Chapter 7:
Modifying Samples in a Range
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
  - **Changing the frequency** of a sound
  - **Splicing** sounds
Using **for** to count with **range**

- Recall the range function:
  ```python
  >>>range (1, 10)   # [1, 2, 3, 4, 5, 6, 7, 8, 9]
  ```

- We can combine the **for** loop with the **range** function to access each sample via its index
  ```python
  for sampleIndex in range (0, getLength(sound)) :
  ```

- Now we have access to each sample and its location within the sound
  - This allows us to access either all of the samples or some of them
Another way of writing array references

- Subscript operator: [ ]
- We use an integer value that represents an index in order to access the element at that index
  - Format: Array[index]
- Example:
  - mySamples [0] #the sample at index 0
- In an array the first element is always at index zero and the last element at index size – 1
  - Size is the number of elements in the array
makeEmptySound

- This function allows us to create a new empty sound.
- We have to specify the length of the sound in samples:
  ```python
  newSound = makeEmptySound(lengthInSamples)
  ```
- The default sampling rate for a new sound is 22,050 samples per second.
- We can also create a new sound by specifying the sampling rate:
  ```python
  newSound = makeEmptySound(length, samplingRate)
  ```
getSamplingRate

- This function allows us to find out the sampling rate of a sound
- It takes a sound as an input and it returns a floating point number that represents the number of samples per second for the given sound
- Example:
  - `rate = getSamplingRate(mySound)`
Modify different sound sections

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

Example: increaseAndDecrease()
Splicing Sounds

- Splicing gets its name from literally cutting and pasting pieces of magnetic tape together.
- Sound splicing: create a new sound made up of “pieces” of other sounds.
- 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.
- Example: `merge()`
Splicing words into a speech

- Another way of splicing sounds is when the words are in the middle of an existing sound and we want to pull them out from there or re-arrange them.

- In order to do that we need to:
  - Find where the end points of words are
    - Use the explorer tool
  - 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”.

<table>
<thead>
<tr>
<th>Word</th>
<th>Ending index</th>
</tr>
</thead>
<tbody>
<tr>
<td>We</td>
<td>15730</td>
</tr>
<tr>
<td>the</td>
<td>17407</td>
</tr>
<tr>
<td>People</td>
<td>26726</td>
</tr>
<tr>
<td>of</td>
<td>32131</td>
</tr>
<tr>
<td>the</td>
<td>33413</td>
</tr>
<tr>
<td>United</td>
<td>40052</td>
</tr>
<tr>
<td>States</td>
<td>55510</td>
</tr>
</tbody>
</table>
General clip and copy functions

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.

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.

Examples: \( \text{clip()} \) and \( \text{copy()} \)
Example: \( \text{splicePreamble()} \)
Abstraction

• General, re-usable functions are a form of abstraction

• abstraction allows us and others to use feature in a language without having to understand the details of how these features are implemented.
  • We just need to know how to use them
Other examples

- We can reverse sounds:
  - The source index starts with the index of the last sample in the source sound
- We can mirror sounds in exactly the same way we mirrored pictures
  - Find the middle point and copy the sample from the left side into the right side

- Examples: `reverse()` and `mirror()`
Creating a library of useful functions

- We will be writing functions that we may want to re-use in other programs
- We can save this functions in a file and then import those functions from other files
- Here is how to do it:

1. Save your reusable functions in a file (fileName.py)
   - Example: soundFunctions.py
2. Write `from media import *` in the first line of the file with the re-usable functions
3. Type this line in the command area prompt to set up the path to the directory where you have the file with the reusable functions:
   ```
   >>> setLibPath(fileNamePath)
   ```
   - Example: `>>> setLibPath(r"C:\MultimediaProgramming")`
4. To use the functions in another file start the file with the line
   ```
   from fileName import *
   ```
   - Example: `from soundFunctions import *`