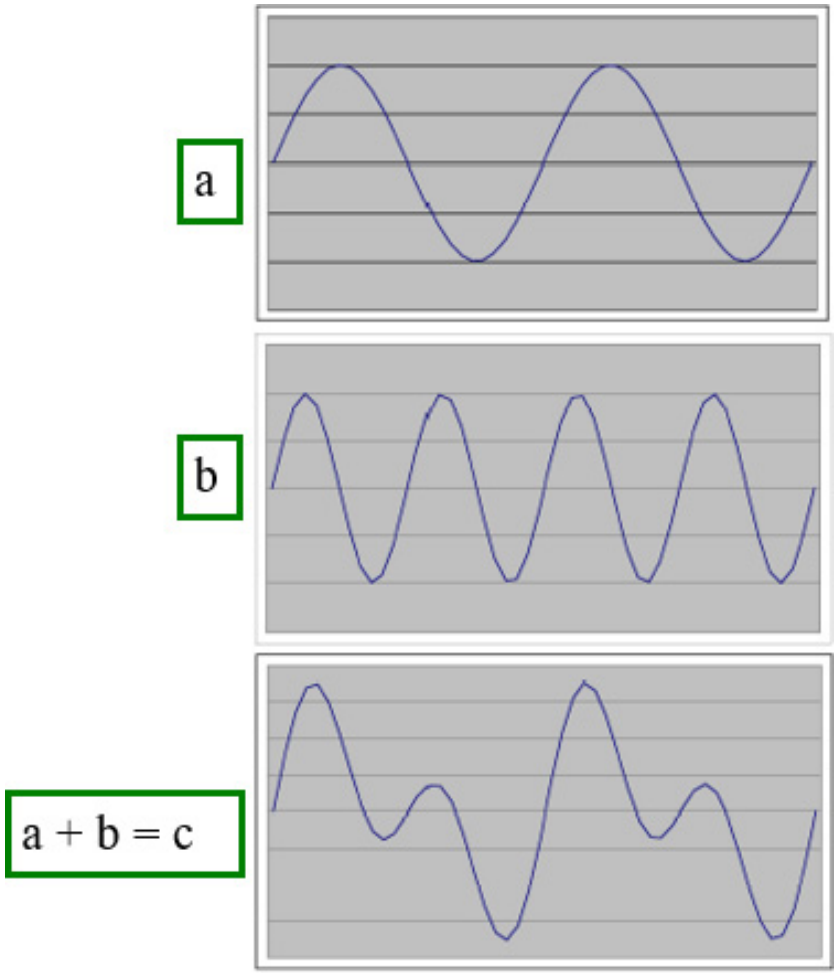
Adding Sounds

- The first two are sine waves with different frequencies
- The third is just the sum of the first two samples
- Notice: always ONE wave



Making More Complex Sounds

- We know that natural sounds are often the combination of multiple sounds.
- Adding waves in physics or math is hard.
- In computer science, it's easy! Simply add the samples at the same index in the two waves:

```
for srcSample in range(0, getLength(source)):
    destValue=getSampleValueAt(dest, srcSample)
    srcValue=getSampleValueAt(source, srcSample)
    setSampleValueAt(source, srcSample, srcValue+destValue)
```


22 Thu Nov 14 Making Sounds by Combining Pieces, part 1/bassoon-c4.wav
Play Entire Sound
Play Before
Play After
Stop
Play Selection
Clear Selection
Start Index: N/A Stop Index: N/A
Current Index: 8000
Sample Value: -2890
The number of samples between pixels: 1
Zoom Out
22 Thu Nov 14 Making Sounds by Combining Pieces, part 1/bassoon-e4.wav
Play Entire Sound
Play Before
Play After
Stop
Play Selection
Clear Selection
Start Index: N/A Stop Index: N/A
Current Index: 8000
Sample Value: 2227
The number of samples between pixels: 1
Zoom Out
-2890 + 2227 = -663
Some arbitrary sample
No File Name
Play Entire Sound
Play Before
Play After
Stop
Play Selection
Clear Selection
Start Index: N/A Stop Index: N/A
Current Index: 8000
Sample Value: -663
The number of samples between pixels: 1
Pearson

Uses for Adding Sounds

- We can mix sounds
  - We even know how to change the volumes of the two sounds, even over time (e.g., fading in or fading out)
- We can create echoes
- We can add sine (or other) waves together to create kinds of instruments/sounds that do not physically exist, but which sound interesting and complex

A Function for Adding Two Sounds

```
def addSoundInto(sound1, sound2):

  for sampleNmr in range(0, getLength(sound1)):
    sample1 = getSampleValueAt(sound1, sampleNmr)
    sample2 = getSampleValueAt(sound2, sampleNmr)
    setSampleValueAt(sound2, sampleNmr, sample1 + sample2)
```

Notice that this adds sound1 and sound2 by adding sound1 **into** sound2

...you can change that

sound2 **into** sound1

...or you can use makeEmptySound() and merge the sounds into a completely new sound object

Making a Chord by Mixing Three Notes

```
>>> c4=makeSound(getMediaPath("bassoon-c4.wav"))
>>> e4=makeSound(getMediaPath("bassoon-e4.wav"))
>>> g4=makeSound(getMediaPath("bassoon-g4.wav"))
>>> addSoundInto(e4,c4)
>>> play(c4)
>>> addSoundInto(g4,c4)
>>> play(c4)
```

Adding Sounds with a Delay

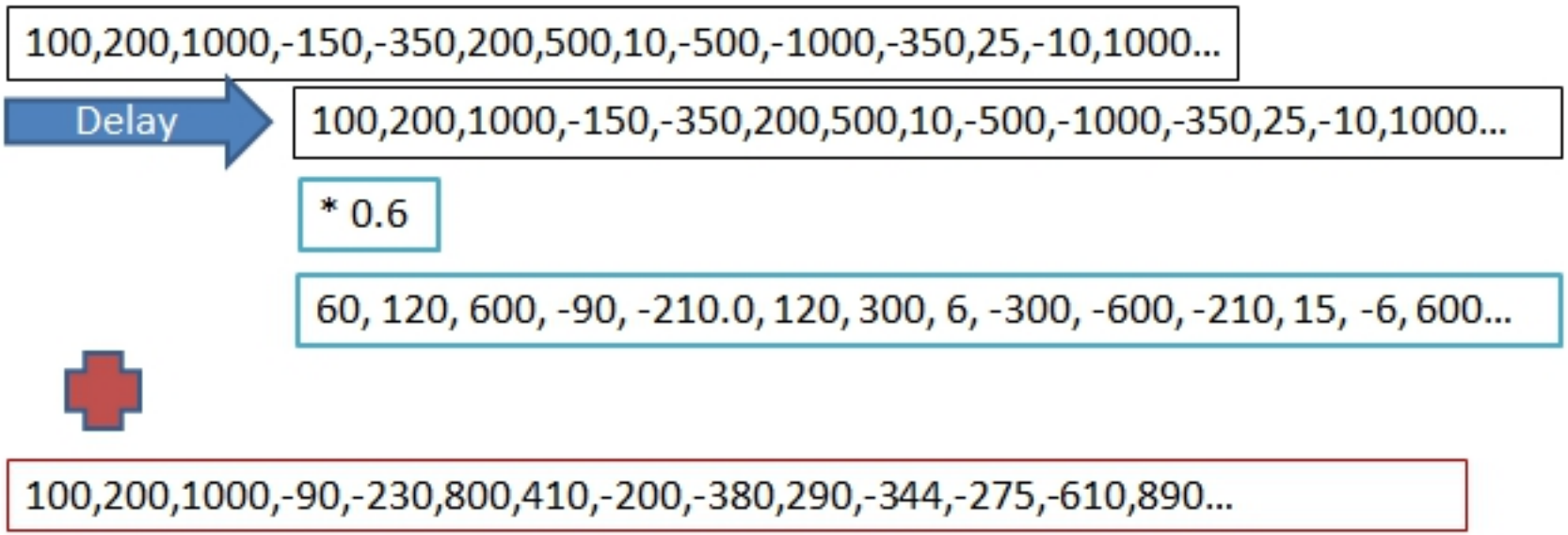
```
def makeChord(sound1, sound2, sound3):
  for index in range(0, getLength(sound1)):
    s1Sample = getSampleValueAt(sound1, index)
    setSampleValueAt(sound1, index, s1Sample )
    if index > 1000:
      s2Sample = getSampleValueAt(sound2, index - 1000)
      setSampleValueAt(sound1, index, s1Sample + s2Sample)
    if index > 2000:
      s3Sample = getSampleValueAt(sound3, index - 2000)
      setSampleValueAt(sound1, index, s1Sample + s2Sample + s3Sample)
```

- Add in sound2 after 1000 samples
- Add in sound3 after 2000 samples

Note that in this version we're adding into sound1!

How the Echo Works

Top row is the samples of our sound. We're adding it to us, but delayed a few samples down, and multiplied to make it softer.



Creating an Echo

```
def echo(sndFile, delay):
  s1 = makeSound(sndFile)
  s2 = makeSound(sndFile)
  for index in range(delay, getLength(s1)):
    echo = 0.6*getSampleValueAt(s2, index-delay)
    combo = getSampleValueAt(s1, index) + echo
    setSampleValueAt(s1, index, combo)
  play(s1)
  return s1
```

This creates a delayed echo sound, multiplies it by 0.6 to make it fainter and then adds it into the original sound.