**
- En fait `len()` peut nous indiquer le nombre d'éléments présents dans n'importe quel ensemble ou séquence (tel que les chaînes de caractères...)

```
>>> greet = 'Hello Bob'
>>> print len(greet)
9
>>> x = [1, 2, 'joe', 99]
>>> print len(x)
4
>>>
```

Utilisation de la fonction `range`

- La fonction `range` nous donne une liste de nombres qui varient entre zéro et un de moins que le paramètre
- Nous pouvons créer un index en boucle en utilisant `for` et un itérateur entier

```
>>> print range(4)
[0, 1, 2, 3]
>>> amis = ['Joseph', 'Glenn', 'Sally']
>>> print len(amis)
3
>>> print range(len(amis))
[0, 1, 2]
>>>
```

Le conte de deux boucles...

```
amis = ['Joseph', 'Glenn', 'Sally']

for ami in amis :
    print 'Bonne année:', ami

for i in range(len(amis)) :
    ami = amis [i]
    print 'Bonne année:', ami
```

```
>>> amis = ['Joseph', 'Glenn', 'Sally']
>>> print len(amis)
3
>>> print range(len(amis))
[0, 1, 2]
>>>
```

Bonne année: Joseph
Bonne année: Glenn
Bonne année: Sally

Concaténer des listes à l'aide du +

- Nous pouvons créer une nouvelle liste en additionnant deux listes existantes

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print c
[1, 2, 3, 4, 5, 6]
>>> print a
[1, 2, 3]
```

Les Listes peuvent être **tranchées** à l'aide du :

```
>>> t = [9, 41, 12, 3, 74, 15]
>>> t[1:3]
[41, 12]
>>> t[:4]
[9, 41, 12, 3]
>>> t[3:]
[3, 74, 15]
>>> t[:]
[9, 41, 12, 3, 74, 15]
```

Souvenez-vous: *Tout*
comme avec les chaînes, le
deuxième nombre est
“**jusqu'à mais sans inclure**”

Méthodes des Listes

```
>>> x = list()
>>> type(x) <type 'list'>
>>> dir(x) ['append', 'count', 'extend', 'index',
'insert', 'pop', 'remove', 'reverse', 'sort']
>>>
```

<http://docs.python.org/tutorial/datastructures.html>

Créer une liste à partir de zéro

- Nous pouvons créer une liste et ensuite y ajouter des éléments en utilisant la méthode `append`
- La liste reste ordonnée et les nouveaux éléments sont ajoutés en queue de liste

```
>>> stuff = list()
>>> stuff.append('book')
>>> stuff.append(99)
>>> print stuff
['book', 99]
>>> stuff.append('cookie')
>>> print stuff
['book', 99, 'cookie']
```

Est-ce que c'est dans (in) une Liste?

- Python fournit deux **opérateurs** qui vous permettent de vérifier si un élément se trouve dans une liste
- Ce sont des opérateurs logiques qui nous donnent en réponse **True** ou **False**
- Ils ne modifient pas la liste

```
>>> some = [1, 9, 21, 10, 16]
>>> 9 in some
True
>>> 15 in some
False
>>> 20 not in some
True
>>>
```


Une **Liste** est une Séquence Ordonnée

- Une **liste** peut comprendre plusieurs éléments et les garder dans l'ordre jusqu'à ce que nous décidions d'en changer l'ordre
- Une **liste** peut être **triée** (i.e., son ordre peut être changer)
- La méthode **sort** (à la différence des chaînes) signifie “**trie/organise-toi**”

```
>>> friends = [ 'Joseph', 'Glenn', 'Sally' ]
>>> friends.sort()
>>> print friends
['Glenn', 'Joseph', 'Sally']
>>> print friends[1]
Joseph
>>>
```

Les Fonctions Internes et les Listes

- Il y a un nombre de **fonctions** dans **Python** qui permettent de définir des **listes** en tant que paramètres
- Vous souvenez-vous des boucles que nous avons contruites? C'est bien plus simple que cela.

```
>>> nums = [3, 41, 12, 9, 74, 15]
>>> print len(nums)
6
>>> print max(nums)
74
>>> print min(nums)
3
>>> print sum(nums)
154
>>> print sum(nums) / len(nums)
25
```

```
total = 0
count = 0
while True :
    inp = raw_input('Entrer un nombre: ')
    if inp == 'fini' : break
    value = float(inp)
    total = total + value
    count = count + 1

moyenne = total / count
print 'moyenne:', moyenne
```

Entrer un nombre: 3
Entrer un nombre: 9
Entrer un nombre: 5
Entrer un nombre: fini
Moyenne: 5.666666666667

```
numlist = list()
while True :
    inp = raw_input('Entrer un nombre: ')
    if inp == 'fini' : break
    value = float(inp)
    numlist.append(value)

moyenne = sum(numlist) / len(numlist)
print 'moyenne:', moyenne
```

Les Chaînes et les Listes : les meilleures amies

```
>>> abc = 'With three words'
>>> stuff = abc.split()
>>> print stuff
['With', 'three', 'words']
>>> print len(stuff)
3
>>> print stuff[0]
With
```

```
>>> print stuff
['With', 'three',
'words']
>>> for w in stuff :
...     print w
...
With
Three
Words
>>>
```

La méthode **Split** fractionne une chaîne en plusieurs morceaux et produit une liste de chaînes. Ces listes de chaînes sont alors à considérer comme étant des mots. Nous pouvons **accéder** à un mot ou à une **chaîne** en particulier en passant par tous les mots.

```
>>> line = 'A lot          of spaces'
>>> etc = line.split()
>>> print etc['A', 'lot', 'of', 'spaces']
>>>
>>> line = 'first;second;third'
>>> thing = line.split()
>>> print thing['first;second;third']
>>> print len(thing)
1
>>> thing = line.split(';')
>>> print thing['first', 'second', 'third']
>>> print len(thing)
3
>>>
```

- Lorsque vous ne spécifiez pas de **délimiteur**, les espaces multiples sont considérés comme étant *un* délimiteur
- Vous pouvez spécifier quel caractère **délimiteur** utiliser dans le **splitting (fractionnement)**

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

```
fhand = open('mbox-short.txt')
for line in fhand:
    line = line.rstrip()
    if not line.startswith('From ') : continue
    words = line.split()
    print words[2]
```

Sat
Fri
Fri
Fri
...

```
>>> line = 'From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008'
>>> words = line.split()
>>> print words
['From', 'stephen.marquard@uct.ac.za', 'Sat', 'Jan', '5', '09:14:16', '2008']
>>>
```

Le motif double Split

- Parfois nous fractionnons une ligne d'une façon, puis nous en sélectionnons une partie que nous allons fractionner davantage

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
words = line.split()  
email = words[1]
```

Le motif double Split

- Parfois nous fractionnons une ligne d'une façon, puis nous en sélectionnons une partie que nous allons fractionner davantage

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
words = line.split()
```

```
email = words[1]          stephen.marquard@uct.ac.za
```


Le motif double Split

- Parfois nous fractionnons une ligne d'une façon, puis nous en sélectionnons une partie que nous allons fractionner davantage

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
words = line.split()
email = words[1]
pieces = email.split('@')
stephen.marquard@uct.ac.za
['stephen.marquard', 'uct.ac.za']
```

Le motif double Split

- Parfois nous fractionnons une ligne d'une façon, puis nous en sélectionnons une partie que nous allons fractionner davantage

```
From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008
```

```
words = line.split()
email = words[1]
pieces = email.split('@')
print pieces[1]
```

```
stephen.marquard@uct.ac.za
['stephen.marquard', 'uct.ac.za']
'uct.ac.za'
```

Résumé

- Le concept de collection
- Les listes et les boucles définies
- Indexer et rechercher
- La mutabilité des listes
- Les fonctions: len, min, max, sum
- Le fractionnement des listes
- Les méthodes de listes: append, remove
- Trier/organiser des listes
- Fractionner des chaînes en listes de mots
- Utiliser la méthode split pour faire l'analyse de chaînes



Remerciements / Contributions



These slides are Copyright 2010- Charles R. Severance (www.dr-chuck.com) of the University of Michigan School of Information and open.umich.edu and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

Translation: Stéphanie Kamidian