Introduction to Computing and Programming in Python:

A Multimedia Approach

Chapter 7:

Modifying Samples in a Range

Chapter Objectives

The media learning goals for this chapter are:

- To splice sounds together to make sound compositions.
- To reverse sounds.
- To mirror sounds.

The computer science goals for this chapter are:

- To iterate an index variable for an array across a range.
- To use comments in programs and understand why.
- To identify some algorithms that cross media boundaries.

Knowing where we are in the sound

- More complex operations require us to know where we are in the sound, which sample
  - Not just process all the samples exactly the same
- Examples:
  - **Reversing** a sound
    - It's just copying, like we did with pixels
  - **Changing the frequency** of a sound
    - Using sampling, like we did with pixels
  - **Splicing** sounds

Using for to count with range

```
>>> print range(1,3)
[1, 2]
>>> print range(3,1)
[]
>>> print range(-1,5)
[-1, 0, 1, 2, 3, 4]
>>> print range(1,100)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, … 99]
```

Increasing volume by *sample index*

```
def increaseVolumeByRange(sound):
  for sampleNumber in range(0, getLength(sound)):
    value = getSampleValueAt(sound, sampleNumber)
    setSampleValueAt(sound, sampleNumber, value *
```

This really is the same as:

```
def increaseVolume(sound):
  for sample in getSamples(sound):
    value = getSample(sample)
    setSample(sample,value * 2)
```

Modify different sound sections

The index lets us modify parts of the sound now - e.g. here we increase the volume in the first half, and then decrease it in the second half.

```
def increaseAndDecrease(sound):
  length = getLength(sound)
  for index in range(0, length/2):
    value = getSampleValueAt(sound,
index)
    setSampleValueAt(sound, index,
value*2)
  for sampleIndex in range(length/2,
length):
```

Array References

Square brackets ([ ]) are standard notation for arrays (or lists). To access a single array element at position `index`, we use `array[index]`

```
>>> myArray = range(0, 100)
>>> print myArray[0]
0
>>> print myArray[1]
1
>>> print myArray[99]
99
```

Splicing Sounds

- Splicing gets its name from literally cutting and pasting pieces of magnetic tape together
- Doing it digitally is easy (in principle), but painstaking
- The easiest kind of splicing is when the component sounds are in separate files.
- All we need to do is copy each sound, in order, into a target sound.
- Here's a recipe that creates the start of a sentence, "Guzdial is …" (You may complete the sentence.)

Splicing whole sound files

```
def merge():
  guzdial =
makeSound(getMediaPath("guzdial.wav"))
  isSound = makeSound(getMediaPath("is.wav"))
  target =
makeSound(getMediaPath("sec3silence.wav"))
  index = 0
  for source in range(0, getLength(guzdial)):
    value = getSampleValueAt(guzdial, source)
    setSampleValueAt(target, index, value)
    index = index + 1
  for source in range(0,
int(0.1*getSamplingRate(target))):
    setSampleValueAt(target, index, 0)
    index = index + 1
  for source in range(0, getLength(isSound)):
    value = getSampleValueAt(isSound, source)
    setSampleValueAt(target, index, value)
    index = index + 1
  normalize(target)
```

How it works

- Creates sound objects for the words “Guzdial”, “is” and the target silence
- Set target’s index to 0, then let each loop increment index and end the loop by leaving index at the next empty sample ready for the next loop
- The 1st loop copies “Guzdial” into the target
- The 2nd loop creates 0.1 seconds of silence
- The 3rd loop copies “is” into the target
- Then we normalize the sound to make it louder

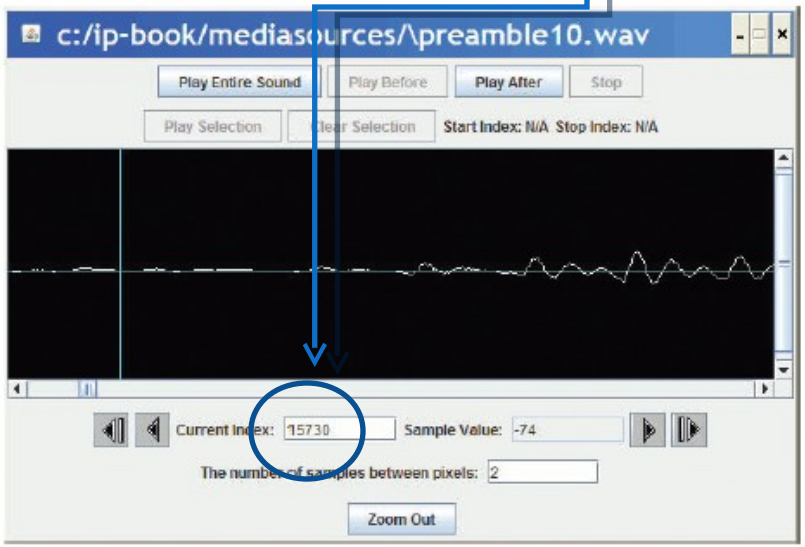
Splicing words into a speech

- Say we want to splice pieces of speech together:
  - We find where the end points of words are
  - We copy the samples into the right places to make the words come out as we want them
  - (We can also change the volume of the words as we move them, to increase or decrease emphasis and make it sound more natural.)

Finding the word end-points

- Using MediaTools and play before/after cursor, we can figure out the index numbers where each word ends
- We want to splice a copy of the word “United” after “We the” so that it says, “We the United People of the United States”.

Word	Ending index
We	15730
the	17407
People	26726
of	32131
the	33413
United	40052
States	55510



Now, it's all about copying

- We have to keep track of the source and target indices, **`srcSample`** and **`destSample`**

```
destSample = Where-the-incoming-sound-should-start
for srcSample in range(startingPoint, endingPoint):
    sampleValue = getSampleValueAt(source, srcSample)
    setSampleValueAt(dest, destSample, sampleValue)
    destSample = destSample + 1
```

The Whole Splice

```
def splicePreamble():
  file = getMediaPath("preamble10.wav")
  source = makeSound(file)
  target = makeSound(file)   # This will be the newly spliced sound
  targetIndex =17408       #  targetIndex starts at just after "We the" in the new sound
  for sourceIndex in range( 33414, 40052):  # Where the word "United" is in the sound
    setSampleValueAt(target, targetIndex,  getSampleValueAt(source, sourceIndex))
    targetIndex = targetIndex + 1
  for sourceIndex in range(17408, 26726):   # Where the word "People" is in the sound
    setSampleValueAt(target , targetIndex, getSampleValueAt(source, sourceIndex))
    targetIndex = targetIndex + 1
  for index in range(0, 1000):                    #Stick some quiet space after that
    setSampleValueAt(target, targetIndex, 0)
    targetIndex = targetIndex + 1
  play(target)                                    #Let's hear and return the result
  return target
```

What's going on here?

- First, set up a source and target.
- Next, we copy "United" (samples 33414 to 40052) after "We the" (sample 17408)
  - That means that we end up at 17408+(40052-33414) = 17408+6638=24046
  - Where does "People" start?
- Next, we copy "People" (17408 to 26726) immediately afterward.
  - Do we have to copy "of" to?
  - Or is there a pause in there that we can make use of?
- Finally, we insert a little (1/1441th of a second) of space – 0's

Word	Ending index
We	15730
the	17407
People	26726
of	32131
the	33413
United	40052
States	55510

What if we didn't do that second copy? Or the pause?

```
def spliceSimpler():
  file = getMediaPath("preamble10.wav")
  source = makeSound(file)
  target = makeSound(file)   # This will be the newly spliced sound
  targetIndex =17408       #  targetIndex starts at just after "We the" in the new sound
  for sourceIndex in range( 33414, 40052):  # Where the word "United" is in the sound
    setSampleValueAt(target, targetIndex,  getSampleValueAt(source, sourceIndex))
    targetIndex = targetIndex + 1

  # Let's hear and return the result
  play(target)
  return target
```

General clip function

We can simplify those splicing functions if we had a general clip method that took a start and end index and returned a new sound clip with just that part of the original sound in it.

```
def clip(source, start, end):
  target = makeEmptySound(end - start)
  tIndex = 0
  for sIndex in range(start, end):
    value = getSampleValueAt(source, sIndex)
    setSampleValueAt(target, tIndex, value)
    tIndex = tIndex + 1
  return target
```

General copy function

We can also simplify splicing if we had a general copy method that took a source and target sounds and copied the source into the target starting at a specified target location.

```
def copy(source, target, start):
  tIndex = start
  for sIndex in range(0, getLength(source)):
    value = getSampleValueAt(source, sIndex)
    setSampleValueAt(target, tIndex, value)
    tIndex = tIndex + 1
```

Simplified preamble splice

Now we can use these functions to insert "United" into the preamble in a much simpler way.

```
def createNewPreamble():
  file = getMediaPath("preamble10.wav")
  preamble = makeSound(file)          # old
preamble
  united = clip(preamble, 33414, 40052) #
"United"
  start = clip(preamble, 0, 17407)       # "We
the"
  end = clip(preamble, 17408, 55510)     # the
rest
  len = getLength(start) + getLength(united)
  len = len + getLength(end)  # length of
everything
  newPre = makeEmptySound(len)         # new
```

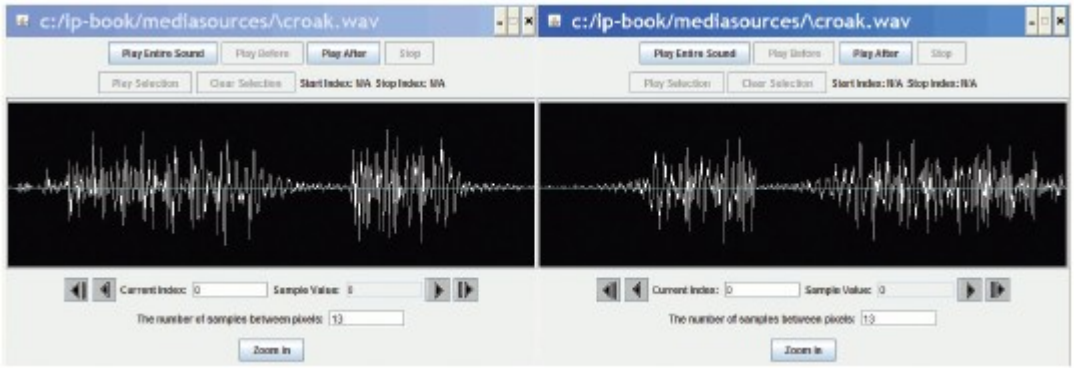
Changing the splice

- What if we wanted to increase or decrease the volume of an inserted word?
  - Simple! Multiply each sample by something as it's pulled from the source.
- Could we do something like slowly increase volume (emphasis) or normalize the sound?
  - Sure! Just like we've done in past programs, but instead of working across *all* samples, we work across only the samples in that sound!

Reversing Sounds

- We can also modify sounds by reversing them

```
def reverse(source):
  target = makeEmptySound(getLength(source))
  sourceIndex = getLength(source) - 1 # start at end
  for targetIndex in range(0, getLength(target)):
    value = getSampleValueAt(source, sourceIndex)
    setSampleValueAt(target, targetIndex, value)
    sourceIndex = sourceIndex - 1 # move backwards
  return target
```



Mirroring

- We can mirror sounds in exactly the same way we mirrored pictures

```
def mirrorSound(sound):
  len = getLength(sound)
  mirrorpoint = len/2
  for index in range(0, mirrorpoint):
    left = getSampleObjectAt(sound, index)
    right = getSampleObjectAt(sound, len-index-1)
    value = getSampleValue(lef
    setSampleValue(right, valu
```