Programmation en Python

Cours 6/8

Laurent Pointal

laurent.pointal @ limsi.fr
@ laposte.net

Source des illustrations: OpenClipArt.org et créations personnelles.

Une application: générer un signal sonore

Objectif

- Créer une **séquence de valeurs numériques** pouvant correspondre à l'*échantillonnage* d'un signal sonore.
- L'enregistrer dans un **fichier son** "wave".
- **Jouer** le fichier son.

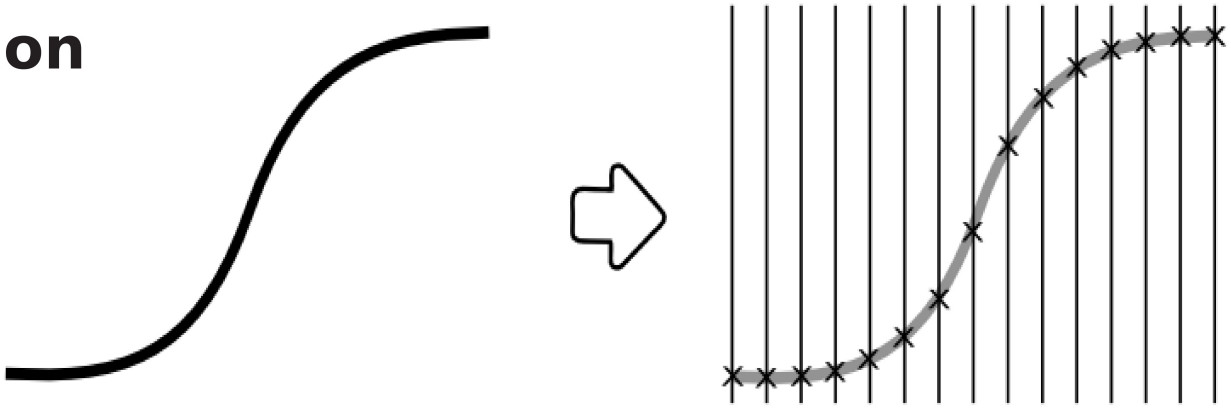
Une partie physique/mathématique

Une partie cuisine informatique

Une partie audition

Signal Sonore

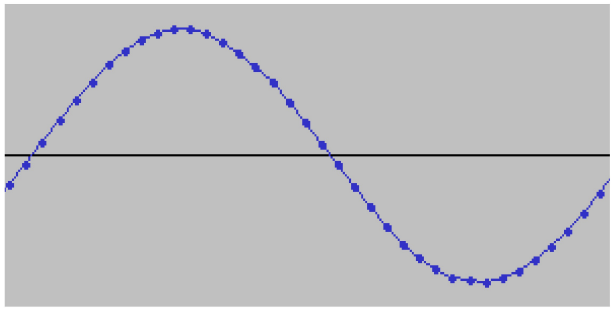
Numérisation



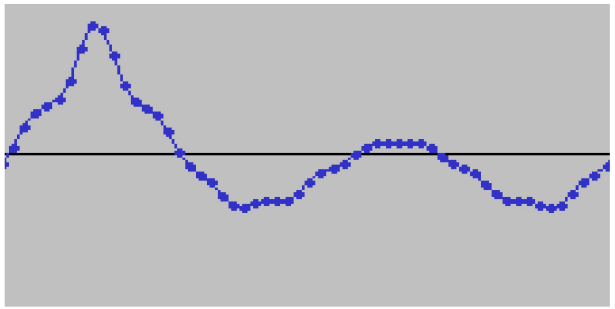
Signal périodique

- Fréquence de base (F_0)
- Décomposition en Harmoniques

Exemples



Sinusoïde simple



Composition
de Sinusoïdes

Synthèse de Fourier - Harmoniques

Fonction **périodique** de fréquence $\mathbf{F_0}$, décomposé en une somme infinie de *fonctions sinusoïdales* de fréquences multiples de F_0 (les ***harmoniques***).

- Avec **coefficients** pour les **sinus** et **cosinus** des harmoniques:

$$h(t)=a_0+\sum_{n=1}^{\infty}\left(a_n\cos(2\pi n F_0 t)\cdot b_n\sin(2\pi n F_0 t)\right)$$

- Avec **coefficient** et **déphasage** des harmoniques:

$$h(t)=a_0+\sum_{n=1}^{\infty}\left(c_n\cos(2\pi n F_0 t+\varphi_n)\right)$$

Si le sujet vous intéresse, voir Transformée de Fourier, en S3, module "proba/analyse harmonique" puis "traitement du signal"

Un lien à voir “Synthèse de Fourier” :

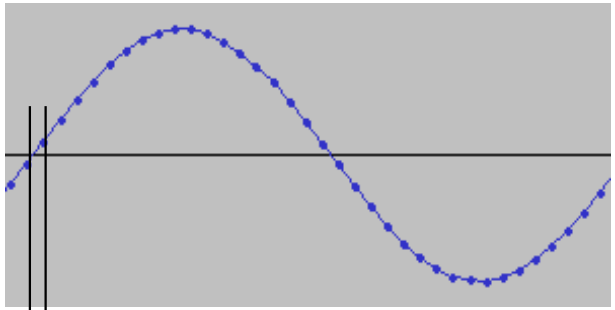
http://www.sciences.univ-nantes.fr/physique/perso/gtulloue/Elec/Fourier/fourier1.html

Créer une sinusoïde - paramètres

Les paramètres pour le calcul pour une harmonique

Durée totale

Période sinusoïde
$T_{harmo}=1/(n*F_0)$



a_n et b_n

Période échantillonnage
$T_{echt}=1/F_{echt}$

Créer une sinusoïde - fonction


```
def creer_harmonique(f0,n,freq_echant,
                     duree_signal=1,an=0,bn=0):
    "Génère un signal sinusoidal."
    echantillons = []
    for i in range(int(freq_echant*duree_signal)) :
        t = i/freq_echant      # en secondes
        a = 2*pi* n*f0 * t     # en radians
        v = an * cos(a) + bn * sin(a)
        echantillons.append(v)
    return echantillons
```

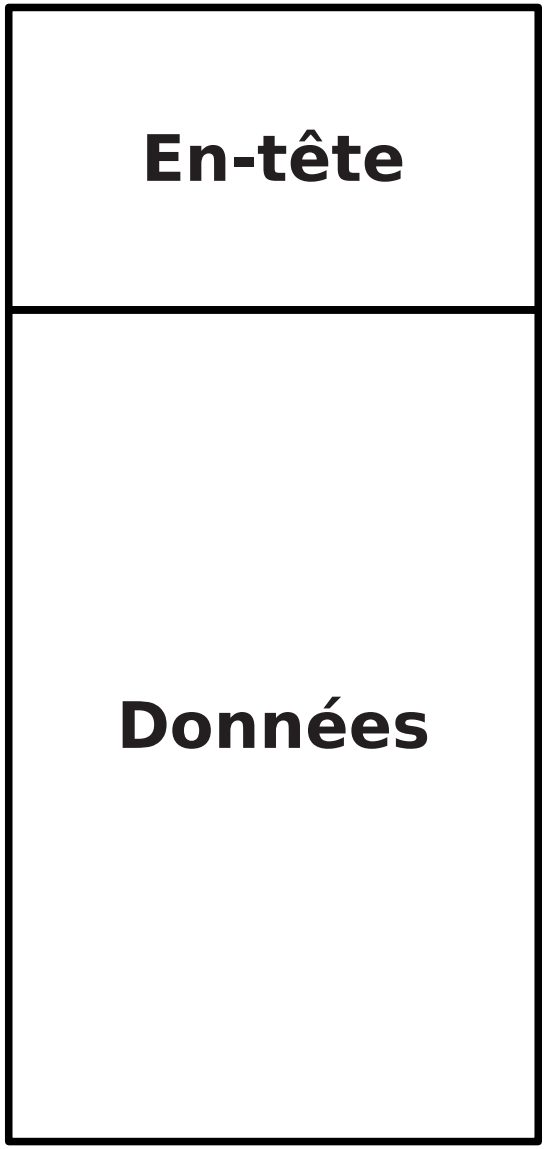
Sommer les sinusoïdes des harmoniques

Recomposer un signal à partir de ses *harmoniques*

Important: avoir la même fréquence d'échantillonnage.

```
def cumuler_signaux(*canaux) :
    "Cumule plusieurs signaux en un seul."
    res = []
    nbcanaux = len(canaux)
    nbecht = len(canaux[0])
    for iecht in range(nbecht) :
        vecht = 0
        for icanal in range(nbcanaux) :
            vecht = vecht + canaux[icanal][iecht]
        res.append(vecht)
    return res
```

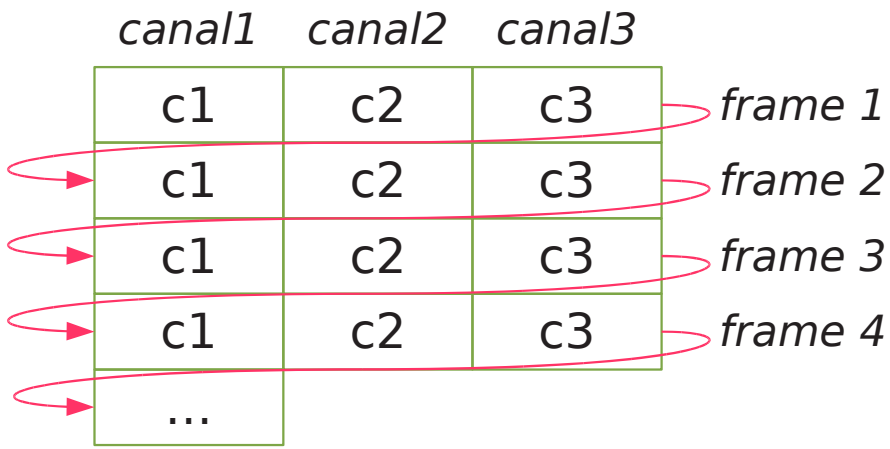
Format fichier son wave (simplifié)



- Nombre de canaux (mono, stéréo, 5ch…)
- Fréquence d'échantillonnage
- Nombres de blocs ("frames")
- …

Amplitudes des échantillons = Nombres entiers signés sur 16 bits (2 octets)
Limites: -32768 à 32767

Plusieurs canaux
➞ valeurs regroupées par bloc ("frame")



Voir: https://ccrma.stanford.edu/courses/422/projects/WaveFormat/

Utiliser le module wave

Éviter d'avoir à faire la "cuisine informatique" soi-même !

➔ **Module standard Python wave**

- **Documentation modules Python**
- **Exemples d'utilisation sur le web**

```
import wave,struct
# Ouverture du fichier à créer, et paramétrage du format WAV.
fson = wave.open(nomfichier, 'w')
fson.setnchannels(nbcanaux) # nombre canaux
fson.setsampwidth(2) # Nombre d'octets par échantillon
fson.setframerate(freqecht) # Fréquence d'échantillonnage.
fson.setcomptype('NONE','non compresse') # Pas de compression.

valeurs = []
for nech in range(nbecht):
    for nc in range(nbcanaux) :
        v = canaux[nc][nech]
        # Transformation valeur pour l'écrire sur 2 octets
        v_rep_binaire = struct.pack("<h", int(v))
        valeurs.append(v_rep_binaire)
valeurs_bloc = b''.join(valeurs)
fson.writeframes(valeurs_bloc)
fson.close()
```

Jouer le son

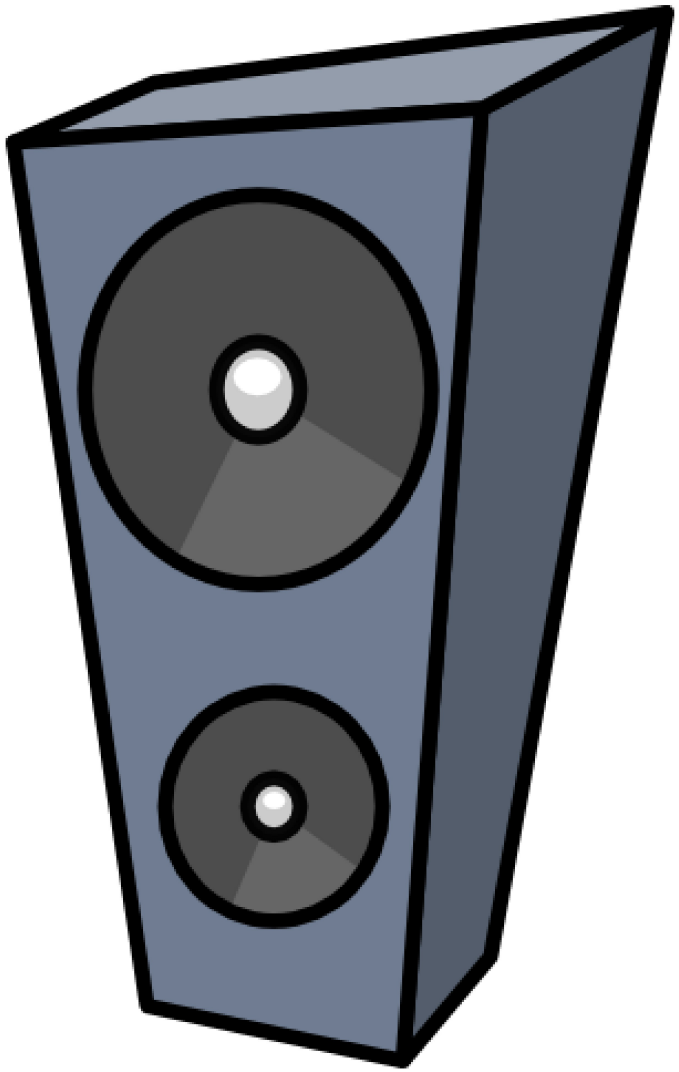
Possible d'utiliser des fonctions dépendantes du système d'exploitation utilisé.

Autre solution : demander au système de lancer l'application qui se charge des fichiers sons.

```
import webbrowser
def jouer_fichier_son(nomfichier) :
    "Lance l'application pour fichier son."
    webbrowser.open(nomfichier)
```

Pour en plus voir l'***aspect du signal***:
logiciel libre **Audacity**

Code & Audition…



Pour aller plus loin

Traitement du signal

Transformée de Fourier, en S3, module "proba/analyse harmonique" puis "traitement du signal"

Psycho-accoustique

La perception du son par le cerveau, avec des démos à écouter
(en) http://www.santafevisions.com/csf/demos/audio/index.htm

Informatique

Logiciels orientés “signal”

- PureData - `http://puredata.info/`
- LabView

Site musiques-rb (création musicale avec Pyhton)

http://www.musiques-rb.org/