L'écriture et la lecture du contenu d'un fichier audio au format WAV est simplifiée par l'utilisation du module `wave`.

Ecriture

Voici un exemple de création d'un fichier audio (PCM 8 bits stéréo 44100 Hz)
avec une forme d'onde sinusoïdale : [embed]http://fsincere.free.fr/isn/python/download/audio/son1.wav[/embe

```
# script audio.py
# (C) Fabrice Sincère ; Jean-Claude Meilland
import wave
import math
import binascii

print("Création d'un fichier audio au format WAV (PCM 8 bits stéréo 44100 Hz)")
print("Son de forme sinusoïdale sur chaque canal\n")

NomFichier = 'son.wav'
Monson = wave.open(NomFichier,'w') # instanciation de l'objet Monson

nbCanal = 2    # stéreo
nbOctet = 1    # taille d'un échantillon : 1 octet = 8 bits
fech = 44100   # fréquence d'échantillonnage

frequenceG = float(input('Fréquence du son du canal de gauche (Hz) ? '))
frequenceD = float(input('Fréquence du son du canal de droite (Hz) ? '))
niveauG = float(input('Niveau du son du canal de gauche (0 à 1) ? '))
niveauD = float(input('Niveau du son du canal de droite (0 à 1) ? '))
duree = float(input('Durée (en secondes) ? '))

nbEchantillon = int(duree*fech)
print("Nombre d'échantillons :",nbEchantillon)

parametres = (nbCanal,nbOctet,fech,nbEchantillon,'NONE','not compressed')# tuple
Monson.setparams(parametres)    # création de l'en-tête (44 octets)

# niveau max dans l'onde positive : +1 -> 255 (0xFF)
# niveau max dans l'onde négative : -1 ->   0 (0x00)
# niveau sonore nul :                0 -> 127.5 (0x80 en valeur arrondi)

amplitudeG = 127.5*niveauG
amplitudeD = 127.5*niveauD

print('Veuillez patienter...')
for i in range(0,nbEchantillon):
    # canal gauche
```

```
    # 127.5 + 0.5 pour arrondir à l'entier le plus proche
    valG = wave.struct.pack('B',int(128.0 + amplitudeG*math.sin(2.0*math.pi*frequenceG*i/fe
    # canal droit
    valD = wave.struct.pack('B',int(128.0 + amplitudeD*math.sin(2.0*math.pi*frequenceD*i/fe
    Monson.writeframes(valG + valD) # écriture frame

Monson.close()

Fichier = open(NomFichier,'rb')
data = Fichier.read()
tailleFichier = len(data)
print('\nTaille du fichier',NomFichier, ':', tailleFichier,'octets')
print("Lecture du contenu de l'en-tête (44 octets) :")
print(binascii.hexlify(data[0:44]))
print("Nombre d'octets de données :",tailleFichier - 44)
Fichier.close()
```

```
>>>
Création d'un fichier audio au format WAV (PCM 8 bits stéréo 44100 Hz)
Son de forme sinusoïdale sur chaque canal

Fréquence du son du canal de gauche (Hz) ? 440.0
Fréquence du son du canal de droite (Hz) ? 880.0
Niveau du son du canal de gauche (0 à 1) ? 1.0
Niveau du son du canal de droite (0 à 1) ? 0.5
Durée (en secondes) ? 2.5
Nombre d'échantillons : 110250
Veuillez patienter...

Taille du fichier son.wav : 220544 octets
Lecture du contenu de l'en-tête (44 octets) :
b'52494646785d030057415645666d74201000000000100020044ac000088580100020008006461746 1545d0300'
Nombre d'octets de données : 220500
>>>
```

Le script crée le fichier `son.wav` dans le répertoire courant (sous Windows c'est normalement le répertoire `C:/PythonXX`, mais on peut aussi choisir un emplacement quelconque en spécifiant le chemin absolu, par exemple `NomFichier = 'F:/Mon dossier/son.wav'`).

Ecouter le son Vous pouvez écouter le fichier `son.wav` avec un lecteur multimédia quelconque (VLC par exemple). Notez que Python sait faire directement une lecture audio avec le module adéquat (module externe `pygame.mixer`, module `ossaudiodev` sous Linux, module `winsound` sous Windows, etc...) :

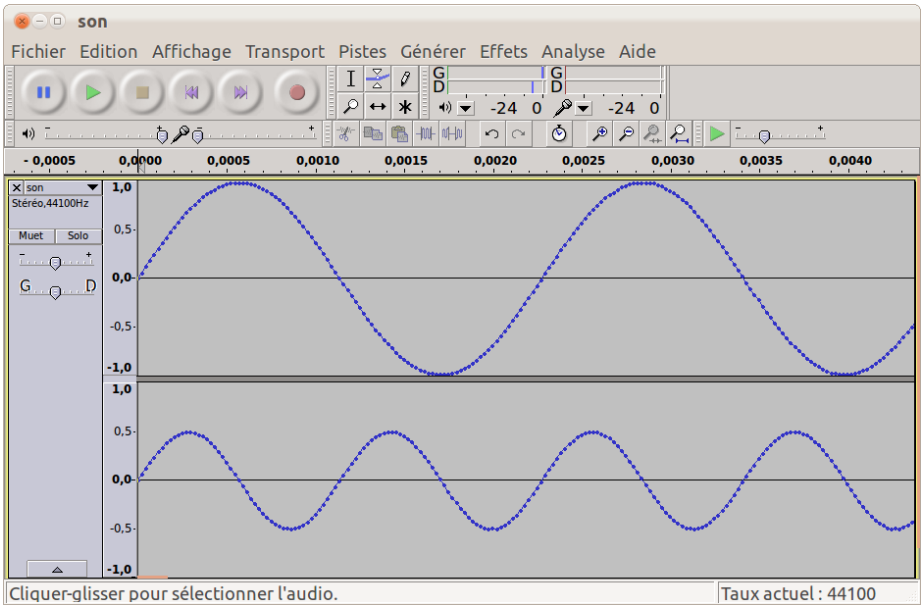
```
# lecture audio (sortie vers la carte son)
import winsound
```

```
winsound.PlaySound('son.wav',winsound.SND_FILENAME)
```

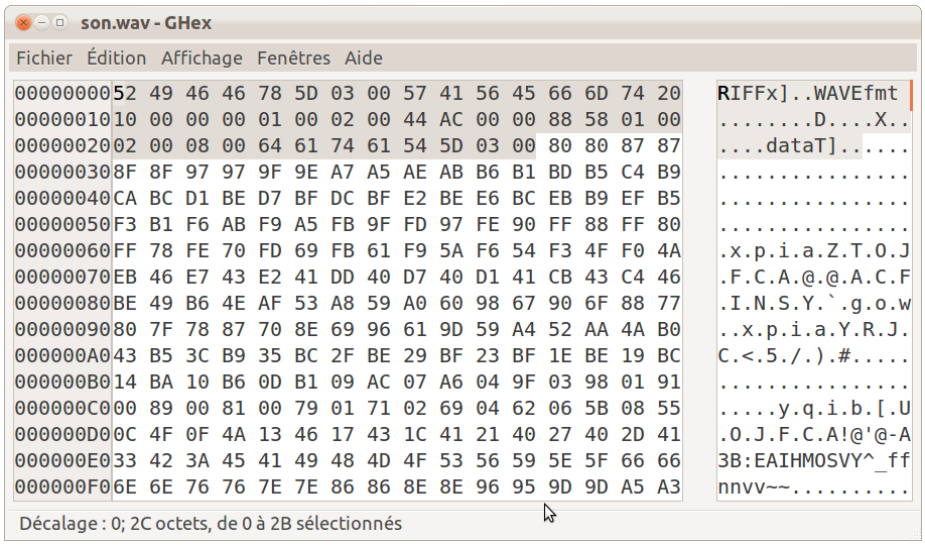
Le module `pygame` est un module externe de création de jeux vidéo en 2D. `pygame` contient un sous module `pygame.mixer` qui permet de charger et de lire des musiques ou des sons dans plusieurs formats (mp3, ogg, wav...). La procédure d'installation de `pygame` se trouve ici.

import pygame pygame.mixer.init() pygame.mixer.Sound("son.wav").play() while pygame.mixer.get_busy(): # lecture en cours pass

Editeur audio On peut compléter l'étude du fichier `son.wav` avec un éditeur de sons tel que Audacity :



Editeur hexadécimal Un éditeur hexadécimal est aussi utile :



Lecture

Le script suivant fonctionne avec tous les fichiers audio au format WAV (PCM sans compression).

```
# (C) Fabrice Sincère
import wave
import binascii

NomFichier = input('Entrer le nom du fichier : ')
Monson = wave.open(NomFichier,'r')  # instanciation de l'objet Monson

print("\nNombre de canaux :",Monson.getnchannels())
print("Taille d'un échantillon (en octets):",Monson.getsampwidth())
print("Fréquence d'échantillonnage (en Hz):",Monson.getframerate())
print("Nombre d'échantillons :",Monson.getnframes())
print("Type de compression :",Monson.getcompname())

TailleData = Monson.getnchannels()*Monson.getsampwidth()*Monson.getnframes()

print("Taille du fichier (en octets) :",TailleData + 44)
print("Nombre d'octets de données :",TailleData)

print("\nAffichage d'une plage de données (dans l'intervalle 0 -",Monson.getnframes()-1,")")

echDebut = int(input('N° échantillon (début) : '))
echFin = int(input('N° échantillon (fin) : '))
```

```
print("\nN° échantillon Contenu")

Monson.setpos(echDebut)
plage = echFin - echDebut + 1
for i in range(0,plage):
    print(Monson.tell(),'\t\t',binascii.hexlify(Monson.readframes(1)))

Monson.close()
```

Avec le fichier créé précédemment, on obtient :

```
>>>
Entrer le nom du fichier : son.wav

Nombre de canaux : 2
Taille d'un échantillon (en octets): 1
Fréquence d'échantillonnage (en Hz): 44100
Nombre d'échantillons : 110250
Type de compression : not compressed
Taille du fichier (en octets) : 220544
Nombre d'octets de données : 220500

Affichage d'une plage de données (dans l'intervalle 0 - 110249 )
N° échantillon (début) : 0
N° échantillon (fin) : 5

N° échantillon  Contenu
0        b'8080'
1        b'8787'
2        b'8f8f'
3        b'9797'
4        b'9f9e'
5        b'a7a5'
>>>
```

Un échantillon est ici constitué de deux octets : le premier correspond à l'échantillon du canal de gauche, le second à celui du canal de droite.

Exercices

Exercice 9.1 Un fichier audio WAV (PCM non compressé) possède les propriétés suivantes : stéréo, 16 bits, 44100 Hz, taille du fichier 97896 octets 1) Quelle est la taille de l'en-tête (en octets) ? 2) Quelle est la taille des données (en octets) ? 3) Quelle est la durée du son de ce fichier (en ms) ? **Exercice 9.2** Bruit blanc A partir du script `audio.py`, écrire le script d'un bruit blanc. Bibliographie : Audacity → Générer → Bruit **Exercice 9.3** Silence ! A partir du script `audio.py`, écrire le script d'un générateur de silence. Bibliographie :

Audacity → Générer → Silence **Exercice 9.4** Code DTMF (dual-tone multi-frequency) Ecrire un générateur de tonalités DTMF. Par exemple :

```
>>>
Numéro de téléphone ? 0619283745

Veuillez patienter...

Le fichier dtmf.wav a été créé
>>>
```

[embed]http://fsincere.free.fr/isn/python/download/audio/dtmf.wav[/embed]
Exercice 9.5 Echo acoustique Ecrire un script qui ajoute un écho. Pour rester simple, on ne traitera que des fichiers audio WAV non compressés, de résolution 8 bits, mono (un seul canal). On ne tiendra pas compte des échos multiples.

```
>>>
Le fichier audio doit avoir le format WAV non compressé, 8 bits, mono

Nom du fichier ? balla.wav

Fréquence d'échantillonnage (en Hz) : 11025
Taille du fichier (en octets) : 38345

Niveau de l'écho (0 à 1) ? 0.3
Retard (en seconde) ? 0.6

Le fichier resultat.wav a été créé avec succès.
>>>
```

[embed]http://fsincere.free.fr/isn/python/download/audio/balla.wav[/embed]
[embed]http://fsincere.free.fr/isn/python/download/audio/resultat.wav[/embed]

Webographie

- Documentation sur le module wave
- WAVEform audio file format
- PCM (Pulse Code Modulation)
- Site officiel du logiciel libre et gratuit Audacity
- Lecteur multimédia
- Editeur hexadécimal
- Code DTMF
- Module pygame.mixer (documentation)