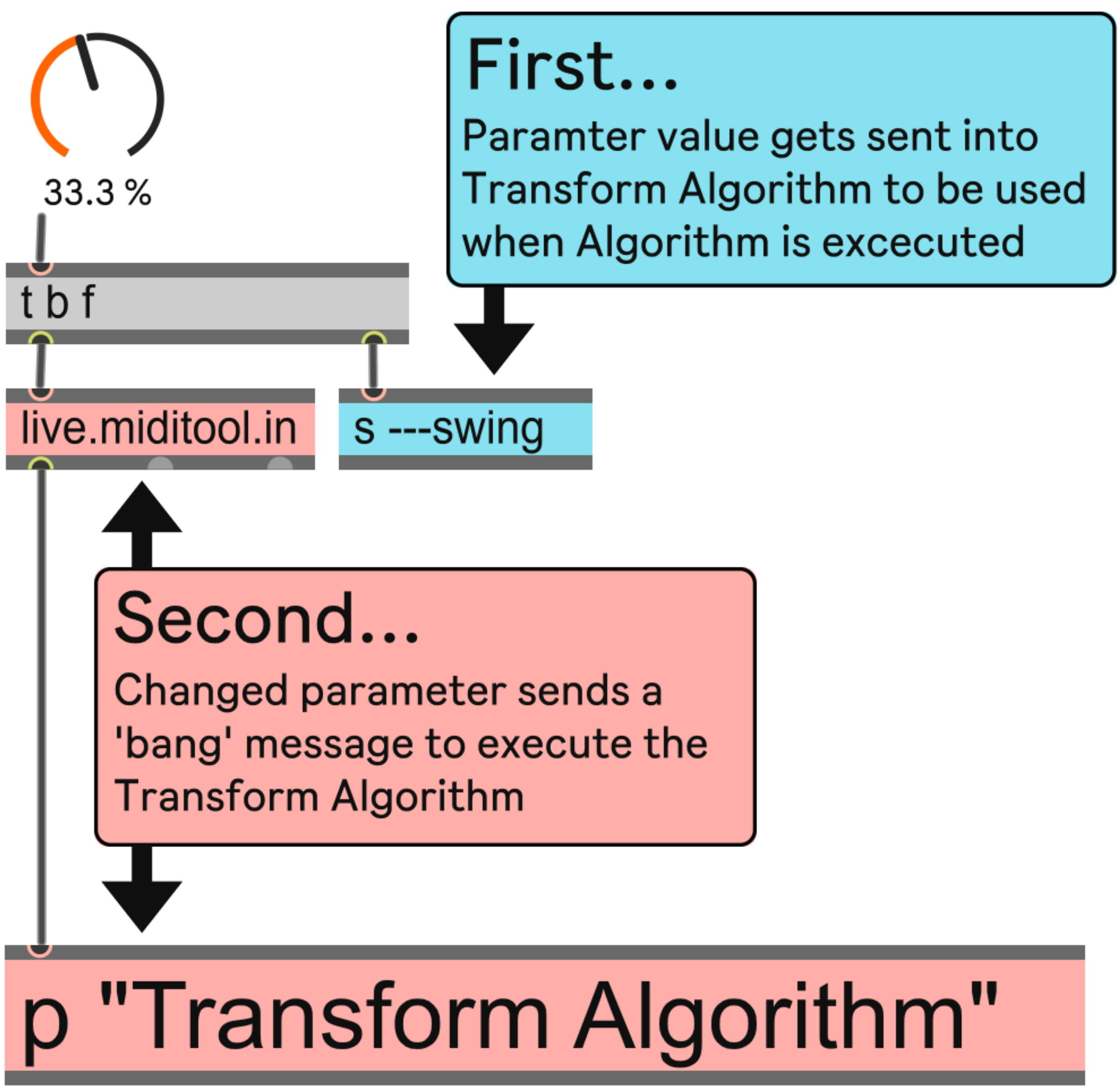
Building a Midi Transformer

STEP 1

Interface parameters get sent to prepare the transform algorithm



STEP 2

Interface parameters trigger the retrieval of note data from Live

The transform algorithm gets executed by sending a “bang” message into the live.miditool.in object, which then outputs an **array** of note data from live.

An **array** is a collection of **elements**, each identified by an **index**. In this case, each element in the array contains information about a MIDI note inside of the clip. If you were to select 8 notes to transform, the live.miditool.in object would output an array with 8 elements… 1 for each note.

Element	note 1 data	note 2 data	note 3 data	note 4 data	note 5 data	note 6 data	note 7 data	note 8 data
Index	0	1	2	3	4	5	6	7

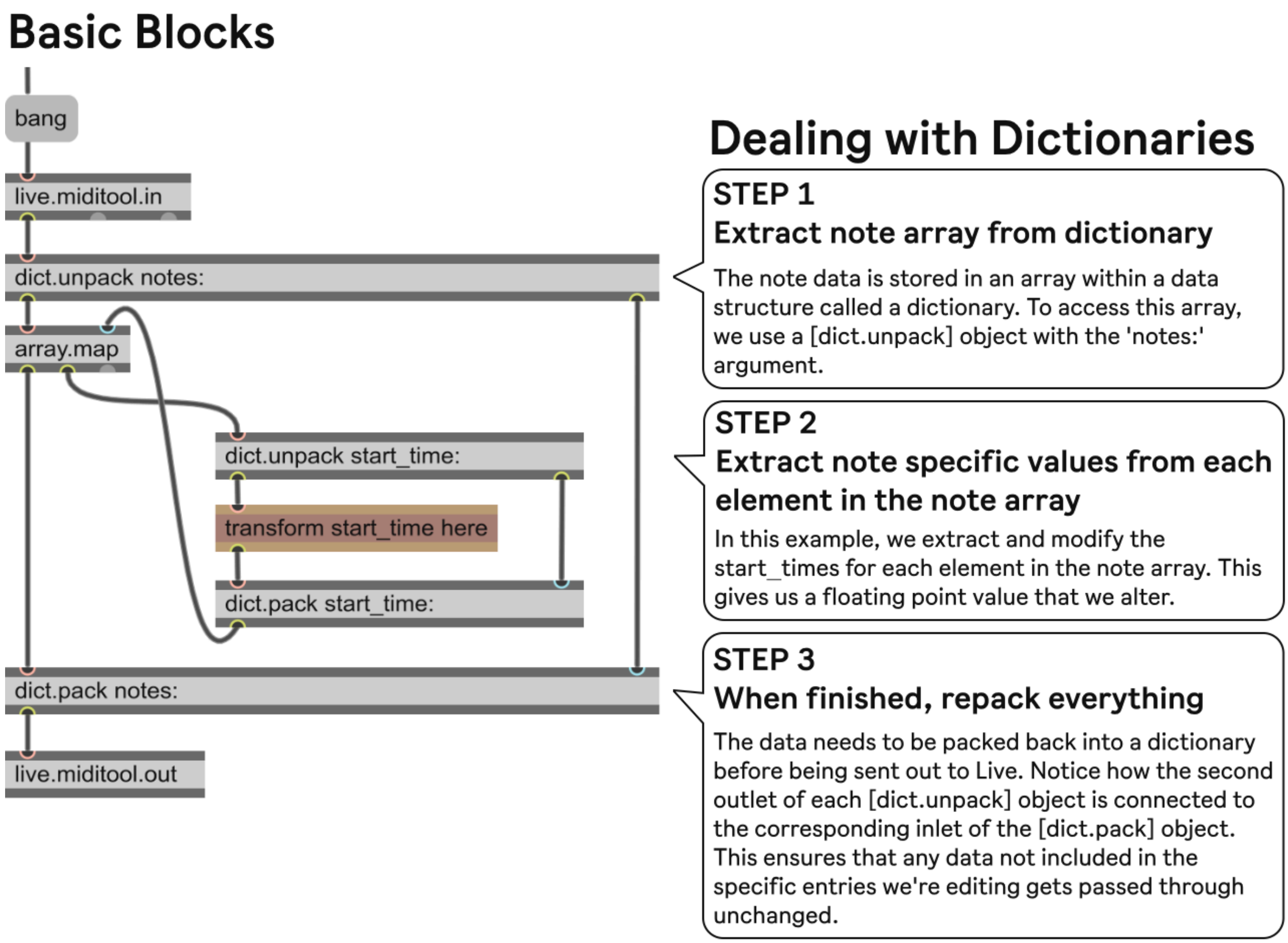
Each element of the array contains the following information…

note_id:	The first note in the clip = 1. The second = 2… This is related to the clip, not the selected notes
pitch:	Pitch of the note (0-127)
start_time:	Where the note is positioned in the clip. The end of the first 1/4 note = 1. The Second = 2…
duration:	The length of the note. So end-time = start_time + duration…
velocity:	The velocity of the note (0-127)
mute:	If the note is disabled in the editor 1, otherwise 0
probability:	The note’s probability parameter value
velocity_deviation:	The note’s velocity deviation parameter value
release_velocity:	The release velocity of the note (0-127)

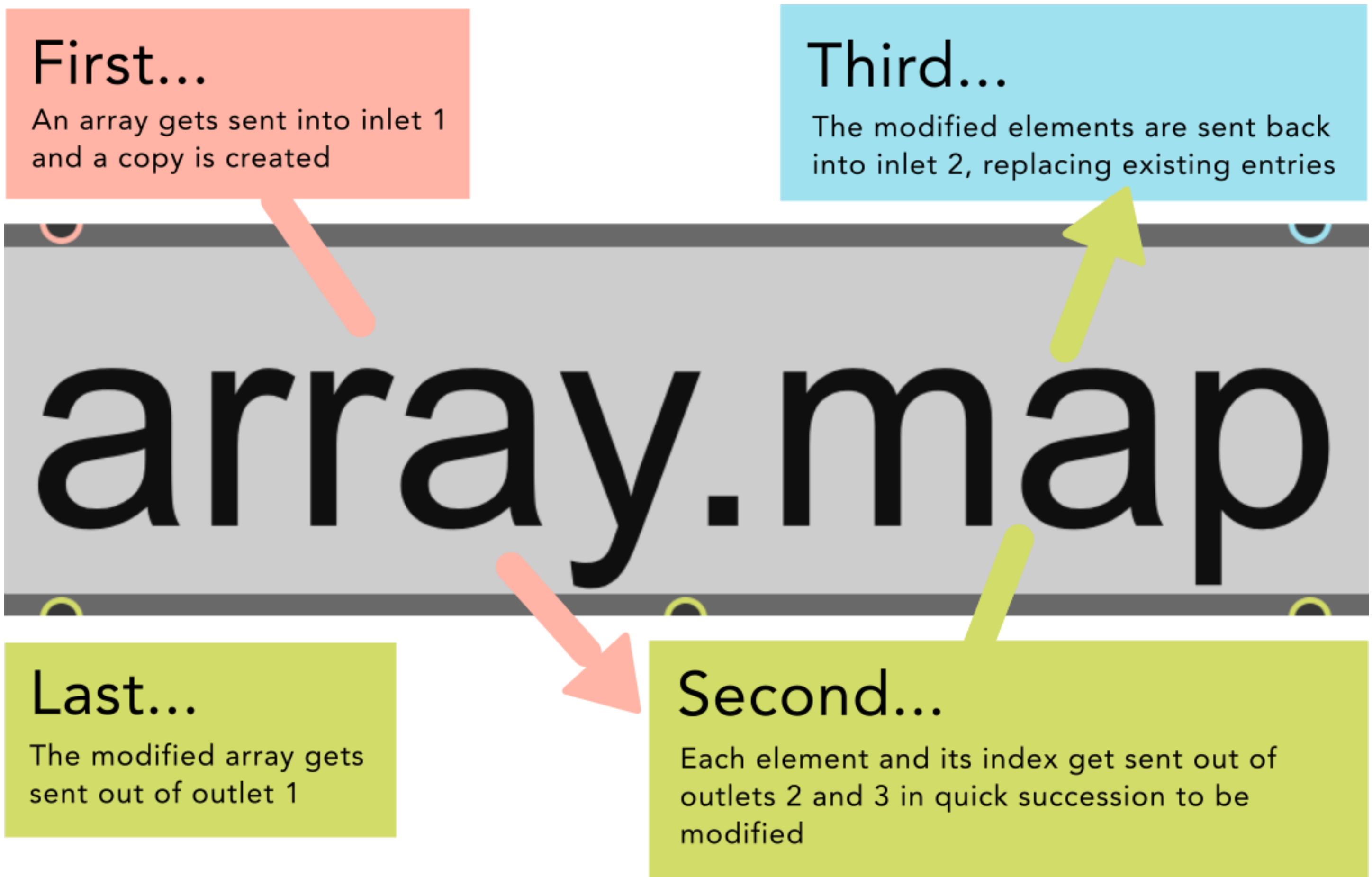
STEP 3

Retrieved note data gets unpacked and transformed

To access and modify this information, we use a series of dict.unpack and array objects. The image below shows the basic blocks required to extract and modify note data from Live.



The array.map object lets us iterate through and modify elements of an array. The image below shows what happens when we send an array into an array.map object.



EXAMPLE PATCH

Here's the Swing Quantization Algorithm from our free Midi Transformer, Swing

[Transform Note Array] patcher from Swing.amxd

This code segment extracts note 'start_times' from arrays of MIDI Note data.

To understand this process, it's helpful to know how these 'start_times' are represented within the arrays.

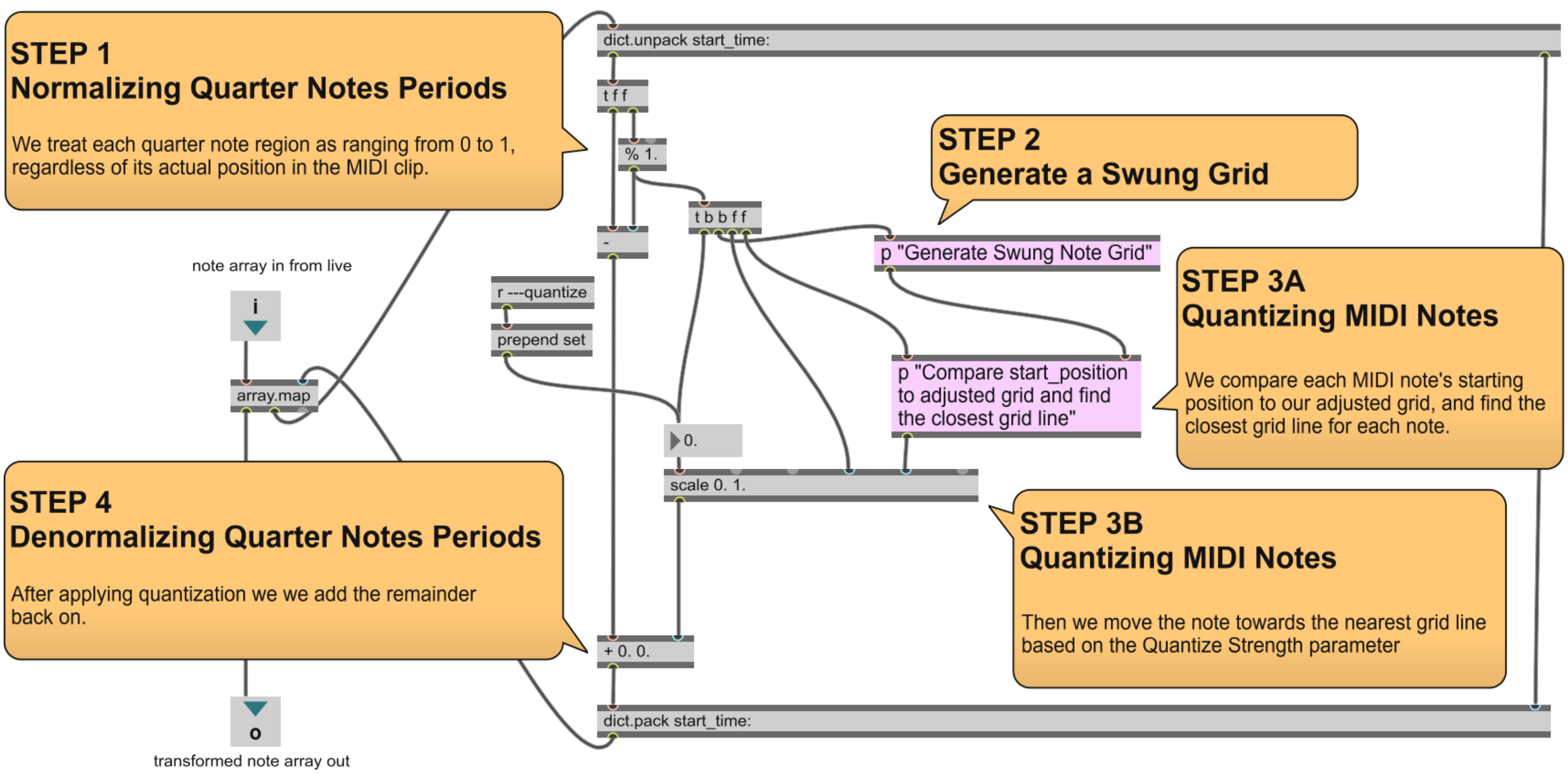
Each notes "start_time" is portrayed as a number relative to the the MIDI Clip inwhich the note resides.

The image below depicts one bar in 4/4 time. Each of the numbers along the top represent quarter note periods of time.

BASIC NOTE LENGTHS

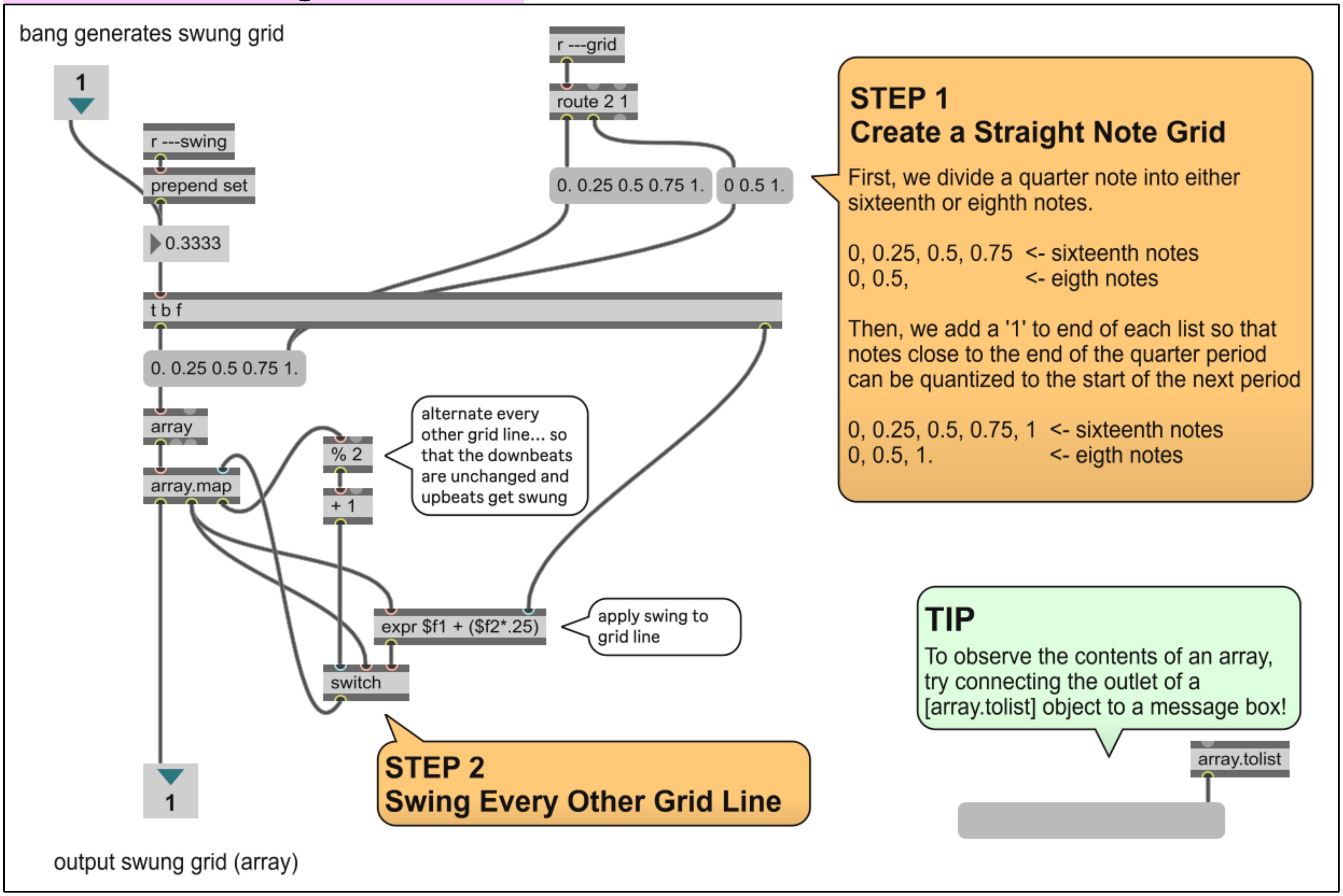
1/4 note = 1.0
1/8 note = 0.5
1/16 note = 0.25

So... if a MIDI note has a start_time of 5, we know that MIDI note starts on the end of the 5th quarter note period

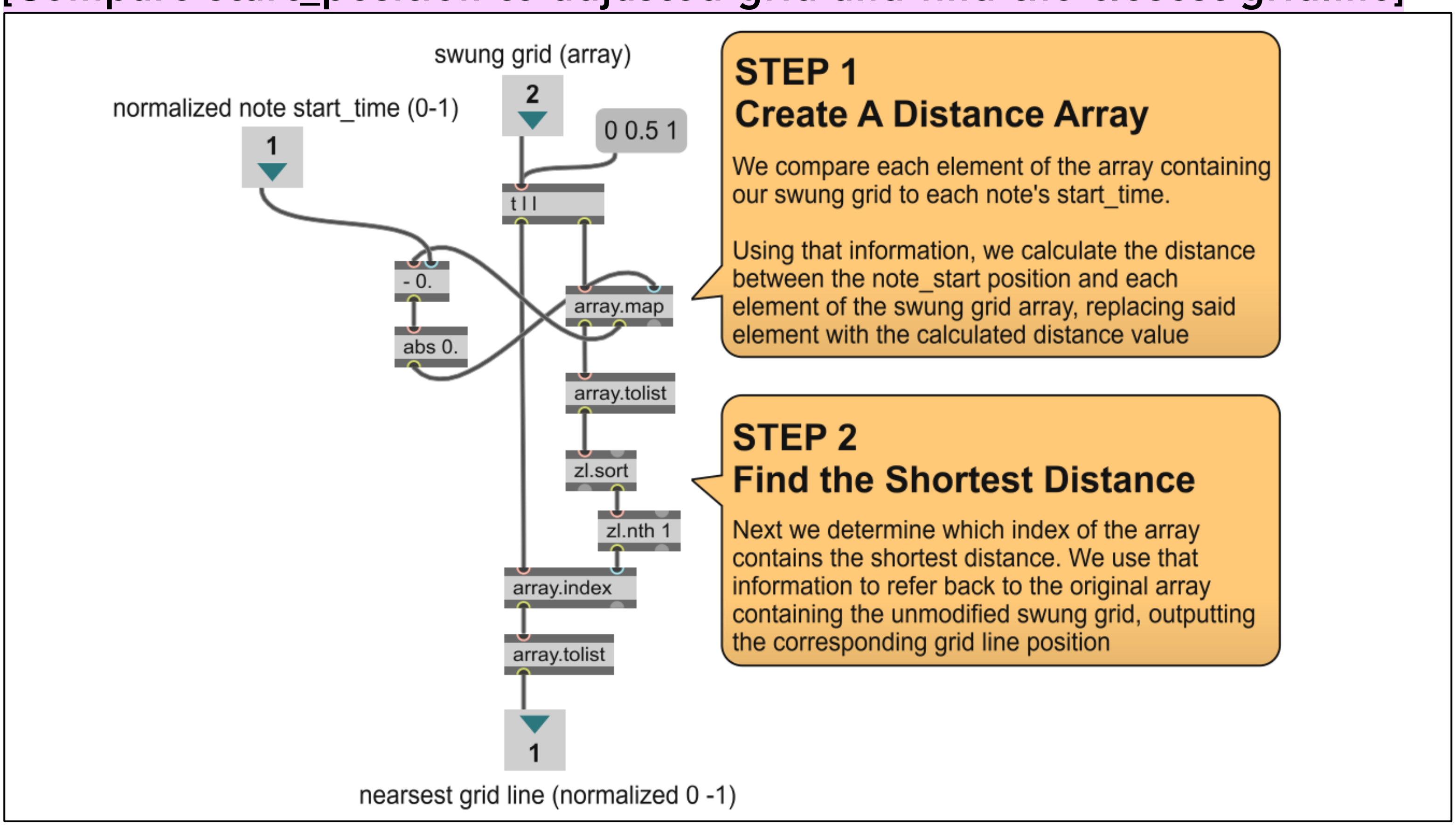


Abstractions on next page

[Generate Swung Note Grid]



[Compare start_position to adjusted grid and find the closest gridline]



Want to learn more?

Dig into Swing… Our free Midi Transformer used in this lesson → windmakeswaves.com/swing

Cycling74 also has a great lesson on Midi tools here…
https://docs.cycling74.com/max8/vignettes/live_miditools?q=array