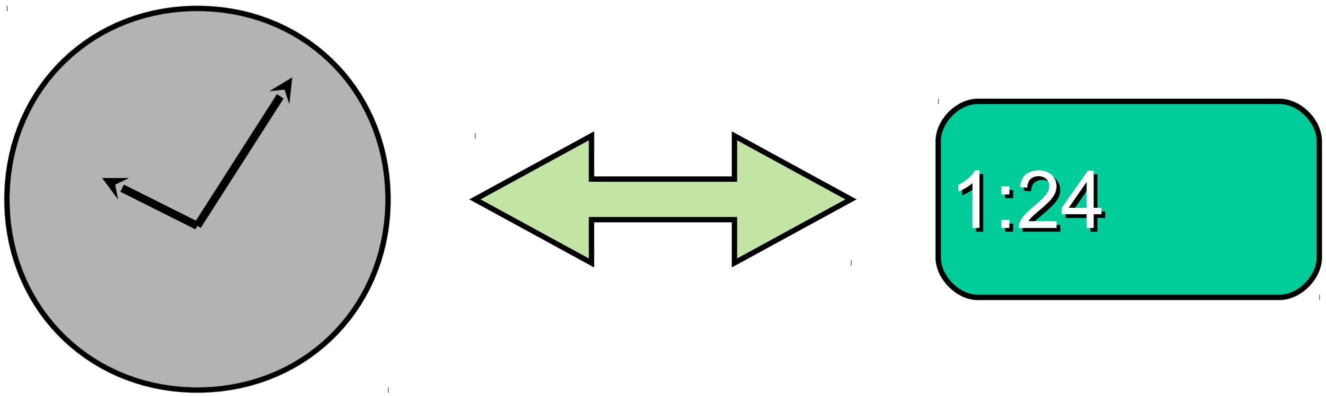
Multimedia Programming

Audio Programming With Python

What is Digital Audio?

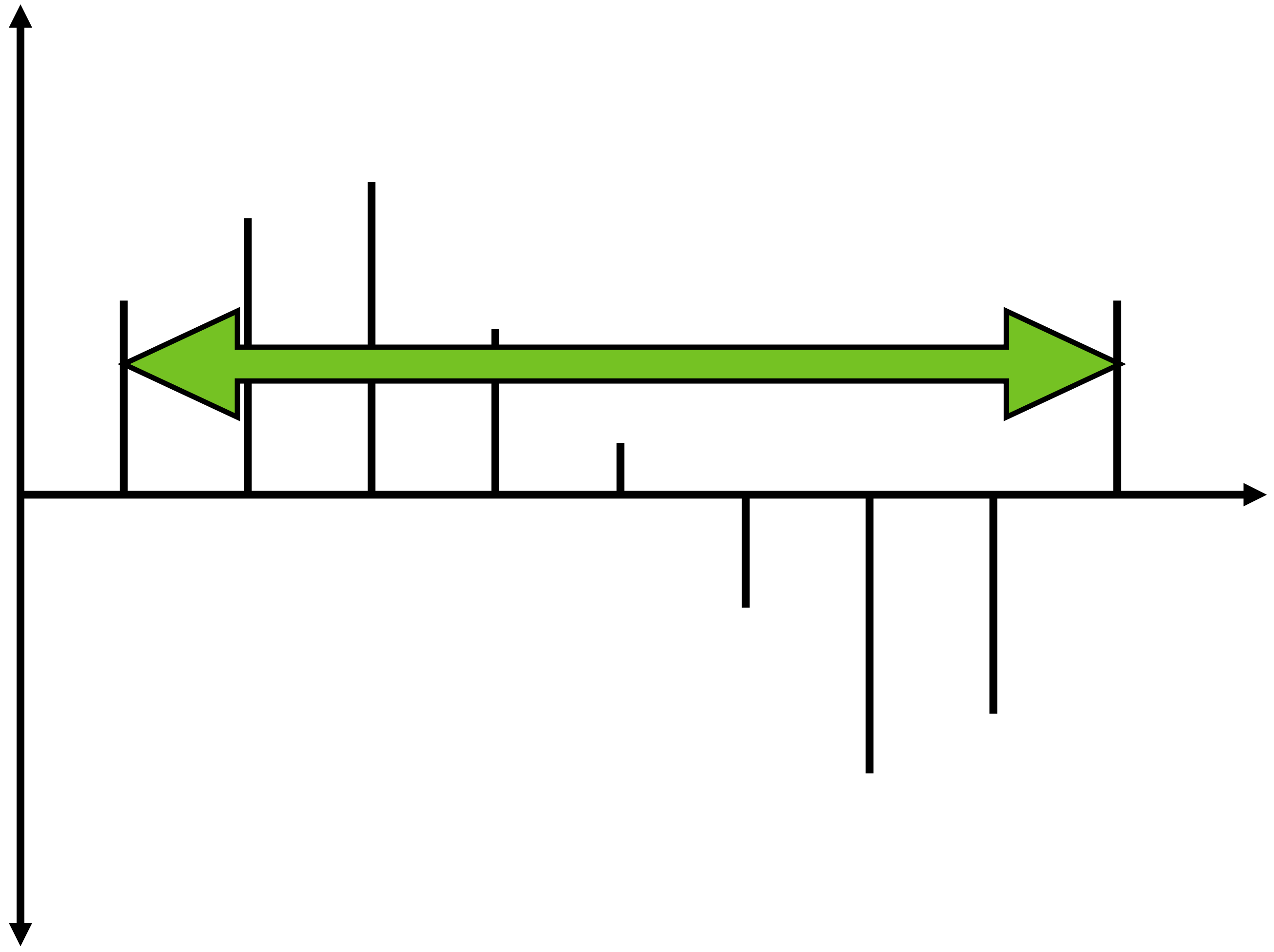
Digital audio is like a digital clock.



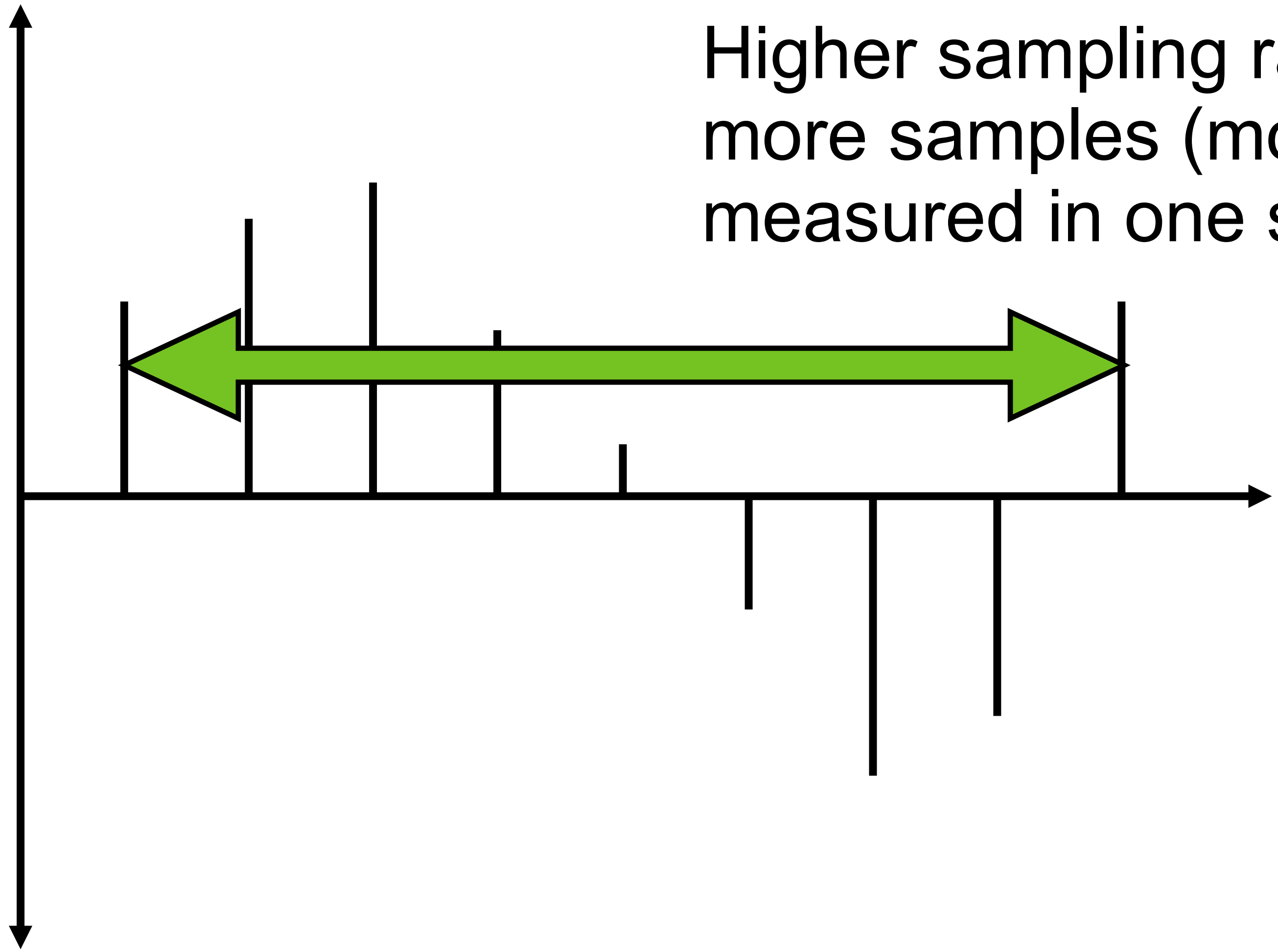
It's a representation of a continuous signal as a series of discrete numbers.

Instead of telling the time, the discrete numbers are measurements of a sound pressure wave at a specific point in time.

The sampling rate is the rate at which the sound wave is measured.

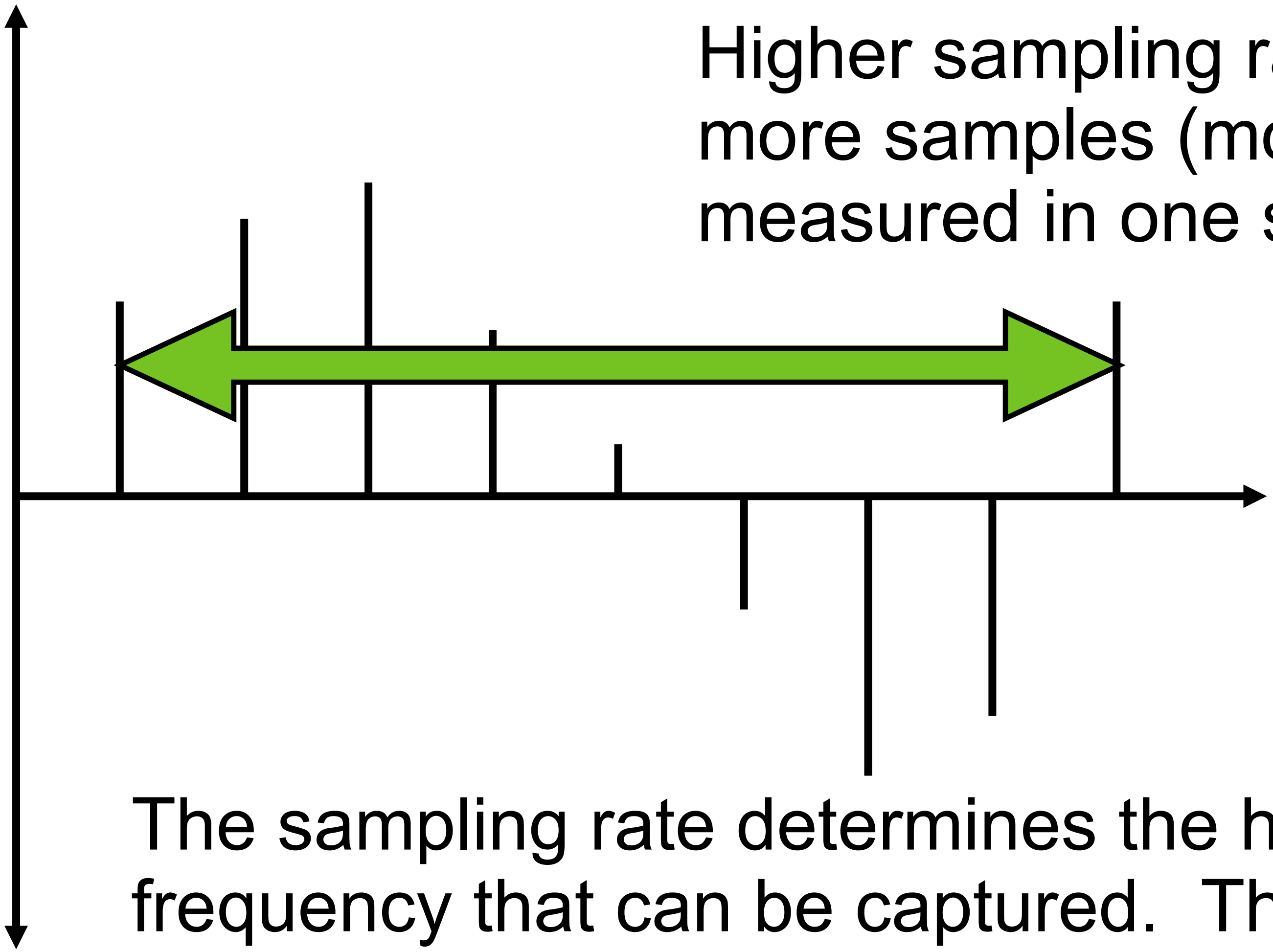


The sampling rate is the rate at which the sound wave is measured.



Higher sampling rates means more samples (more data) is measured in one second.

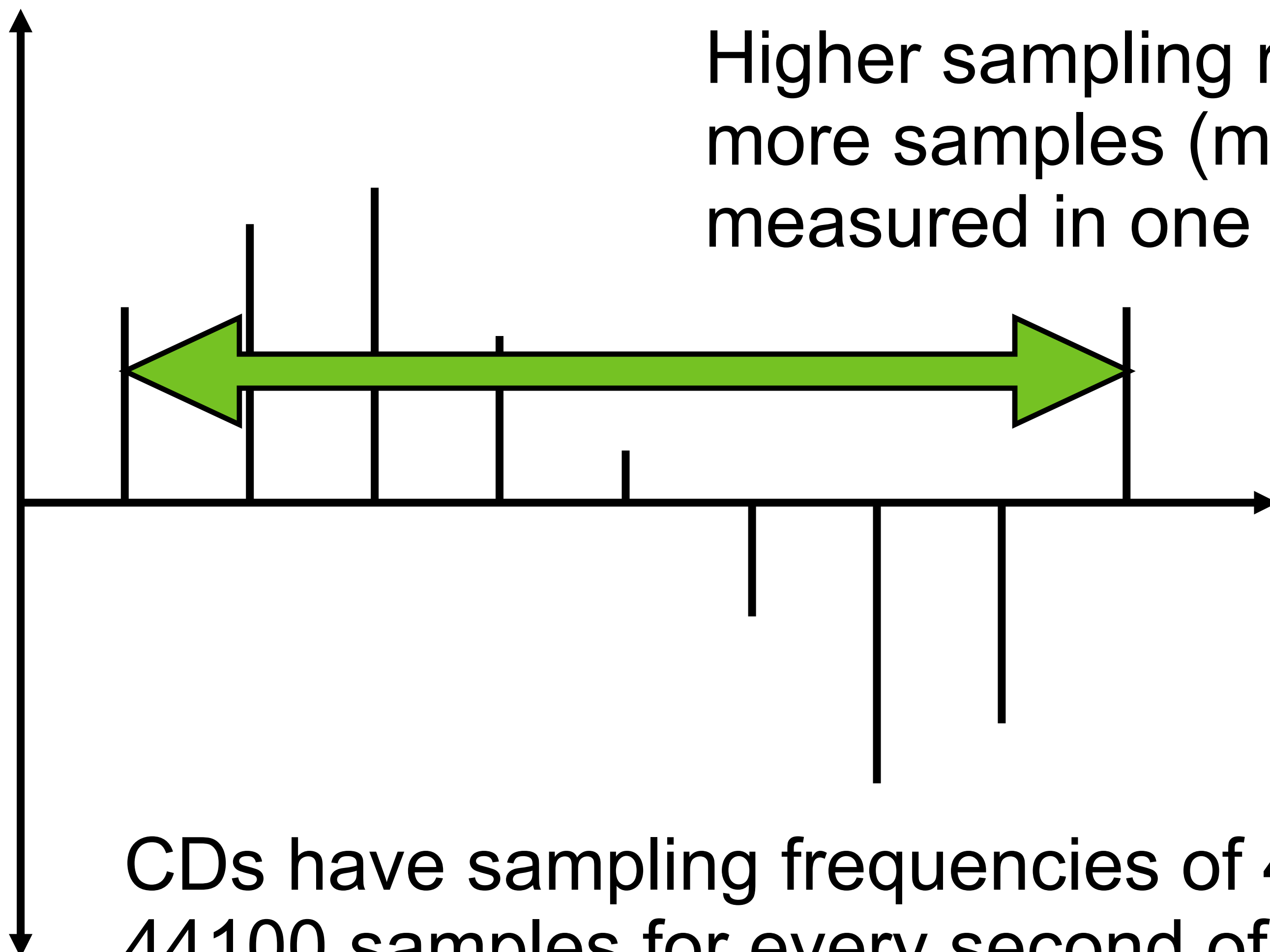
The sampling rate is the rate at which the sound wave is measured.



Higher sampling rates means more samples (more data) is measured in one second.

The sampling rate determines the highest audio frequency that can be captured. The faster the sampling rate, the higher the audio frequency.

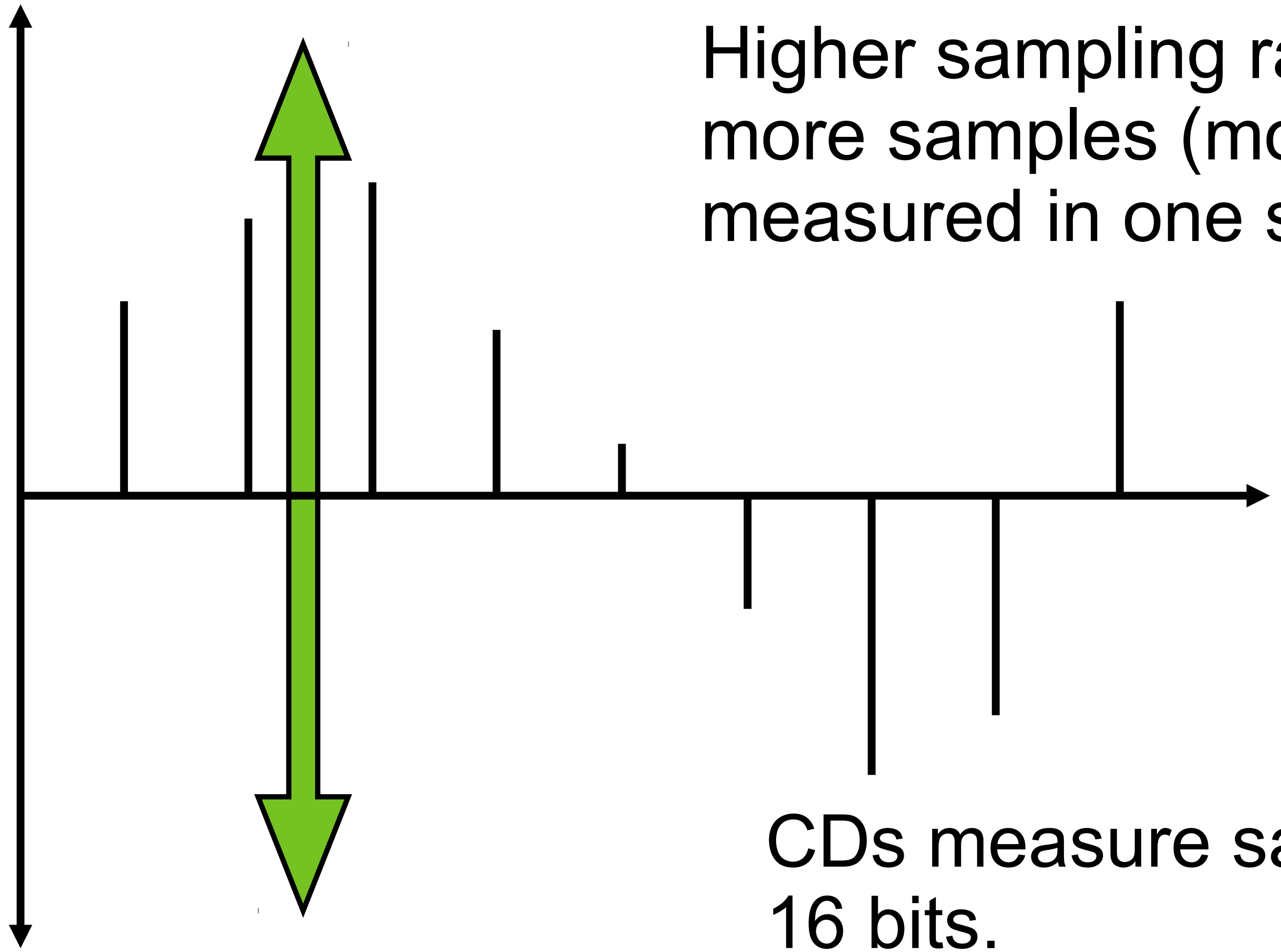
The sampling rate is the rate at which the sound wave is measured.



Higher sampling rates means more samples (more data) is measured in one second.

CDs have sampling frequencies of 44.1 kHz.
44100 samples for every second of audio.

The sampling rate is the rate at which the sound wave is measured.



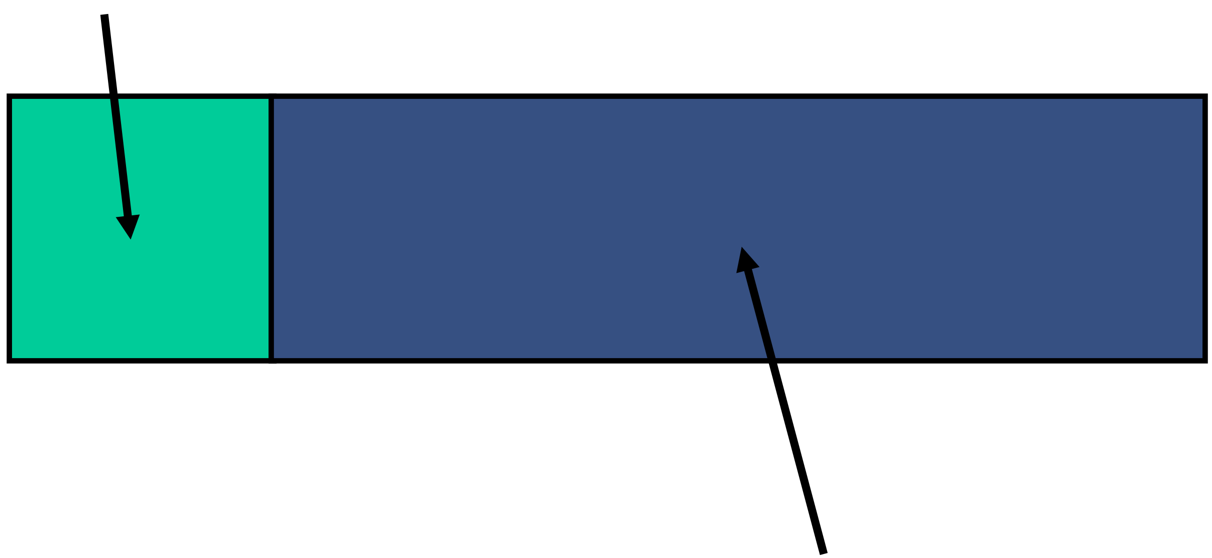
Higher sampling rates means more samples (more data) is measured in one second.

CDs measure samples with 16 bits.

How is Audio Stored?

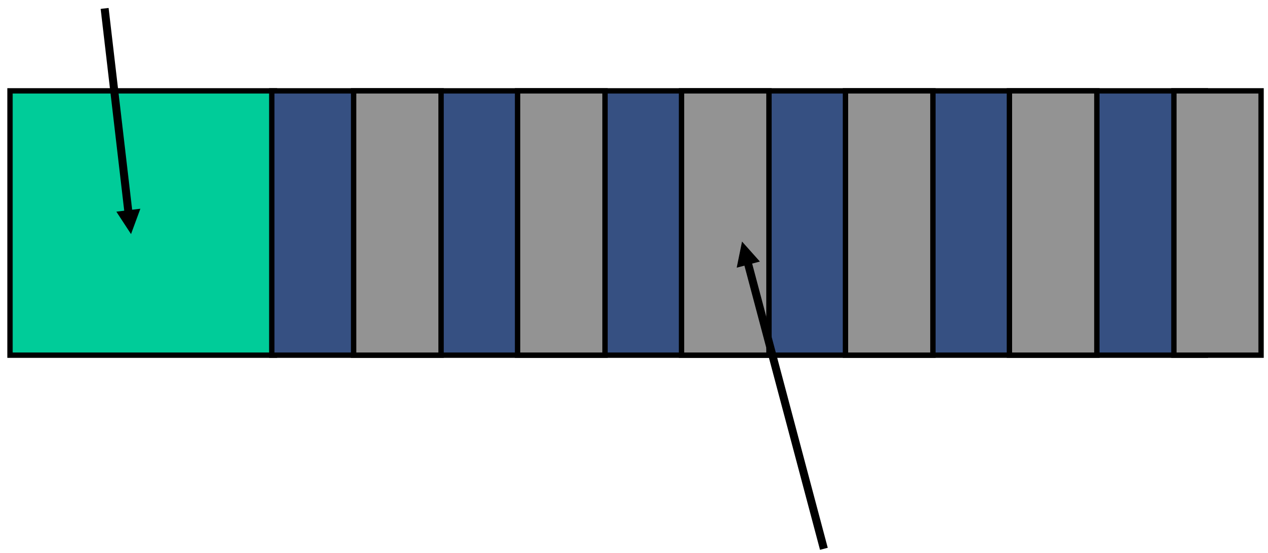
Audio is commonly stored in the wav file format.

The wav format has header information at the beginning which tells the computer what the sampling rate and bit depth is for the digital audio samples.



After the header are the audio samples.

The header also lists how many channels of audio there are in the file.



If there is more than one channel of audio (stereo audio is 2 channels), short sections of each audio channel are alternated.

File formats like mp3 don’t store a list of samples.

The samples undergo psychoacoustic compression to reduce the number of total bytes needed to represent the audio recording.

But the file format similar to a wav file as there is header information and then audio data.

But you need the right codec to decode the data into audio samples.

Audio Processing in Python: audiolab

We will cover how to read in audio data from a file, play that audio data from Python, and output audio data to a file.

We will use the Python module **audiolab**, but will quickly review some other available modules at the end.

Python has very limited built-in functionality for audio, so it's necessary to use third party modules.

audiolab is a simple set of modules with fewer dependencies than other options, but is also less interactive and less flexible.

To install audiolab

Go to

http://cournape.github.com/audiolab/installing.html

for full documentation.

Binary can be found at

http://pypi.python.org/pypi/scikits.audiolab/

or you can install from source at

http://cournape.github.com/audiolab/

To install audiolab

You will need NumPy

If you install from the binary, libsndfile is included.

If you install from source, you will also need to install libsndfile

http://www.mega-nerd.com/libsndfile

Why are we using audiolab?

- Has a MATLAB-like interface
- Can read and write files using data in NumPy arrays
- Can play back audio (but can't do anything else while audio is playing: blocking)
- It is limited to reading and writing wav, aiff, ircam, ogg, au, and flac files only

Read an Audio File

MATLAB

This is what we would type in MATLAB to read a wav file.

```
[y, Fs, nbits, opts] = wavread(filename)
```

The audio data.

The sampling rate of the file.

The bit depth of the file.

This could be other information about the file included in the header.

The path to the file.

audiolab

This is what we would type in Python to read a wav file if we want to use the MATLAB-like interface

Import the method from the subpackage.

```
from scikits.audiolab import wavread
data, fs, enc = wavread('test.wav')
```

The NumPy array of the audio data.

The sampling rate of the audio file.

The encoding format.

The path to the wav file.

audiolab

This is what we would type in Python to read a wav file if we want to use the Sndfile object and not the MATLAB-like interface.

```
import numpy
from scikits.audiolab import Sndfile
```

Import the necessary packages.

audiolab

This is what we would type in Python to read a wav file if we want to use the Sndfile object and not the MATLAB-like interface.

```
import numpy
from scikits.audiolab import Sndfile

f = Sndfile('test.wav', 'r')
```

Create a Sndfile object with the path to the wav file

audiolab

This is what we would type in Python to read a wav file if we want to use the Sndfile object and not the MATLAB-like interface.

```
import numpy
from scikits.audiolab import Sndfile

f = Sndfile('test.wav', 'r')
fs = f.samplerate
nc = f.channels
enc = f.encoding
data_length = f.nframes
```

Read in the header information if you will need it later

audiolab

This is what we would type in Python to read a wav file if we want to use the Sndfile object and not the MATLAB-like interface.

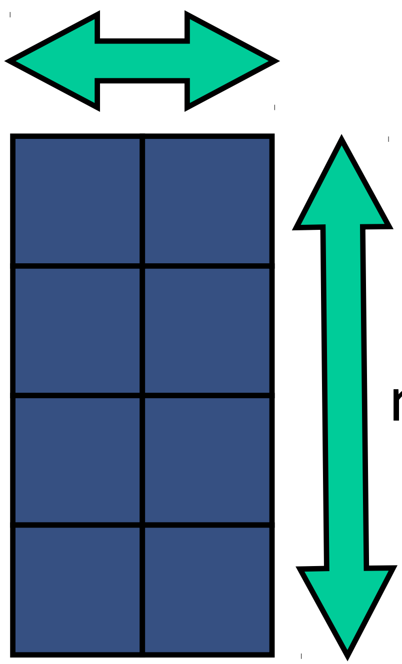
```
import numpy
from scikits.audiolab import Sndfile

f = Sndfile('test.wav', 'r')
fs = f.samplerate
nc = f.channels
enc = f.encoding
data_length = f.nframes

data = f.read_frames(1000, dtype=numpy.float32)
```

Read the first 1000 frames and store in a NumPy array

audiolab

```
f = Sndfile('test.wav', 'r')
f.read_data(num_of_frames, dtype=numpy.float32)
```

returns a NumPy array



Each column is one audio channel, so the number of columns is determined by `f.nc`

The number of rows is determined by the number of samples read in.

audiolab

Once the audio data is read into a NumPy array, we can use NumPy and other packages as we would for any other NumPy array.

```
import numpy as np

# data is a numpy array containing audio frames
# find the maximum value of this signal
sample_max = abs(max(data))
```

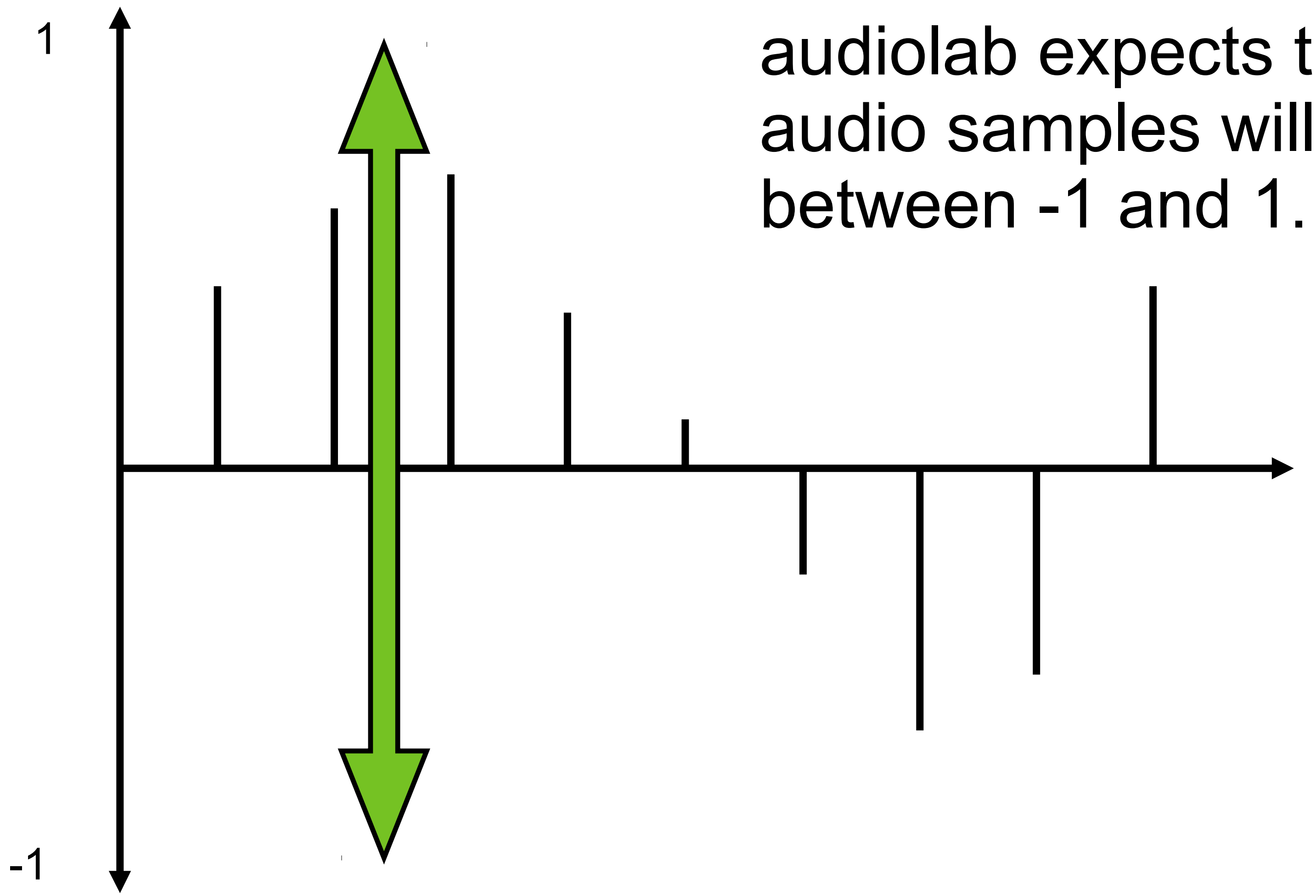
audiolab

Once the audio data is read into a NumPy array, we can use NumPy and other packages as we would for any other NumPy array.

```
import numpy as np

# data is a numpy array containing audio frames
# find the maximum value of this signal
sample_max = abs(max(data)

# find the RMS of the first channel of the signal
rms = np.power(np.sum(np.power(data[:0], 2)), 0.5)
```

Play an Audio Buffer

When trying to play audio or write to an audio file, audiolab expects that the audio samples will be between -1 and 1.



audiolab

```
# Output one second of stero noise
# with a sampling rate of 48 kHz
```

audiolab

```
# Output one second of stero noise
# with a sampling rate of 48 kHz
import numpy as np
from scikits.audiolab import play
```

Import the required packages.

software carpentry

audiolab

```
# Output one second of stero noise
# with a sampling rate of 48 kHz
import numpy as np
from scikits.audiolab import play
```

Create an array of random values.

```
my_noise = 2 * np.random.random_sample(2, 48000) - 1
play(my_noise, fs=48000)
```

Two channels and 48000 samples, so 1 second of audio at a sampling rate of 48 kHz.

audiolab

```
# Output one second of stero noise
# with a sampling rate of 48 kHz
import numpy as np
from scikits.audiolab import play


my_noise = 2 * np.random.random_sample(2, 48000) - 1
play(my_noise, fs=48000)
```

The audio samples can only be floating point values within [-1, 1]. Values outside that range will clip.

The default sampling rate is 44100, so if you want a different sampling rate you need to supply a keyword argument.

Writing an Array to File

audiolab

```
import numpy as np
from scikits.audiolab import Format, Sndfile
```

Import the required packages.

audiolab

```
import numpy as np
from scikits.audiolab import Format, Sndfile

# create one second of stereo noise
data = 2 * np.random.random_sample(2, 48000) - 1
```

Create data to output.

audiolab

```
import numpy as np
from scikits.audiolab import Format, Sndfile

# create one second of stereo noise
data = 2 * np.random.random_sample(2, 48000) - 1

# Create a Sndfile object for writing WAV files
# sampled at 48 kHz
format = Format('wav')
f = Sndfile('foo.wav', 'w', format, 2, 48000)
```

Format object describes the header information needed for a WAV file

audiolab

```
import numpy as np
from scikits.audiolab import Format, Sndfile

# create one second of stereo noise
data = 2 * np.random.random_sample(2, 48000) - 1

# Create a Sndfile object for writing WAV files
# sampled at 48 kHz
format = Format('wav')
f = Sndfile('foo.wav', 'w', format, 2, 48000)
```

'w' for writing to file.

The number of channels.

Sampling rate.

audiolab

```
import numpy as np
from scikits.audiolab import Format, Sndfile

# create one second of stereo noise
data = 2 * np.random.random_sample(2, 48000) - 1

# Create a Sndfile object for writing WAV files
# sampled at 48 kHz
format = Format('wav')
f = Sndfile('foo.wav', 'w', format, 2, 48000)

# Write the data and close the file
f.write_frames(data)
f.close()
```

Other Available Libraries

swmixer has more dependencies, but is more powerful

brian hears provides a lot of key functionality for psychoacoustics research

swmixer

Installation and documentation at
http://pypi.python.org/pypi/SWmixer and
http://code.google.com/p/pygalaxy/wiki/SWmixer
Dependencies: PyAudio, NumPy, MAD, PyMAD
(MAD and PyMAD for mp3 support)

File formats: wav and mp3 if you install MAD and pyMAD
Can read and write to file from numpy arrays
Can playback audio (non-blocking), so can be used in interactive applications with more playback control
Can stream, so good for large files
Can record from a microphone

brian hears

To install use easy_install brian
Documentation available at
http://www.brain-simulator.org/docs/reference-hears.html
Dependencies: NumPy, SciPy, Pylab (SymPy optional), PyGame for audio playback

File formats: wav and aiff
Can generate test tones easily
Good for psychoacoustics work
Has built-in functions for plots such as spectrograms
Need PyGame to play sounds

How to choose?

- Do I need to play audio?
- Interactive application?
- wav and/or also mp3?
- Large files or a lot of files at the same time?

If yes to any, then **swmixer**!

- Am I doing auditory modelling or psychoacoustics research?
- Not concerned with memory constraints?
- Only wav or aiff?

If yes to most, then **brian hears**!

If you just need to read in audio files and write them out to file again after some simple processing, then **audiolab** will suffice.

All of the libraries use NumPy as the format to store the audio samples

So you can mix and match the libraries to suit your needs

This requires a certain comfort-level with Python…

software carpentry

created by

Becky Stewart

June 2011