

# Aller plus loin avec Python

## Python au service du scientifique

Denis Steckelmacher  
dsteckel@ulb.ac.be  
[public.steckdenis.be/ulb/](http://public.steckdenis.be/ulb/)

9 mars 2015

# Table des matières

Pourquoi Python ?

Rappels de Python

Travailler confortablement

Calcul matriciel avec NumPy

Tracé de fonctions avec PyPlot

Aller encore plus loin

# Pourquoi Python ?

Pourquoi Python ?

Rappels de Python

Travailler confortablement

Calcul matriciel avec NumPy

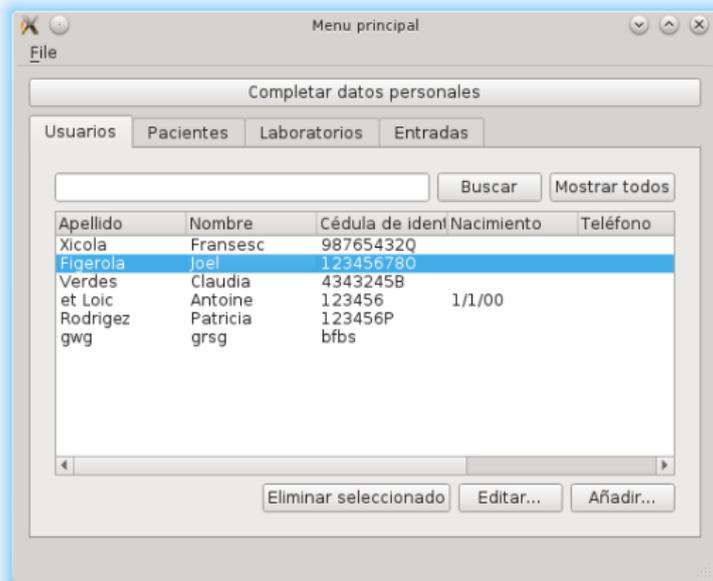
Tracé de fonctions avec PyPlot

Aller encore plus loin

# Pourquoi Python ?

- ▶ Facile (pour de la programmation)
- ▶ Semblable à des dizaines d'autres langages
- ▶ Utile aux informaticiens (applications, sites web)
- ▶ Permet le calcul scientifique

# Interface graphique en Python





## D-Cubes

- Accueil
- Nos produits
- Qui sommes-nous ?
- Le Logiciel Libre
- Paiement et livraison
- Contact

### Configuration de l'ordinateur

Voici maintenant l'étape pendant laquelle vous allez pouvoir configurer votre ordinateur. Quoi que vous fassiez, votre ordinateur sera utilisable dès le déballage. N'oubliez tout de même pas que certaines options peuvent vous être utiles, comme le Wifi ou une plus grande quantité de mémoire vive.

Disque dur :	250 Go
<i>Stockage de l'ordinateur, sur lequel se trouvent vos fichiers</i>	
RAM :	1 Gio
<i>Mémoire vive de l'ordinateur.</i>	
Baie CD :	(aucun)
<i>Emplacement pour un lecteur CD, DVD ou Bluray</i>	
Processeur :	Intel D410 (1x1,66 Ghz)
<i>Coeur, cerveau de l'ordinateur.</i>	

**Accessoires externes :**

*Accessoires fournis avec votre ordinateur, pour plus de simplicité*

- Dongle WiFi a/b/g (+ 11.00 euros)
- Souris P5/2 (+ 4.00 euros) (Ne permet pas la livraison)
- Clavier belge 105 touches (+ 6.00 euros) (Ne permet pas la livraison)
- Ecran 19" 1366x768 (+ 90.00 euros) (Ne permet pas la livraison)

Pour un total de 189.99 € TTC

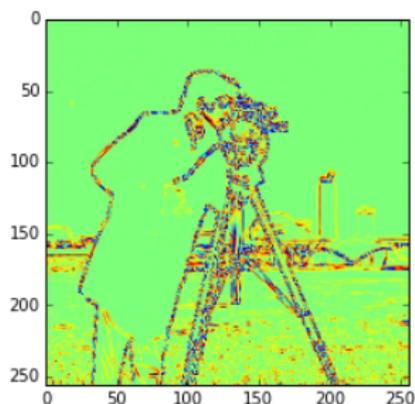
Certaines options, si sélectionnées, vous obligent à renoncer à vous faire livrer. Ce sont des «options de facilité», qui ne sont pas nécessaires, des composants que vous savez acheter vous-même dans le commerce sans avoir à payer plus cher. La plus-value de ces options ne justifie pas la forte augmentation des frais de port qu'elles engendrent.

# Traitement d'images et de matrices

```
In [8]: import scipy.ndimage as ndi
image = ndi.imread("/home/steckdenis/Documents/Cours/Cours/steckdenis/steckdenis.jpg")
image = image.astype(int16)

A = ndi.filters.convolve(image, [[1, 0, -1], [1, 0, -1], [1, 0, -1]])
B = ndi.filters.convolve(image, [[1, 1, 1], [0, 0, 0], [0, 0, 0]])
plt.imshow(A*A + B*B)
```

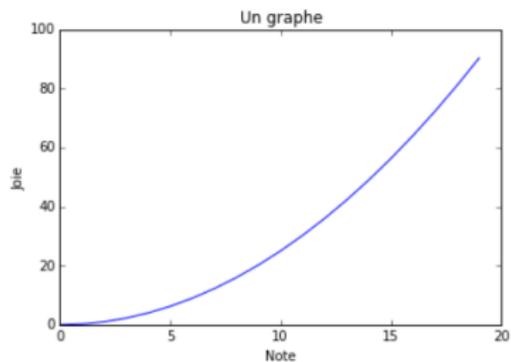
Out[8]: <matplotlib.image.AxesImage at 0x7f7b0a91a4e0>



# Graphes

```
def f(x):  
    return x*x / 4  
  
X = range(20)  
Y = [f(x) for x in X]  
  
plot(Y)  
title("Un graphe")  
xlabel("Note")  
ylabel("Joie")
```

<matplotlib.text.Text at 0x7f7b0a4e8518>



# Rappels de Python

Pourquoi Python ?

Rappels de Python

Travailler confortablement

Calcul matriciel avec NumPy

Tracé de fonctions avec PyPlot

Aller encore plus loin

# Généralités

1. Inventé en 1991, version 3.4.3 sortie le 25 février 2015
2. Langage généraliste
3. Concepts (variables, boucles, fonctions) communs à R, Matlab, Fortran, C++, Java, PHP, etc
4. Conçu pour ne pas être plus compliqué que nécessaire

# Expressions

```
>>> 3  
3
```

```
>>> 3 * (2 + 3)  
15
```

# Variables

```
>>> pi = 3.1415926
```

```
>>> pi
```

```
3.1415926
```

```
>>> pi * 2
```

```
6.2831852
```

# Variables

```
>>> a = 9
>>> a = 2 * a - 4
>>> a
14
```

# Variables

```
>>> a = 9
>>> a = 2 * 9 - 4
>>> a
14
```

# Listes

```
>>> a = ["pomme", "poire", "pêche"]
>>> a
["pomme", "poire", "pêche"]
>>> a[1]
"poire"
```

## Listes

```
>>> a = ["pomme", "poire", "pêche"]
>>> a.append("prune")
>>> a
["pomme", "poire", "pêche", "prune"]
>>> a[2] = "pastèque"
>>> a
["pomme", "poire", "pastèque", "prune"]
```

## Fonctions (appel)

```
>>> delta(1, 4, -2 * 2)  
32
```

## Fonctions (définition)

```
def delta(a, b, c):  
    return b * b - 4 * a * c
```

## Fonctions (fonctionnement)

```
>>> delta(1, 4, -2 * 2)
```

## Fonctions (fonctionnement)

```
def delta(a, b, c):  
    return b * b - 4 * a * c
```

## Fonctions (fonctionnement)

```
def delta(a = 1, b = 4, c = -2 * 2):  
    return b * b - 4 * a * c
```

## Fonctions (fonctionnement)

```
def delta(a = 1, b = 4, c = -2 * 2):  
    return 4 * 4 - 4 * 1 * -2 * 2
```

## Fonctions (fonctionnement)

```
def delta(a = 1, b = 4, c = -2 * 2):  
    return 32
```

## Fonctions (fonctionnement)

```
>>> delta(1, 4, -2 * 2)
```

# Fonctions (fonctionnement)

```
>>> 32
```

```
32
```

## Fonctions (fonctionnement)

```
>>> delta(1, 4, -2 * 2) + 1  
33
```

# Modules

- ▶ Toute la puissance de son langage est dans ses fonctions
- ▶ Si une fonction vous permet de conquérir le monde, l'appeler vous fera conquérir le monde
- ▶ En Python, les fonctions sont rassemblées dans des modules
- ▶ Modules pour les maths, les réseaux, les interfaces graphiques, les images, etc

# Contenu d'un module

math

acos

acosh

asin

asinh

atan

atanh

ceil

...

## Modules (utilisation)

```
>>> import math
>>> math.sin(3.141)
0.0005926535550994539
```

```
>>> import datetime
>>> maintenant = datetime.datetime.now()
>>> maintenant
datetime.datetime(2015, 3, 4, 20, 33, 4, 725546)
>>> maintenant.year
2015
```

# IPython

Pourquoi Python ?

Rappels de Python

**Travailler confortablement**

Calcul matriciel avec NumPy

Tracé de fonctions avec PyPlot

Aller encore plus loin

## IP[y]: IPython Interactive Computing

1. <http://www.ipython.org>
2. Entrer du code Python et voir immédiatement son effet
3. Possibilité du modifier du code (si on a fait une faute)
4. Enregistrement du code terminé dans un fichier .py

# IPython (Python de base)

```
import math
```

```
angle = float(input('Entrez un angle : '))  
print(math.sin(angle))
```

Entrez un angle :

# IPython (Python de base)

```
import math
```

```
angle = float(input('Entrez un angle : '))  
print(math.sin(angle))
```

```
Entrez un angle : 2.394793  
0.6792936154326202
```

# NumPy

Pourquoi Python ?

Rappels de Python

Travailler confortablement

Calcul matriciel avec NumPy

Tracé de fonctions avec PyPlot

Aller encore plus loin

# NumPy

- ▶ Base du calcul scientifique en Python
- ▶ Gère les matrices de manière efficace
- ▶ Code compact et très rapide

## Créer un vecteur

```
>>> from numpy import *
>>>
>>> vecteur = array([1, 2, 3])
>>> vecteur
array([1, 2, 3])
>>> vecteur.shape
(3,)
```

## Opérations sur les vecteurs

```
>>> vecteur * 2
array([2, 4, 6])
>>> vecteur + 4
array([5, 6, 7])
>>> vecteur.sum()
6
>>> vecteur.mean()
2.0
>>> vecteur.repeat(2)
array([1, 1, 2, 2, 3, 3])
```

## Créer une matrice

```
>>> from numpy import *
>>>
>>> matrice = array([1, 2, 3], [4, 5, 6])
>>> matrice
array([1, 2, 3],
       [4, 5, 6])
>>> matrice.shape
(2, 3)
```

# Opérations sur les matrices

- ▶ Les mêmes que sur les vecteurs
- ▶ Multiplication matricielle
- ▶ Inversion

## Multiplication de matrices

```
>>> matrice * 2
array([2,  4,  6],
       [8, 10, 12])
>>> vecteur = array([[1, 2, 3]])
>>> vecteur = vecteur.transpose()
>>> vecteur
array([[1],
       [2],
       [3]])
>>> dot(matrice, vecteur)
array([[14],
       [32]])
```

## Inversion de matrices

```
>>> matrice = mat([[1, 2], [3, 4]])  
>>> matrice  
matrix([[1, 2],  
        [3, 4]])  
>>> matrice.I  
matrix([[ -2. ,  1. ],  
        [ 1.5, -0.5]])
```

## Masques (création)

```
>>> matrice
array([1, 2, 3],
      [4, 5, 6])
>>> matrice > 2
array([[False, False,  True],
      [ True,  True,  True]])
```

# Masques (création)

matrice

<b>1</b>	<b>2</b>	<b>3</b>
<b>4</b>	<b>5</b>	<b>6</b>

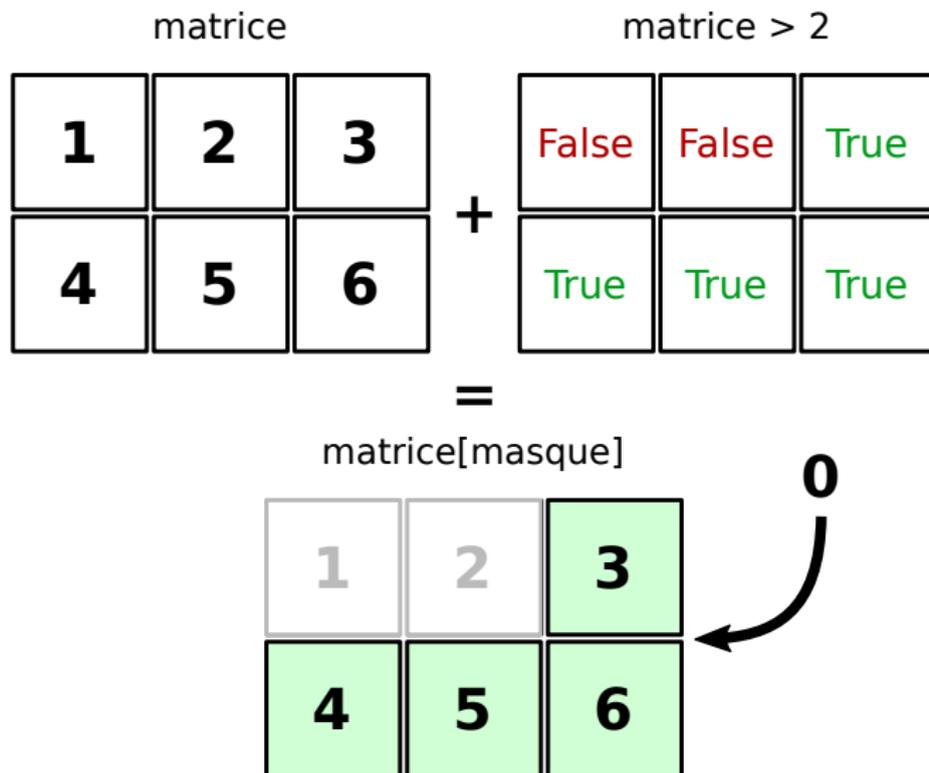
matrice > 2

False	False	True
True	True	True

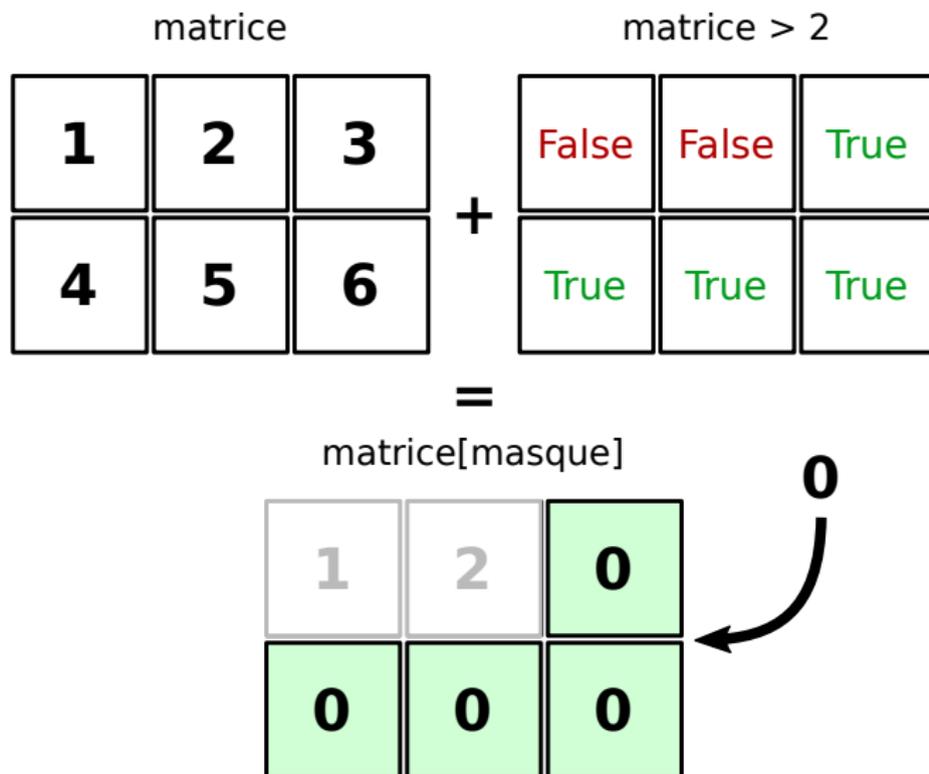
## Masques (utilisation)

```
>>> matrice
array([1, 2, 3],
      [4, 5, 6])
>>> masque = matrice > 2
>>> matrice[masque]
array([3, 4, 5, 6])
>>> matrice[masque] = 0
>>> matrice
array([[1, 2, 0],
      [0, 0, 0]])
```

# Masques (utilisation)



# Masques (utilisation)



## Slicing

```
>>> matrice
array([1, 2, 3],
      [4, 5, 6])
>>> matrice[0, 0]
1
>>> matrice[0, 1]
2
>>> matrice[1, 1:3]
array([5, 6])
>>> matrice[1, 0:3:2]
array([4, 6])
```

# Slicing

```
>>> matrice
array([1, 2, 3],
      [4, 5, 6])
>>> matrice[1, 1:3] = -6
>>> matrice
array([[ 1,  2,  3],
       [ 4, -1, -1]])
```

# PyPlot

Pourquoi Python ?

Rappels de Python

Travailler confortablement

Calcul matriciel avec NumPy

Tracé de fonctions avec PyPlot

Aller encore plus loin

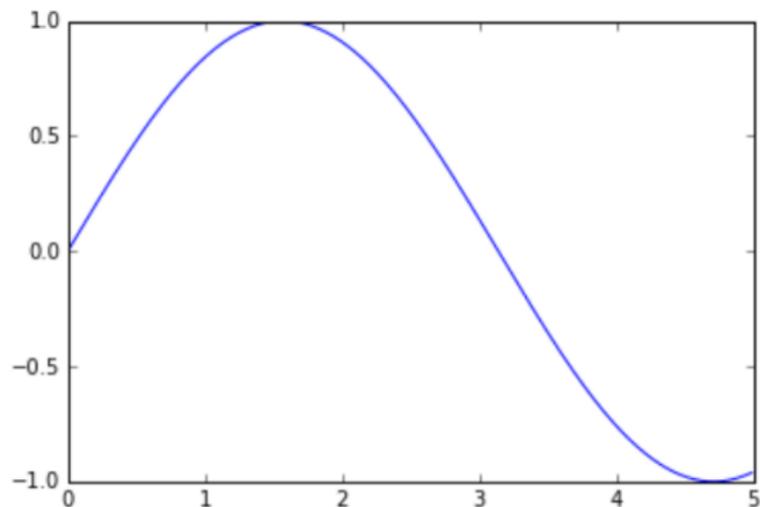
# PyPlot

- ▶ Tracé de fonctions
- ▶ Tracé de graphes en général (barres, camemberts, etc)
- ▶ Fonctions et arguments compatibles avec Matlab
- ▶ Permet également l'affichage d'images
- ▶ Très bonne intégration avec IPython

## Dessin simple

```
X = array(range(500))  
X = X / 100  
|  
plot(X, sin(X))
```

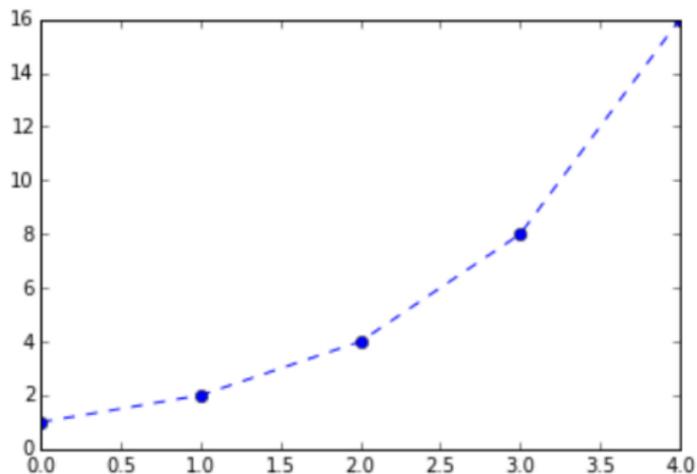
[<matplotlib.lines.Line2D at 0x7f24903b78d0>]



## Dessin personnalisé

```
plot([1, 2, 4, 8, 16], linestyle='--', marker='o')
```

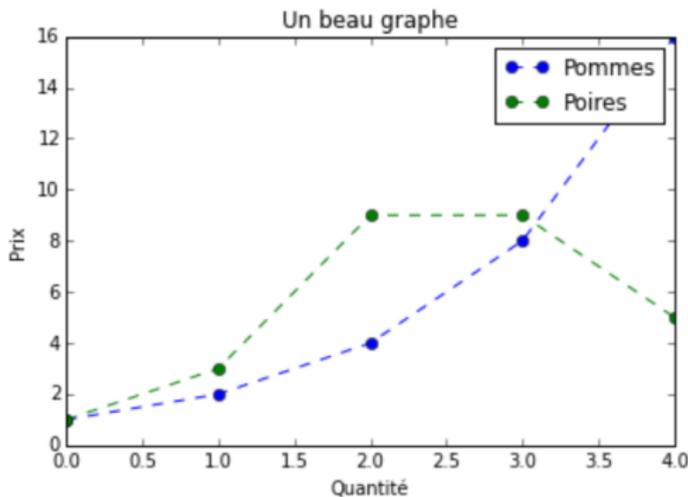
```
[<matplotlib.lines.Line2D at 0x7f2490123a58>]
```



# Dessin personnalisé

```
title('Un beau graphe')
plot([1, 2, 4, 8, 16], linestyle='--', marker='o')
plot([1, 3, 9, 9, 5], linestyle='--', marker='o')
legend(['Pommes', 'Poires'])
xlabel('Quantité')
ylabel('Prix')
```

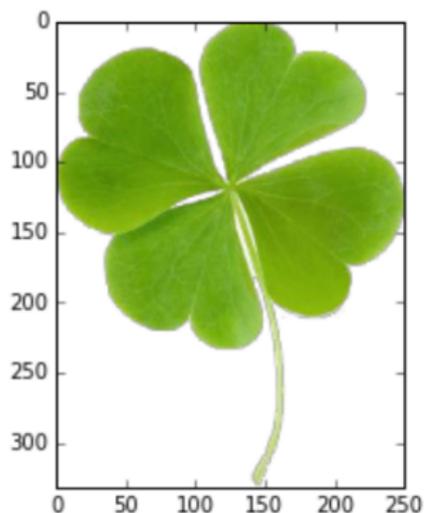
<matplotlib.text.Text at 0x7f2490031ef0>



## Lire et afficher des images

```
image = imread('/home/steckdenis/test.png')  
imshow(image)
```

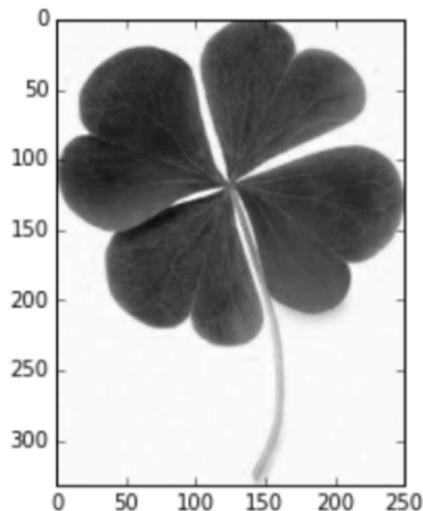
<matplotlib.image.AxesImage at 0x7f248be60b38>



# Les images sont des matrices

```
image = imread('/home/steckdenis/test.png')  
image = image[:, :, 0]  
imshow(image, cmap=cm.gray)|
```

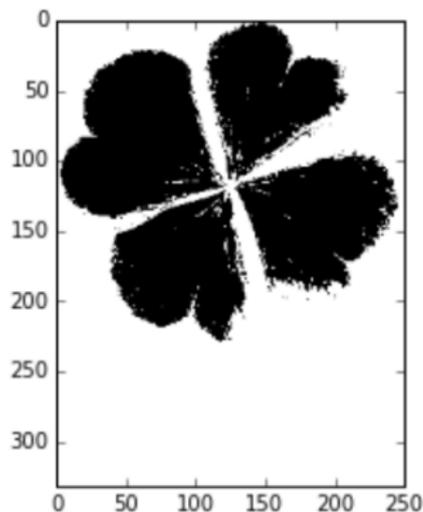
<matplotlib.image.AxesImage at 0x7f248a23ea20>



## Les images sont des matrices

```
image = imread('/home/steckdenis/test.png')  
image = image[:, :, 0]  
imshow(image > 0.5, cmap=cm.gray)
```

<matplotlib.image.AxesImage at 0x7f248bbbccc0>



# Aller encore plus loin

Pourquoi Python ?

Rappels de Python

Travailler confortablement

Calcul matriciel avec NumPy

Tracé de fonctions avec PyPlot

Aller encore plus loin

# Interfaces graphiques avec Qt

- ▶ Bibliothèque professionnelle, développée par Nokia
- ▶ Google Earth, Photoshop Elements et d'autres sont basés sur Qt
- ▶ Interfaces graphiques, Internet, bases de données, traitement d'images, ...

# Sites web avec Django

- ▶ Facile à utiliser et très puissant
- ▶ Mais le développement de sites web reste très compliqué

# Vers l'infini et au delà

- ▶ Python existe depuis très longtemps et est très utilisé
- ▶ Si vous voulez quelque-chose, ça existe
- ▶ QGIS (Géographie), divers outils mathématiques et scientifiques codés en Python
- ▶ Des programmes, des jeux (y compris 3D) et des sites web sont codés en Python

# Liens

- ▶ **IPython** : [www.ipython.org](http://www.ipython.org)
- ▶ **NumPy** : [www.numpy.org](http://www.numpy.org)
- ▶ **PyPlot** : [www.matplotlib.org/api/pyplot\\_api.html](http://www.matplotlib.org/api/pyplot_api.html)
- ▶ **Qt** (Qt en C++) : [www.qt-project.org](http://www.qt-project.org)
- ▶ **PyQt** (Qt en Python) :  
[www.riverbankcomputing.com/software/pyqt/download5](http://www.riverbankcomputing.com/software/pyqt/download5)
- ▶ **Django** : [www.djangoproject.com](http://www.djangoproject.com)