PYTHON FOR MUSICIANS(2)

NEHAL JAIN

QCFINANCE.IN

CONTENT

Abstraction

- Introduction
- Musical Form and Custom Functions
- Recording & Uploading Sounds
- Making Custom Beats: makeBeat
- Looping
- String Operations
- Musical Repetition
- Debugging Logic
- Evaluating Correctness: Part Two

INTRODUCTION

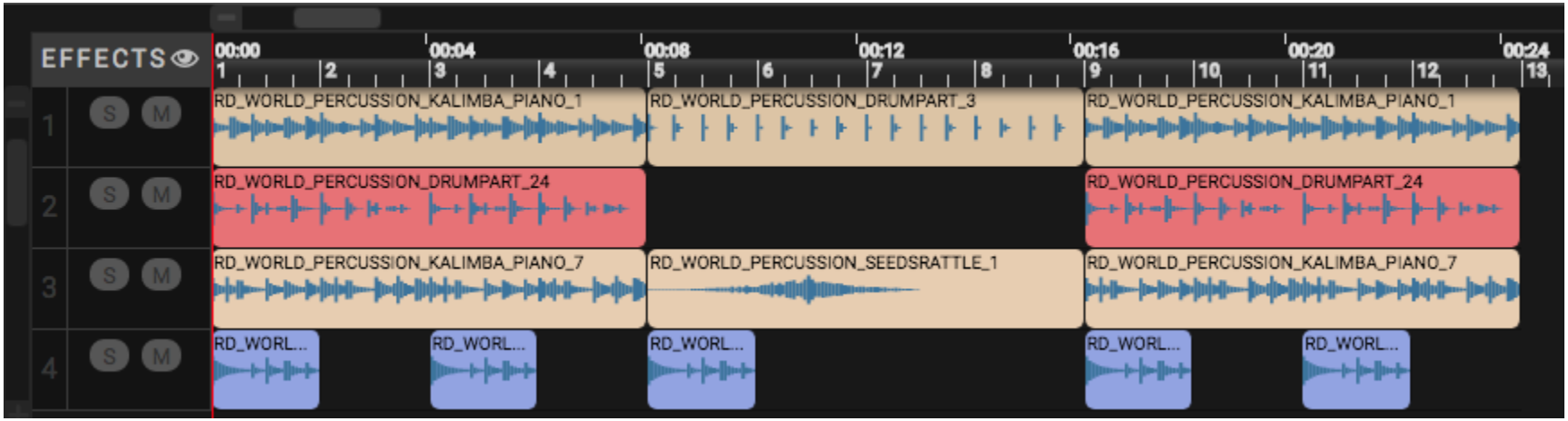
- You aren’t limited to using the functions within the EarSketch API, like fitMedia() and setEffect().
- You can make new functions to do anything you want.
- These custom functions allow you to group together lines of code to form a single instruction for the computer. This process is known as **abstraction**.
- Custom functions provide the basis for creating longer, more complex pieces in EarSketch and for creating music that combines repetition with contrast, a cornerstone of most musical genres.

Musical Form and Custom Functions

- Several measures that express an idea or feeling make up a **section**.
- Songs that contain multiple sections allow for variety and structure, or **form**.
- Intros, Verses, Choruses, and Outros are examples of sections that contribute to form.

A-B-A Form

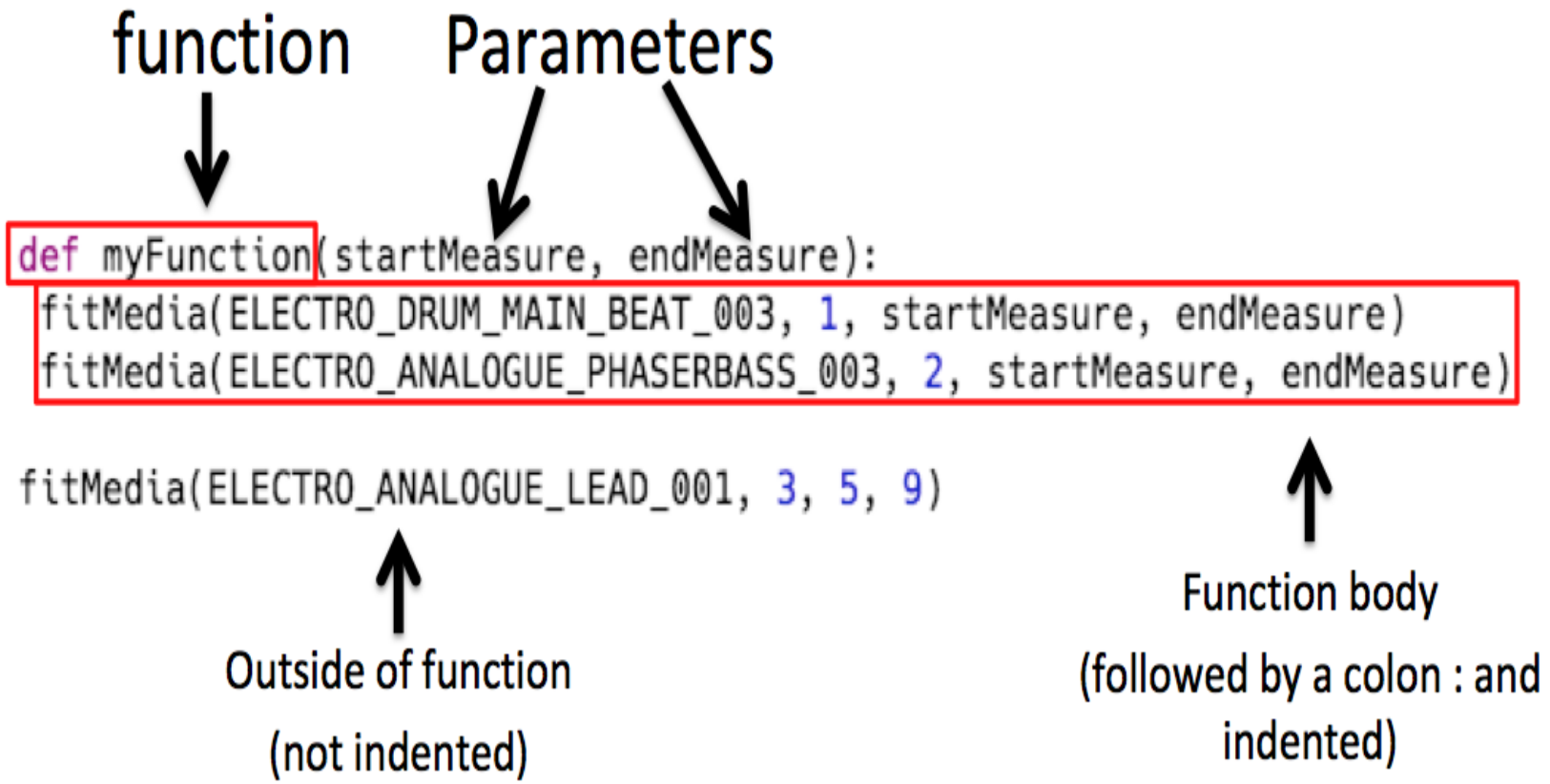
- The most common form is A-B-A, as it tends to work well musically. The B section adds variety, while returning to the A section invokes familiarity. The code below creates an ABA form:
- **Section A**: measures 1-4.
- **Section B**: measures 5-8. Features contrasting sounds to Section A.
- **Section A (repeated)**: measures 9-12.
- The sparser sound of the B section above provides contrast. This can usually be a achieved by using fewer instruments or cutting out drum parts. Using different effects, samples, or rhythms can also add contrast.



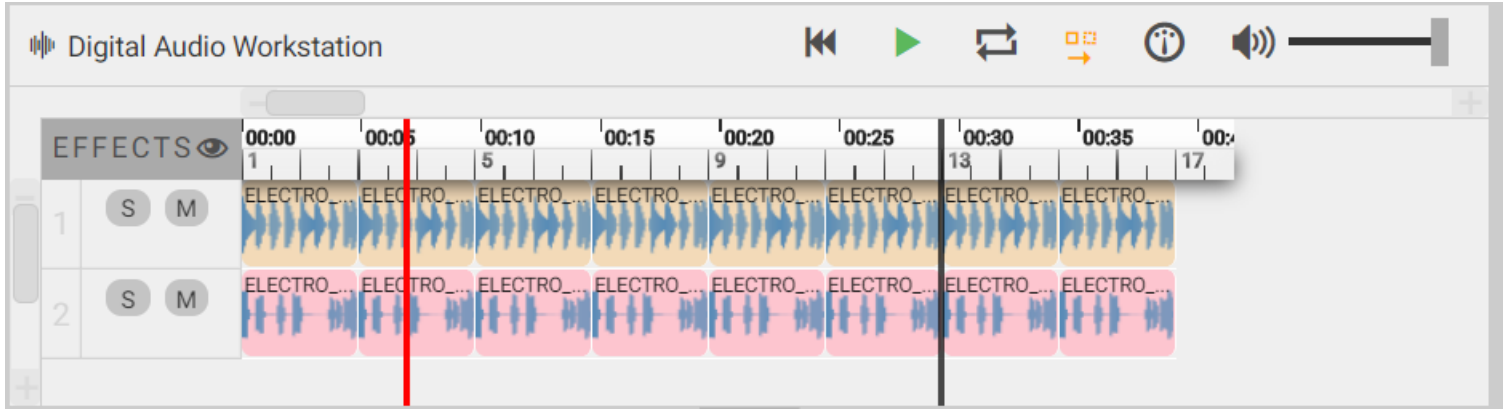
```
# python code
#
# script_name: A-B-A Form
#
# author: The EarSketch Team
#
# description: A song with A and B sections
#
#
#

#Setup
from earsketch import *
init()
setTempo(120)

#Music

# Create an A section

fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_1, 1, 1, 5) # main
fitMedia(RD_WORLD_PERCUSSION_DRUMPART_24, 2, 1, 5) # drums
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_7, 3, 1, 5) # bassline
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, 1, 2) # backing
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, 3, 4) # backing repeated

# Create a 4 measure B section between measures 5 and 9

fitMedia(RD_WORLD_PERCUSSION_DRUMPART_3, 1, 5, 9) # sparse drums
fitMedia(RD_WORLD_PERCUSSION_SEEDSRATTLE_1, 3, 5, 9) # rattling
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, 5, 6) # backing

# Then back to section A at measure 9

fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_1, 1, 9, 13) # main
fitMedia(RD_WORLD_PERCUSSION_DRUMPART_24, 2, 9, 13) # drums
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_7, 3, 9, 13) # bassline
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, 9, 10)  # backing
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, 11, 12) # backing repeated

#Finish
finish()
```

Custom Functions

- Allows you to write your own functions and avoid repetitive code. You can give them any name and run them anywhere.
- Below, we define and call a function named myFunction(). It has two parameters, meaning it takes two arguments when called.
- These parameters (startMeasure and endMeasure) hold the arguments being passed to the function, so they can be used inside the function body. A custom function can have as many parameters as necessary.


function
Parameters
def myFunction(startMeasure, endMeasure):
fitMedia(ELECTRO_DRUM_MAIN_BEAT_003, 1, startMeasure, endMeasure)
fitMedia(ELECTRO_ANALOGUE_PHASERBASS_003, 2, startMeasure, endMeasure)
fitMedia(ELECTRO_ANALOGUE_LEAD_001, 3, 5, 9)
Outside of function
(not indented)
Function body
(followed by a colon : and
indented)

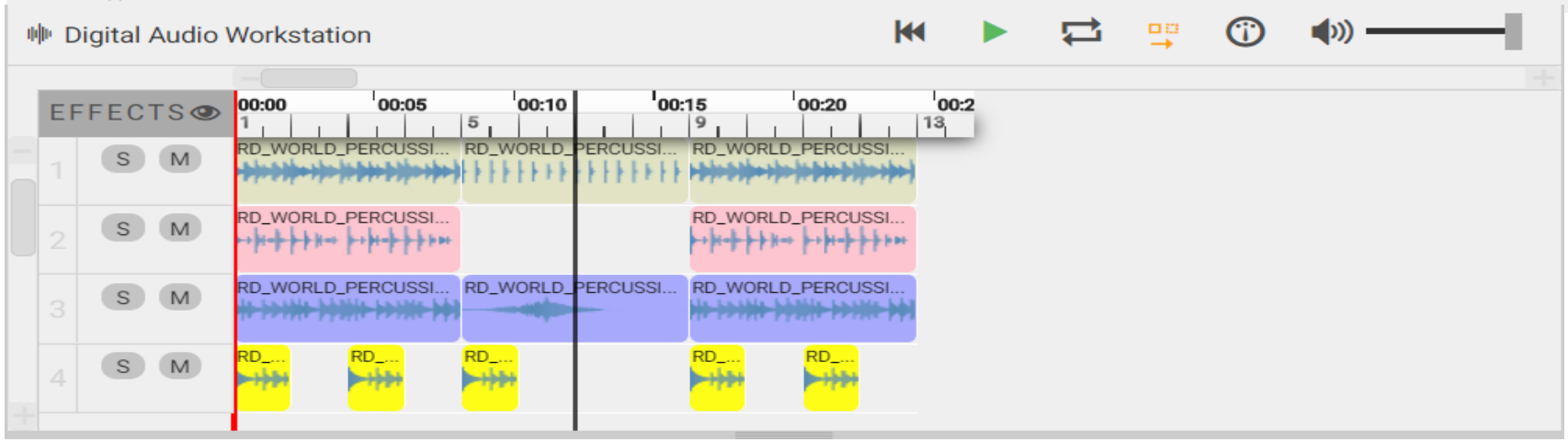
```
#Setup
from earsketch import *
init()
setTempo(100)

#Music

# Defining our new function with two parameters
def myFunction(startMeasure, endMeasure):
 fitMedia(ELECTRO_DRUM_MAIN_BEAT_003, 1, startMeasure, endMeasure)
 fitMedia(ELECTRO_ANALOGUE_PHASERBASS_003, 2, startMeasure, endMeasure)

# Calling our function, passing it two arguments: 1 and 17.
myFunction(1, 17)

#Finish
finish()
```



Improved A-B-A

- Let’s apply this new construct to the previous ABA example to make the code more modular.
- We will make functions for section A and B, and then call them in the order they should appear in the music: ABA

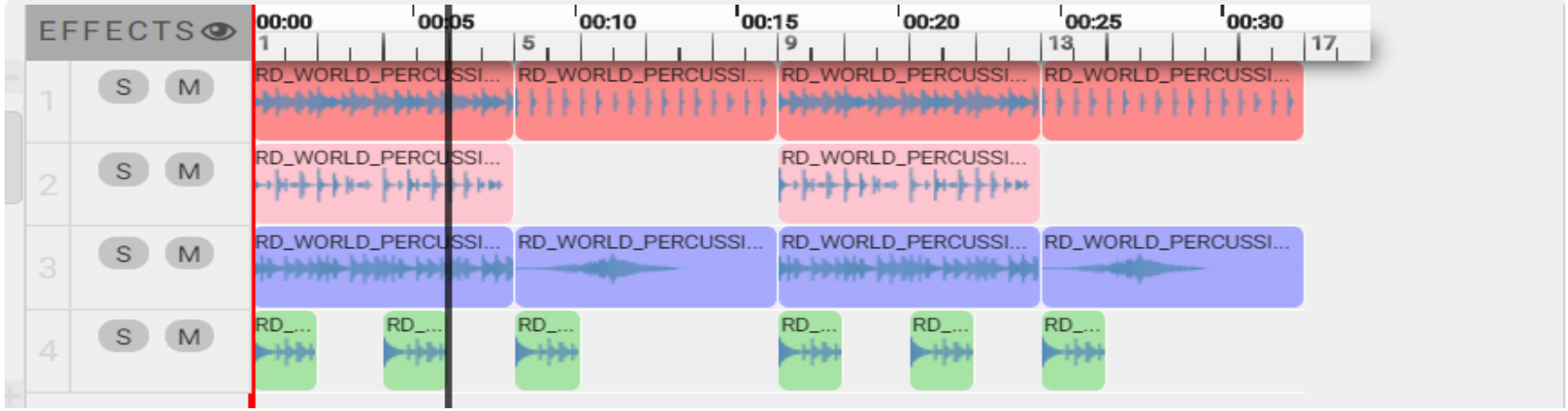
```
#Setup
from earsketch import *
init()
setTempo(120)

#Music

# A section

def sectionA(startMeasure, endMeasure): # create an A section, placing music from startMeasure (inclusive) to endMeasure (exclusive)
  fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_1, 1, startMeasure, endMeasure) # main
  fitMedia(RD_WORLD_PERCUSSION_DRUMPART_24, 2, startMeasure, endMeasure) # drums
  fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_7, 3, startMeasure, endMeasure) # bassline
  fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, startMeasure, startMeasure + 1) # backing
  fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, startMeasure + 2, startMeasure + 3) # backing repeated

# B section

def sectionB(startMeasure, endMeasure):
  fitMedia(RD_WORLD_PERCUSSION_DRUMPART_3, 1, startMeasure, endMeasure) # sparse drums
  fitMedia(RD_WORLD_PERCUSSION_SEEDSRATTLE_1, 3, startMeasure, endMeasure) # rattling
  fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, startMeasure, startMeasure + 1) # backing

# Setting up an ABA musical form through function calls
sectionA(1, 5)
sectionB(5, 9)
sectionA(9, 13)

#Finish
finish()
```



A-B-A-B form WITHOUT functions

- Now you might be thinking that there isn't that much of a difference between the examples with function and without function.
- However, you will notice a difference as you call the function in the code more often.
- Let's now try A-B-A-B form.
- You will notice that the 2nd example, which uses function, is shorter than the 1st example, which does not use function.
- As you see in the following examples, functions allow you to write an efficient code even as the script becomes more complex.
-

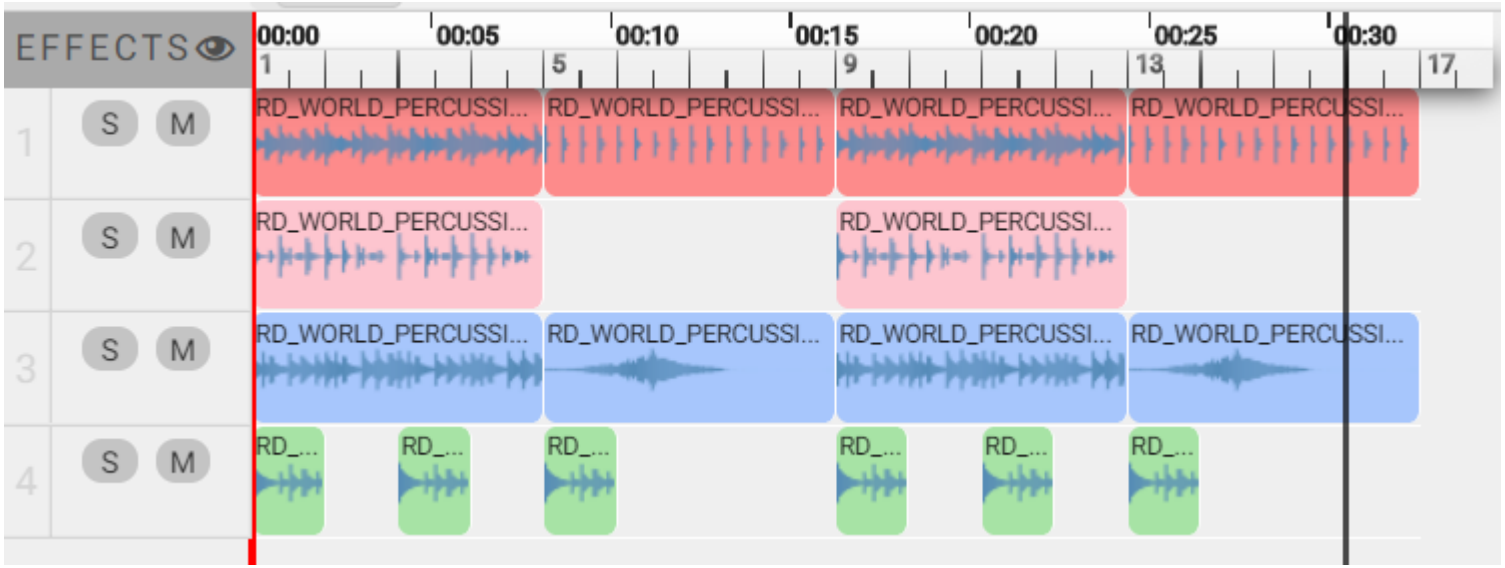
```
#Setup
from earsketch import *
init()
setTempo(120)

#Music

# Create an A section

fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_1, 1, 1, 5) # main
fitMedia(RD_WORLD_PERCUSSION_DRUMPART_24, 2, 1, 5) # drums
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_7, 3, 1, 5) # bassline
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, 1, 2) # backing
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, 3, 4) # backing repeated

# Create a 4 measure B section between measures 5 and 9

fitMedia(RD_WORLD_PERCUSSION_DRUMPART_3, 1, 5, 9) # sparse drums
fitMedia(RD_WORLD_PERCUSSION_SEEDSRATTLE_1, 3, 5, 9) # rattling
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, 5, 6) # backing

# Back to section A at measure 9

fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_1, 1, 9, 13) # main
fitMedia(RD_WORLD_PERCUSSION_DRUMPART_24, 2, 9, 13) # drums
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_7, 3, 9, 13) # bassline
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, 9, 10)  # backing
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, 11, 12) # backing repeated

# Then back to section B at measure 13.  The code is starting to look a lot messier when you're not using functions.

fitMedia(RD_WORLD_PERCUSSION_DRUMPART_3, 1, 13, 17) # sparse drums
fitMedia(RD_WORLD_PERCUSSION_SEEDSRATTLE_1, 3, 13, 17) # rattling
fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, 13, 14) # backing

#Finish
finish()
```



A-B-A-B form WITH functions

```
#Setup
from earsketch import *
init()
setTempo(120)

#Music

# A section

def sectionA(startMeasure, endMeasure): # create an A section, placing music from startMeasure (inclusive) to endMeasure (exclusive)
  fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_1, 1, startMeasure, endMeasure) # main
  fitMedia(RD_WORLD_PERCUSSION_DRUMPART_24, 2, startMeasure, endMeasure) # drums
  fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_7, 3, startMeasure, endMeasure) # bassline
  fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, startMeasure, startMeasure + 1) # backing
  fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, startMeasure + 2, startMeasure + 3) # backing repeated

# B section

def sectionB(startMeasure, endMeasure):
  fitMedia(RD_WORLD_PERCUSSION_DRUMPART_3, 1, startMeasure, endMeasure) # sparse drums
  fitMedia(RD_WORLD_PERCUSSION_SEEDSRATTLE_1, 3, startMeasure, endMeasure) # rattling
  fitMedia(RD_WORLD_PERCUSSION_KALIMBA_PIANO_3, 4, startMeasure, startMeasure + 1) # backing

# Setting up an ABA musical form through function calls
sectionA(1, 5)
sectionB(5, 9)
sectionA(9, 13)
sectionB(13, 17) # adding another section B only requires one more line of code if you use a function

#Finish
finish()
```

A-B-A-B form WITH functions



Note-

Custom functions can be called in any order, allowing us to play with form. Using different parameters for each function call enables more complex forms, an improvement over simple repetition.

Abstraction

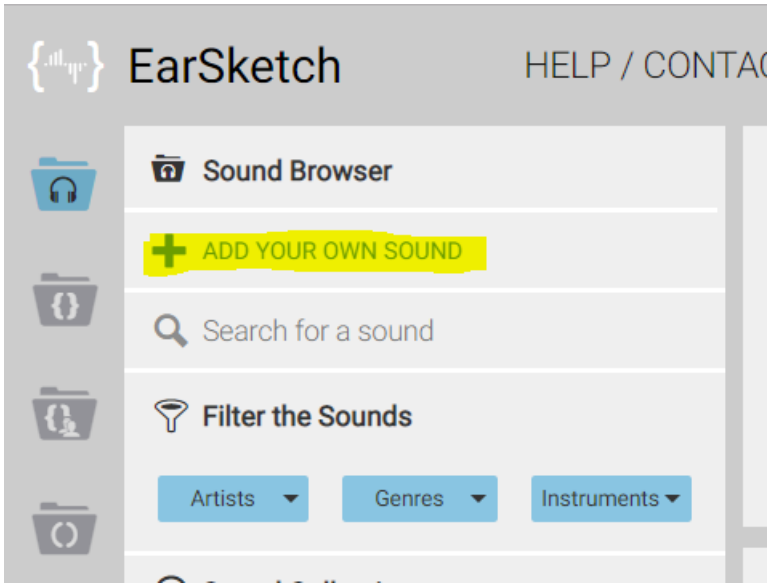
- In programming we can create abstractions, just as we group musical ideas into sections.
- An **abstraction** is a bundling of ideas to form a single concept.
- Functions are one kind of abstraction used in computer science.
- They pack multiple statements into one tool so they can be easily referred to.
- They also help manage the complexity of a program; the user doesn't have to worry about what is in the function body.
- Abstractions can make the form of a program more clear, which is helpful when writing and debugging large programs.

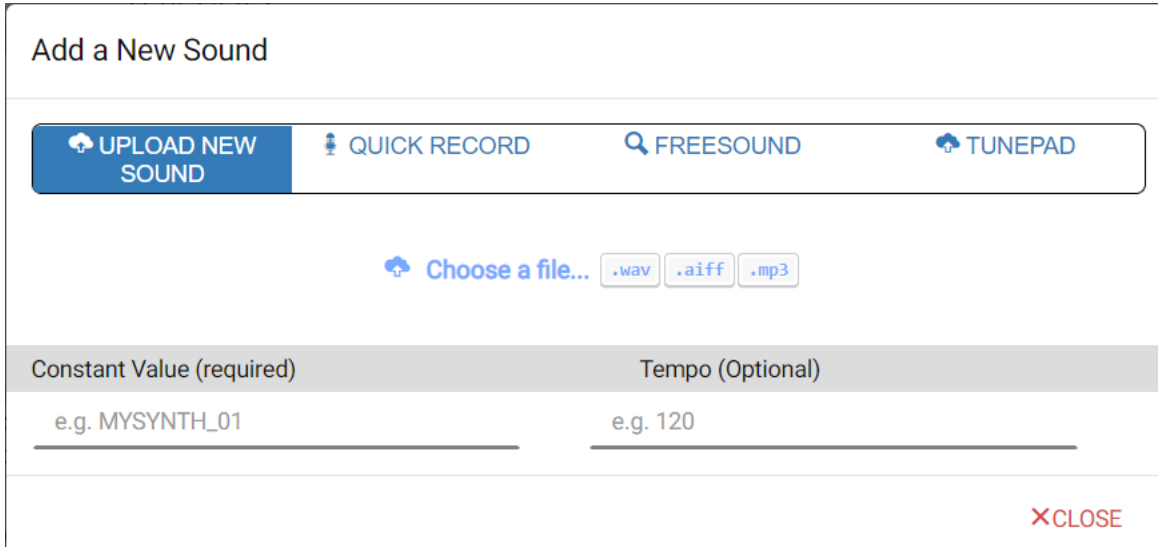
Summary

- **Sections** are related musical units consisting of multiple measures. Each expresses an idea or feeling.
- The structure and variety found within a song is known as its **form**. The most common musical form is A-B-A.
- **Custom functions** are unique functions written by the programmer to accomplish a specific task. Custom functions are an effective way to code sections, helping to avoid repetitive code. They are named by the programmer and can be called anywhere in a script.
- An **abstraction** is the bundling of ideas to form a single, often less complex, concept. Functions are an example of abstraction.

Recording & Uploading Sounds

- EarSketch offers the ability to upload your own audio through the Sounds Browser.
- Clicking on the "**Add Your Own Sound**" link will open a window and present you with two options: **File Upload**, and **Quick Record**.
- **File Upload** allows you to choose audio files (.mp3, .aiff, and so on) already on your computer, and
- **Quick Record** lets you record short clips directly into the EarSketch library.


EarSketch
HELP / CONTAC
Sound Browser
ADD YOUR OWN SOUND
Search for a sound
Filter the Sounds
Artists
Genres
Instruments


Add a New Sound
UPLOAD NEW SOUND
QUICK RECORD
FREESOUND
TUNEPAD
Choose a file...
.wav
.aiff
.mp3
Constant Value (required)
Tempo (Optional)
e.g. MYSYNTH_01
e.g. 120
CLOSE

Processes and Memory