

# Notions avancées

En construction. Les liens sont juste donnés. Une introduction et un exemple devrait être proposé pour chaque rubrique, et le nombre de ces rubriques augmenté.

## Itérateurs

### Générateurs et "yield"

- <http://fr.openclassrooms.com/informatique/cours/pratiques-avancees-et-meconnues-en-python/es-generateurs-2>
- <http://feldboris.alwaysdata.net/blog/python-les-iterateurs-et-les-generateurs-fr.html>
- <https://wiki.python.org/moin/Generators>
- <http://sahandsaba.com/combinatorial-generation-using-coroutines-in-python.html>
- <http://code.activestate.com/recipes/580628-pluggable-python-generators/>
- [Processing Large Data Sets With Yield and Generators](#)

### Liste en compréhension

- [http://fr.wikipedia.org/wiki/Liste\\_en\\_compr%C3%A9hension](http://fr.wikipedia.org/wiki/Liste_en_compr%C3%A9hension)
- <http://www.pythonforbeginners.com/basics/list-comprehensions-in-python>
- <http://fgallaire.flext.net/comprehension-de-liste-en-python-map-filter/>, remplacement de map() et filter()
- [http://www.python-course.eu/list\\_comprehension.php](http://www.python-course.eu/list_comprehension.php), yc suppression de lambda et reduce()
- <https://www.datacamp.com/community/tutorials/python-list-comprehension>
- <https://gist.github.com/bearfrieze/a746c6f12d8bada03589>
- [A Beginner's Guide to Python List Comprehensions](#) (Jonathan Hsu, Medium, 23/04/2020)

### Transformations et manipulations de chaînes (string)

- `translate()` : transformation sur base de correspondances entre des caractères (y compris des caractères spéciaux (\n, \t, \r,...)
  - `string.punctuation` peut être utilisé pour enlever la ponctuation
- `replace()` : remplacement d'une sous-chaîne
- `split()` : découpe en une liste de sous-chaînes. Le caractère utilisé par défaut est l'espace. Le nombre de découpe peut être précisé. `rsplit()` permet de commencer par la droite.
- `partition()` : découpe particulière → `myString.partition("search string")` renvoie un tuple de trois sous-chaînes : ( texte précédent, première occurrence de la sous-chaîne cherchée, texte à la suite)
- `strip()`, `lstrip()`, `rstrip()` : élimine des caractères en début et/ou fin, par défaut les espaces
- `zfill(n)` : ajoute des zéros devant pour arriver à une longueur donnée
- `' '.join()` : joint des sous-chaînes en liste pour créer une chaîne en ajoutant ' ' (dans cet exemple) comme sous-chaîne additionnelle

- `upper()` : capitalise
- `lower()`
- `title()` : capitalise les premières lettres des mots
- `swapcase()` : inverse capitales et bas de casse
- `startswith('fizz')`, `endswith('buzz')`, `'fizz buzz'` in `test_string` : renvoie True ou False

## Expressions rationnelles (régulières)

- `import re`
- <http://howchoo.com/g/zdvmogrIngz/python-regexes-findall-search-and-match>
- <https://docs.python.org/2/howto/regex.html>
- <http://linuxfr.org/news/travailler-avec-des-expressions-rationnelles>
- [https://fr.wikipedia.org/wiki/Expression\\_rationnelle](https://fr.wikipedia.org/wiki/Expression_rationnelle)
- ...

## Décorateurs

- [http://www.python-course.eu/python3\\_memoization.php](http://www.python-course.eu/python3_memoization.php)

## Context managers

## Programmation orienté objet

Page dédiée : [Programmation Python Orientée Objet](#)

- Exemples simples :
  - <http://nbviewer.ipython.org/url/bender.astro.sunysb.edu/classes/python-science/lectures/python-classes.ipynb>
  - <http://jeffknupp.com/blog/2014/06/18/improve-your-python-python-classes-and-object-oriented-programming/>


## Divers

- Définir ses propres types : stack, queue, tree + algorithms : [Data Structures & Algorithms in Python](#) by Papa Moryba Kouate, Aug, 2020, Towards Data Science

## Closures

- <http://stackoverflow.com/questions/36636/what-is-a-closure>
- <http://programmers.stackexchange.com/questions/40454/what-is-a-closure>

# Programmation fonctionnelle

- [Functional Programming in Python](#) (  )
- [Map, filter, reduce](#) :
  - [How To Replace Your Python For Loops with Map, Filter, and Reduce - Write more semantic code with functional programming](#)

## Performances, temps d'exécution, ...

- [Making Python Programs Blazingly Fast](#), 01/01/2020

## Tests unitaires

## Python Scripting

- [How to Run Python Scripts](#)

## Interface utilisateur graphique (GUI)

- [How to build your first Desktop Application in Python](#) Ampofo Amoh - Gyebi, Medium, 12/12/2020

From:  
<https://dvillers.umons.ac.be/wiki/> - **Didier Villers, UMONS - wiki**

Permanent link:  
[https://dvillers.umons.ac.be/wiki/teaching:progappchim:notions\\_avancees?rev=1610449396](https://dvillers.umons.ac.be/wiki/teaching:progappchim:notions_avancees?rev=1610449396)

Last update: **2021/01/12 12:03**

