Calcul Numérique en Mécanique
Python

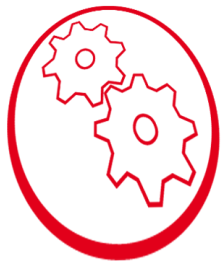
R. Mozul & L. Daridon

Université de Montpellier - CNRS

2021

Sommaire

- Tableaux *numpy*
- Indexation
- Constructeurs
- Opérateurs/Vectorisation
- Duck typing



Tableaux *numpy*

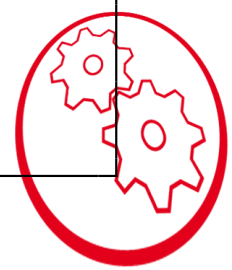
Les tableaux stockent :

- des données homogènes
- en un nombre fixé
- dont la forme peut changer

Ces contraintes assurent beaucoup d'éfficacité dans le traitement des données.

Tableau

```
>>> import numpy as np
>>> a = np.array( [12.,  5., 3.14, 1.1 ] )
>>> a.size
4
>>> a.shape
(4,)
>>> a[0]
12.0
>>> a[1] = 13
>>> a
array([ 12.  , 13.  ,  3.14,  1.1 ])
>>> a.shape = [2,2]
>>> a
array([[ 12.  , 13.  ],
       [ 3.14,  1.1 ]])
>>> a.size = 6
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: attribute 'size' of 'numpy.ndarray' objects is not writable
>>> a[0] = 'toto'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: could not convert string to float: 'toto'
```



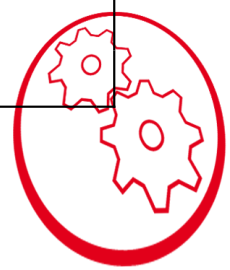
Indexation

L'indexation fonctionne de la même manière pour les listes et les tableaux:

- l'indice est un entier commençant par 0
- l'indexation se fait [begin:end:step]
- l'indice end est exclu de la sélection
- si l'indice i est négatif, il est compté comme $n - i$ où n est la taille
- si on omet des valeurs, les valeurs par défaut sont $[0 : n : 1]$
- les : sont optionnels
- chaque dimension du tableau peut utiliser cette notation, séparé par des ,

Tableau

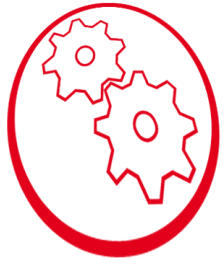
```
>>> import numpy as np
>>> a = np.arange(12)
>>> a[-2]
10
>>> a[:3]
array([0, 1, 2])
>>> a[-3:]
array([9, 10, 11])
>>> a[::4]
array([0, 4, 8])
>>> a.shape = [3,4]
>>> a[::2,1::2]
array([[ 1,  3],
       [ 9, 11]])
>>> for r in a:
...     print(r)
...
[0 1 2 3]
[4 5 6 7]
[ 8  9 10 11]
```



Constructeurs

Les constructeurs les plus courants:

- depuis une liste np.array([..])
- np.zeros et np.ones pour créer des tableaux de $0.$ ou $1.$
- np.arange pour créer un tableau d'entier comme range
- np.linspace pour créer un tableau de réels entre un borne inférieure et supérieure
- np.random pour créer un tableau de réels avec des valeurs aléatoires



Constructeurs

```
>>> import numpy as np
>>> np.zeros(3)
array([0., 0., 0.])
>>> np.ones([2,3])
array([[0., 0., 0.],
       [0., 0., 0.]])
>>> np.arange(3)
array([0, 1, 2])
>>> np.arange(1,4,2)
array([1, 3])
>>> np.linspace(0.,12.,6)
array([  0. ,   2.4,   4.8,   7.2,   9.6,  12. ])
```

Opérateurs

Les opérateurs courant de python:

- sont: + - * / **
- fonctionnent terme à terme
- peuvent fonctionner avec une valeur et un tableau
- peuvent fonctionner avec deux tableaux de même forme

Il existe de nombreuses autres fonctionnalités utiles:

- méthodes : min, max, argmin, argmax, sort
- fonction : dot, tensordot
- sous-module : linalg (calcul norm, inversion de matrices)

Opérateurs

```
>>> import numpy as np
>>> a = np.zeros(20)
>>> a = a+5
>>> b = a * np.random.random(20)
>>> a.shape = [4,5]
>>> b.shape = [5,4]
>>> a + b
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: operands could not be broadcast together with shapes (4,5) (5,4)
```