Découverte de Numpy et Matplotlib

- Numpy
- Matplotlib
- Import
- Tableaux et attributs

Alexandre Gramfort

Programmation Python 2 : Découverte de Numpy et Matplotlib

Alexandre Gramfort

Qu'est-ce-que NumPy?

- `NumPy` est une librairie utilisée dans presque tous les projets de calcul numérique sous Python
- `NumPy` fournit des structures de données performantes pour la manipulation de vecteurs, de matrices, de tenseurs. On parlera de tableau ou `array` en Anglais
- `NumPy` est écrit en C et en Fortran d'où ses performances élevées lorsque les calculs sont vectorisés (formulés comme des opérations sur des tableaux)

Qu'est-ce-que matplotlib?

- `matplotlib` est une librairie pour la génération de graphiques en 2D et 3D
  - syntaxe très proche de celle de Matlab
  - supporte texte et étiquettes en LaTeX
  - sortie de qualité dans divers formats (PNG, PDF, SVG, EPS...)
  - interface graphique intéractive pour explorer les figures

Découverte de Numpy et Matplotlib

Numpy
Matplotlib
Import
Tableaux et attributs

Pour utiliser `NumPy` et `matplotlib` il faut commencer par les importer :

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

np et **plt** sont les abréviations d'import classiques.

Alexandre Gramfort

Création de tableaux (arrays) `numpy`

Plusieurs possibilités:

- à partir de listes ou n-uplets Python
- en utilisant des fonctions dédiées, telles que `arange`, `linspace`, etc.
- par chargement à partir de fichiers

A partir de listes

Au moyen de la fonction `np.array`:

In [3]:
```
# un vecteur à partir d'une liste Python
v = np.array([1, 3, 2, 4])
print(v)
print(type(v))
```

```
[1 3 2 4]
<type 'numpy.ndarray'>
```

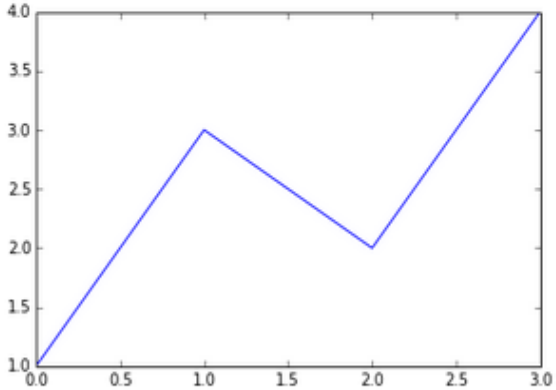
Découverte de Numpy et Matplotlib

Numpy
Matplotlib
Import
Tableaux et attributs

Alexandre Gramfort

On peut alors visualiser ces données avec matplotlib:

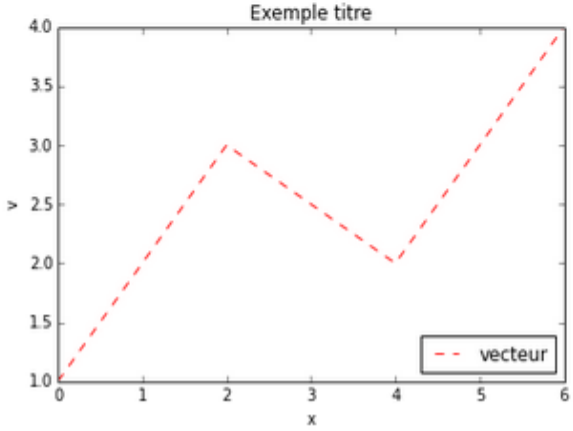
In [4]:
```
plt.figure()
plt.plot(v)
plt.show()
```



On peut omettre `plt.show()`, lorsque la fonction `plt.ion()` a été appelée ; c'est le cas dans Spyder.

Un peu plus compliqué:

In [5]:
```
x = np.array([0,2,4,6])
plt.figure()
plt.plot(x, v, 'r--', label='vecteur')
plt.xlabel('x')
plt.ylabel('v')
plt.title('Exemple titre')
plt.legend(loc='lower right')
plt.show()
```



Alexandre Gramfort

Découverte de Numpy et Matplotlib

Numpy
Matplotlib
Import
Tableaux et attributs

Alexandre Gramfort

Tableau de dimension 2 (matrice):

In [6]:
```
# une matrice: l'argument est une liste emboitée
M = np.array([[1, 3], [2, 4]])
print(M)
```

```
[[1 3]
 [2 4]]
```

In [7]:
```
# accéder à un élément
print(M[0, 0])
print(M[1, 1])
```

```
1
4
```

La variable `M` est aussi du type `ndarray`

In [8]:
```
print(type(M))
```

```
<type 'numpy.ndarray'>
```

L'attribut `shape`

`v` et `M` diffèrent par leur taille:

```
In [9]: print(v.shape)
```

```
(4,)
```

```
In [10]: print(M.shape)
```

```
(2, 2)
```

```
In [11]: print(type(M.shape))
```

```
<type 'tuple'>
```

L'attribut `ndim`

il permet de savoir quel est la dimension d'un ndarray

```
In [12]: print(v.ndim)
```

```
1
```

```
In [13]: print(M.ndim)
```

```
2
```

L'attribut `size`

il permet de savoir quel est le nombre d'éléments dans un `ndarray`

```
In [14]: print(v.size)
```

```
4
```

```
In [15]: print(M.size)
```

```
4
```

Exemple d'un tableau à 3 dimensions

In [16]:
```
T = np.array([[[1, 3, 2, 4]]])
print(T)
```

```
[[[1 3 2 4]]]
```

In [17]:
```
print(T.shape)
```

```
(1, 1, 4)
```

In [18]:
```
print(T.ndim)
```

```
3
```

In [19]:
```
print(T.size)
```

```
4
```

In [20]:
```
np.size(T)
```

Out[20]: 4