



FORMATION PYTHON

BIBLIOTHÈQUE SCIENTIFIQUE

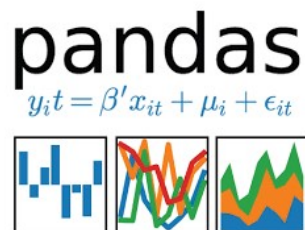
EN PYTHON

Mustapha HAIN

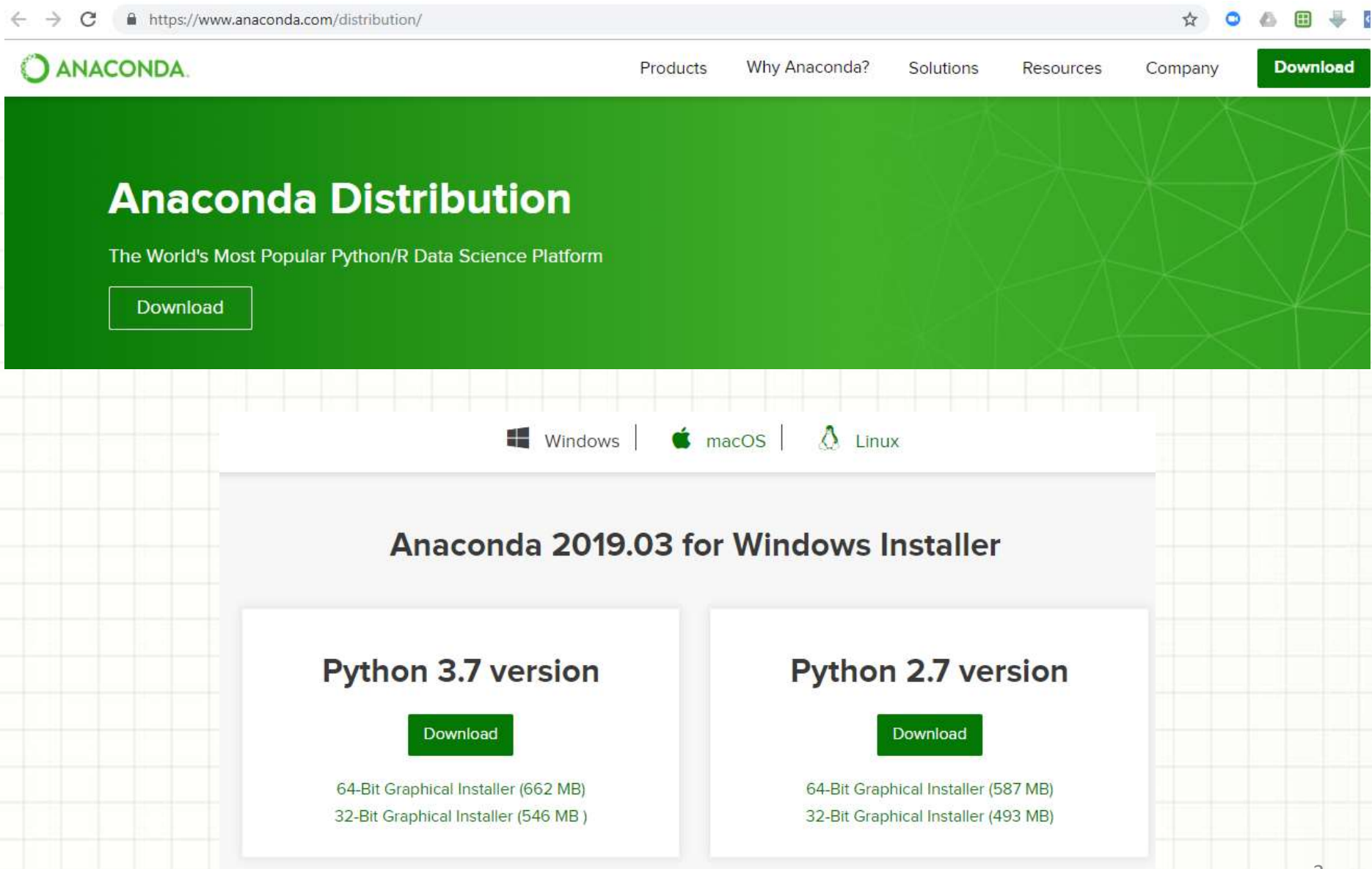
INTRODUCTION

Python possède des librairies pour à peu près tout ce que vous pouvez imaginer, et notamment:

- Numpy et Scipy pour les calculs matriciels
- Pandas pour gérer les données (les charger, appliquer des opérations d'algèbre relationnelle, etc.)
- Matplotlib et Seaborn pour la visualisation
- Scikit-learn pour les algorithmes IA
- Tensorflow et PyTorch pour le deep learning.



ENVIRONNEMENT DU TRAVAIL



The screenshot shows the Anaconda Distribution website. The header includes the Anaconda logo and navigation links: Products, Why Anaconda?, Solutions, Resources, Company, and a prominent Download button. The main banner features the text 'Anaconda Distribution' and 'The World's Most Popular Python/R Data Science Platform' with a Download button. Below this, there are tabs for Windows, macOS, and Linux. The Windows tab is selected, displaying 'Anaconda 2019.03 for Windows Installer'. Under this, there are two options: 'Python 3.7 version' and 'Python 2.7 version'. Each option has a Download button and lists the available installers: 64-Bit Graphical Installer and 32-Bit Graphical Installer with their respective sizes.

← → ↻ <https://www.anaconda.com/distribution/> ☆ ⓘ ⚙ ⌵

ANACONDA Products Why Anaconda? Solutions Resources Company **Download**

Anaconda Distribution

The World's Most Popular Python/R Data Science Platform

[Download](#)

Windows | macOS | Linux

Anaconda 2019.03 for Windows Installer

Python 3.7 version

[Download](#)

64-Bit Graphical Installer (662 MB)
32-Bit Graphical Installer (546 MB)

Python 2.7 version

[Download](#)

64-Bit Graphical Installer (587 MB)
32-Bit Graphical Installer (493 MB)

ENVIRONNEMENT DU TRAVAIL

Anaconda Navigator

File Help

ANACONDA NAVIGATOR

Upgrade Now

Sign in to Anaconda Cloud

Home

Environments

Projects (beta)

Learning

Community

Documentation

Developer Blog

Feedback



Applications on

Anaconda3

Channels

Refresh



jupyterlab

0.27.0

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Launch



jupyter notebook

5.0.0

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Launch



qtconsole

4.3.1

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Launch



rstudio

1.1.383

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Launch



spyder

3.2.4

Scientific PYTHON Development Environment. Powerful Python IDE with advanced editing, interactive testing,



glueviz

0.12.4

Multidimensional data visualization across files. Explore relationships within and among related datasets.



orange3

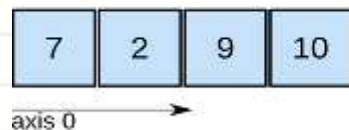
3.11.0

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows

DEFINITION

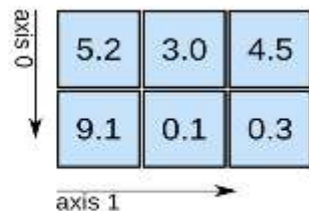
La bibliothèque NumPy (<http://www.numpy.org/>) permet d'effectuer des calculs avancés sur les tableaux/ Matrices avec Python.

1D array



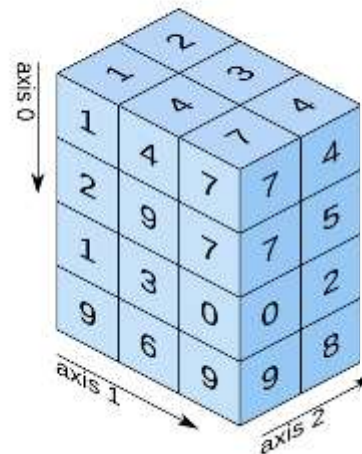
shape: (4,)

2D array



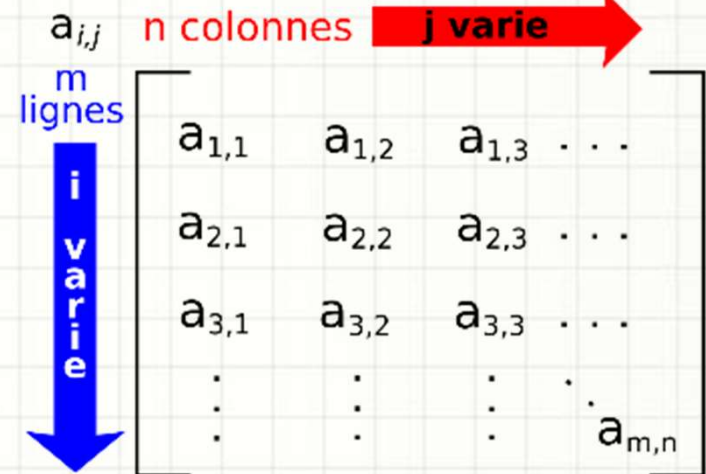
shape: (2, 3)

3D array



shape: (4, 3, 2)

matrice $m \times n$



```
import numpy as np  
a = np.array([[1, 2, 3], [4, 5, 6]])  
print(a)
```

```
print(a.size)  
print(a.ndim)  
print(a.shape)  
print(type(a))
```

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

```
6  
2  
(2, 3)  
<class 'numpy.ndarray'>
```




LA BIBLIOTHÈQUE NUMPY

Fonctions statistiques

```
import numpy as np
a=np.array([[2,4,6],
           [4,11,5],
           [4,6,12]])

print(a)
print(a.min())
print(a.max())
print(a.mean())
a.sort()
print(a)
```

```
[[ 2  4  6]
 [ 4 11  5]
 [ 4  6 12]]

2
12
6.0
[[ 2  4  6]
 [ 4  5 11]
 [ 4  6 12]]
```

- Numpy est un package pour Python spécialisé dans la manipulation des tableaux (array).
- Le package propose un grand nombre des fonctions d'accès rapide aux données (ex. recherche, extraction), pour les manipulations diverses (ex. tri), pour les calculs (ex. calcul statistique).
- tous les éléments d'un array doivent être du même type.



LA BIBLIOTHÈQUE NUMPY

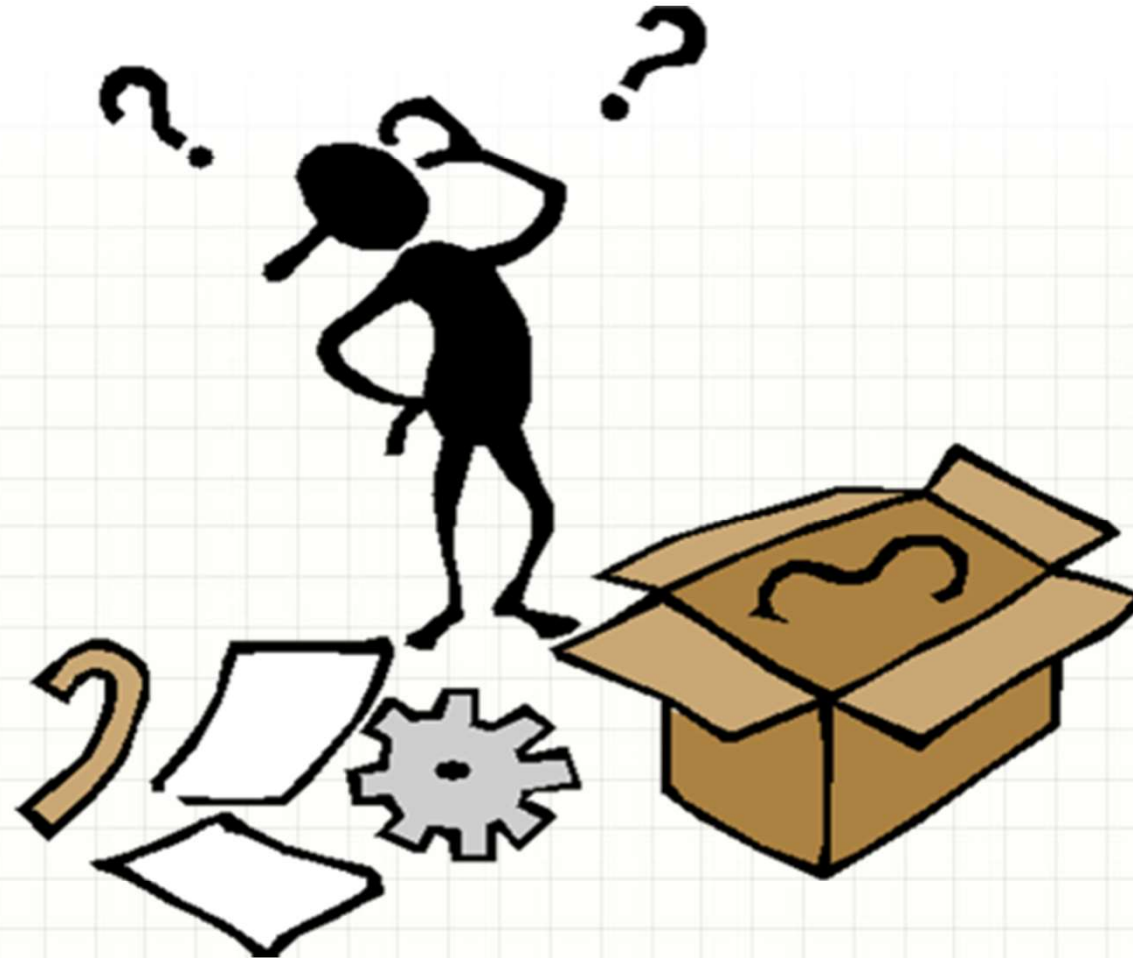
Fonctions diverses

x^{**n}	x à la puissance n, exemple : x^{**2}
numpy.sqrt(x)	racine carrée
numpy.exp(x)	exponentielle
numpy.log(x)	logarithme népérien
numpy.abs(x)	valeur absolue
numpy.sign(x)	signe

Fonctions mathématiques avec NumPy

Fonctions trigonométriques	
numpy.sin(x)	sinus
numpy.cos(x)	cosinus
numpy.tan(x)	tangente
numpy.arcsin(x)	arcsinus
numpy.arccos(x)	arccosinus
numpy.arctan(x)	arctangente

Atelier x :



Travaillons ensemble-

Arange function Return evenly spaced values within a given interval.

```
numpy.arange([start, ]stop, [step, ]dtype=None, *, like=None)
```

```
import numpy as np  
a = np.arange(24)  
print(a)
```

Reshaping we can add or remove dimensions or change number of elements in each dimension.

```
b = a.reshape(4,2,3)  
print b
```

LA BIBLIOTHÈQUE NUMPY

Fonctions Spécifiques

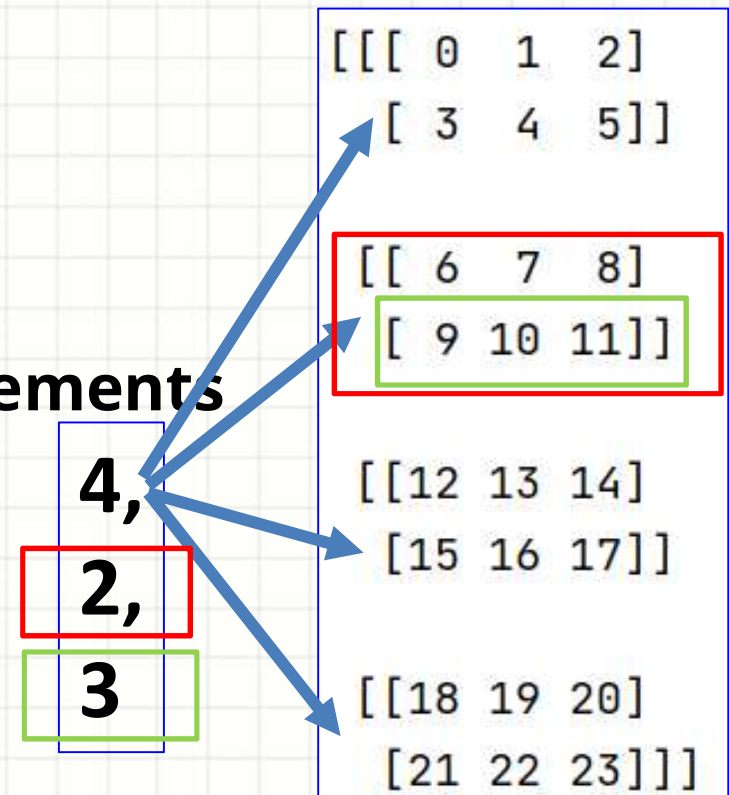
Arange function Return evenly spaced values within a given interval.

`numpy.arange([start,]stop, [step,]dtype=None, *, like=None)`

```
import numpy as np  
a = np.arange(24)  
print(a)
```

Reshaping we can add or remove dimensions or change number of elements in each dimension.

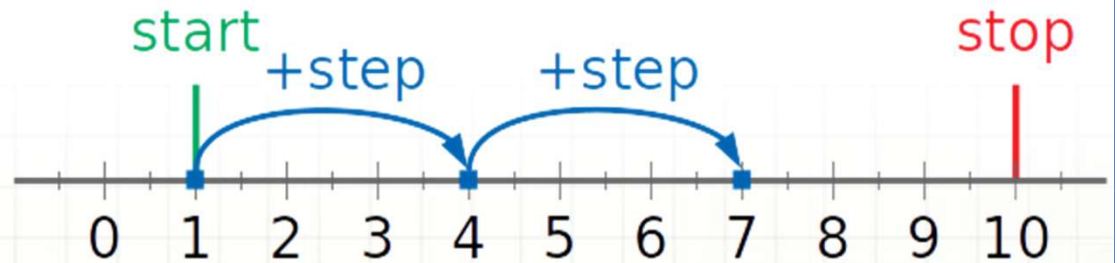
```
b = a.reshape(4,2,3)  
print b
```



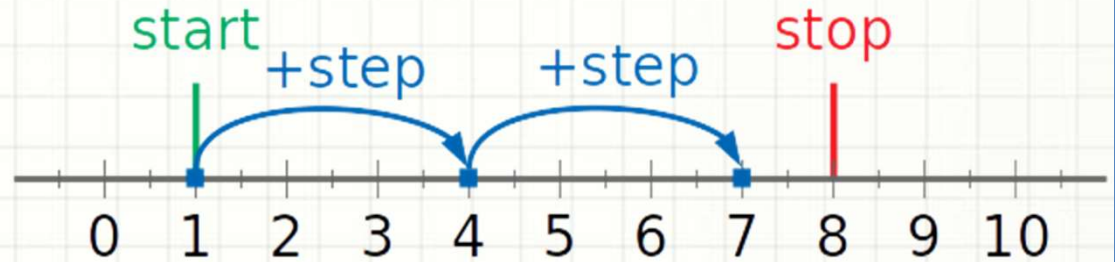
LA BIBLIOTHÈQUE NUMPY

Fonctions Spécifiques

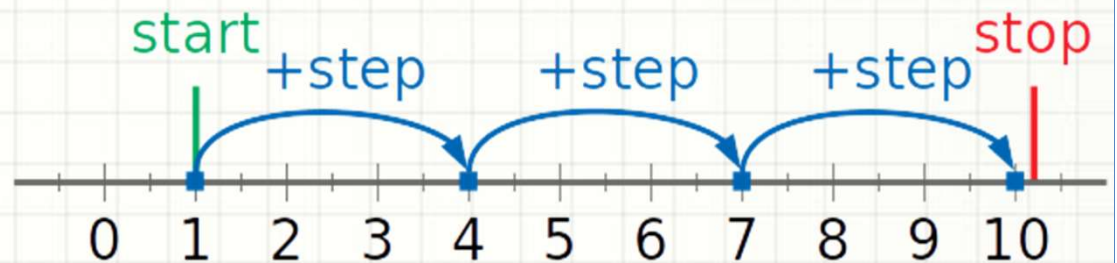
```
>>> np.arange(1, 10, 3)  
array([1, 4, 7])
```



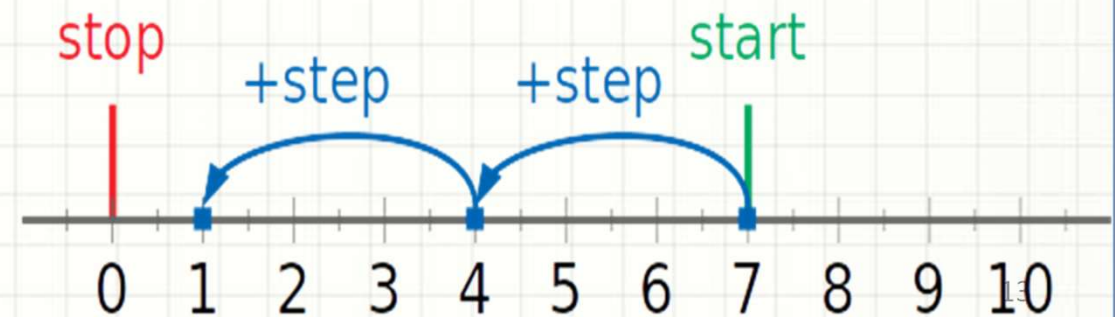
```
>>> np.arange(1, 8, 3)  
array([1, 4, 7])
```



```
>>> np.arange(1, 10.1, 3)  
array([1., 4., 7., 10.])
```



```
>>> np.arange(7, 0, -3)  
array([7, 4, 1])
```



LA BIBLIOTHÈQUE NUMPY

Fonctions Spécifiques

A Python slice object is constructed by giving start, stop, and step parameters to the built-in slice

```
a = np.arange(10)
```

```
print(a) → [0 1 2 3 4 5 6 7 8 9]
```

```
s = slice(2,7,2)
```

```
print(s) → slice(2, 7, 2)
```

```
print(a[s]) → [2 4 6]
```

```
a = np.arange(20)
```

```
print(a) →
```

```
s = slice(10,16,5)
```

```
print(s) →
```

```
print a[s] →
```



Array indexing is the same as accessing an array element

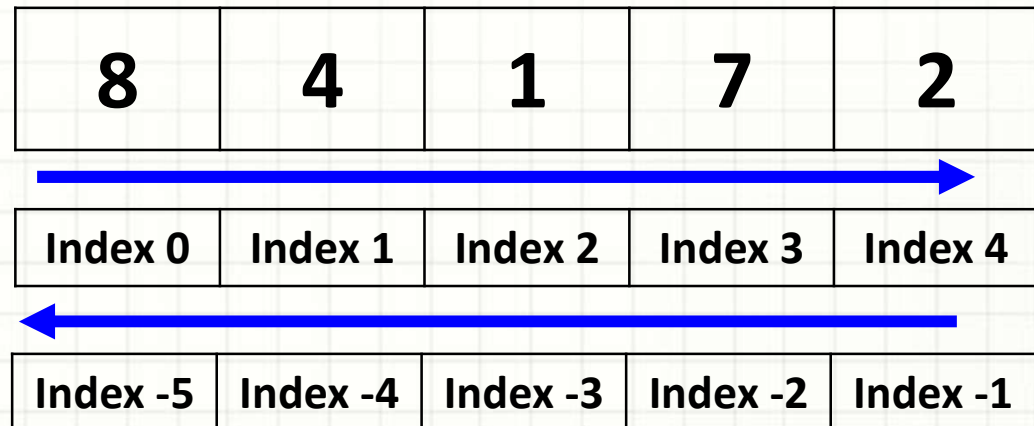
```
import numpy as np
```

```
arr =  
np.array([8,4,1,7,2])
```

```
print(arr[0])      8
```

```
print(arr[2])      1
```

```
print(arr[-2])     7
```



```
s=arr[-2]+arr[2]-arr[1]  
print(s)
```



NumPy - Advanced Indexing

```
import numpy as np
```

```
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
```

```
print(arr[0, 1])
```

```
print(arr[1, 4])
```

```
print(arr[1, -1])
```

Last element from 2nd dim

2nd element on 1st dim

5th element on 2nd dim

```
import numpy as np
```

```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
```

```
print(arr[0, 1, 2])
```

`arr[0, 1, 2]` prints the value 6.

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

`[[7, 8, 9], [10, 11, 12]]`

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

4 5 6



```
import numpy as np
```

```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
```

```
print(arr[0, 3, 2])
```

```
print(arr[1, 0, 2])
```



LA BIBLIOTHÈQUE NUMPY

Advanced Indexing

```
import numpy as np
```

```
x = np.arange(10)
```

```
print(x)
```

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

```
print(x[2:7])
```

[2 3 4 5 6]

```
print(x[2:7:1])
```

[2 3 4 5 6]

```
print(x[2:7:2])
```

[2 4 6]

```
print(x[7:1:-3])
```

[7 4]



LA BIBLIOTHÈQUE NUMPY

Advanced Indexing

```
import numpy as np
import numpy as np
print(np.arange(12).reshape(3,4))

print(np.arange(12).reshape(2,6))

print(np.arange(10).reshape(2,5))
print(np.arange(10).reshape(5,2))
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]
```



GÉNÉRATION DES VALEURS

Nombres avec pas

np.linspace(3, 9, 10)

Nb valeur

À partir

Jusqu'au

```
[ 3.,  3.66666667,  4.33333333,  5.,  
 5.66666667,  6.33333333,  7.,  
 7.66666667,  8.33333333,  9. ]
```

Nombres aléatoires

La fonction `numpy.random.random()` permet d'obtenir des nombres compris entre 0 et 1 par tirage aléatoire avec une loi

uniforme.

np.random.random()

```
0.5540884899329033
```

np.random.random(3)

```
array([ 0.86431861,  0.88519197,  
 0.30663316])
```

np.random.random((2,3))

```
([[ 0.66265691,  0.39385577,  0.09319192],  
 [ 0.43483474,  0.42859904,  0.79189574]])
```

numpy.zeros

Returns a new array of specified size, filled with zeros.

```
import numpy as np  
x = np.zeros(5)  
print (x)
```

[0. 0. 0. 0. 0.]

```
print(np.zeros(2, dtype=int))
```

[0 0]

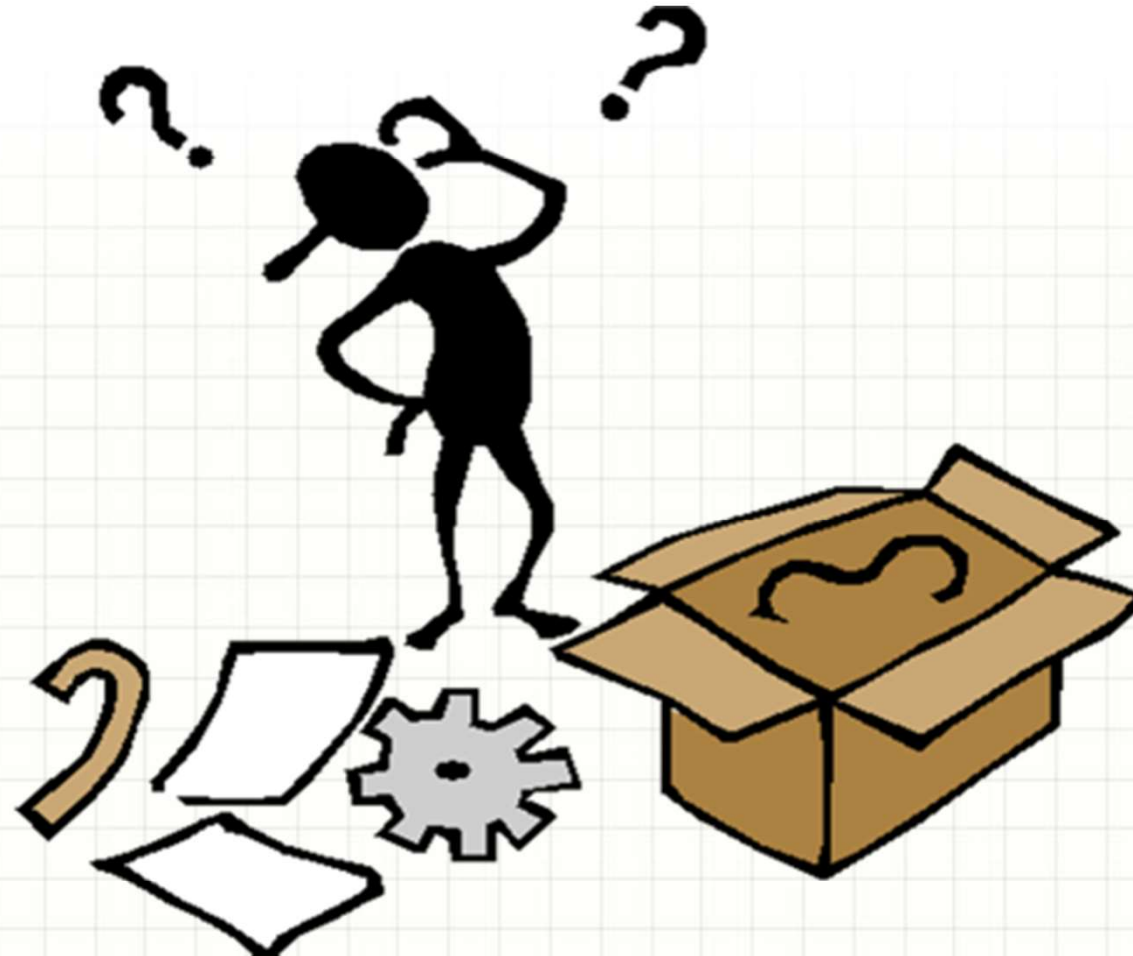
```
print(np.zeros([2, 2],  
dtype=int))
```

[[0 0]
[0 0]]

```
print(np.zeros([3, 3]))
```

[[0. 0. 0.]
[0. 0. 0.]
[0. 0. 0.]]

Atelier x :



Travaillons ensemble-



0	1	2	3	4	5
10	11	12	13	14	15
20	21	22	23	24	25
30	31	32	33	34	35
40	41	42	43	44	45
50	51	52	53	54	55