python™

Le langage Python

damien.andre@unilim.fr, www.yakuru.fr/~dada/csma.zip

14 Mai 2017

SPCTS Science des Procédés Céramiques et de Traitements de Surface

ENSCI ÉCOLE NATIONALE SUPÉRIEURE DE CÉRAMIQUE INDUSTRIELLE

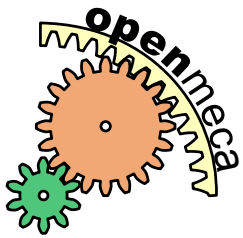
cnrs dépasser les frontières

Mes activités...

Activités scientifiques et d'enseignement à l'interface entre la *thermo-mécanique des matériaux céramiques*, les *méthodes numériques* et la *programmation*.

la technopole ester à Limoges

Des projets informatiques (libres)...



logiciel Openmeca

application pyrfda

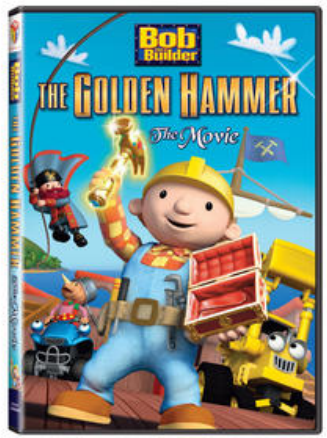
application pydic

code de calcul GranOO

➥ *pyrfda*, entièrement développé en Python, servira de fil rouge

Le concept du « golden hammer »...

Première fois énoncée par un psychologue américain en 1966 : *« I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail. »*



Appliqué au développement informatique, cette loi est devenue un *anti-pattern* (mauvaise pratique). Cette mauvaise pratique consiste à l'application systématique (voir obsessionnelle) d'une technologie familière à un grand nombre de problème.

➥ Nous ne choisissons pas toujours l'outil le plus efficace !

Python, plus qu'un simple outil...

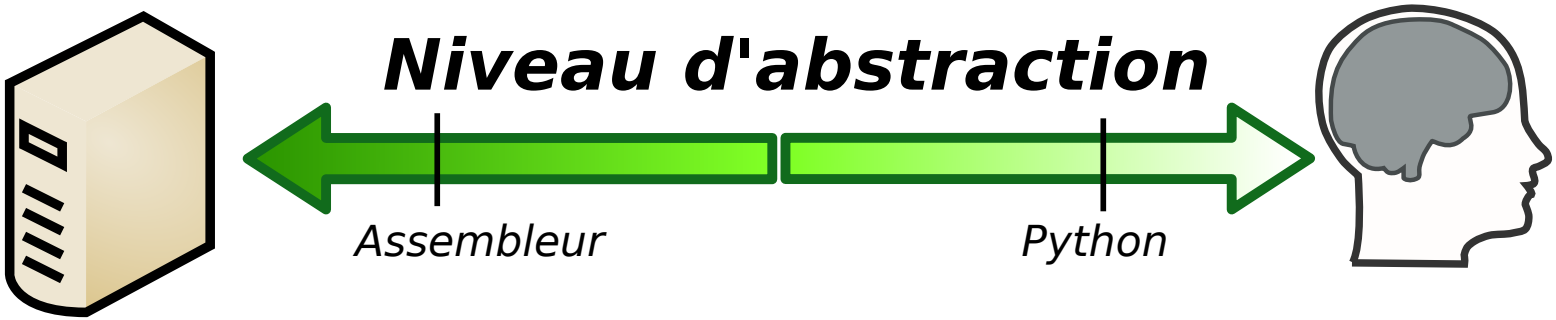
Python offre toute la puissance et la souplesse d'un **langage d'un programmation**. C'est donc un *meta-outil* qui permet de **forger** des outils spécifiques. L'outil développé est donc toujours **adapté** à la tâche a effectuée.

Or, il existe un grand nombre de langage de programmation qui, en théorie, sont tous équivalents (ils sont dits *turing complets*).

➥ Alors, Parmi tous les langages disponibles pourquoi choisir Python ?

Pourquoi Python ?

C'est un langage de dernière génération avec un *fort niveau d'abstraction*




```
1 tab = ['C++', 'Python', 'Java']
2 if 'Python' in tab:
3     print 'hello giens!'
```

➥La syntaxe de Python est **intuitive**

Pourquoi Python ?

Car python est un langage **interprété** qui lui confère une grande souplesse d'utilisation ⚠ problème de performance ! [1]

```
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> msg = 'hello giens! '
>>> print msg*4
hello giens! hello giens! hello giens! hello giens!
>>>
```

L'interpréteur interactif Python

➥ Python est un langage facile à mettre en oeuvre

1. Why python is slow ? https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/

Pourquoi Python ?

Car Python est un **outil libre**, disponible sous toutes les **plateformes** qui fédère une trés **vaste communauté** d'utilisateur.

PyPi [2], le dépôt tiers officiel de Python recense plus de 105 875 package !

Parmi ces dépôts, se trouvent *matplotlib*, *numpy* et *scipy* qui forment un véritable couteau suisse pour le calcul scientifique...

2. pypi.python.org

Traitement scientifique avec Python

Traçons la fonction $f = sin(x)$ avec *matplotlib* et *numpy*

```
1  # import numpy and matplotlib
2  import matplotlib.pyplot as plt
3  import numpy as np
4
5  # init values
6  x = np.linspace(0., 2*np.pi, 100)
7  y = np.sin(x)
8
9  # plot them
10 plt.figure()
11 plt.plot(x,y, 'o-')
12 plt.show()
```

Partie pratique

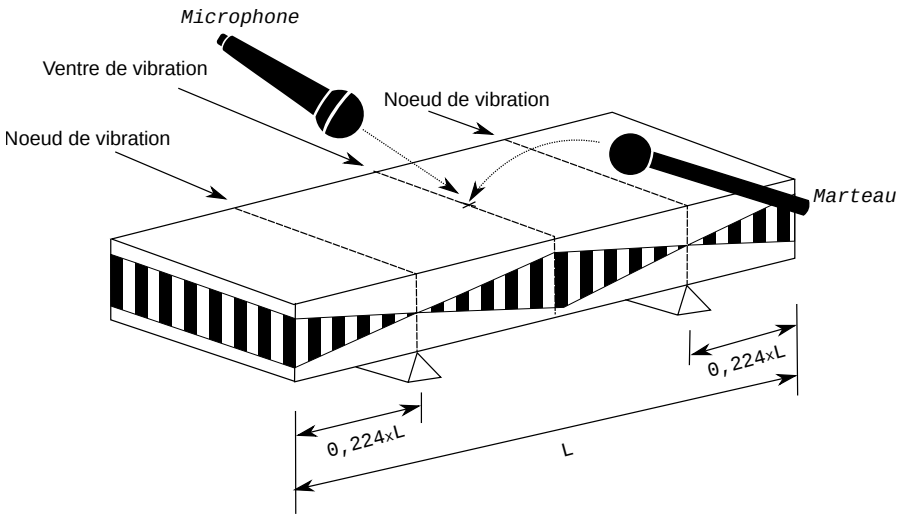
Zoom sur l'application pyrfda

La technique « RFDA »

La technique RFDA (Resonant Frequency Damping Analysis) ou IET (Impulse Excitation Technique) [3] est une méthode expérimentale permettant de mesurer les constantes d'élasticité (E,G) à partir du son produit par une plaque lorsque celle-ci est impactée.

3. *Standard Test Method for Dynamic Young's Modulus, Shear Modulus, and Poisson's Ratio by Impulse Excitation of Vibration* 2005

Dispositif expérimental en flexion



Montage en flexion

Calcul du module de Young E :

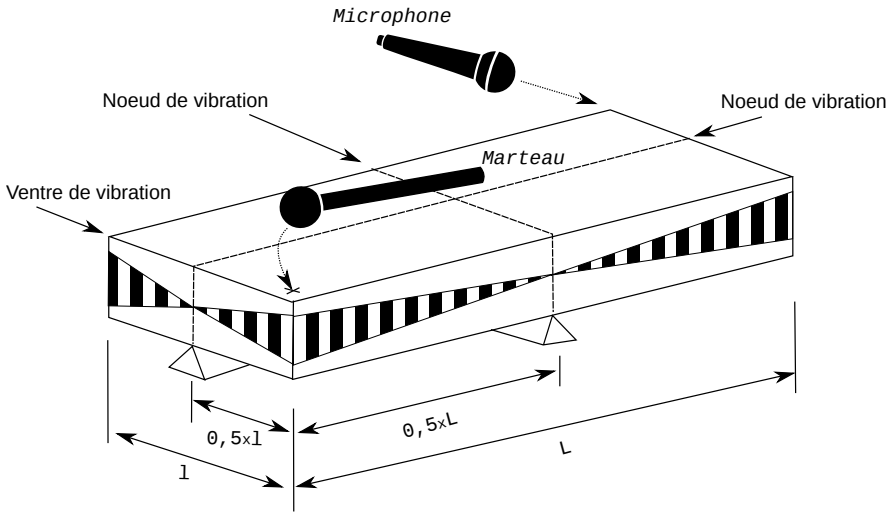
$$E = 0,9465 \left(\frac{m f_f^2}{b} \right) \left(\frac{L^3}{t^3} \right) T$$

Où T est un facteur correctif empirique :

$$T = 1 + 6,585 \left(\frac{t}{L} \right)^2$$

avec E le module de Young, m la masse, f_f la fréquence résonance en flexion, b la largeur, L la longueur, t l'épaisseur de l'échantillon

Dispositif expérimental en torsion



Montage en torsion

Calcul du module de Cisaillement G :

$$G = \frac{4Lmf_t^2}{bt}\frac{B}{1+A} \qquad (1)$$

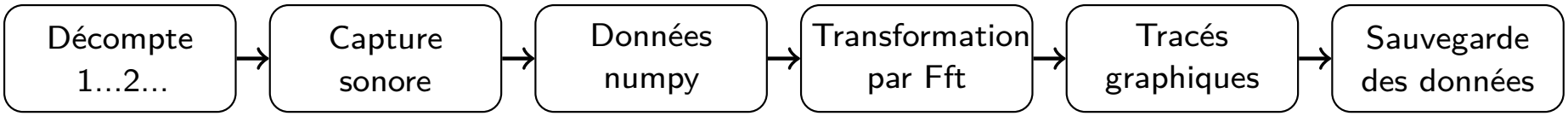
Où A et B sont des facteurs correctifs empiriques (voir annexe)

avec E le module de Young, m la masse, f_t la fréquence résonance en torsion, b la largeur, L la longueur, t l'épaisseur de l'échantillon

Vue générale de l'algorithme

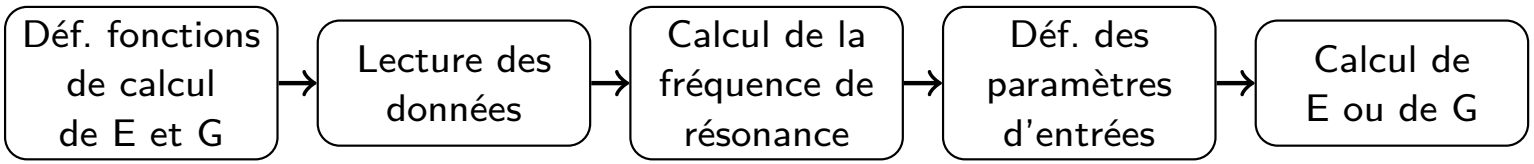
Partie 1/2, fichier `listen.py`

Acquisition + traitement + sauvegarde des données

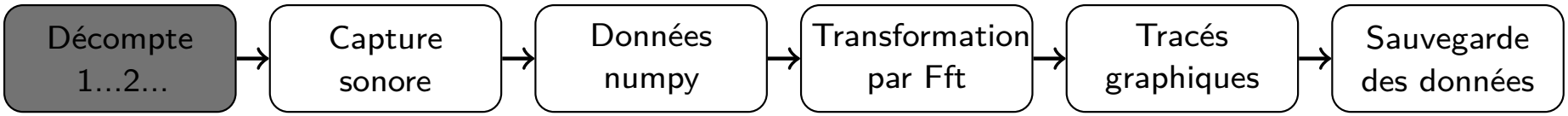


Partie 2/2, fichier `compute.py`

Lecture des données + traitement + calcul de E ou G



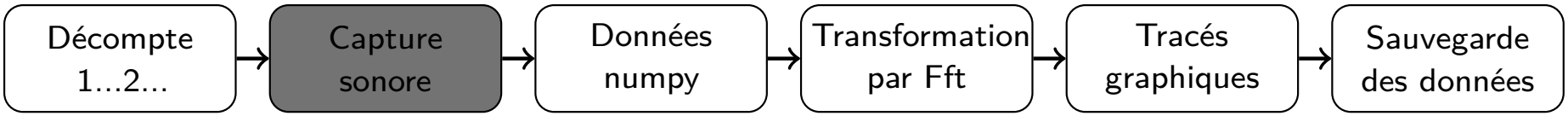
« `listen.py` », focus sur...




```
1 print '#### PART 0: Display delay before capturing sound'
2 import time
3 DELAY = 3 # INPUT: delay before capturing (s)
4
5 for i in range(DELAY):
6     print "start recording sound in", DELAY - i, "second(s)"
7     time.sleep(1)
8 print "start recording..."
```

Utilisation de la librairie *time* permettant de geler l'exécution du programme. Cette action est répétée `DELAY` fois.

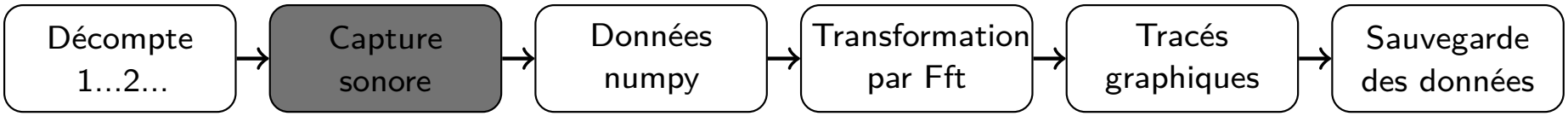
« `listen.py` », focus sur...




```
1 import pyaudio
2 RATE       = 44100  # INPUT: capture frequency (Hz)
3 TIME       = 2.     # INPUT: capture duraction (s)
4 CHUNKSIZE = 1024    # INPUT: buffer size (16 bit)
5 p = pyaudio.PyAudio()
6 stream = p.open(format=pyaudio.paInt16, channels=1, rate=RATE,
      input=True, frames_per_buffer=CHUNKSIZE)
```

Utilisation de la librairie *pyaudio* qui permet de manipuler les entrées/sorties audio. *pyaudio* est une interface python pour piloter la carte son.

« `listen.py` », focus sur...

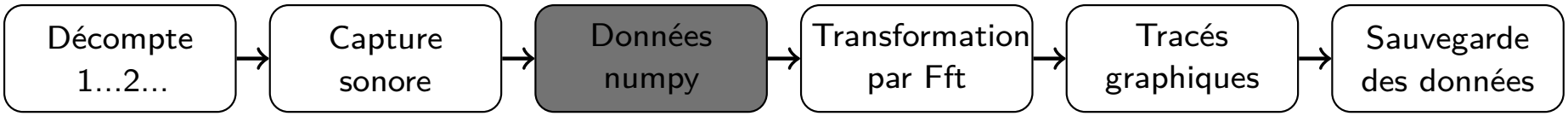



```
1 data = []
2 while len(data) < int((RATE / CHUNKSIZE) * TIME):
3     buf = stream.read(CHUNKSIZE)
4     data.append(buf)
5 # and close stream
6 stream.stop_stream();stream.close();p.terminate()
```

Le signal audio est stocké dans un buffer (mémoire tampon) `buf`. Une boucle permet de « pousser » le contenu de ces buffer dans une liste python `data`.

⚠ `data` est une véritable soupe de bits !

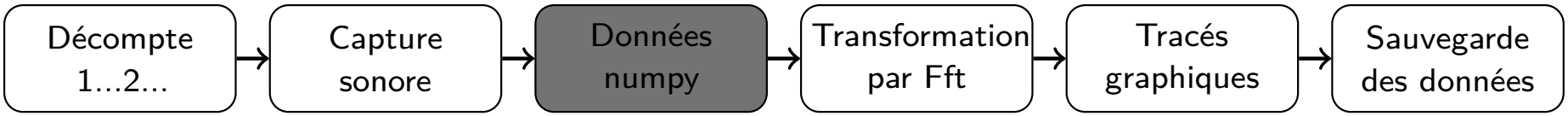
« `listen.py` », focus sur...




```
1 import numpy as np
2 signal = [np.fromstring(f, dtype=np.int16) for f in data]
```

Utilisation des listes de compréhension pour balayer la liste `data` et recréer une nouvelle liste. Les listes de compréhension permettent de réaliser « à la volée » des traitements sur les éléments d'une liste. Ici, la fonction `np.fromstring` est utilisée pour convertir chacun des éléments de la liste `data` en tableau d'entier 16 bits.

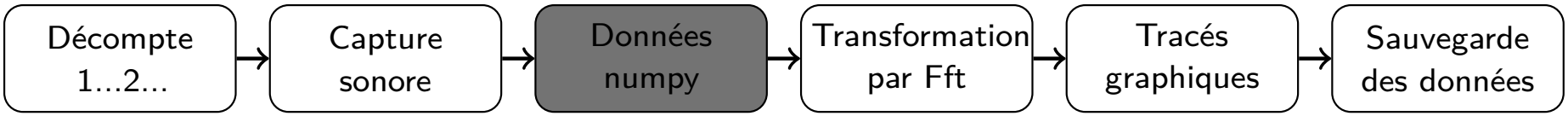
« `listen.py` », focus sur...

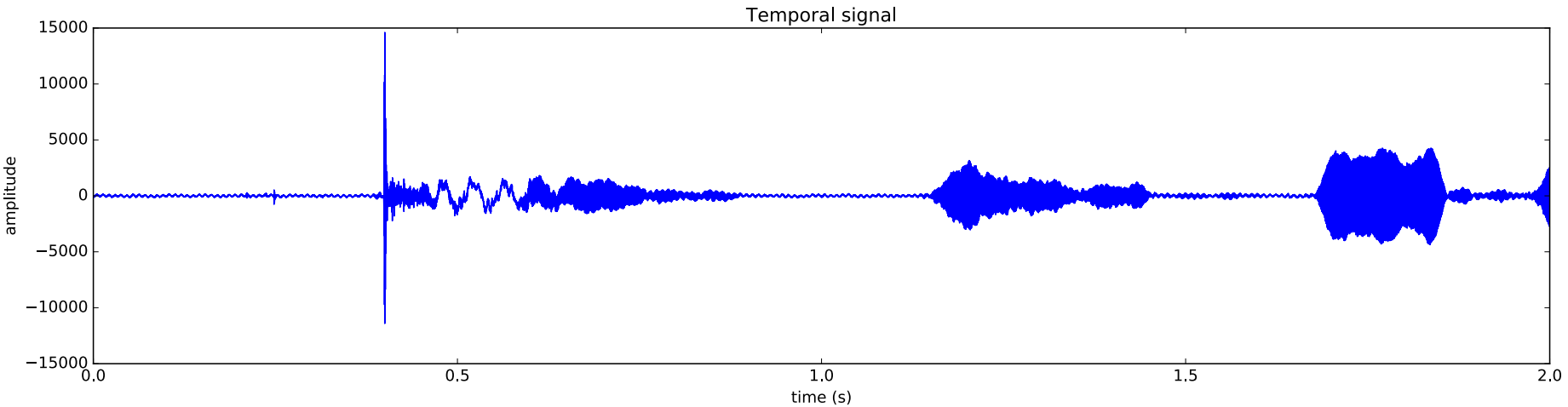



```
1 signal = np.hstack(signal)
2 time   = np.linspace(0, TIME, num=len(signal))
```

Utilisation de la fonction `np.hstack` pour concaténer un tableau 2D en un tableau 1D. Utilisation de la fonction `np.linspace` pour créer une séquence de valeurs qui représente le temps écoulé.

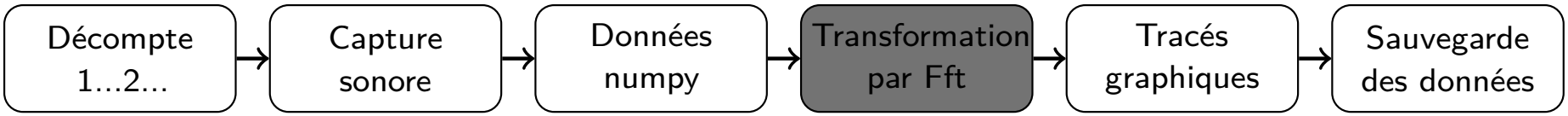
« `listen.py` », focus sur...





On peut ensuite réaliser le tracé graphique du signal en fonction du temps avec *matplotlib*

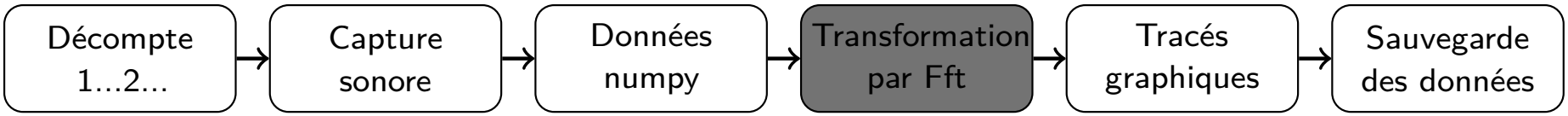
« `listen.py` », focus sur...




```
1 print '#### PART 3: spectral analysis of the audio signal'
2 yf = np.fft.fft(signal)
```

Utilisation de la fonction `np.fft.fft` pour transformer le signal temporel en signal fréquentiel à l'aide de la transformée de Fourier rapide. Le résultat `yp` est un tableau de nombres complexes.

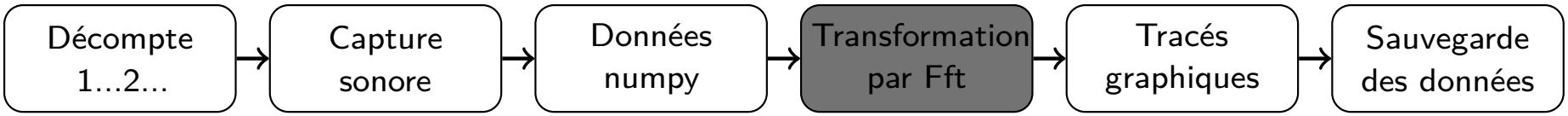
« `listen.py` », focus sur...




```
1 N = len(signal)
2 T = 1./RATE                          # the period of data
3 psd  = (T/N) * np.abs(yf)**2         # compute power spectral density
4 frq = np.fft.fftfreq(N, d=T)         # and the related frequencies
```

Afin de déterminer la fréquence de résonance, la densité spectrale de puissance est calculée à l'aide de la formule $\Gamma_x = |X|^2 T$. Les fréquences sont ensuite calculées à l'aide de la fonction `np.fft.fftfreq`

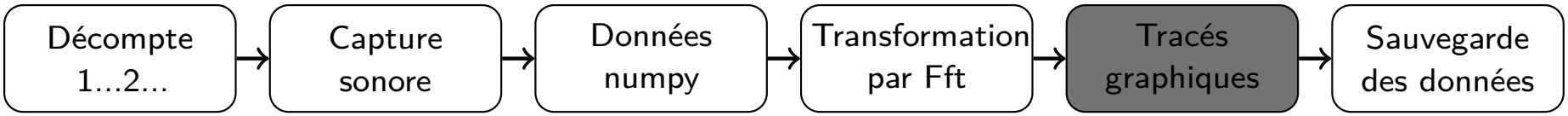
« `listen.py` », focus sur...




```
1 frq = frq[:N/2] # remove negative frequencies
2 psd = psd[:N/2] # and the related values of psd
```

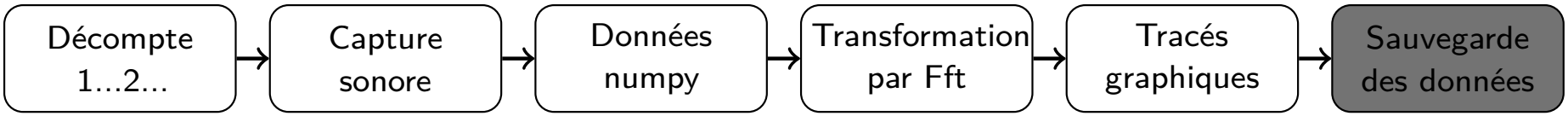
Finalement, seules les « moitiés droites » des tableaux sont conservées car les gauches contiennent les fréquences négatives. Ces opérations sont réalisées à l'aide du *slicing* python (tranchage).

« listen.py », focus sur...




```
1  print '#### PART 4: plot the results if wanted'
2  from matplotlib import pyplot as plt
3  plt.figure()
4  plt.subplot(2,1,1)
5  plt.plot(time, signal)
6  plt.xlabel('time (s)');plt.ylabel('amplitude')
7  plt.title("Temporal signal")
8
9  plt.subplot(2,1,2)
10 plt.plot(frq, psd)
11 plt.xlabel('frquency (Hz)');plt.ylabel('power spectral density')
12 plt.title("PSD versus frquency")
13 plt.show()
```

« `listen.py` », focus sur...



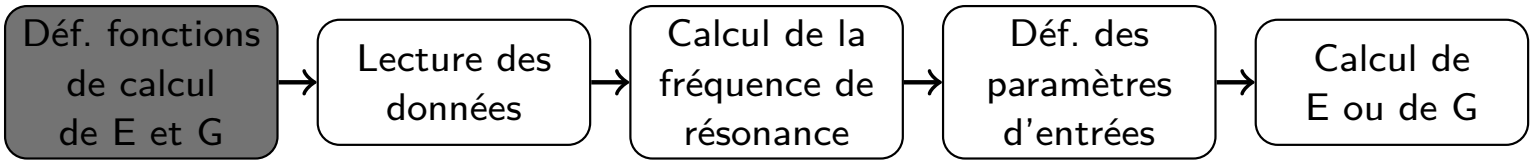

```
1 print '#### PART 5: save signals'
2 np.savez('results', frq=frq, psd=psd)
3 print '-> save results.npz , run "compute.py" script now'
```

Création d'un fichier `results.npz` dans le répertoire courant

Partie pratique (2/2)

Zoom sur l'application pyrfda

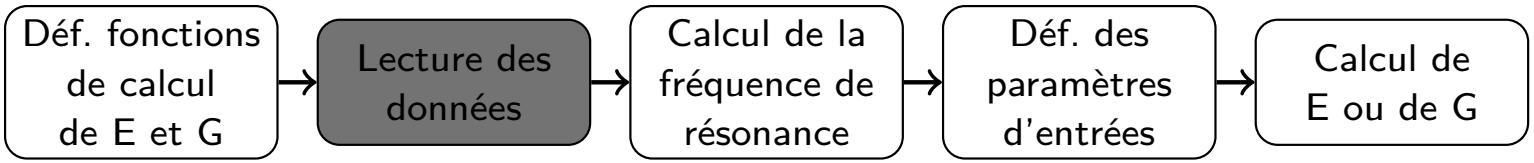
« compute.py », focus sur...




```
1  print '#### PART 0: define the function to compute E & G'
2  def compute_E(b, t, L, m, fb):
3      T=1. + 6.585*(t/L)**2.
4      return 0.9465*(m*(fb**2)/b)*((L/t)**3.)*T
5
6  def compute_G(b, t, L, m, ft):
7      B=((b/t)+(t/b))/(4.*(t/b)-2.52*((t/b)**2)+0.21*((t/b)**6))
8      A=(0.5062-0.8776*(b/t)+0.3504*((b/t)**2)-0.0078*((b/t)**3))
9      A/=(12.03*(b/t)+9.892*((b/t)**2))
10     return ((4.*L*m*(ft**2))/(b*t))*(B/(1+A))
```

➥ Application des formules pour calculer E et G

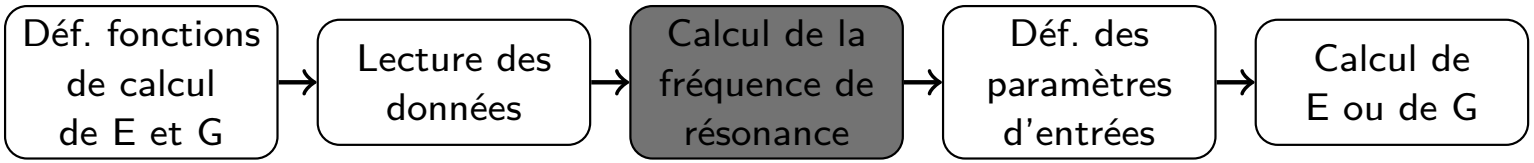
« compute.py », focus sur...




```
1 print '#### PART 1: read result file'
2 import numpy as np
3 npzfile = np.load('results.npz')
4 frq = npzfile['frq']
5 psd = npzfile['psd']
```

`npzfile` est un dictionnaire python qui permet d'associer des données, *ici des tableaux numpy*, à des clés, *ici des chaînes de caractères*.

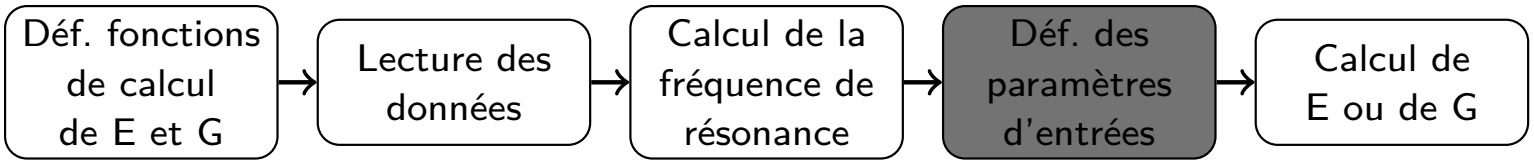
« compute.py », focus sur...




```
1 print '#### PART 2: compute the max psd and the related
      frequency'
2 rank = psd.argmax()
3 maxf = frq[rank]
4 print 'the max frequency is {0:.2f} Hz'.format(maxf)
```

La méthode `argmax()` renvoie l'index de la valeur maximale contenue dans le tableau numpy `psd`. L'opérateur crochet `[]` permet de renvoyer la valeur de la fréquence associée à cet index.

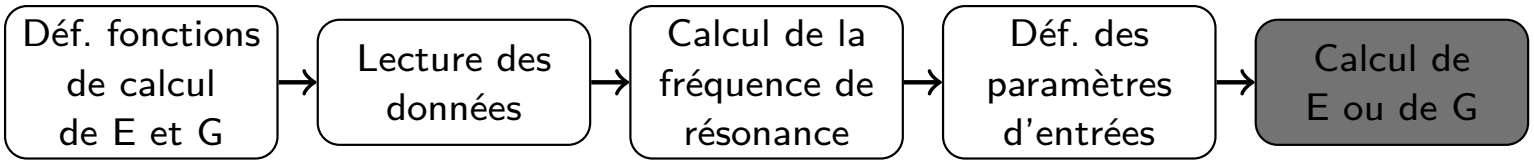
« compute.py », focus sur...




```
print '#### PART 3: input settings'
b     = 0.05  # INPUT: sample width     (m)
t     = 0.004 # INPUT: sample thickness (m)
L     = 0.15  # INPUT: sample length    (m)
m     = 0.081 # INPUT: sample mass      (kg)
test  = 'b'   # INPUT: 'b'=bending or 't'=torsion
```

Permet à l'utilisateur de rentrer les caractéristiques de l'échantillon

« compute.py », focus sur...



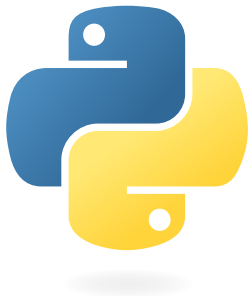

```
1 print '#### PART 4: compute result (E or G)'
2 if test is 'b':
3     E = compute_E(b, t, L, m, maxf)
4     print "The Young's modulus E is {0:.2f} GPa".format(E*1e-9)
5 elif test is 't':
6     G = compute_G(b, t, L, m, maxf)
7     print "The shear modulus G is {0:.2f} GPa".format(G*1e-9)
8 else:
9     print "Sorry, I can't understand"
```

Utilisation de tests conditionnels, puis application de la méthode de calcul

Conclusion

Python est un langage de programmation :

- facile à apprendre avec une syntaxe intuitive,
- facile à mettre en oeuvre grâce à son interpréteur,
- libre, gratuit et très populaire dans la communauté scientifique,
- qui possède de très nombreuses librairies(modules),
- élégant... whate else ?



Bon dev ! !

Annexes

Calcul du module de cisaillement

$$G = \frac{4Lmf_t^2}{bt}\frac{B}{1+A}$$

où

$$A = \frac{0,5062 - 0,8776\left(\frac{b}{t}\right) + 0,3504\left(\frac{b}{t}\right)^2 - 0.0078\left(\frac{b}{t}\right)^3}{12,03\left(\frac{b}{t}\right) + 9.892\left(\frac{b}{t}\right)^2}$$

et

$$B = \frac{\left(\frac{b}{t}\right) + \left(\frac{t}{b}\right)}{4\frac{t}{b} - 2,52\frac{t}{b}^2 + 0,21\left(\frac{b}{t}\right)^6}$$

avec G le module de cisaillement, m la masse, f_t la fréquence de résonance en torsion, b la largeur, L la longueur et t l'épaisseur de l'échantillon

Merci de votre attention !

Références

Bibliographiques

Références

Standard Test Method for Dynamic Young's Modulus, Shear Modulus, and Poisson's Ratio by Impulse Excitation of Vibration (2005). Standard. West Conshohocken, PA, 19428-2959 USA : ASTM.