Canlı Kodlama Yöntemiyle Müzik Üretimi Pratikleri

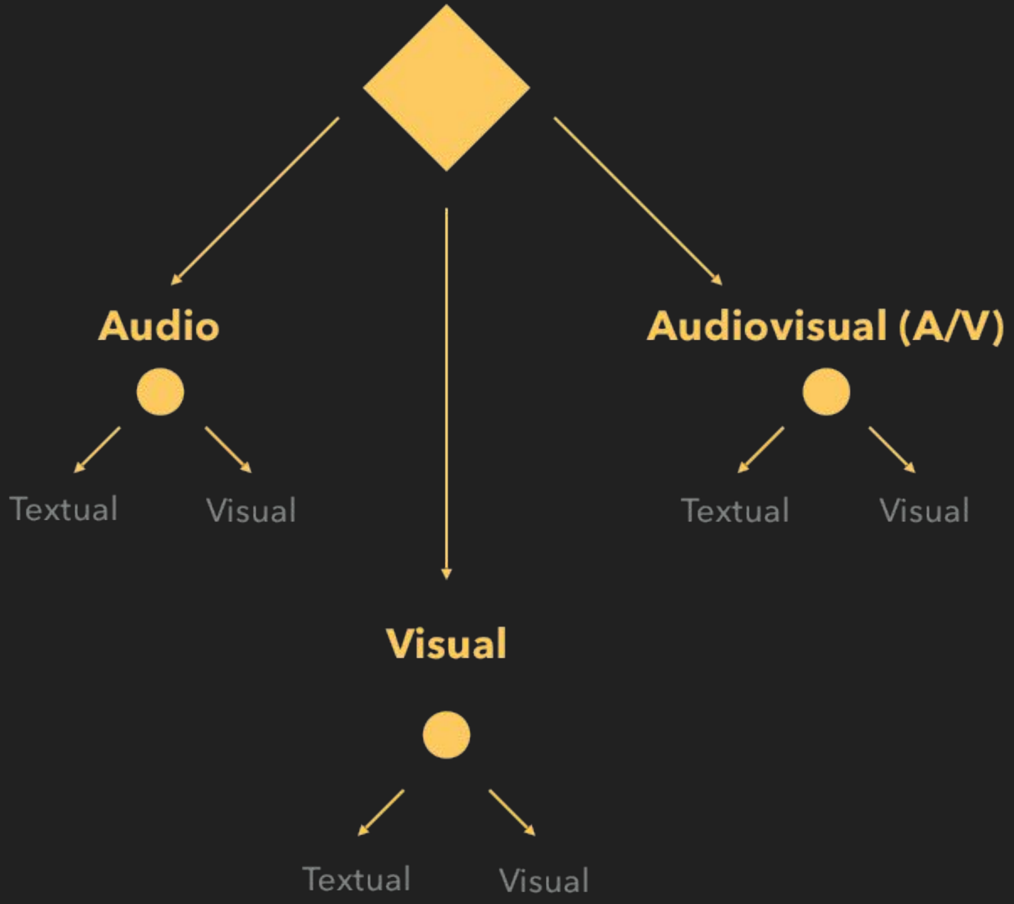
RAW

- www.rawlivecoding.com
- facebook.com/rawlivecoding
- instagram.com/rawlivecoding
- spotify/RAW
- soundcloud.com/rawlivecoding

```
]
jux (iter 4) $ smash 4 [1,2,3
karlsruhe.tidal   All
Data.Maybe| tidal> t
` UTC,89.637343,1.1
tidal> tidal> tidal>
l.VolcaKeys Sound.O
Tidal.VolcaKeys Soun
```

What is Live Coding?

Live Coding


Audio
Textual
Visual
Visual
Textual
Visual
Audiovisual (A/V)
Textual
Visual

Brief History of Live Coding

- 2004, Changing grammars, University of Fine Arts of Hamburg (HfbK)
- 2004, TOPLAP Organization
- 2011, Algorave (N. Collins, A. McLean)

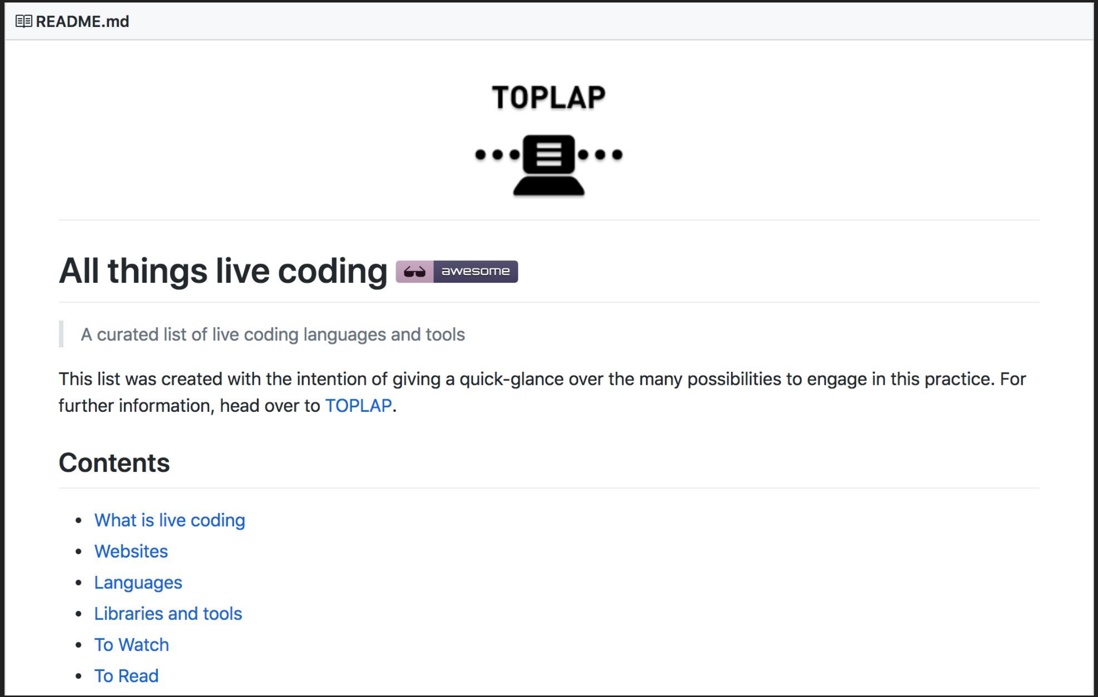
http://toplap.org

https://toplap.org/wiki/ManifestoDraft

Algorave

Live Coding Environments

https://github.com/toplap/awesome-livecoding



README.md

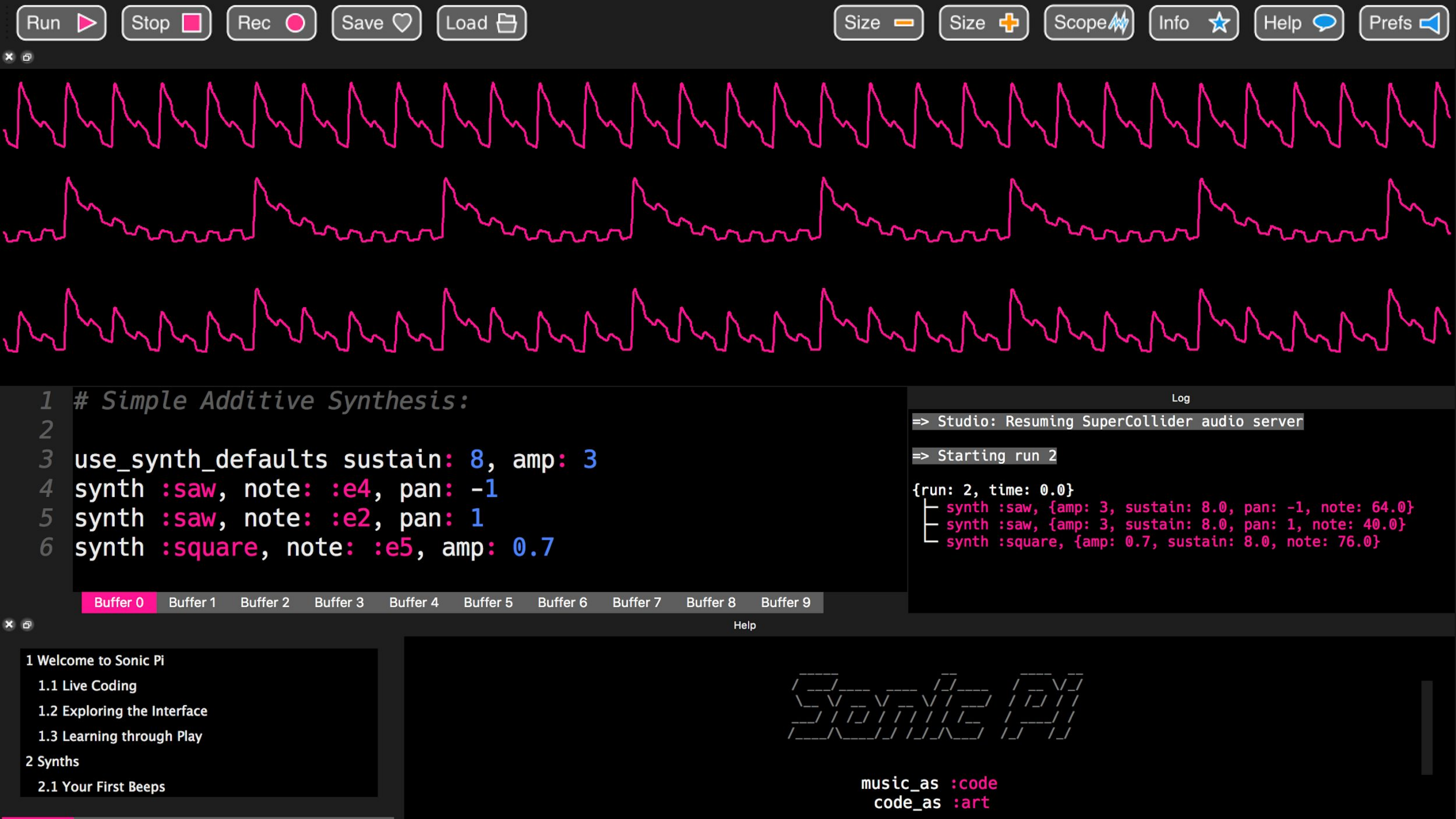
TOPLAP

All things live coding

> A curated list of live coding languages and tools

This list was created with the intention of giving a quick-glance over the many possibilities to engage in this practice. For further information, head over to TOPLAP.

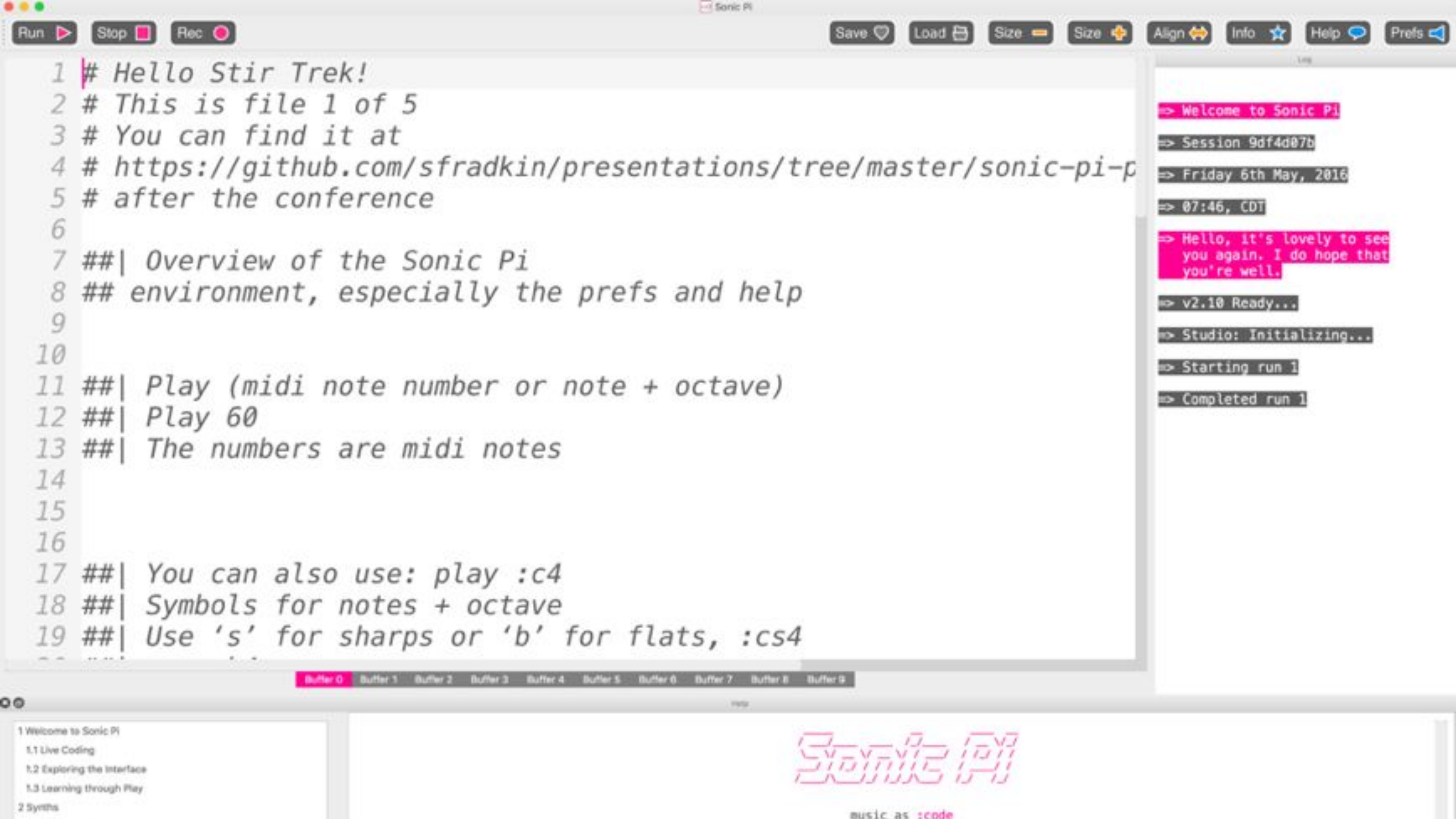
Contents



```
# Simple Additive Synthesis:

use_synth_defaults sustain: 8, amp: 3
synth :saw, note: :e4, pan: -1
synth :saw, note: :e2, pan: 1
synth :square, note: :e5, amp: 0.7
```

```
=> Studio: Resuming SuperCollider audio server

=> Starting run 2

{run: 2, time: 0.0}
 ├─ synth :saw, {amp: 3, sustain: 8.0, pan: -1, note: 64.0}
 ├─ synth :saw, {amp: 3, sustain: 8.0, pan: 1, note: 40.0}
 └─ synth :square, {amp: 0.7, sustain: 8.0, note: 76.0}
```

```
music_as :code
 code_as :art
```



```
# Hello Stir Trek!
# This is file 1 of 5
# You can find it at
# https://github.com/sfradkin/presentations/tree/master/sonic-pi-p
# after the conference

##| Overview of the Sonic Pi
## environment, especially the prefs and help



##| Play (midi note number or note + octave)
##| Play 60
##| The numbers are midi notes



##| You can also use: play :c4
##| Symbols for notes + octave
##| Use 's' for sharps or 'b' for flats, :cs4
```

```
=> Welcome to Sonic Pi

=> Session 9df4d07b

=> Friday 6th May, 2016

=> 07:46, CDT

=> Hello, it's lovely to see
   you again. I do hope that
   you're well.

=> v2.10 Ready...

=> Studio: Initializing...

=> Starting run 1

=> Completed run 1
```

Sonic Pi

music_as :code

How it works?

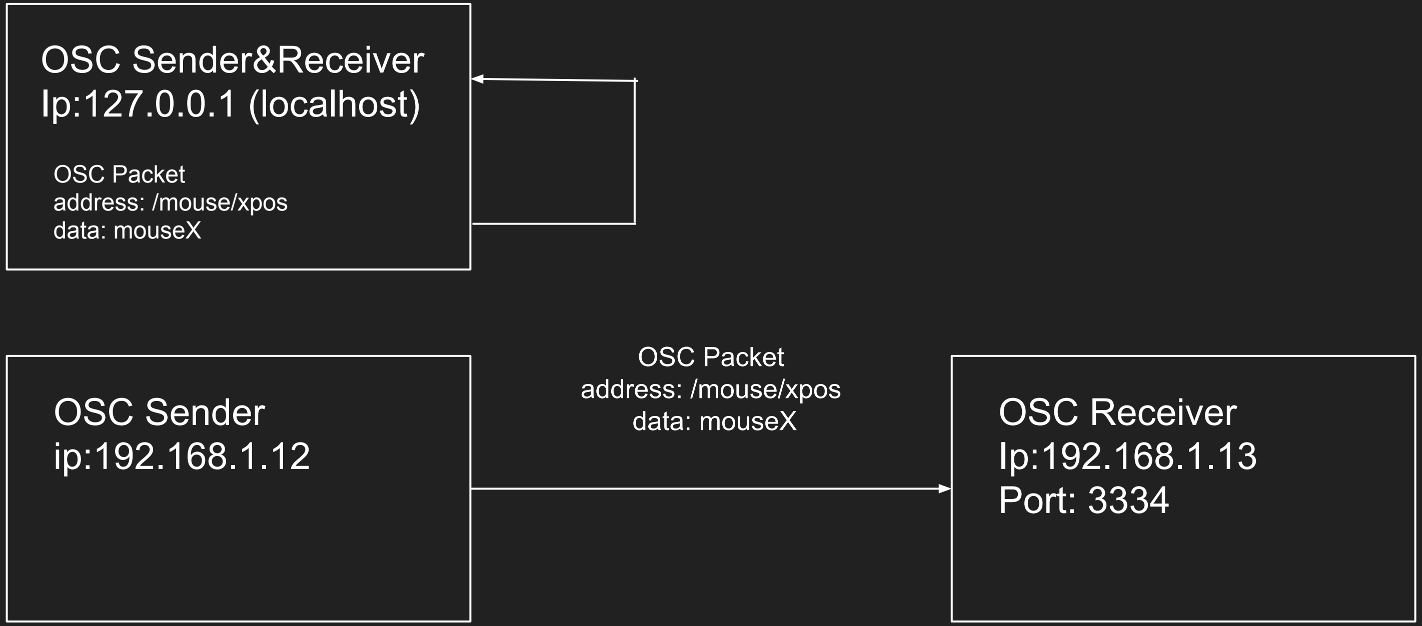
- Based on SuperCollider
- Sonic Pi GUI sends/receives OSC messages to server
- Server sends messages to SC

OSC (Open Sound Control)

- Ongoing research project by Berkeley Center for New Music and Audio Technology (CNMAT)
- Open Sound Control (OSC) is an open,
- Transport-independent,
- Message-based protocol based on UDP
- Developed for communication among computers, sound synthesizers, and other multimedia devices.

OSC Communication

For more than one devices, you need a local network, each device need to be delivering unique ip’s..
Be aware if there is any firewall restriction



OSC Formatting

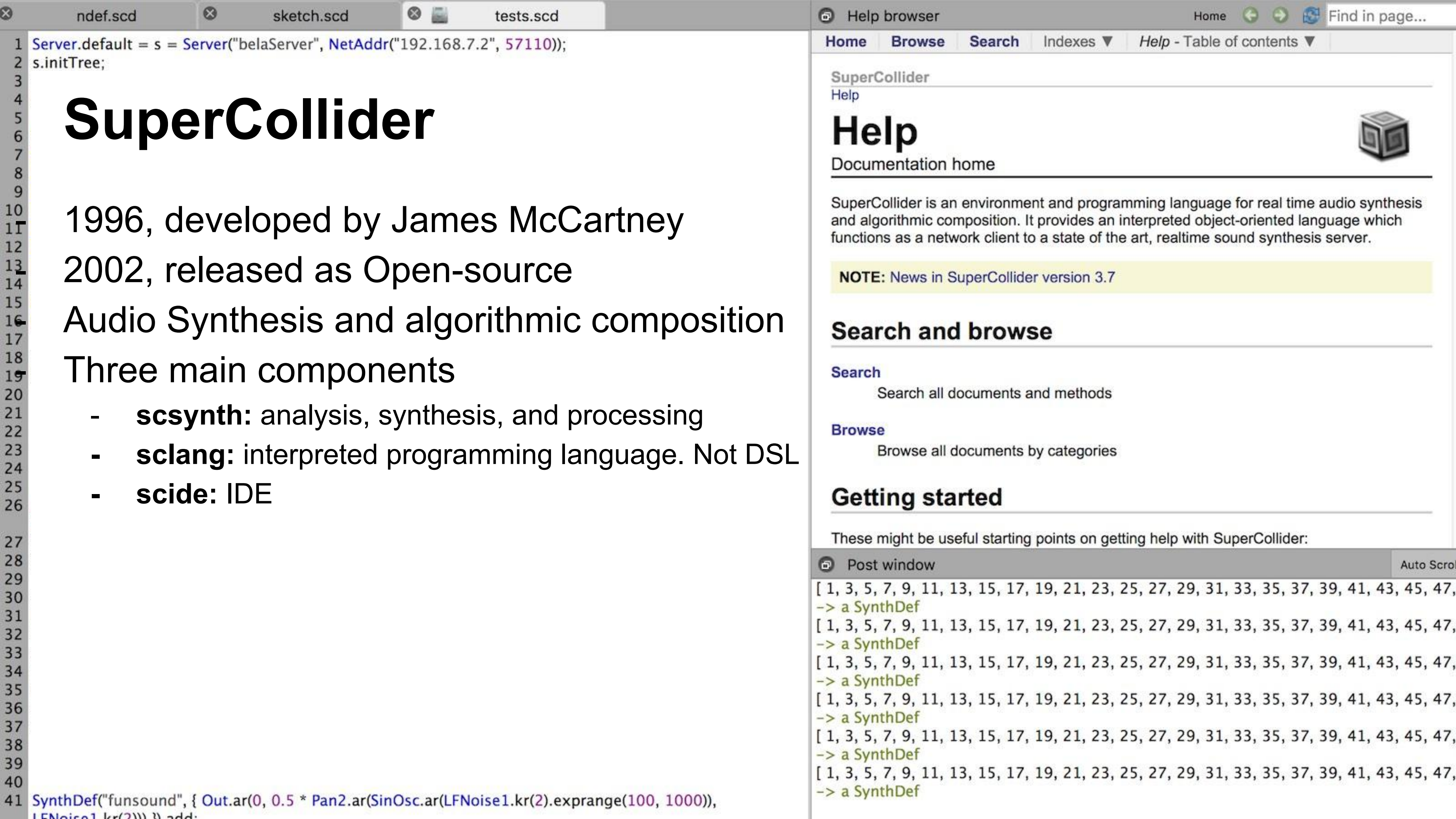
All OSC data is composed of the following fundamental data types:

Int32 i.e. 3, 5, 188

Float32 3.4, 2.7, 56.8

OSC-string “hello world”

The unit of transmission of OSC is an OSC Packet. Every OSC Packet requires an address and a data information.



ndef.scd sketch.scd tests.scd

```
Server.default = s = Server("belaServer", NetAddr("192.168.7.2", 57110));
s.initTree;
```

SuperCollider

1996, developed by James McCartney

2002, released as Open-source

Audio Synthesis and algorithmic composition

Three main components

- **scsynth:** analysis, synthesis, and processing
- **sclang:** interpreted programming language. Not DSL
- **scide:** IDE

```
SynthDef("funsound", { Out.ar(0, 0.5 * Pan2.ar(SinOsc.ar(LFNoise1.kr(2).exprange(100, 1000)),
LFNoise1.kr(2))) }).add;
```

Help browser

SuperCollider
Help

Help

Documentation home

SuperCollider is an environment and programming language for real time audio synthesis and algorithmic composition. It provides an interpreted object-oriented language which functions as a network client to a state of the art, realtime sound synthesis server.

NOTE: News in SuperCollider version 3.7

Search and browse

Search
Search all documents and methods

Browse
Browse all documents by categories

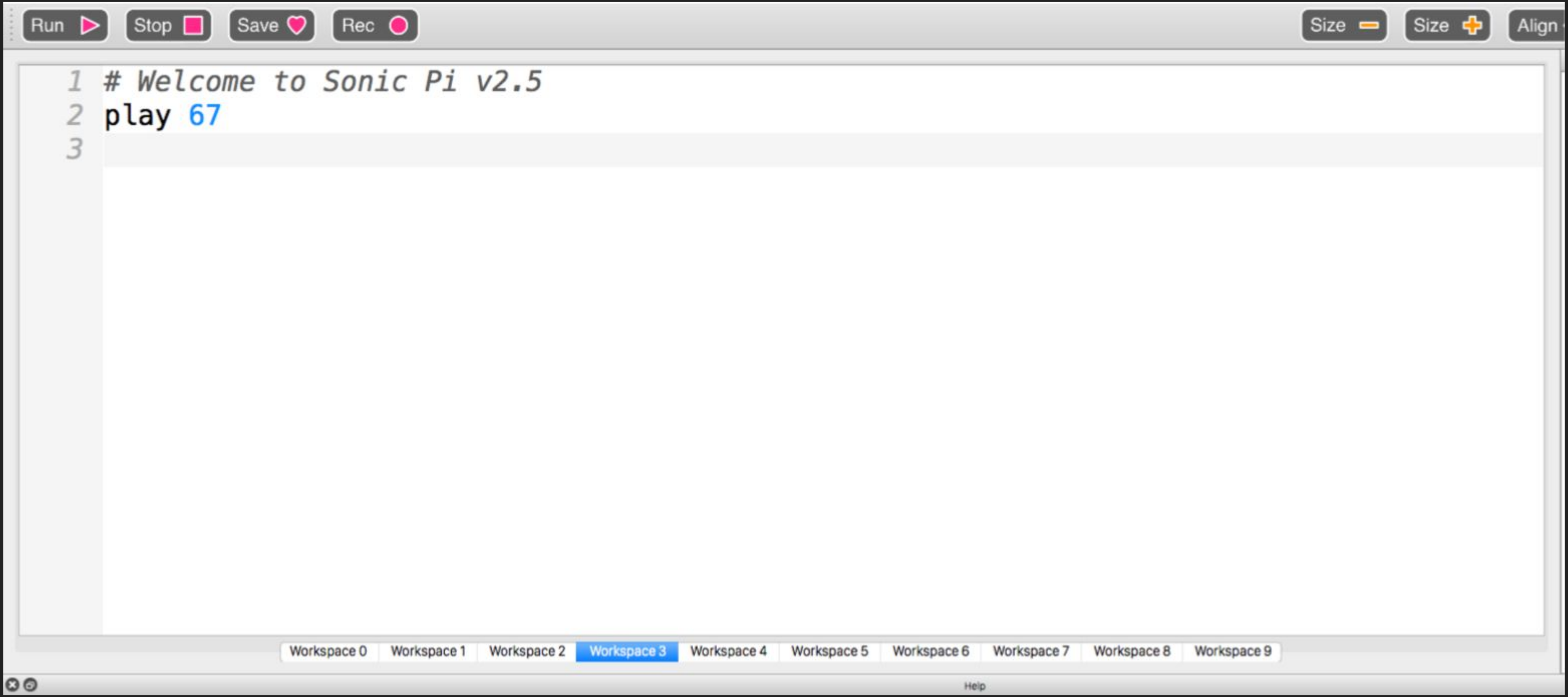
Getting started

These might be useful starting points on getting help with SuperCollider:

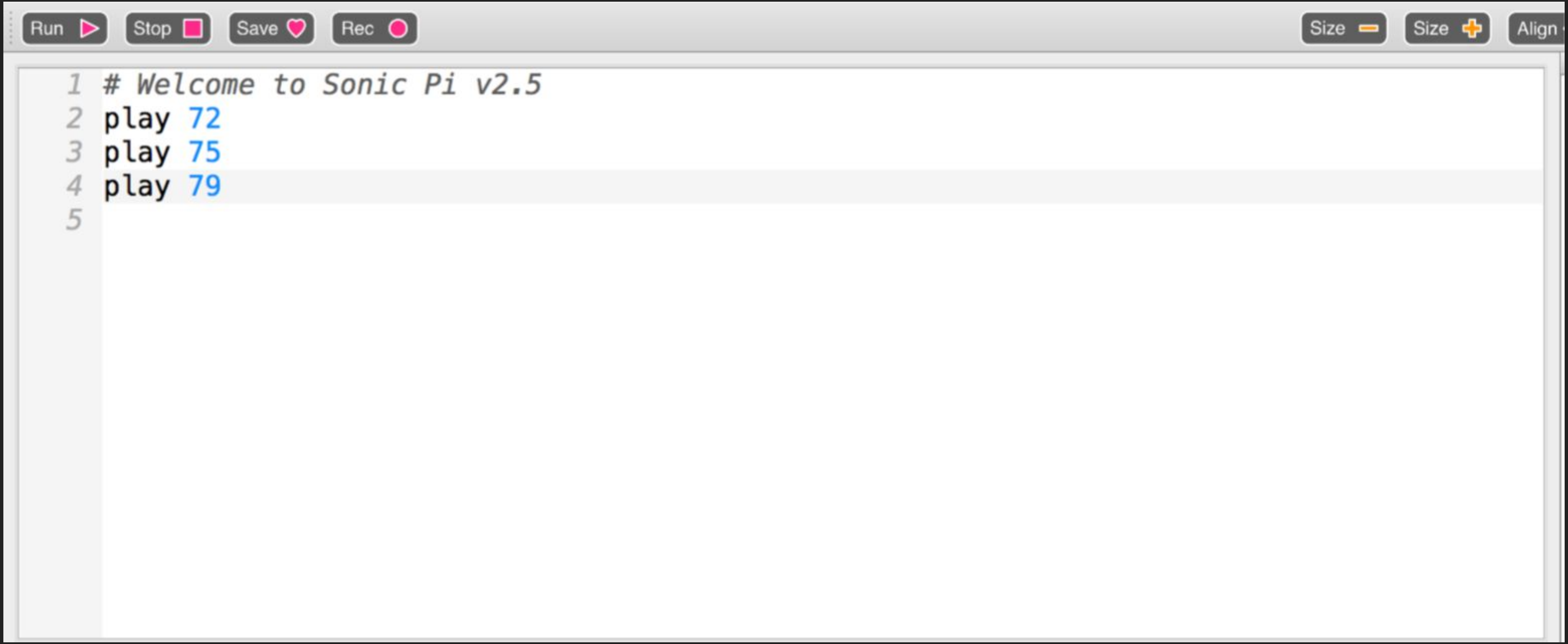
Post window

```
[ 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47,
-> a SynthDef
[ 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47,
-> a SynthDef
[ 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47,
-> a SynthDef
[ 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47,
-> a SynthDef
[ 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47,
-> a SynthDef
[ 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47,
-> a SynthDef
```

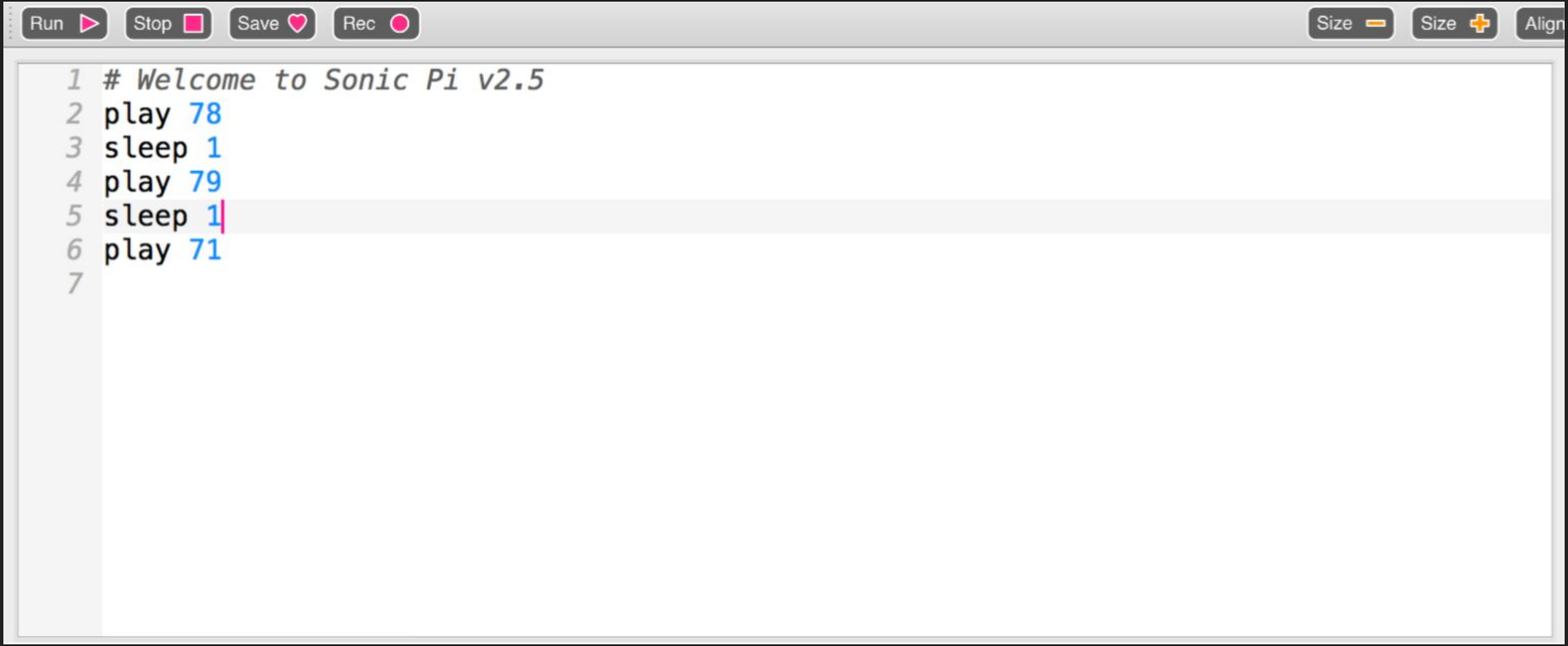
https://sonic-pi.net



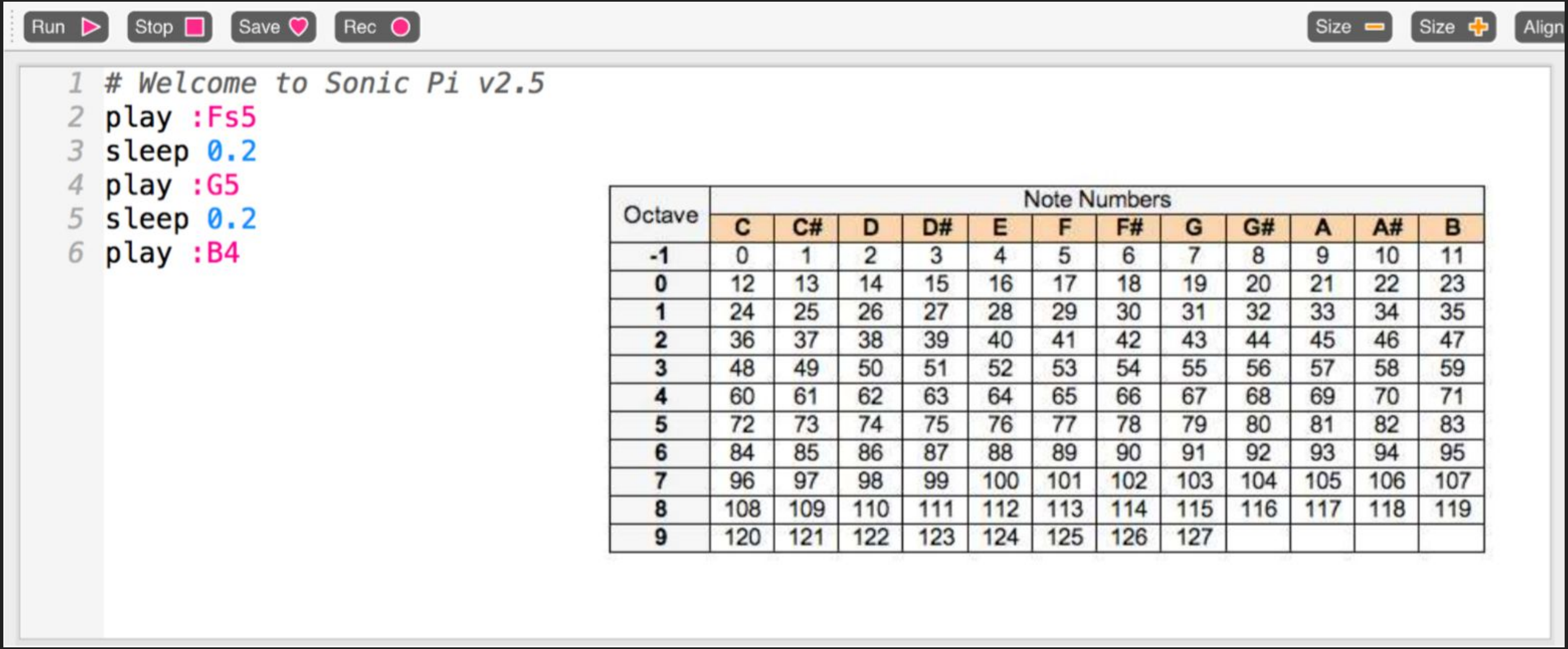
```
# Welcome to Sonic Pi v2.5
play 67
```



```
# Welcome to Sonic Pi v2.5
play 72
play 75
play 79
```

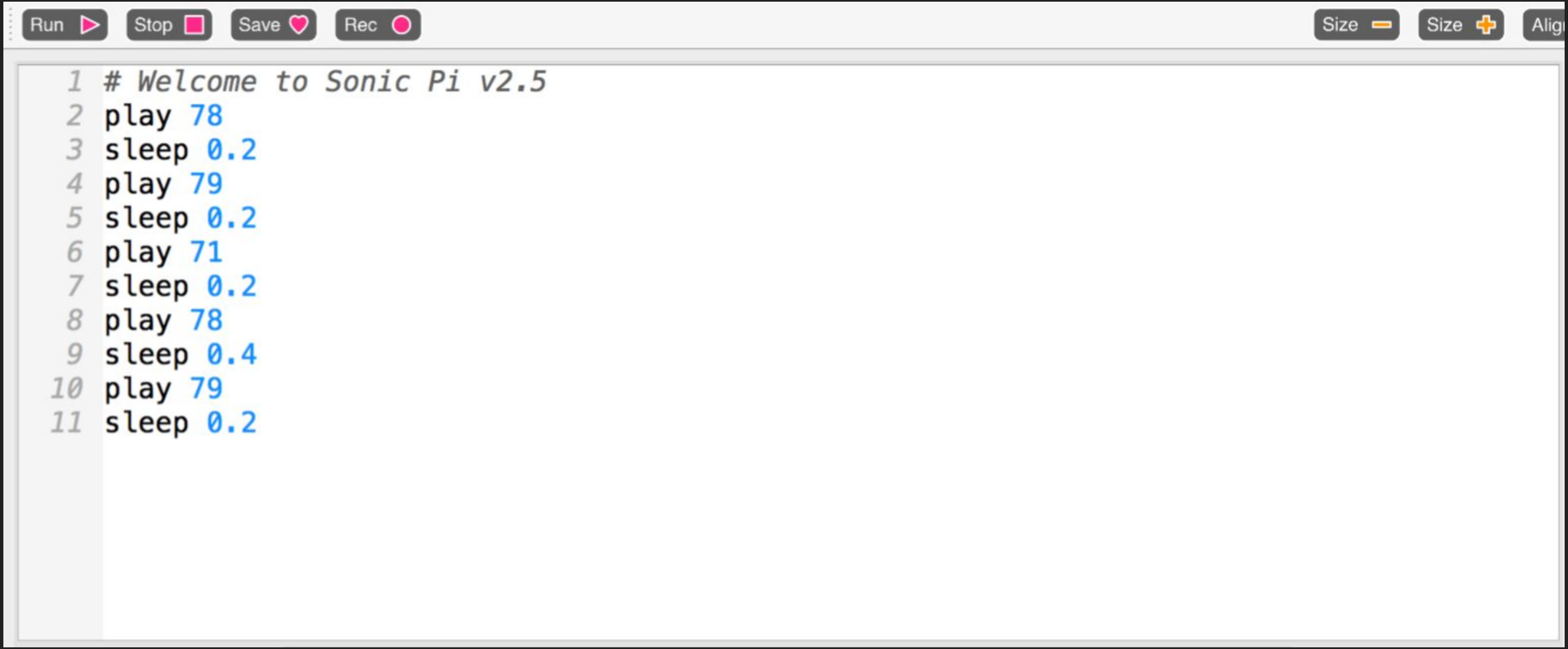


```
# Welcome to Sonic Pi v2.5
play 78
sleep 1
play 79
sleep 1
play 71
```

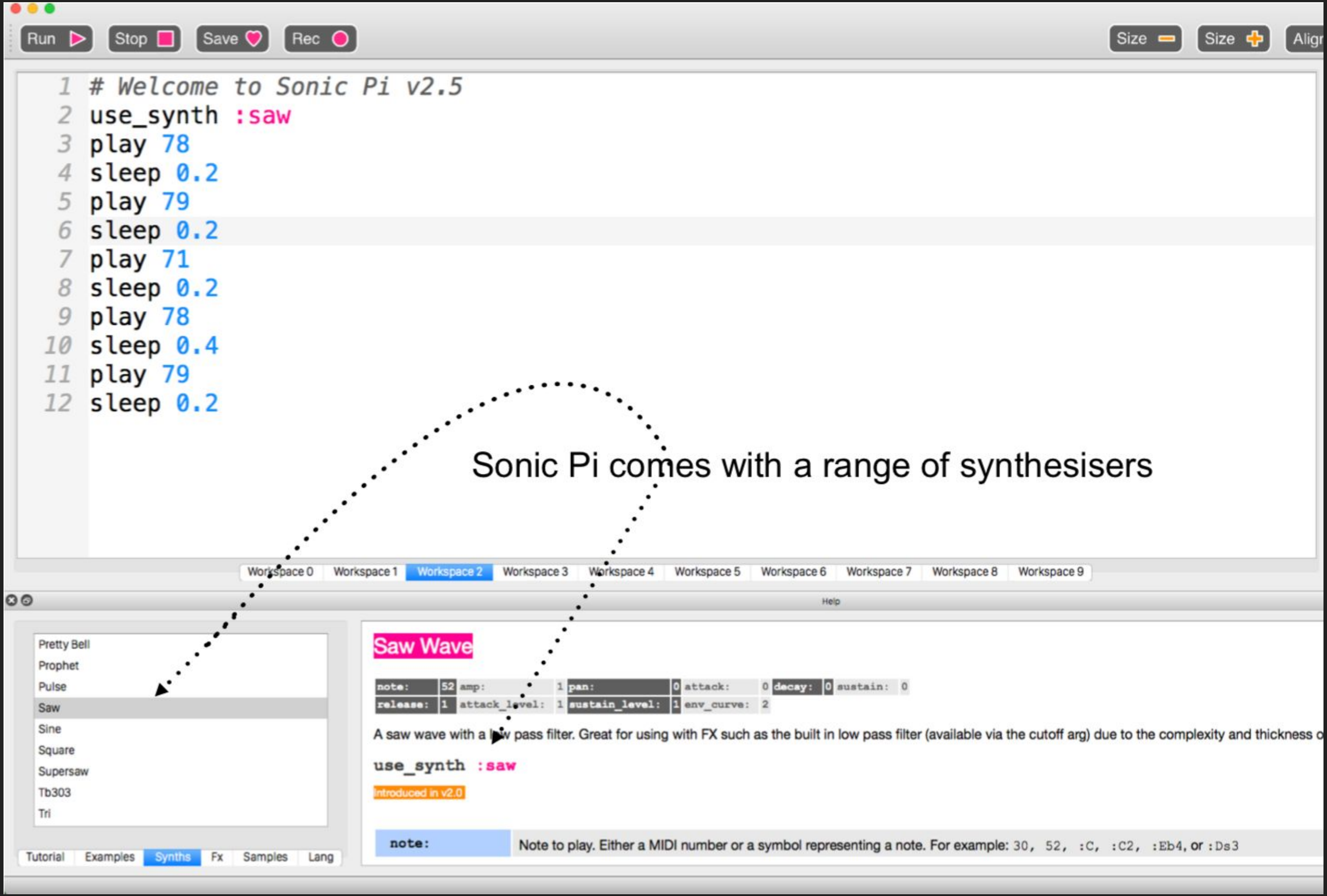


```
# Welcome to Sonic Pi v2.5
play :Fs5
sleep 0.2
play :G5
sleep 0.2
play :B4
```

Octave	Note Numbers											
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-1	0	1	2	3	4	5	6	7	8	9	10	11
0	12	13	14	15	16	17	18	19	20	21	22	23
1	24	25	26	27	28	29	30	31	32	33	34	35
2	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59
4	60	61	62	63	64	65	66	67	68	69	70	71
5	72	73	74	75	76	77	78	79	80	81	82	83
6	84	85	86	87	88	89	90	91	92	93	94	95
7	96	97	98	99	100	101	102	103	104	105	106	107
8	108	109	110	111	112	113	114	115	116	117	118	119
9	120	121	122	123	124	125	126	127				



```
# Welcome to Sonic Pi v2.5
play 78
sleep 0.2
play 79
sleep 0.2
play 71
sleep 0.2
play 78
sleep 0.4
play 79
sleep 0.2
```



```
# Welcome to Sonic Pi v2.5
use_synth :saw
play 78
sleep 0.2
play 79
sleep 0.2
play 71
sleep 0.2
play 78
sleep 0.4
play 79
sleep 0.2
```

Sonic Pi comes with a range of synthesisers

Saw Wave

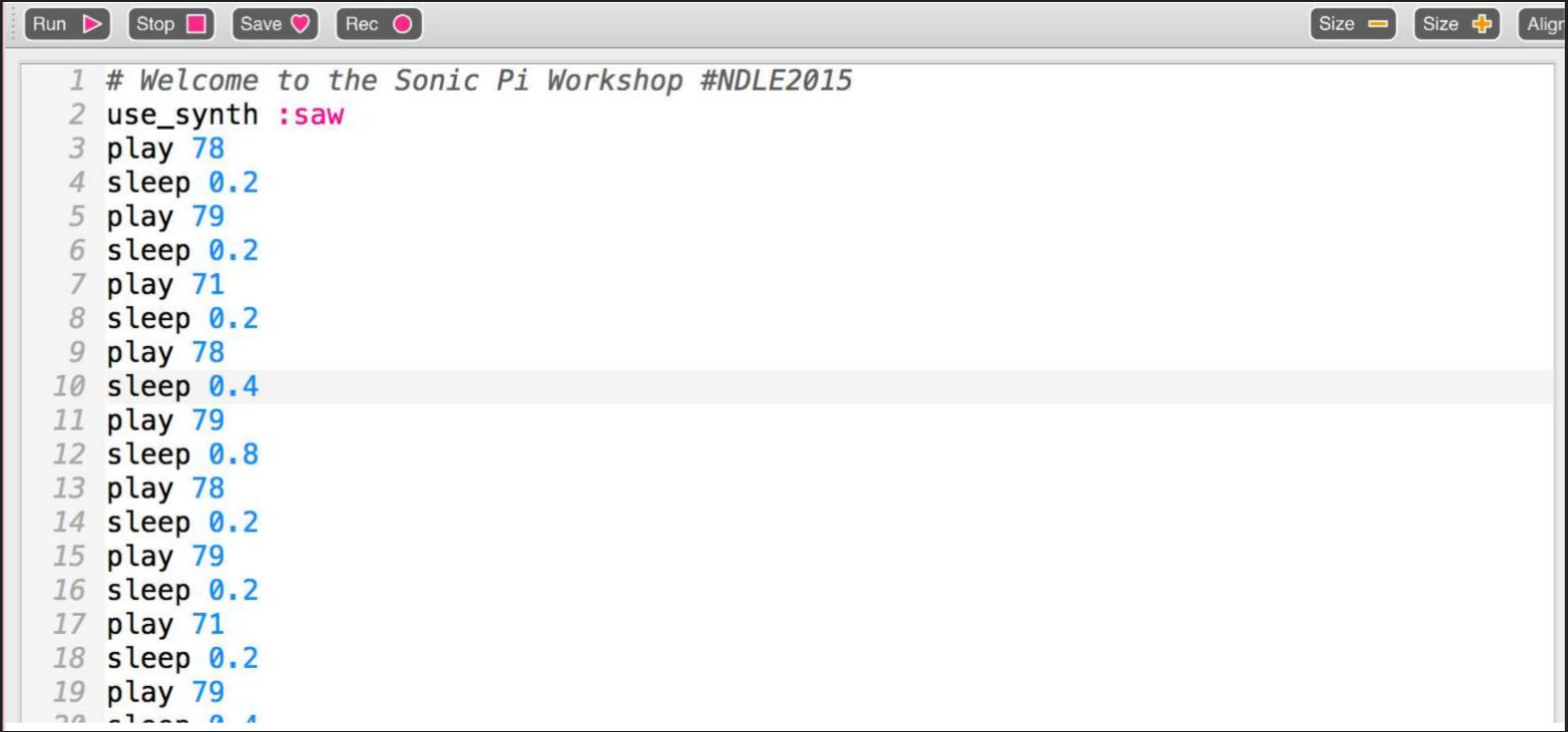
note:	52	amp:	1	pan:	0	attack:	0	decay:	0	sustain:	0
release:	1	attack_level:	1	sustain_level:	1	env_curve:	2				

A saw wave with a low pass filter. Great for using with FX such as the built in low pass filter (available via the cutoff arg) due to the complexity and thickness o

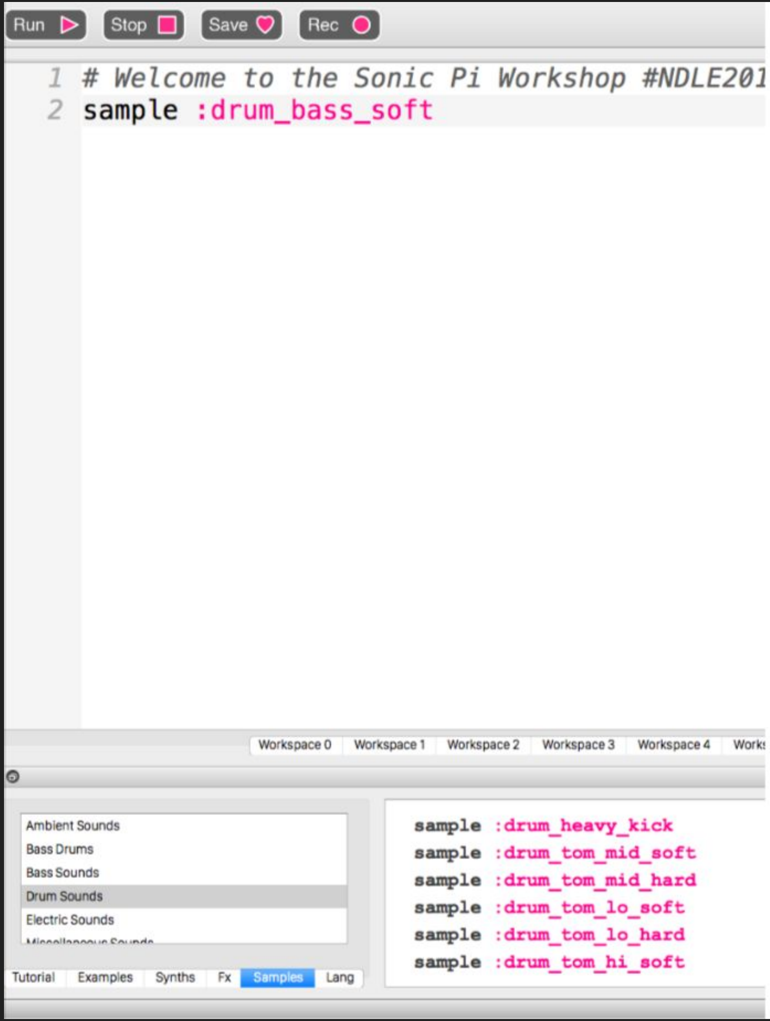
```
use_synth :saw
```

Introduced in v2.0

note:	Note to play. Either a MIDI number or a symbol representing a note. For example: `30, 52, :C, :C2, :Eb4`, or `:Ds3`



```
# Welcome to the Sonic Pi Workshop #NDLE2015
use_synth :saw
play 78
sleep 0.2
play 79
sleep 0.2
play 71
sleep 0.2
play 78
sleep 0.4
play 79
sleep 0.8
play 78
sleep 0.2
play 79
sleep 0.2
play 71
sleep 0.2
play 79
```

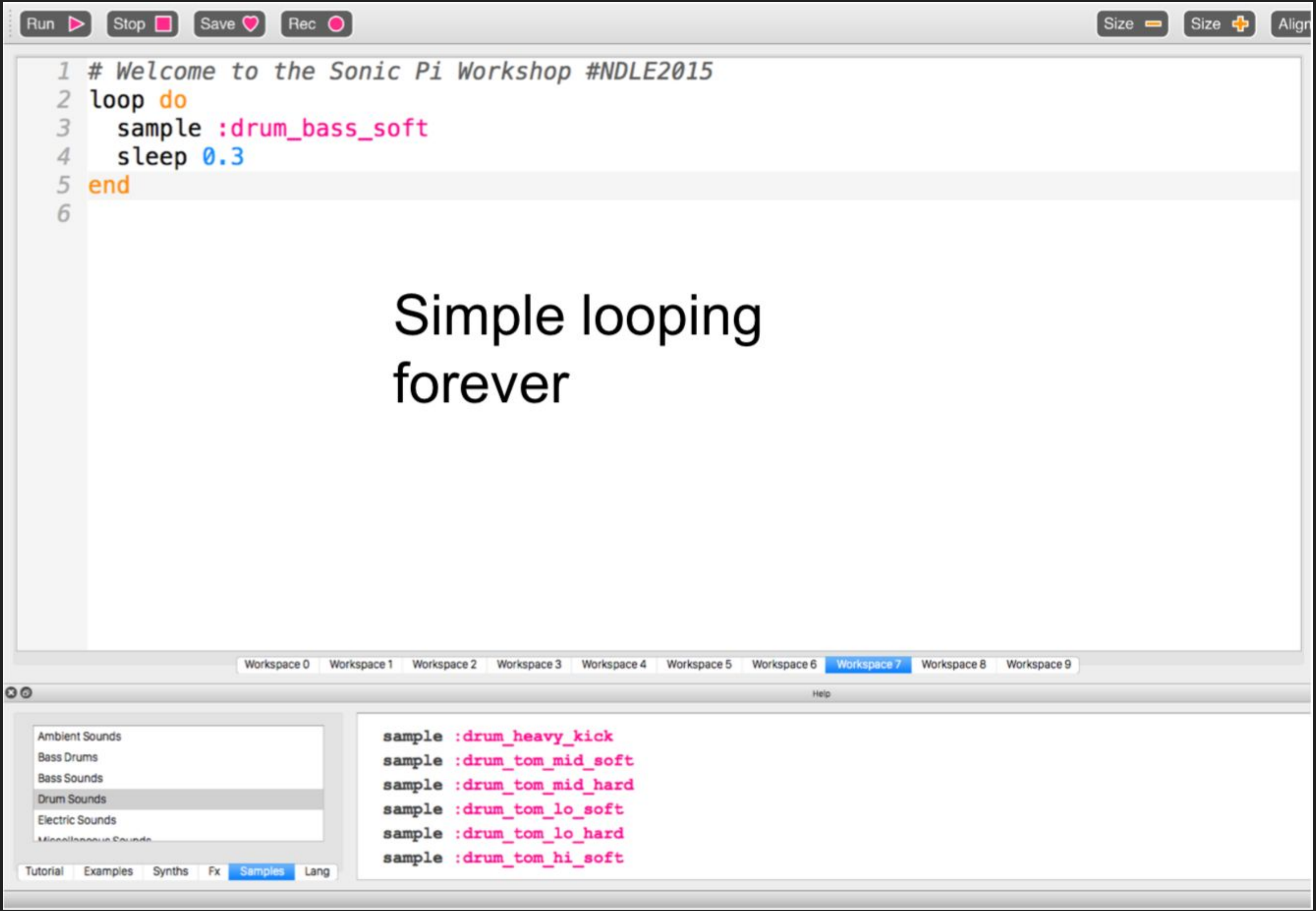


```
# Welcome to the Sonic Pi Workshop #NDLE201
sample :drum_bass_soft
```

```
sample :drum_heavy_kick
sample :drum_tom_mid_soft
sample :drum_tom_mid_hard
sample :drum_tom_lo_soft
sample :drum_tom_lo_hard
sample :drum_tom_hi_soft
```

Open a new workspace and try some samples

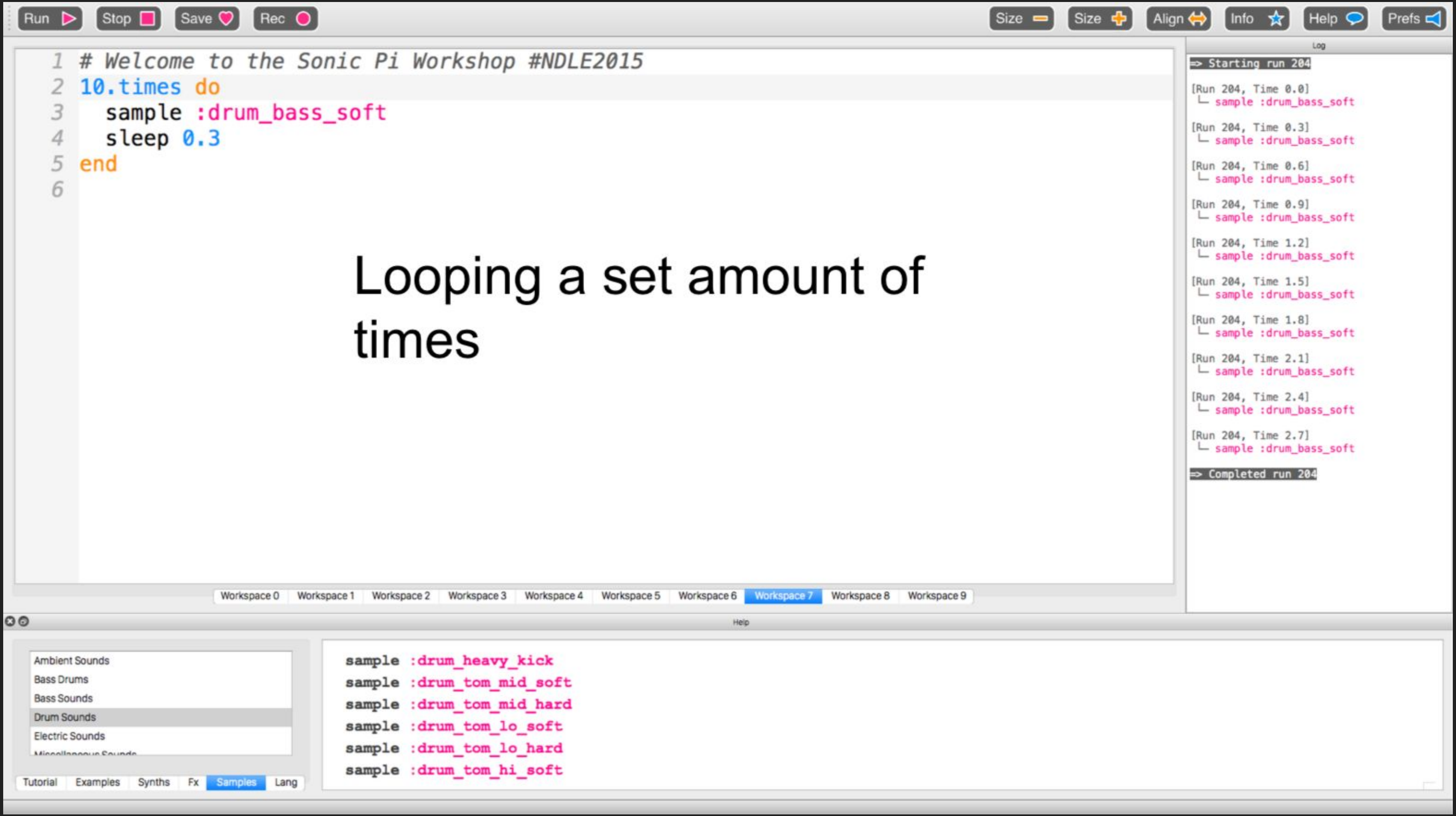
There are lots listed in the sample section, it also shows you how to write them in your code



```
# Welcome to the Sonic Pi Workshop #NDLE2015
loop do
  sample :drum_bass_soft
  sleep 0.3
end
```

Simple looping forever

```
sample :drum_heavy_kick
sample :drum_tom_mid_soft
sample :drum_tom_mid_hard
sample :drum_tom_lo_soft
sample :drum_tom_lo_hard
sample :drum_tom_hi_soft
```



```
# Welcome to the Sonic Pi Workshop #NDLE2015
10.times do
  sample :drum_bass_soft
  sleep 0.3
end
```

Looping a set amount of times

```
=> Starting run 204

[Run 204, Time 0.0]
 └ sample :drum_bass_soft

[Run 204, Time 0.3]
 └ sample :drum_bass_soft

[Run 204, Time 0.6]
 └ sample :drum_bass_soft

[Run 204, Time 0.9]
 └ sample :drum_bass_soft

[Run 204, Time 1.2]
 └ sample :drum_bass_soft

[Run 204, Time 1.5]
 └ sample :drum_bass_soft

[Run 204, Time 1.8]
 └ sample :drum_bass_soft

[Run 204, Time 2.1]
 └ sample :drum_bass_soft

[Run 204, Time 2.4]
 └ sample :drum_bass_soft

[Run 204, Time 2.7]
 └ sample :drum_bass_soft

=> Completed run 204
```

```
sample :drum_heavy_kick
sample :drum_tom_mid_soft
sample :drum_tom_mid_hard
sample :drum_tom_lo_soft
sample :drum_tom_lo_hard
sample :drum_tom_hi_soft
```

Live Loops

```
live_loop :foo do
 play 60
 sleep 1
end
```

Live Loops + synth + sound samples

```
live_loop :foo do
 use_synth :prophet
 play :c1, release: 8,
cutoff: rrand(70, 130)
 sleep 8
end

live_loop :bar do
 sample :bd_haus
 sleep 0.5
end
```

Load external sound samples

```
# Raspberry Pi, Mac, Linux

sample "/Users/sam/Desktop/my-sound.wav", rate: 0.5, amp: 0.3



# Windows

sample "C:/Users/sam/Desktop/my-sound.wav", rate: 0.5, amp: 0.3
```

Live Loops to try

```
live_loop :arp do
 play (scale :e3, :minor_pentatonic).tick, release: 0.1
 sleep 0.125
end
```

List & Arrays

play 52

play 55

play 59

try...

play [52, 55, 59] or play [:E3, :G3, :B3]

Chords

```
play chord(:E3, :minor)
```

Steve Reich: Violin Phase

Related Links

1. **Sonic Pi Essentials Book** -
2. https://www.raspberrypi.org/magpi/issues/essentials-sonic-pi-v1/
3. **TOPLAP** - http://toplap.org
4. **Live Code Slack** - http://live-code-slack.herokuapp.com/
5. **Algorave** - http://algorave.com
6. **Sonic Pi on Github** - https://github.com/samaaron/sonic-pi
7. **Sonic Pi Google Group** - https://groups.google.com/forum/#!forum/sonic-pi
8. **Sonic Pi on Gitter.im** - https://gitter.im/samaaron/sonic-pi