Earsketch Overview

Georgia Institute of Technology

Mr. Michaud

Description of EarSketch

- Programming Environment
- Python Based
- Web Based App or Installed System of Software
- API built in Python for Music Mixing
- NSF Funded project to encourage computational interest through the mixing and sharing of music.
- Curriculum and Social Media Site
- earsketch.gatech.edu
- earsketch.gatech.edu/earsketch2

What we will learn:

- Music Remixing
  - Digital Audio Workstation
  - Tracks (Layers)
  - Measures
  - “Slices” of Music
  - Rhythms
  - Instrumentation
  - Effects

- Computer Science
  - Python Programming
  - Sequential Steps
  - Functions
  - Arguments
  - Variables
  - Integers
  - Strings
  - For Loops

Remix: Three Types

1)Taking sections of different songs and combining them into one song.

2)Taking instrumental tracks and combining them into a new song
Drums, Bass, Piano, Melody / Vocal

3)Taking Instrument sounds or “clips” and arranging them into new rhythmic cells.

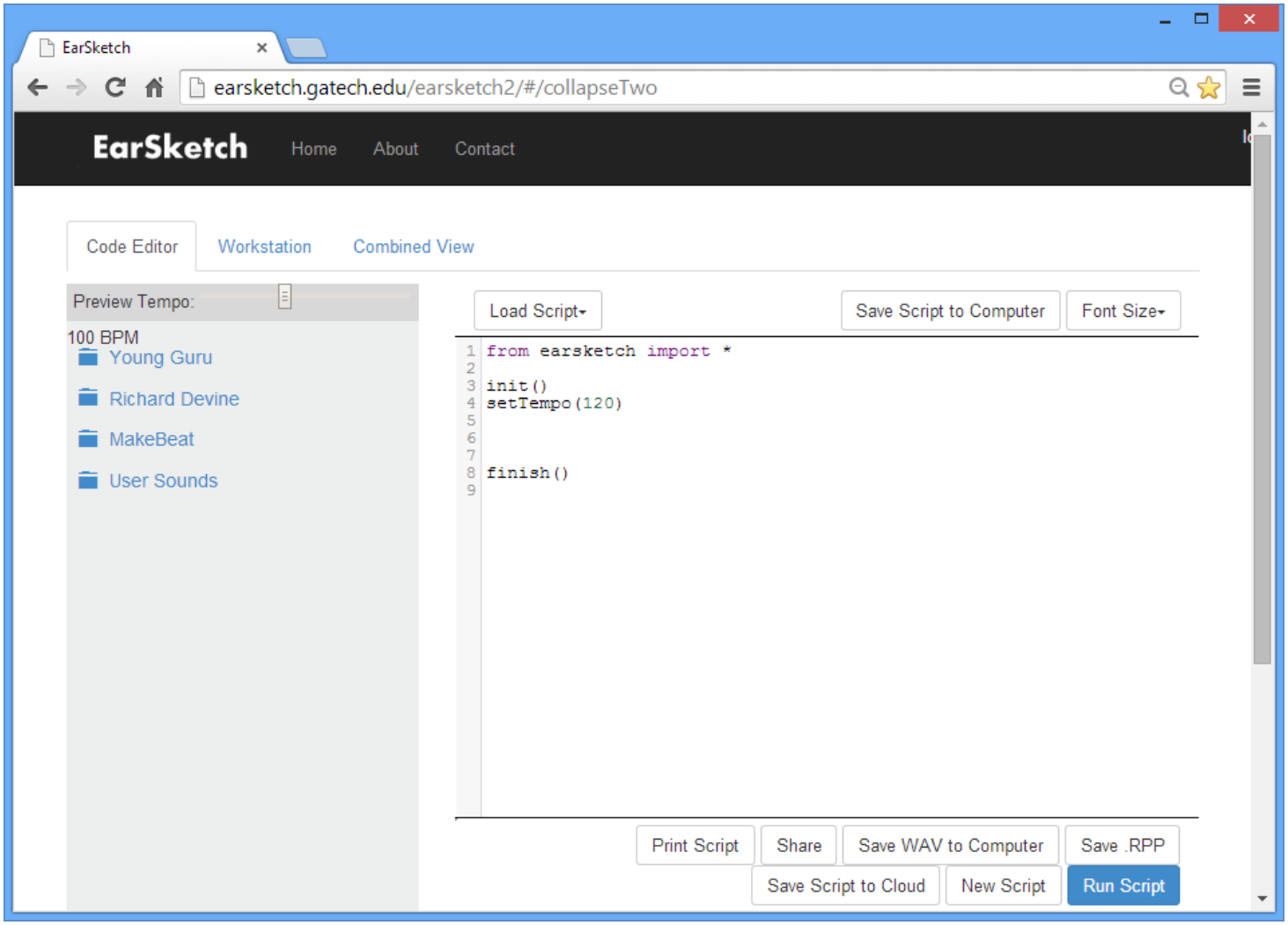
Online EarSketch Components

- Music Clip Library: Preview and Search for EarSketch Music Samples

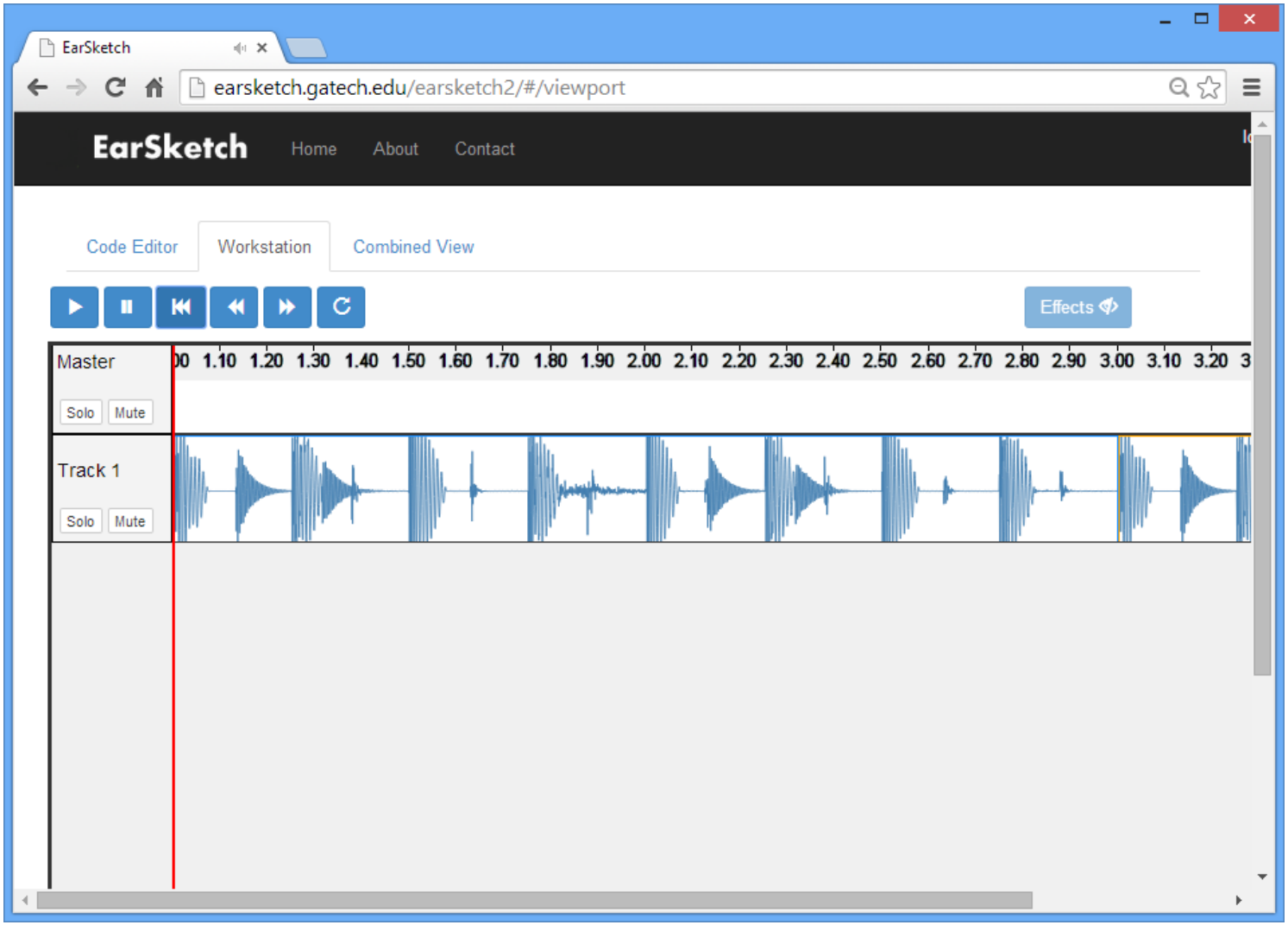
- Code Editor: Where you program in python to mix music

- Music Workstation: Where music clips are mixed on tracks and measures for playback.

Online IDE: Code Editor



Online IDE: Workstation View



File Types and Music

.wav

WAV files are best for when you wish to save the file on your local computer.

WAV files preserve the quality of audio that you hear in Reaper.

WAV files use about 10 MB for 1 minute of Music

.mp3

MP3 files are best for when you wish to share your song with others on the Internet.

MP3 files have somewhat reduced quality, but attain a smaller file size (about 10% the size of a WAV file). (1 MB for 1 minute of music)

Programming with Python

- Python – text based programming language
- Sequential List of Instructions for Computer to Follow
- Used by Google, Nasa, Robotics . . .
- Example . . . Class Demonstration

Essential Elements we will use in Python:

- **Comments**
  ```
  # This is a comment – meant for Humans
  ```
- **Includes** – loading preset methods or data
  ```
  from earsketch import *
  ```
- **Methods** – telling the computer "what to do"
  ```
  fitMedia(drums, 1, 1, 5)
  ```
- **Variables** – Names for information stored by program
  ```
  Beat1 = "0+++0+++0+0+0+++"
  ```

Include Statements

- Must put these two lines towards the beginning of all your EarSketch remix programs:

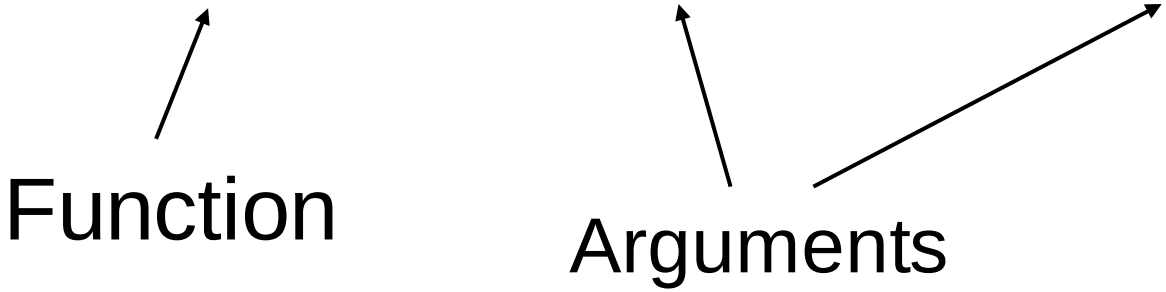
```
from earsketch import *
```

- These put the necessary “preloaded” instructions into your program.

Functions and Arguments

- **Function** – Tell the Computer what to do
- **Arguments** – details for the Method
- Example:

```
doExercise(JumpingJacks, 5)
```



EarSketch Python Functions

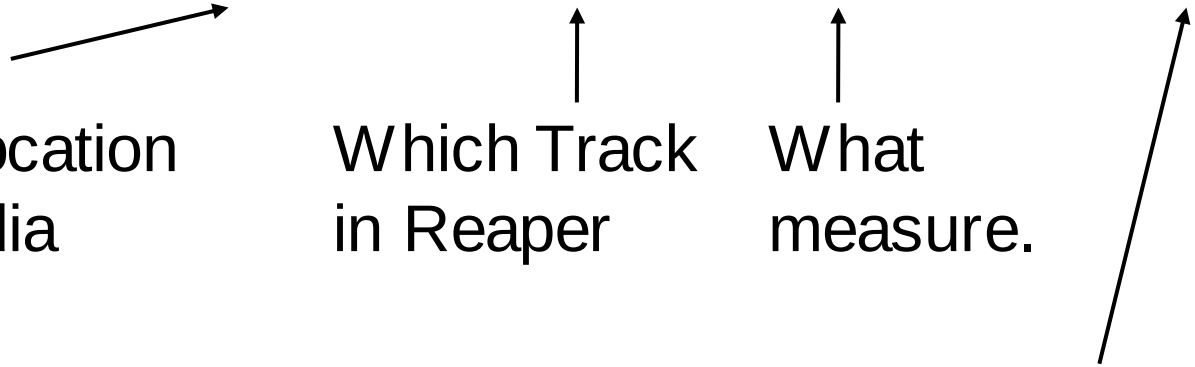
- `init()`
  Start New Reaper File
- `setTempo(120)`
  Beats per minute in remix
- `println(“Hello”)`-
  Prints message in console

EarSketch Python Functions

- `insertMedia(file, track, measure, scale)`
- `insertMediaSection(file, track, location, start, end, scale)`
- `fitMedia(file, track, start, end)`
- `makeBeat(file, track, measure, beatString)`

“InsertMedia” Method

```
insertMedia(file, track, measure, Scale)
```



File Location of Media Sound

Which Track in Reaper

What measure.

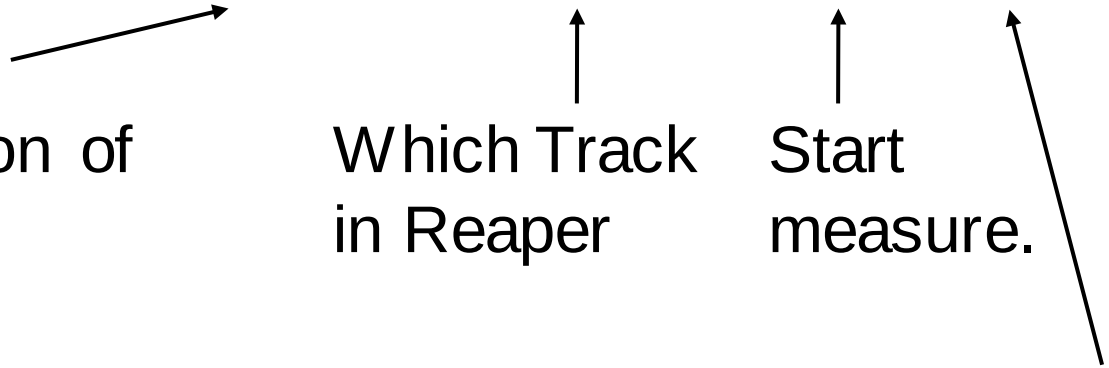
Scale: True → Matches Tempo
False → Do Not Match Tempo
Usually not used

Example:

```
insertMedia(HIP_HOP_DRUMS1_2M, 1, 1)
```

“fitMedia” Function

```
fitMedia(file, track, start, end)
```



Location of Media Sound

Which Track in Reaper

Start measure.

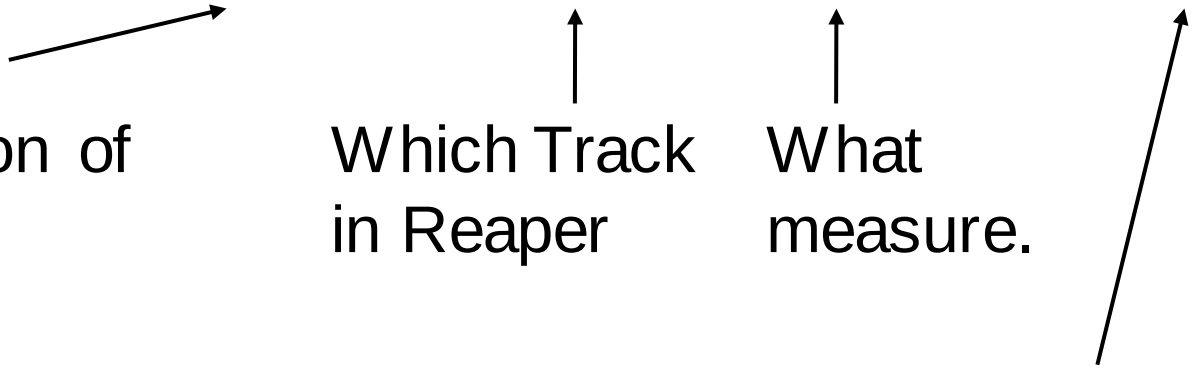
End Measure

Example:

```
fitMedia(HIP_HOP_DRUMS1_2M, 1, 1, 9)
```

“makeBeat” Method

```
makeBeat(file, track, measure, BeatString)
```



Location of Media Sound

Which Track in Reaper

What measure.

Example: “0+++0+++0+0+0+++”

Example:

```
makeBeat(drums, 1, 1, “0+0+0+++00-00+++”)
```

EarSketch Sample

```
'''
Day 1 Sample Project 1
A very simple EarSketch script
'''

from earsketch import *

# initialize Reaper
init()
setTempo(120)

# Add a music file to track 1
insertMedia(DRUM_N_BASS_DRUMS2_1M, 1)

# Set the project cursor to be
# at the beginning of the file
finish()
```

EarSketch Sample

```
'''
Day 1 Sample Project 1
A very simple EarSketch script
'''

from earsketch import *

# initialize Reaper
init()
setTempo(120)

# Add a music file to track 1
insertMedia(DRUM_N_BASS_DRUMS2_1M, 1)

# Set the project cursor to be
# at the beginning of the file
finish()
```

Multi Line Comments

Include Statement: Imports EarSketch Functions

Single Line Comment

init() - Sets Up Reaper

SetTempo() - Rate or speed of music

InsertMedia() - places music sample in track

finish() - required to “wrap up” EarSketch Code.

Variables

Variables are reusable “places” where we can store numbers or words (we call them “strings” in programming)

Examples:

```
myAge = 17
MyPhone = “404-219-1234”
birthdayMonth = “February”
```

Class Examples:

```
# Variables
food =
drink =
student1 =
student2 =
area =

# Main Program
eatLunch(student1, front)
eatLunch(student2, back)
```

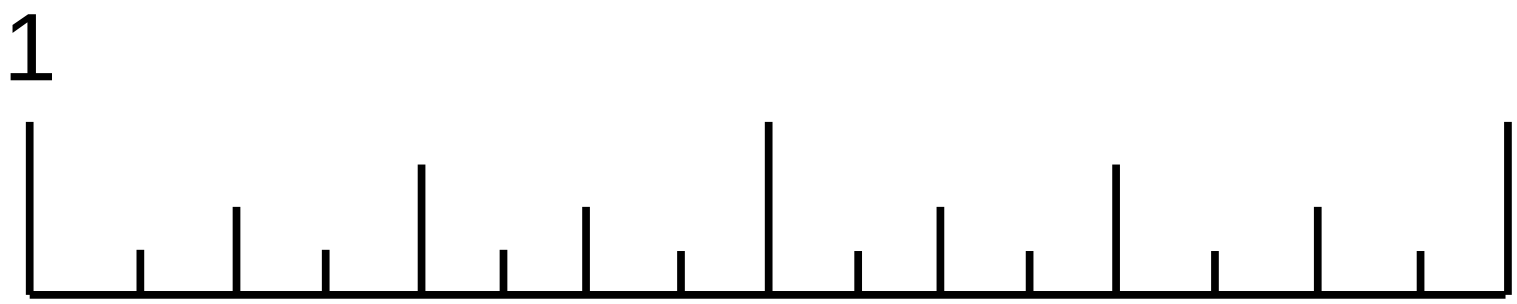
```
# Variables
drums = HIP_HOP_DRUMS1_2M
bass = HIP_HOP_BASS4_4M
start = 1

# Render Music
insertMedia(drums, 1, start)
insertMedia(bass, 2, start)
```

```
# Variables
drums = HIP_HOP_DRUMS1_2M
bass = HIP_HOP_BASS4_4M
start = 1
End = 9

# Render Music
fitMedia(drums, 1, start, end)
fitMedia(bass, 2, start, end)
```

Sections of a Measure

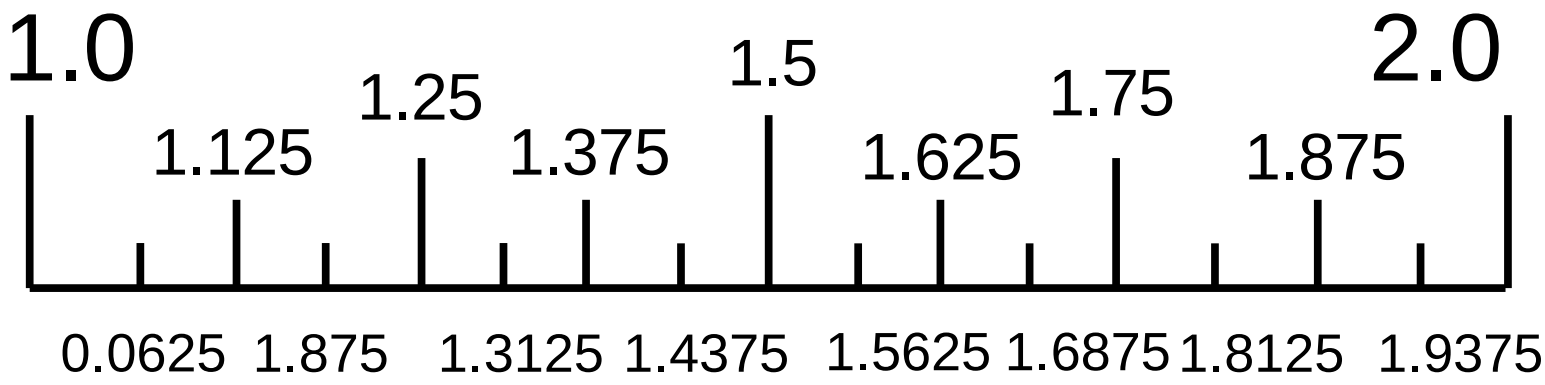


Each measure of music is made up of 16 “subBeats” or divisions. This is like the hash marks in a ruler. We can measure and place music in these subBeats to create different rhythm patterns.

Mixing at the subBeat level allows the composer to create interesting and driving rhythms that add energy to the remix.

We can program these subBeat rhythms two ways:
1. Using math and floating point numbers to represent places in the subBeat. And then using insertMediaSection() or fitMedia().
2. Using string notation and the “makeBeat()” method

Sections of a Measure: Using Floating Point numbers and math operators



For Loops:
Do something more than once

```
for count in range(4):
    insertMedia(ELEKTRO_HOUSE_DRUMS2_2M, 1)
```

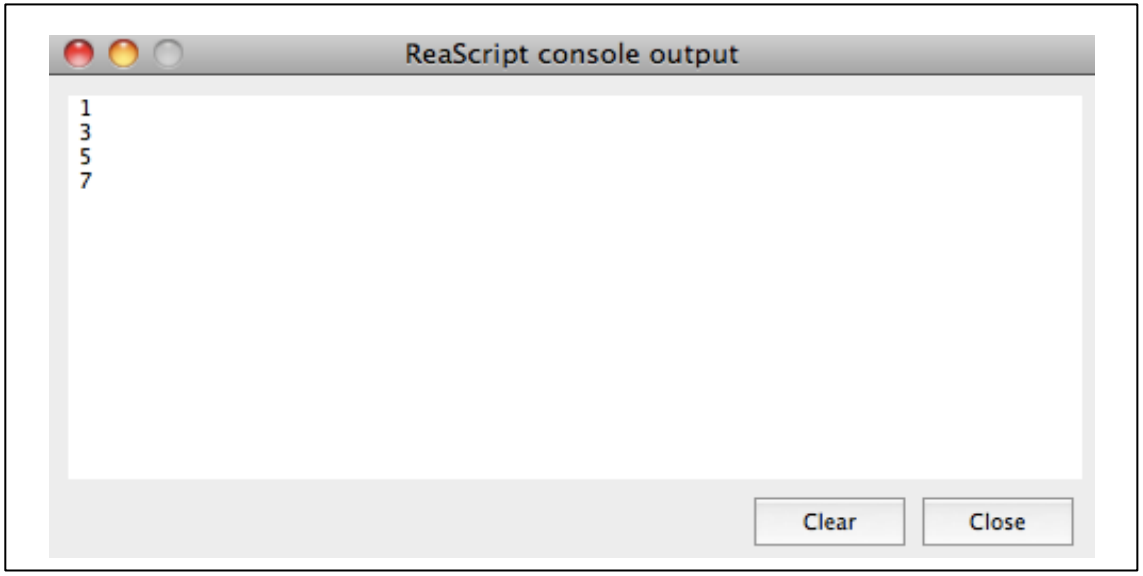
Means the same as:

```
insertMedia(ELEKTRO_HOUSE_DRUMS2_2M, 1)
insertMedia(ELEKTRO_HOUSE_DRUMS2_2M, 1)
insertMedia(ELEKTRO_HOUSE_DRUMS2_2M, 1)
insertMedia(ELEKTRO_HOUSE_DRUMS2_2M, 1)
```

For Loops: Skip Counting

```
path = HIP_HOP_DRUM_FOLDER

for count in range(1, 9, 2):
    randomFile = selectRandomFile(path)
    fitMedia(randomFile, 1, count, count + 1)
    println(count)
```



For Loops: Skip Counting Challenge:

Write For Loops to meet the following conditions:

Track 1: Place a Measure of music every 4 measures:

1 2 3 4 ***5*** 6 7 8 ***9*** 10 11 12 ***13*** 14 15 16

Track 2: Place a Measure of music every 2 measures

1 2 ***3*** 4 ***5*** 6 ***7*** 8 ***9*** 10 ***11*** 12 ***13*** 14 ***15*** 16

Track 3: Place a Measure of music on multiples of 4 measures

1 2 3 ***4*** 5 6 7 ***8*** 9 10 11 ***12*** 13 14 15 ***16***

Track 4: Your choice to make everything stick together!

For Loops: Skip Counting

```
fillDrum = HIP_HOP_SYNTHDRUMS2_2M
beat = "0+++0+++0-000+00"

for measure in range(1, 9, 2):
   makeBeat(fillDrum, 1, measure, beat)
```

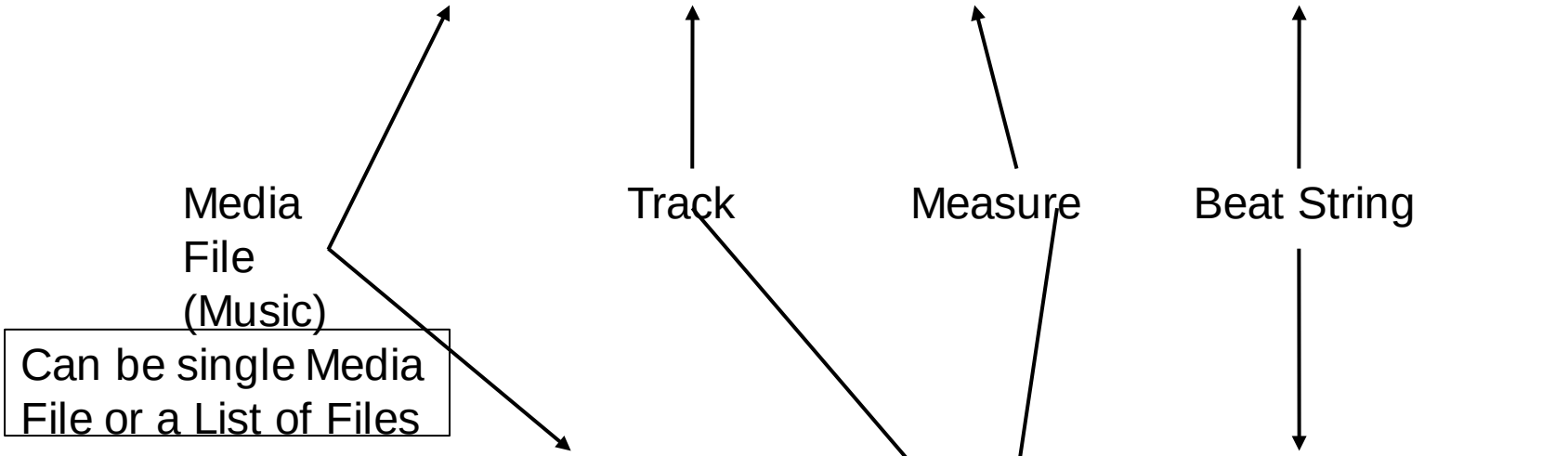
measure is the "index variable" = assigned values from the range()

(1, 9, 2) means start counting at 1,
end before 9 [meaning 8] and skip count by 2:

(1, 3, 5, 7)

makeBeat() Method

```
makeBeat(Media, Track, Measure, BeatString)
```




```
makeBeat(ELEKTRO_HOUSE_DRUMS3_2M, 1, 1, “0+++0++00+0+0-00”)
```

Means: Put rhythm beat pattern “0+++0++00+0 on track 1, measure 1 using the ELEKTRO_HOUSE_DRUMS3_2M media sound.

Beat String notation

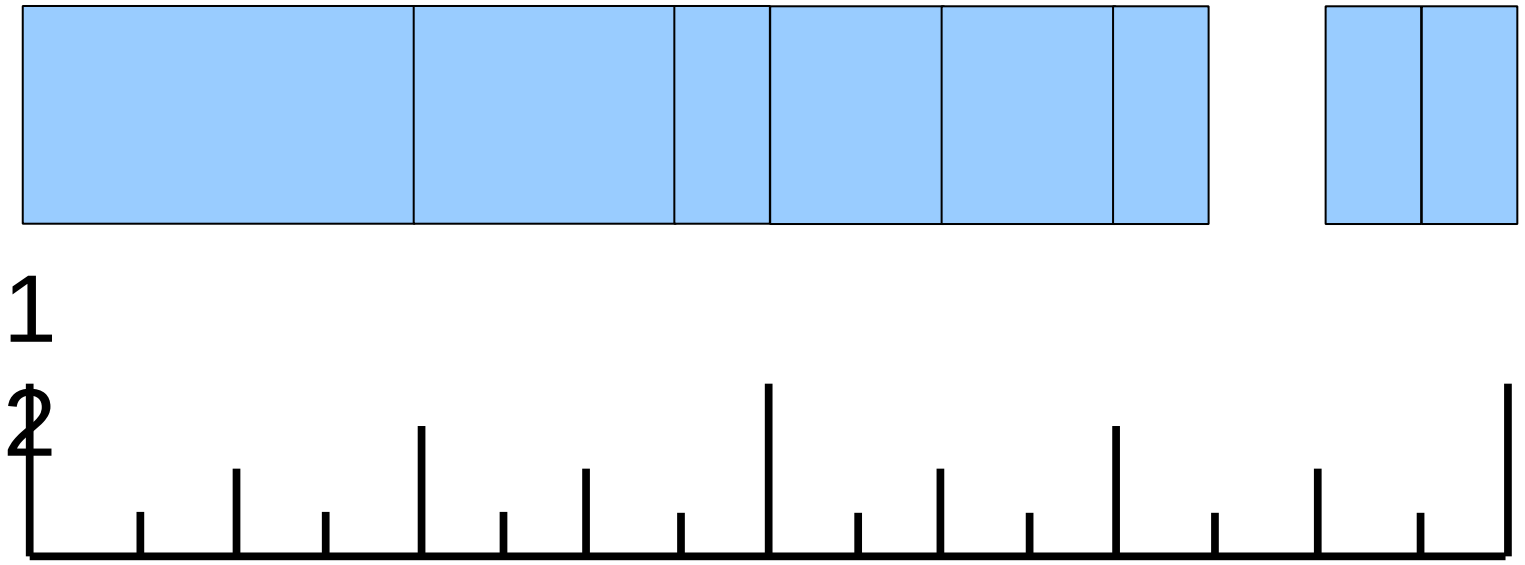
"0, 1, 2, 3 . . . " = Which Media Sound you want for the segment of beat. Correspond to placement in a List that is one based.

Note: 0 will refer to a sound if it is the only media file in the argument.

"+" Means extend or loop the Media sound $1/16^{th}$ of a measure.

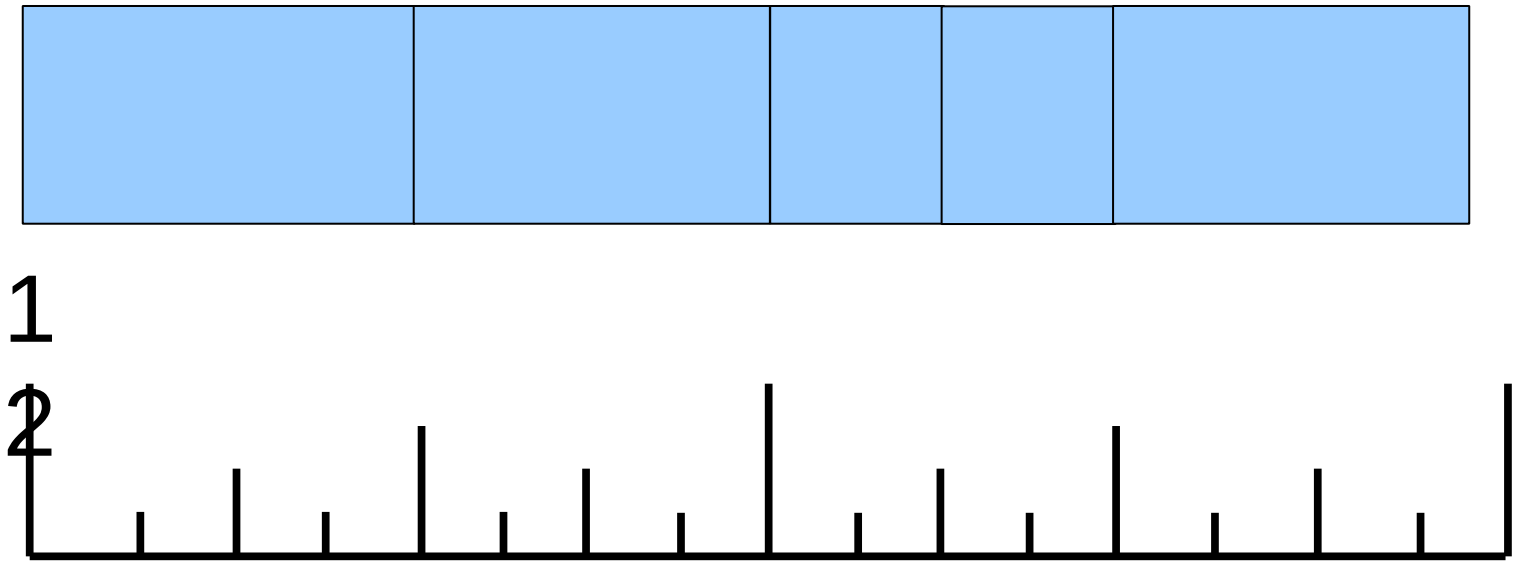
"-" Means $1/16^{th}$ measure of rest.

“0+++0++00+0+0-00”



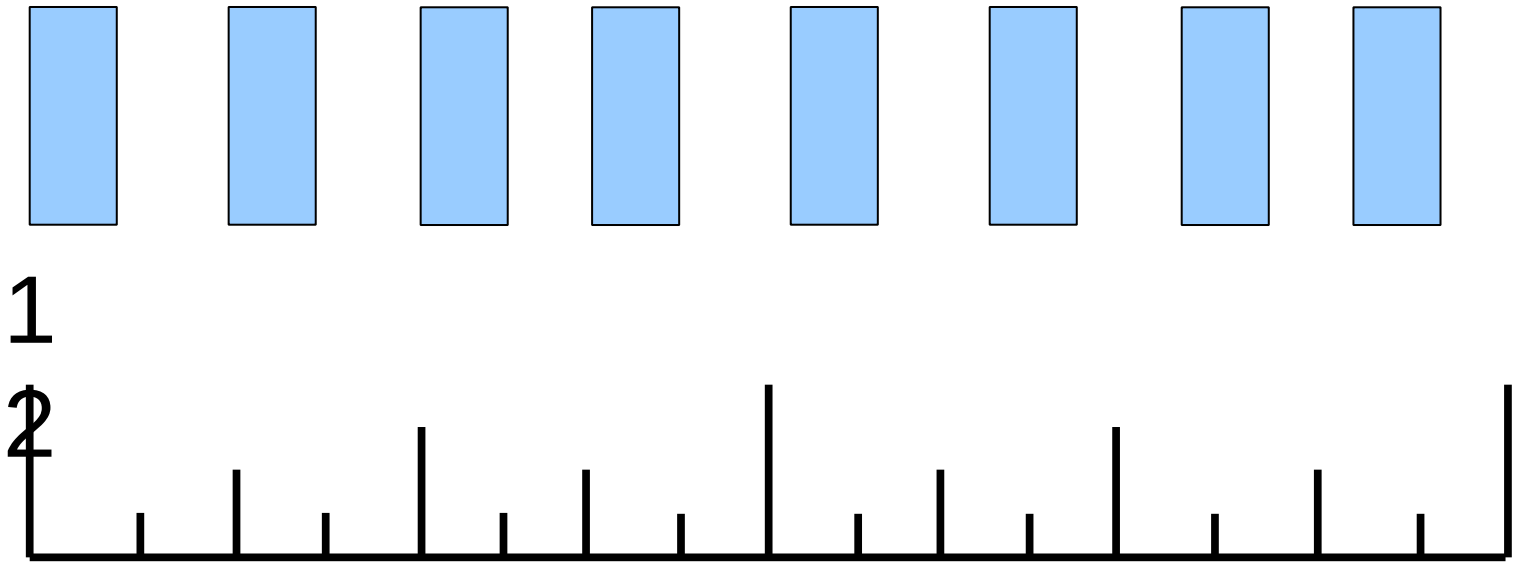

```
makeBeat(ELEKTRO_HOUSE_DRUMS3_2M, 1, 1, “0+++0++00+0+0-00”)
```

“0+++0+++0+0+0+++”



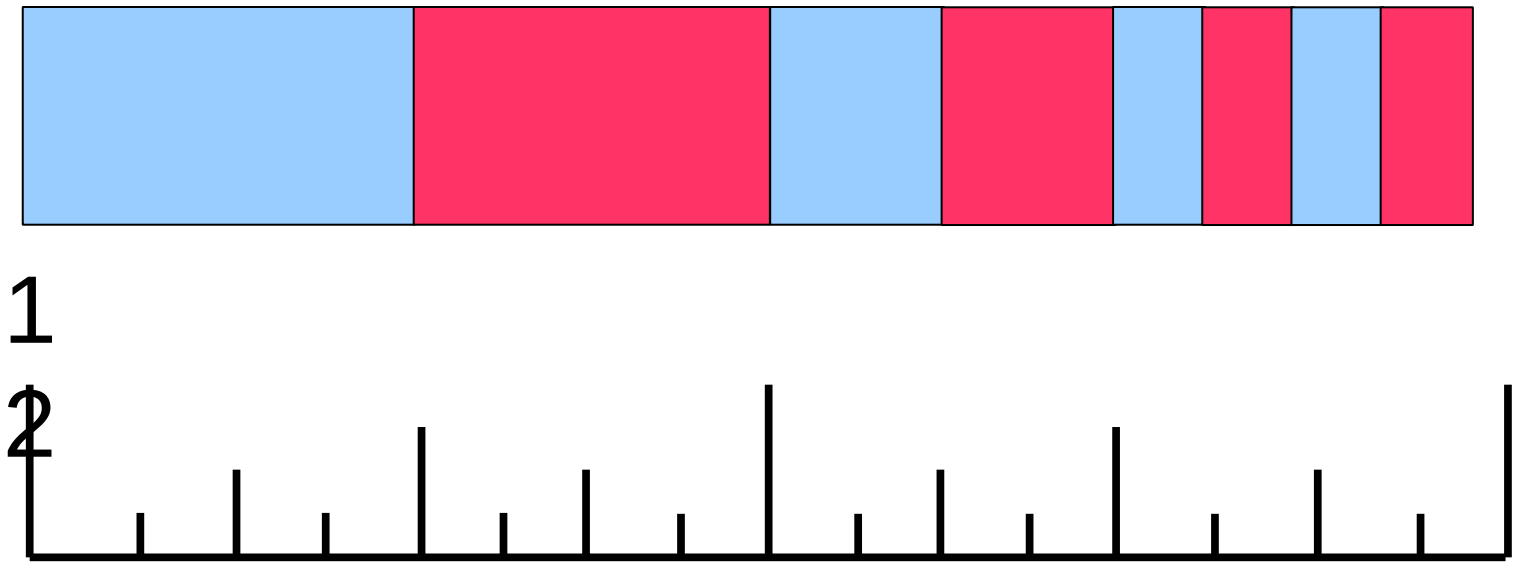

```
makeBeat(ELEKTRO_HOUSE_DRUMS3_2M, 1, 1, “0+++0+++0+0+0+++”)
```

“0-0-0-0-0-0-0-0-”



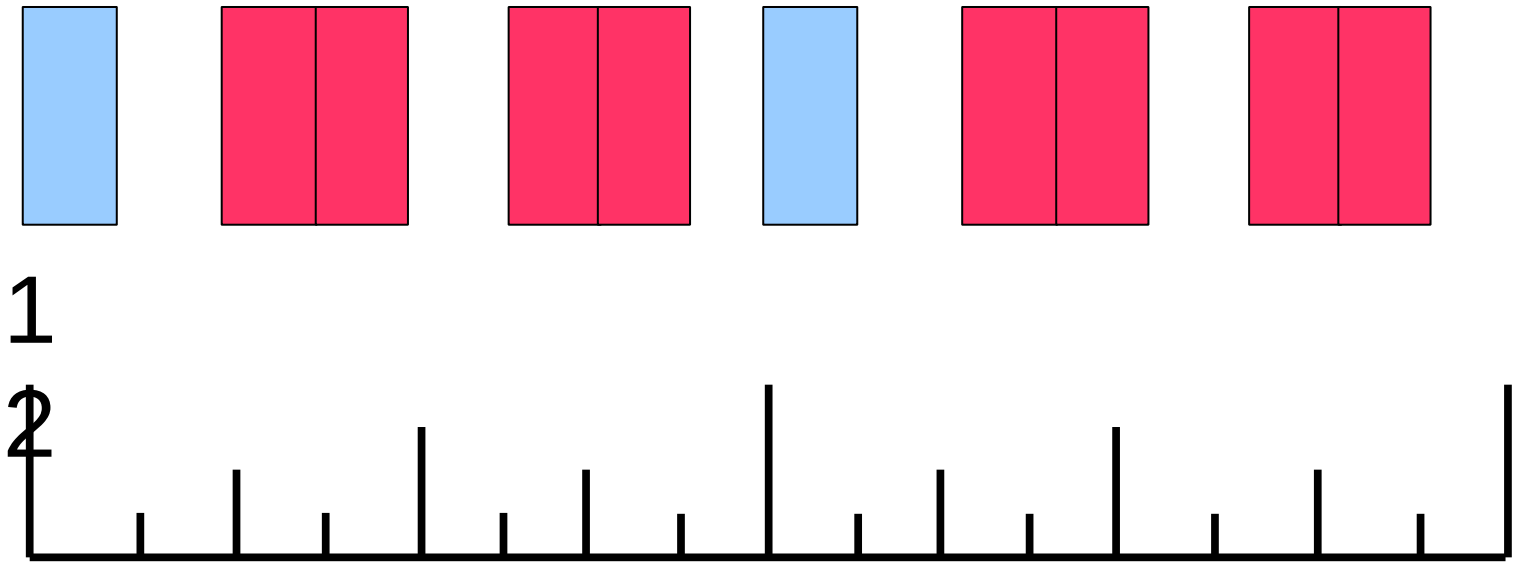

```
makeBeat(ELEKTRO_HOUSE_DRUMS3_2M, 1, 1, “0-0-0-0-0-0-0-0-”)
```

“0+++1+++0+1+0101”




```
MusicList = [ELEKTRO_HOUSE_DRUMS3_2M,
HIP_HOP_DRUMS4_2M]
makeBeat(MusicList, 1, 1, “0+++1+++0+1+0101”)
```

“0-11-11-0-11-11-”




```
MusicList = TECHNO_ALARMCALLDRUMS_2M,
TECHNO_ALMIGHTYDRUMS_2M]
makeBeat(MusicList, 1, 1, “0-11-11-0-11-11-”)
```

Sample makeBeat()

```
from earsketch import *

init()
setTempo(120)

beat = "0-00-00-0+++0+0+"
drum = DRUM_N_BASS_DRUMS1_4M

makeBeat(drum, 1, 1, beat)

finish()
```

Sample makeBeat() with Loop

```
from earsketch import *

init()
setTempo(120)

beat = "0-00-00-0+++0+0+"
drum = DRUM_N_BASS_DRUMS1_4M

for measure in range(1, 9):
    makeBeat(drum, 1, measure, beat)

finish()
```

What Measures?

```
from earsketch import *

init()
setTempo(120)

beat = "0-00-00-0+++0+0+"
drum = DRUM_N_BASS_DRUMS1_4M

for measure in range(5, 13):
    makeBeat(drum, 1, measure, beat)

finish()
```

What Measures?

```
from earsketch import *

init()
setTempo(120)

beat = "0-00-00-0+++0+0+"
drum = DRUM_N_BASS_DRUMS1_4M

for measure in range(1, 9, 2):
    makeBeat(drum, 1, measure, beat)

finish()
```

What Measures?

```
from earsketch import *

init()
setTempo(120)

beat = "0-00-00-0+++0+0+"
drum = DRUM_N_BASS_DRUMS1_4M

for measure in range(1, 17, 4):
    makeBeat(drum, 1, measure+1, beat)

finish()
```

Will this work?

```
from earsketch import *

init()
setTempo(120)

beat = "0-00-00-0+++0+0+"
drum = DRUM_N_BASS_DRUMS1_4M

for m in range(1, 17, 4):
    makeBeat(drum, 1, m+1, beat)

finish()
```

Functions

- Eb Major Scale in Quarter Notes
- When the Saints Go Marching In
- Blues Scale in Bb
- 4/4 Time on Drums

- Functions Summarize Actions into single “calls”
- In EarSketch we use functions to define forms in Music

Functions: Recycle and Reuse!

```
jazzDrums = HIP_HOP_JAZZDRUMS1_4M
synthDrums = HIP_HOP_SYNTHDRUMS3_2M
keys = TECHNO_ELDERSONSYNTH_2M
scratch = HIP_HOP_DJSCRATCH2_2M

fitMedia(jazzDrums, 1, 1, 9)
fitMedia(synthDrums, 2, 1, 9)
fitMedia(keys, 3, 1, 9)

for measure in range(1, 9):
    if measure % 4 == 0:
        fitMedia(scratch, 4, measure, measure + 1)
```

Functions: Recycle and Reuse!

```
def sectionA(start, end):

    jazzDrums = HIP_HOP_JAZZDRUMS1_4M
    synthDrums = HIP_HOP_SYNTHDRUMS3_2M
    keys = TECHNO_ELDERSONSYNTH_2M
    scratch = HIP_HOP_DJSCRATCH2_2M

    fitMedia(jazzDrums, 1, start, end)
    fitMedia(synthDrums, 2, start, end)
    fitMedia(keys, 3, start, end)

    for measure in range(start, end):
        if measure % 4 == 0:
            fitMedia(scratch, 4, measure, measure + 1)
```

Now I can use this section anywhere!

```
sectionA(1, 9)
sectionA(17, 25)
```

Copyright Rights – if you created it!

- to make copies
- to make derivative works (a new work based on the original – like a movie adaptation of a book)
- to distribute copies
- to perform publicly
- to display publicly
- to digitally transmit

What is owned in Music?

- “Concept” – Melody, Harmony, Notes
- Lyrics
- Sheet Music
- Any recordings
- Live Performances / Performing Rights
- Instrumentalists / performers

Fair Use

- **(1) The purpose and character of the use.** This is also the part of fair use that covers things like parody and satire.
- **(2) The nature of the copyrighted work.** Basically, fair use is less likely if the original work is fiction rather than nonfiction, and if it is published rather than unpublished. Note that putting something on the Internet counts as being published.
- **(3) Amount used.** The more of an original work you use, the less likely it is to be fair use. So quoting one line from a book in a book review is probably fair use. Copying an entire book but changing one word and republishing it is probably not.
- **(4) Market harm**

Creative Commons

- "… you have to put my name on it." - Attribution (BY)
- "… you can't change it at all." – No Derivatives (ND)
- "… you can't make money from it." – Non-Commercial (NC)
- "… you have to share whatever new thing you make under the same license." – Share-Alike (SA)

Installed EarSketch Components

1. REAPER – Digital Audio Workstation

   Learn the essentials of inserting and editing Music in Graphic User Interface Environment

2. EarSketch – Python API

   Learn how to use computer programming and python code to create Music. Explore how programming and code allows us to create new musical ideas.

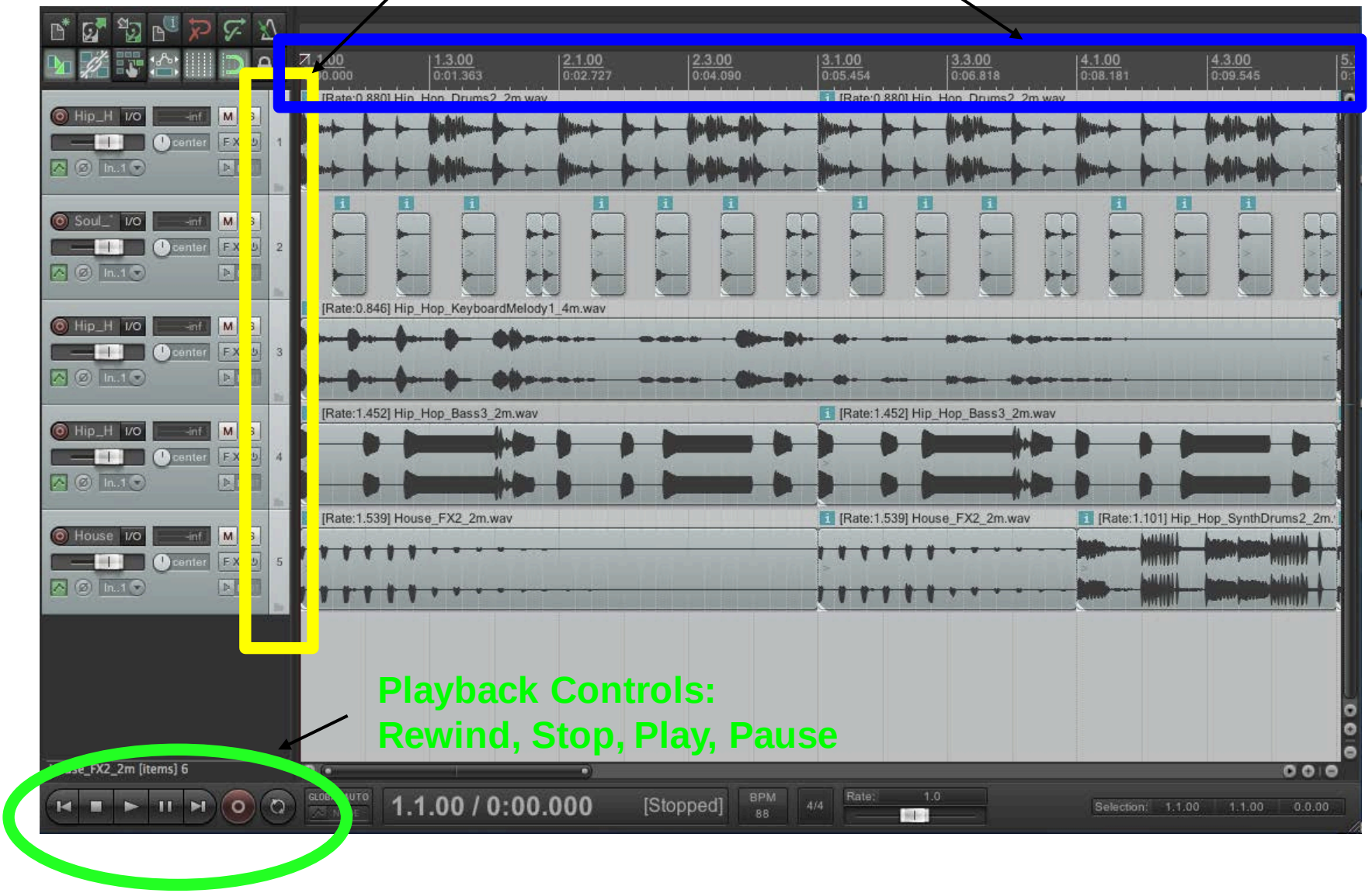
3. EarSketch Website:

   Share our Music and remix ideas to create new music

Digital Audio Workstation: Reaper

- Tracks: Timeline where you put Audio Files
  Usually one instrument or section per track.
- PlayBack Controls:
  Volume/Pan
  Play / Stop / Record
  Action Window – where you run Scripts
- Reaper is similar to Audacity or Garage Band
- Reaper is used by professionals in Studio work.

Reaper


Tracks: Hold Sounds
Measures: Timeline
1.3.00
0:01.363
2.1.00
0:02.727
2.3.00
0:04.090
3.1.00
0:05.454
3.3.00
0:06.818
4.1.00
0:08.181
4.3.00
0:09.545
[Rate:0.880] Hip_Hop_Drums2_2m.wav
[Rate:0.846] Hip_Hop_KeyboardMelody1_4m.wav
[Rate:1.452] Hip_Hop_Bass3_2m.wav
[Rate:1.539] House_FX2_2m.wav
[Rate:1.101] Hip_Hop_SynthDrums2_2m.
Hip_H
Soul_
House
1.1.00 / 0:00.000
[Stopped]
BPM
88
4/4
Rate:
1.0
Selection:
Playback Controls:
Rewind, Stop, Play, Pause

Komodo: Python IDE

- IDE: **I**ntegrated **D**evelopment **E**nvironment

- Program that gives the user a space to type in and save programs. Like a "Word Processor" for programming. Most IDE's will color code text as you type it in to help you with syntax and context. We will use Komodo for Python and Reaper.