

# Audio In Python

Fall '18

# Modules:

## Built-In audio functionality

- PyAudio (record and play audio)
- Aubio (get pitches, beat detection)
- Pydub + Audio Segment (change volume, concat, stack, etc)

## More Complex (need to do audio manipulation yourself)

- Analyseffi (mac)
- SoundAnalyse (windows)
- Threading

# Common Projects

- Audio/music visualizer
- Sheet music reader
  - Create your own music sheets
- Games using sound input
- Mostly used with other modules to create cool things!

# Cool Things You Can Do With Sound

- Pitch Detection
- Beat Detection
- Volume Detection
- Getting rid of noise
- Threading with the rest of your project
- Does anyone have any ideas for audio projects right now?

# Past Projects

- Pulse

- Audio visualizer
- <https://www.youtube.com/watch?v=QLwTMGOUm10>

- Composition Software

- Music composition software
- <https://www.youtube.com/watch?v=P3Qar1B66Yc> (basic)
- <https://www.youtube.com/watch?v=yGkZrPUFBc4> (advanced)

- Screaming Bird

- Funny use of audio
- <https://www.youtube.com/watch?v=6lRaLpRxF9Y>

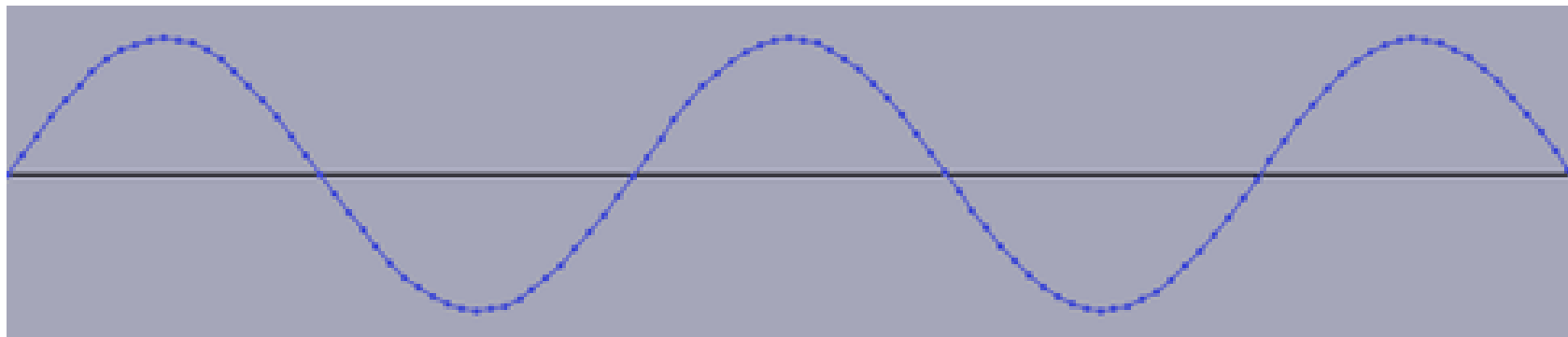
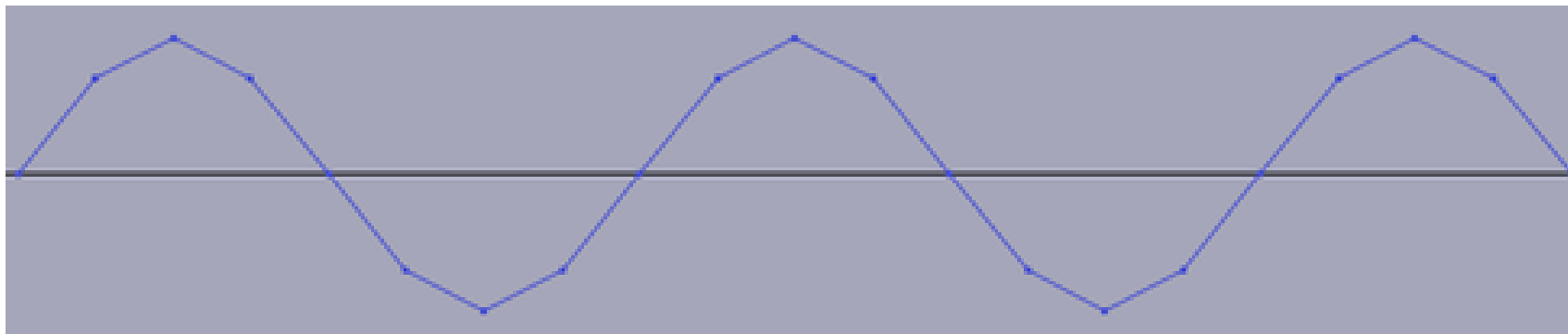
# How is music represented digitally: The WAV file

- Developed by Microsoft and IBM
- Simpler methods: Sounds are waves
- Why we care: It's easiest to get PyAudio to work with wav files
- Waves have amplitude and frequency
- Amplitude = volume
- Frequency = pitch

# Definitions: Chunks and Samples

- Chunk: piece of data storing information about sound
- Chunk size: Size of music data in bytes
- Channel: A singular waveform in autodata (ex: mono, stereo, surround)
- Sample: Scalar value representing amplitude of wave
- Frame: Snapshot of all samples at a given time
- Sampling Rate: Number of samples of data for each second (44100 Hz)

# Some Waves



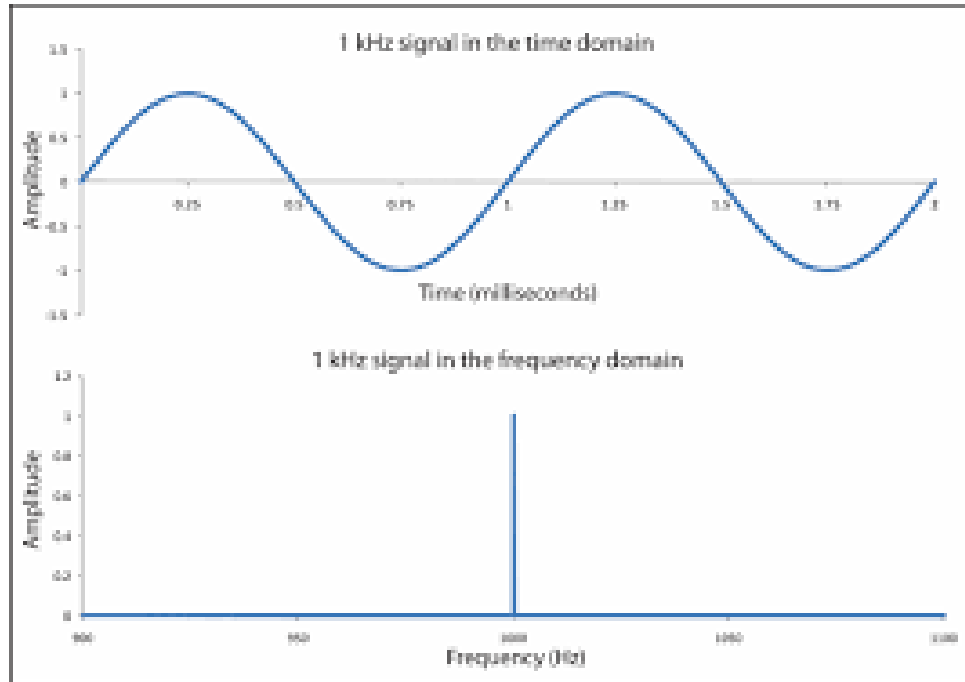


# How to get pitches? DTFT vs. DFT

$$X(e^{j\Omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\Omega n}$$

$$X_k = \sum_{n=0}^{N-1} x[n] \cdot e^{-j2\pi \frac{kn}{N}}$$

# Time Domain to Frequency Domain



# Challenges

- If there is a lot of data, slow computation  $O(n^2)$ ,  $n$  outputs, each evaluates a sum of  $n$  terms
  - Algorithms to make it faster: FFT, fastest at  $O(n \log n)$ 
    - Cooley-Turkey
    - Prime factor
    - Research them!
- Noise!
  - Your sound data isn't going to be perfect
  - Minimizing background noise
  - Humans don't sing perfect pitches

Let's do a demo!

# Analyseffi -- What it Does

- uses numpy and detect\_pitch (an internal function from the file analyze.py)
- use pyaudio to store the read CHUNK in a variable (data = stream.read(CHUNK))
- use numpy to properly format data (x = numpy.fromstring(data))
- use detect\_pitch to find the frequency being played (freq=detect\_pitch(x))
- from here, it is up to you to store the frequency and sort through the raw data

## Attendance

[https://goo.gl/forms/yas  
WOZEk6Dww28VK2](https://goo.gl/forms/yasWOZEk6Dww28VK2)

aykilinc@andrew.cmu.edu