# CSCI128 Lab 8

**Chris Greencorn**

## Lab 8

**1. Write a function *increaseVolumeNamed* that takes a file name as input then play the louder sound.**

```
def increaseVolumeNamed(filename):
  sound = makeSound(filename)
  for sample in getSamples(sound):
    value = getSampleValue(sample)
    setSampleValue(sample,value * 3)

  play(sound) # 3x louder = +9db
```

**2. Write a function that takes two inputs: A sound to increase in volume, and a multiplier. Use the multiplier as how much to increase the amplitude of the sound samples. Can we use this same function to both increase and decrease the volume?**

```
def increaseVolumeByFactor(filename,factor):
  sound = makeSound(filename)
  for sample in getSamples(sound):
    value = getSampleValue(sample)
    setSampleValue(sample,value * factor)
  play(sound)
```

The function can be used to increase/decrease the volume, but not by simply multiplying by a negative integer. Multiplying by a negative inverts inverts the phase of the of the waveform but increases the amplitude by the same factor. Using a percentage of 1 will decrease the volume, ex. 0.9 or 0.25.

**3. Rewrite the following so that you normalize the first second of a sound, then slowly decrease the sound in steps of 1/5 for each following second. (How many samples are in a second? getSamplingRate is the number of samples per second for the given sound.)**

Original:

```
def increaseAndDecrease(sound):
    for sampleIndex in range(1,getLength(sound)/2):
        value = getSampleValueAt(sound,sampleIndex)
        setSampleValueAt(sound,sampleIndex,value * 2)
    for sampleIndex in range(getLength(sound)/2,getLength(sound)+1):
        value = getSampleValueAt(sound,sampleIndex)
        setSampleValueAt(sound,sampleIndex,value * 0.2)
```

Rewritten:

```
def normalizeAndDrop(filename):
  # Normalizing First Second
  sound = makeSound(filename)
  samplesPerSecond = getSamplingRate(sound)
  sps = samplesPerSecond
  loudest = getSampleValueAt(sound,0)
  for sampleIndex in range(1,sps):
    if loudest < getSampleValueAt(sound,sampleIndex):
      loudest = getSampleValueAt(sound,sampleIndex)
    normfactor = 32767.0/loudest
    normlevel = normfactor * getSampleValueAt(sound,sampleIndex)
    setSampleValueAt(sound,sampleIndex,normlevel)
  # Dealing with the rest of the audio clip
  factor = 0.8
  for sampleIndex in range(sps,getLength(sound)):
    if sampleIndex % sps == 0:
      factor = factor
      value = getSampleValueAt(sound,sampleIndex)
      setSampleValue(sampleIndex,value*factor)
    else:
      factor = factor - 0.2
      value = getSampleValueAt(sound,sampleIndex)
      setSampleValue(sampleIndex,value*factor)

  play(sound)
```