Sonic Pi Workshop

Author: Mihir Gupta

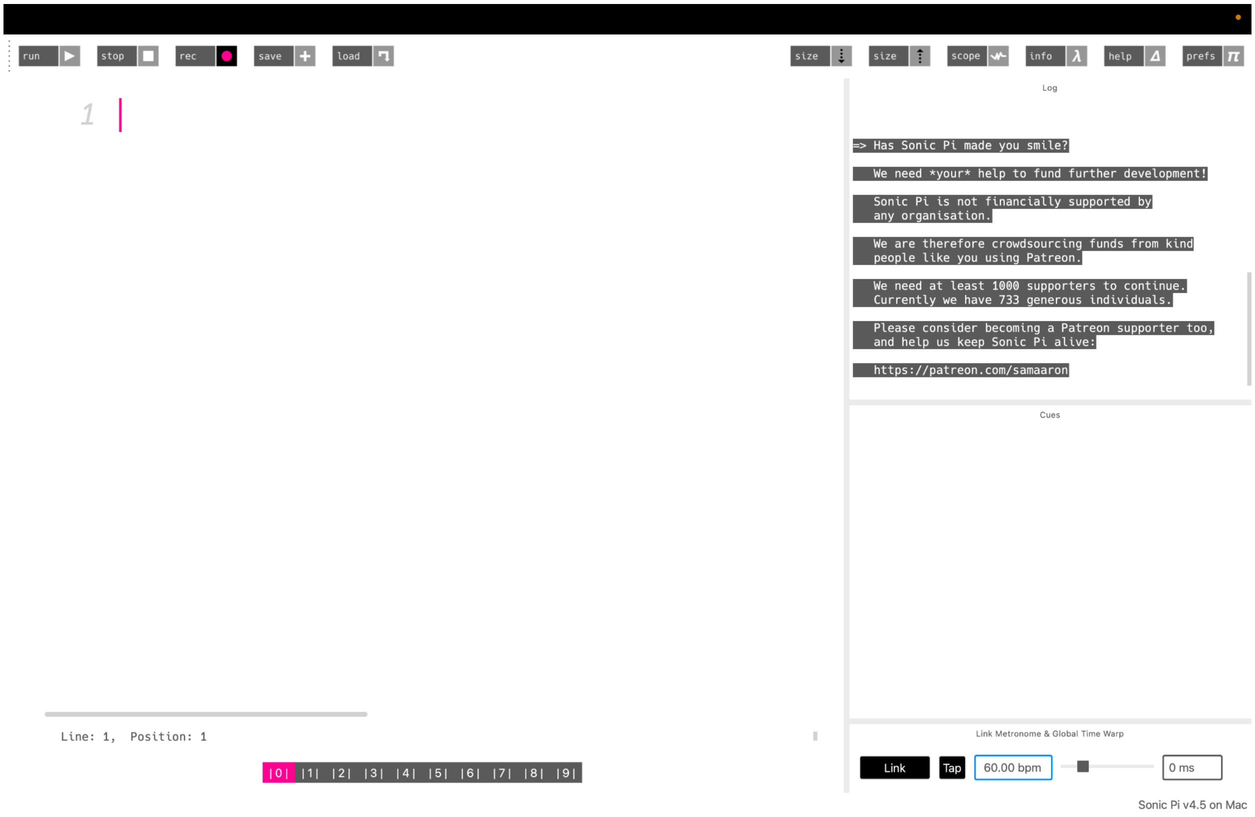
For more information go to phippysonicband.com

Welcome to the Sonic Pi workshop where you will learn how to code in the programming language Ruby and produce your music. Feel free to grab a friend as you go through the workshop! All you need to do this workshop is a computer with Sonic Pi installed.

Step 1: Navigating the UI*

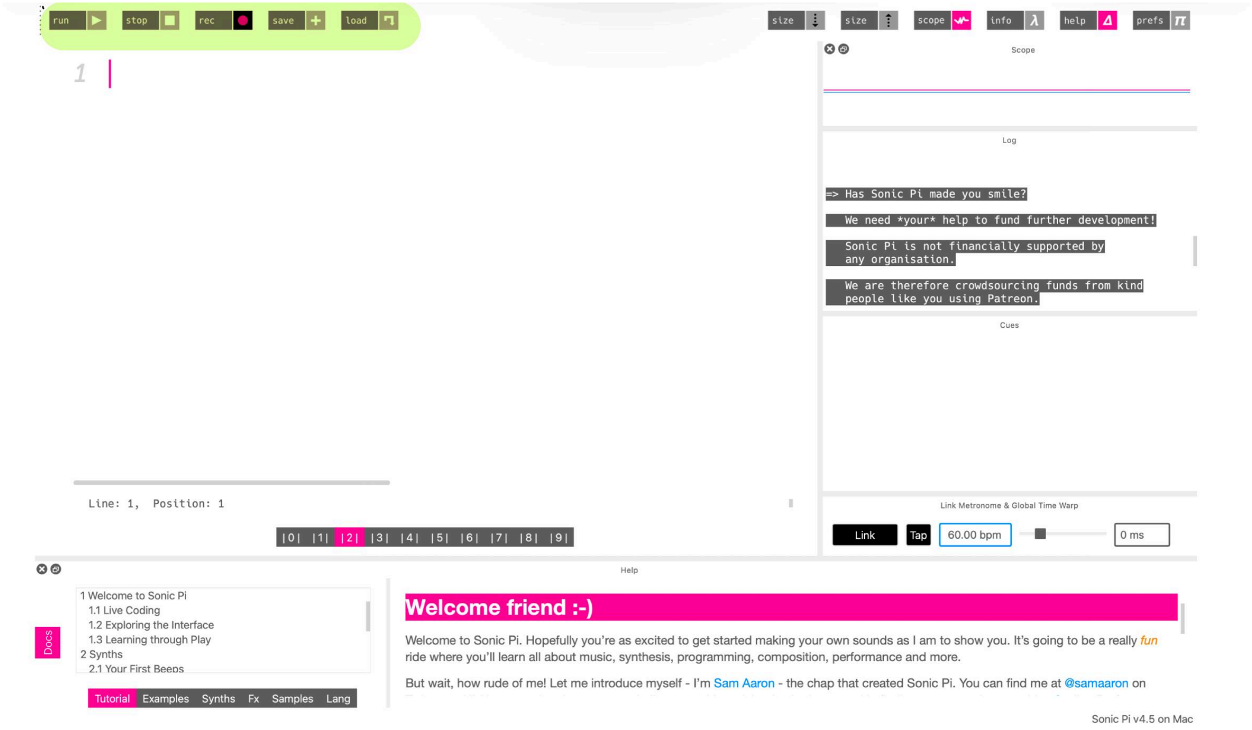
***Before starting, make sure you have downloaded and opened the appropriate version of Sonic Pi!**

Once you have Sonic Pi open, your screen should look like this:



```
=> Has Sonic Pi made you smile?

   We need *your* help to fund further development!

   Sonic Pi is not financially supported by
   any organisation.

   We are therefore crowdsourcing funds from kind
   people like you using Patreon.

   We need at least 1000 supporters to continue.
   Currently we have 733 generous individuals.

   Please consider becoming a Patreon supporter too,
   and help us keep Sonic Pi alive:

   https://patreon.com/samaaron
```

This workshop will use this screenshot throughout. On the top left of your screen, you have your basic controls. Press the run button to play the music you've coded, the stop button to stop, the record button to record, the save button to save your code, and the load to load a previous save.



```
=> Has Sonic Pi made you smile?

   We need *your* help to fund further development!

   Sonic Pi is not financially supported by
   any organisation.

   We are therefore crowdsourcing funds from kind
   people like you using Patreon.
```

```
Line: 1,  Position: 1
```

1 Welcome to Sonic Pi
1.1 Live Coding
1.2 Exploring the Interface
1.3 Learning through Play
2 Synths
2.1 Your First Beeps

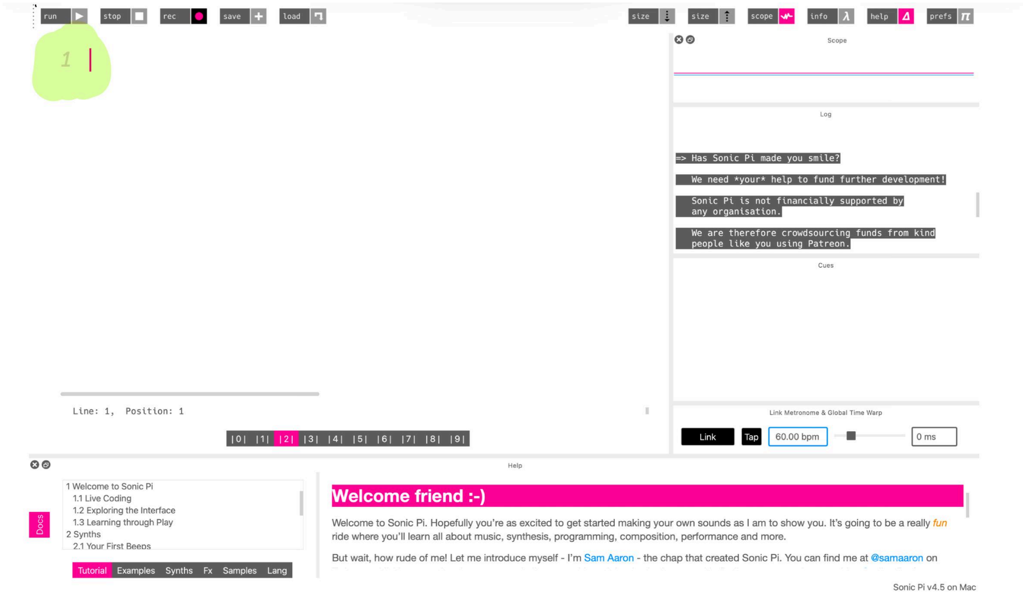
Welcome friend :-)

Welcome to Sonic Pi. Hopefully you're as excited to get started making your own sounds as I am to show you. It's going to be a really *fun* ride where you'll learn all about music, synthesis, programming, composition, performance and more.

But wait, how rude of me! Let me introduce myself - I'm Sam Aaron - the chap that created Sonic Pi. You can find me at @samaaron on

Sonic Pi v4.5 on Mac

The big white box with the number 1 is the terminal window, where you will write all your code.



```
=> Has Sonic Pi made you smile?

   We need *your* help to fund further development!

   Sonic Pi is not financially supported by
   any organisation.

   We are therefore crowdsourcing funds from kind
   people like you using Patreon.
```

```
Line: 1,  Position: 1
```

1 Welcome to Sonic Pi
1.1 Live Coding
1.2 Exploring the Interface
1.3 Learning through Play
2 Synths
2.1 Your First Beeps

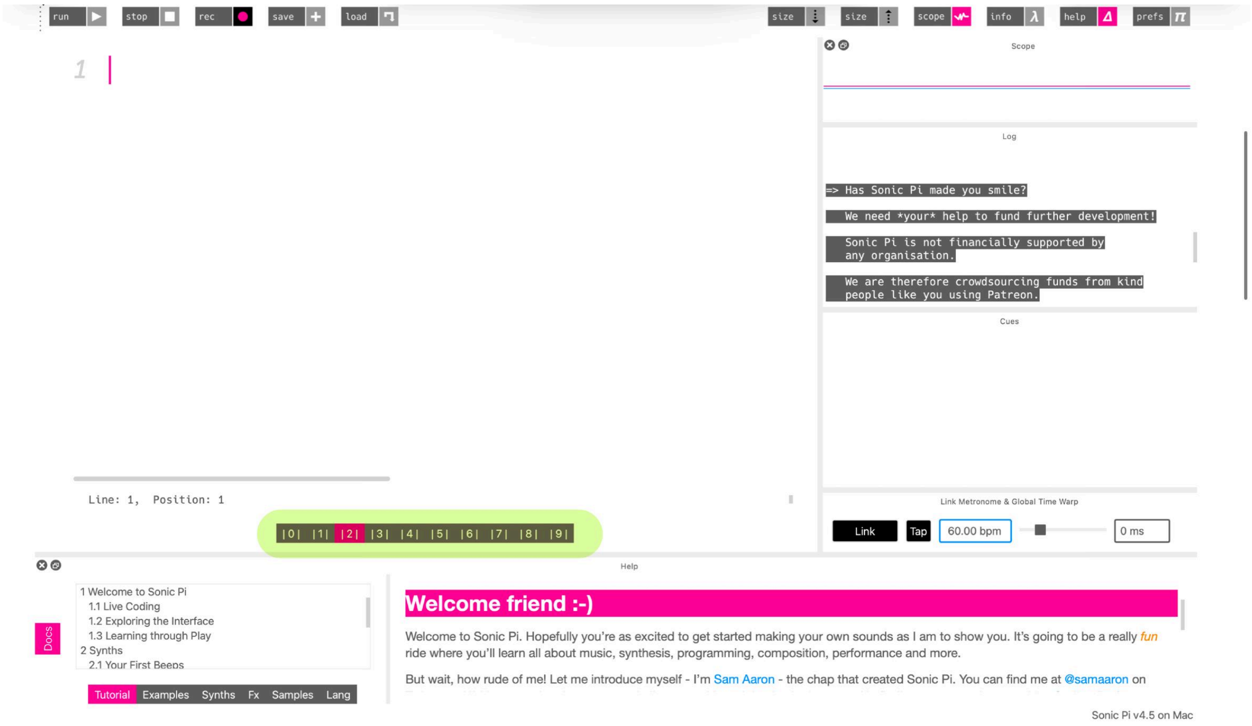
Welcome friend :-)

Welcome to Sonic Pi. Hopefully you're as excited to get started making your own sounds as I am to show you. It's going to be a really *fun* ride where you'll learn all about music, synthesis, programming, composition, performance and more.

But wait, how rude of me! Let me introduce myself - I'm Sam Aaron - the chap that created Sonic Pi. You can find me at @samaaron on

Sonic Pi v4.5 on Mac

Right below the terminal window is a list of numbers. Click on these numbers to switch between different terminal windows, automatically saving your work in its separate window.



```
=> Has Sonic Pi made you smile?

   We need *your* help to fund further development!

   Sonic Pi is not financially supported by
   any organisation.

   We are therefore crowdsourcing funds from kind
   people like you using Patreon.
```

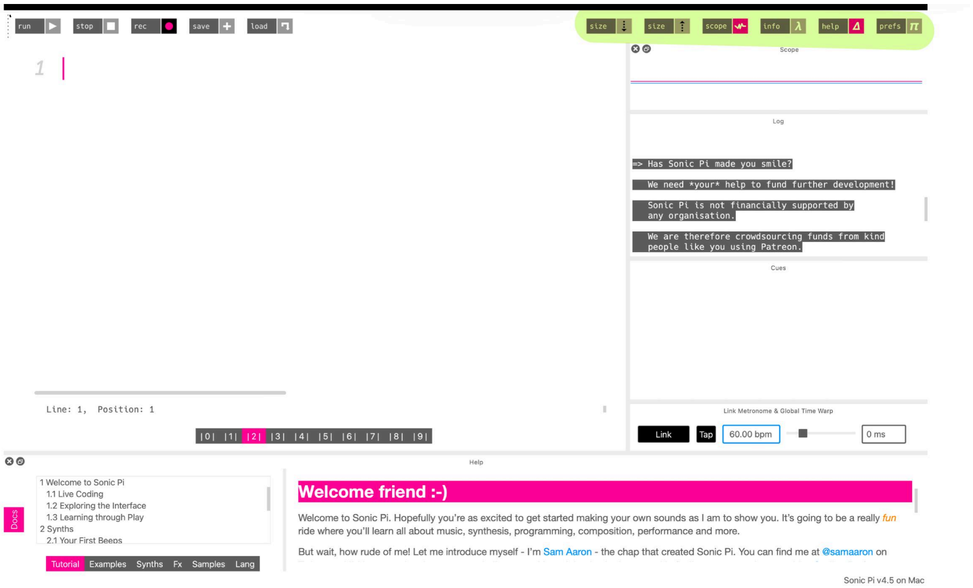
1 Welcome to Sonic Pi
1.1 Live Coding
1.2 Exploring the Interface
1.3 Learning through Play
2 Synths
2.1 Your First Beeps

Welcome friend :-)

Welcome to Sonic Pi. Hopefully you're as excited to get started making your own sounds as I am to show you. It's going to be a really *fun* ride where you'll learn all about music, synthesis, programming, composition, performance and more.

But wait, how rude of me! Let me introduce myself - I'm Sam Aaron - the chap that created Sonic Pi. You can find me at @samaaron on

You have some more controls on the top right across the screen from the basic controls. Here you have the option to zoom in and out in the terminal window with the different size buttons and receive help, among other things.



```
=> Has Sonic Pi made you smile?

   We need *your* help to fund further development!

   Sonic Pi is not financially supported by
   any organisation.

   We are therefore crowdsourcing funds from kind
   people like you using Patreon.
```

1 Welcome to Sonic Pi
1.1 Live Coding
1.2 Exploring the Interface
1.3 Learning through Play
2 Synths
2.1 Your First Beeps

Welcome friend :-)

Welcome to Sonic Pi. Hopefully you're as excited to get started making your own sounds as I am to show you. It's going to be a really *fun* ride where you'll learn all about music, synthesis, programming, composition, performance and more.

But wait, how rude of me! Let me introduce myself - I'm Sam Aaron - the chap that created Sonic Pi. You can find me at @samaaron on

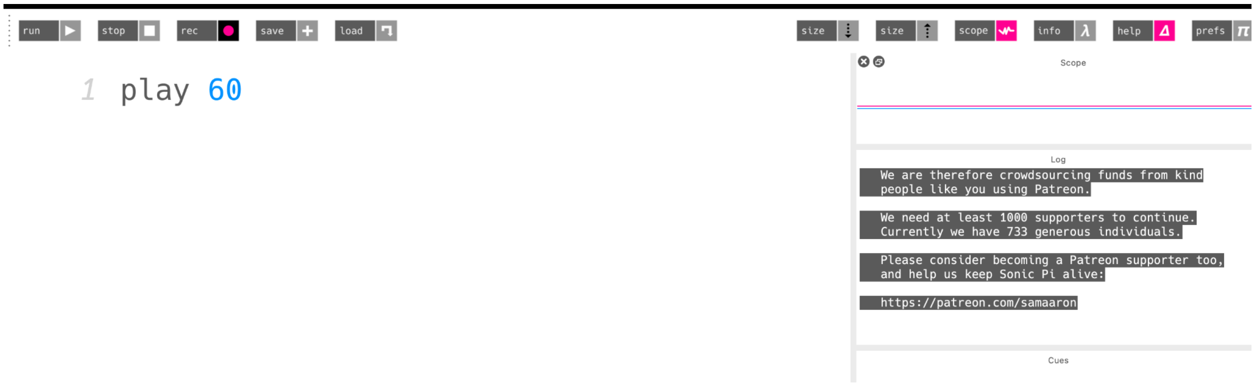
Step 2: Learning the Basics

Plays and Sleeps

Now that you are familiar with how to get around Sonic Pi, let's start writing some code. Begin by writing the following code in the terminal:

```
play 60
```

The SonicPi console will look like this:



```
play 60
```

```
We are therefore crowdsourcing funds from kind
people like you using Patreon.

We need at least 1000 supporters to continue.
Currently we have 733 generous individuals.

Please consider becoming a Patreon supporter too,
and help us keep Sonic Pi alive:

https://patreon.com/samaaron
```

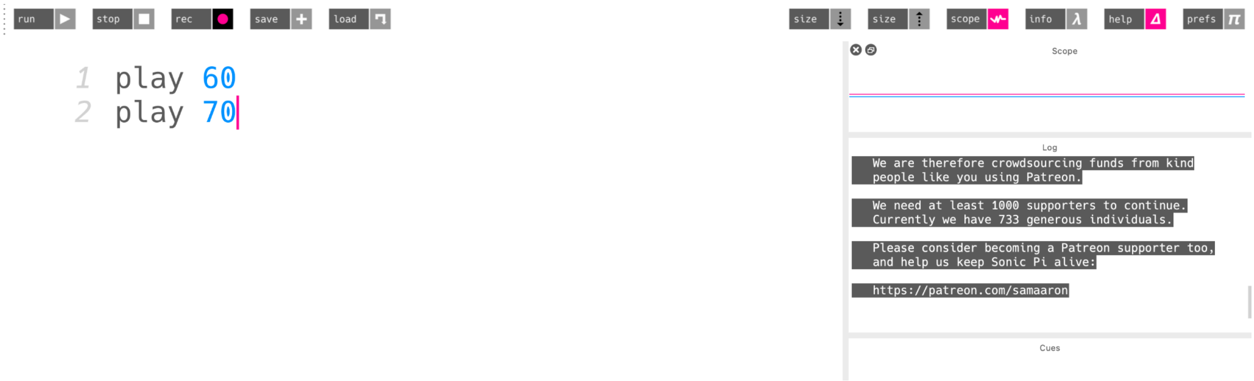
Press run after to find out what it does!

After pressing play, you should've heard a beep. This is because the play is a SonicPi command which plays a sound. But how do you determine which sound? The number after play determines what sound will be played. In this case, we have the number 60 after play, so it will play the sound of the 60th note on the piano because the piano is the default instrument of Sonic Pi. Now try with some different numbers!

After you’ve tried with different values, try putting multiple play commands in a row like so:

```
play 60
play 70
```

Here’s what your screen should look like:



```
play 60
play 70
```

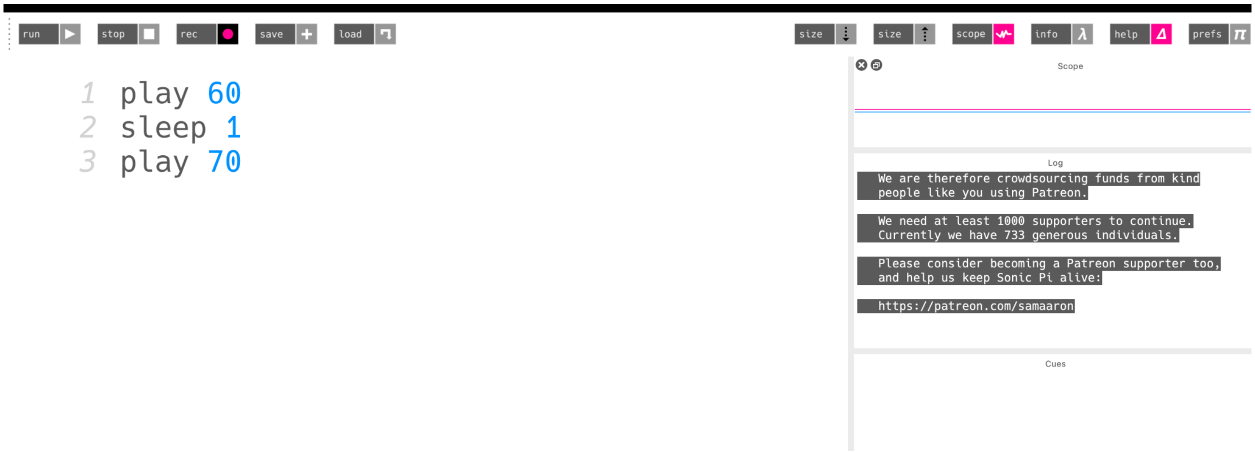
```
We are therefore crowdsourcing funds from kind
people like you using Patreon.

We need at least 1000 supporters to continue.
Currently we have 733 generous individuals.

Please consider becoming a Patreon supporter too,
and help us keep Sonic Pi alive:

https://patreon.com/samaaron
```

After pressing play, you should have heard the 2 notes play simultaneously. But what if you want to play them one after the other? In that case, you need to use a sleep.

Use a sleep by putting it in between the 2 notes, like so:

```
play 60
sleep 1
play 70
```

Here’s what your screen should look like:



```
play 60
sleep 1
play 70
```

```
We are therefore crowdsourcing funds from kind
people like you using Patreon.

We need at least 1000 supporters to continue.
Currently we have 733 generous individuals.

Please consider becoming a Patreon supporter too,
and help us keep Sonic Pi alive:

https://patreon.com/samaaron
```

Essentially, the sleep command tells the computer to sleep for 1 second, which results in a period of 1 second where no sound is being played. In music theory, we call this a rest. Of course, you can change the number to shorten the rest period.

Number and Letter notes

In music, we don't distinguish notes based on numbers, like we did earlier. We use letters, and they range from A to G. So, we can establish some equivalent values to some numbers that you might've used earlier. Here is the full list:

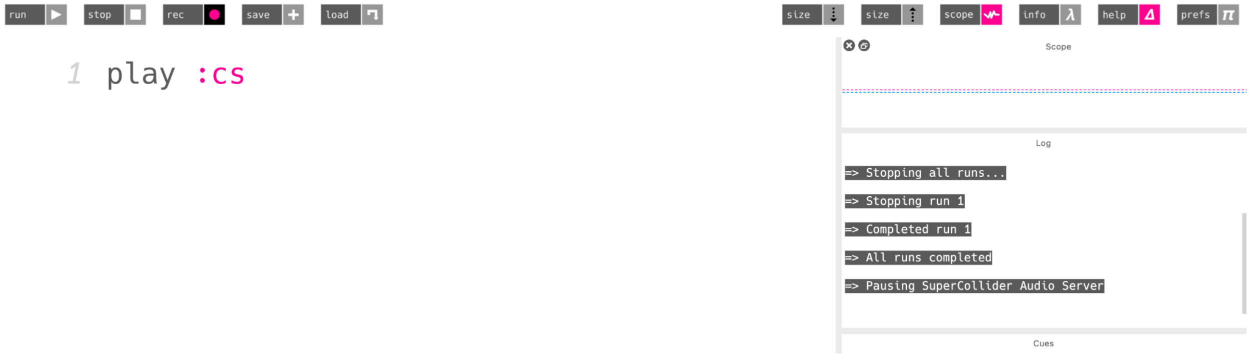
play :a = play 69
play :b = play 71
play :d = play 62
play :e = play 64
Play :f = play 65
play :g = play 67

There are also concepts known as sharps (s) or flats (b). In music, sharps and flats are used to increase and decrease the pitch of a note, but not to the point where it becomes a new note. For example, A sharp would be a higher-pitched A note, but not high enough to the point where it is the same pitch as B.

You can write this in Sonic Pi like so:

```
play :cs
```

Here's what your screen should look like:



```
play :cs
```

```
=> Stopping all runs...
=> Stopping run 1
=> Completed run 1
=> All runs completed
=> Pausing SuperCollider Audio Server
```

You should've heard the note C sharp. This only works with notes, so don't try doing this:

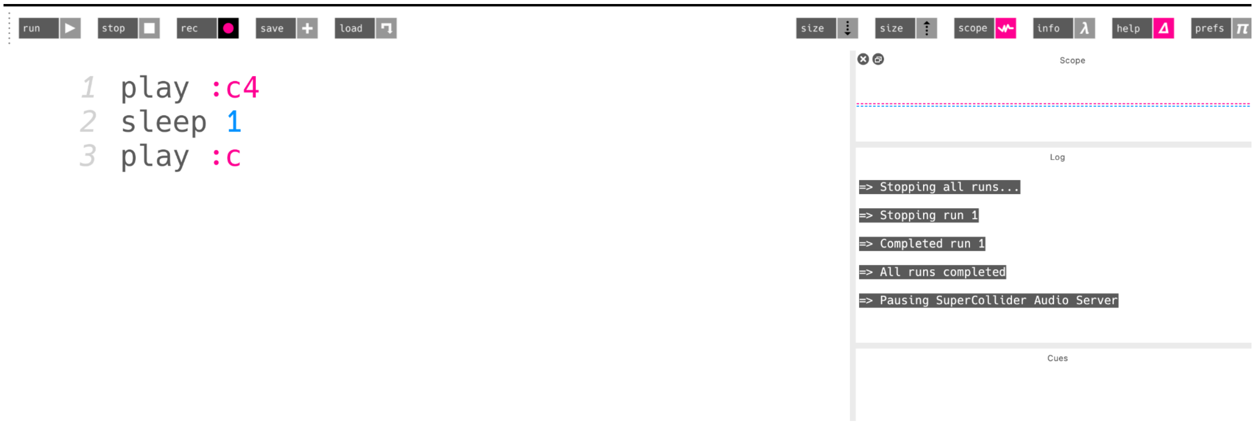
```
play 60b
```

Octaves

Now that you're comfortable playing different notes, let's explore how to change the octave. An octave is a set of eight notes, and in Sonic Pi, we can change the octave by adjusting the number after the play command, assuming that you are using the command for a letter note rather than a key on the piano. Additionally, in Sonic Pi, whenever you use the play command for a letter note, it automatically starts in the 4th octave. This means that if you were to compare play :c with play :c4, they would sound the same, while play :c3 would be a lower pitch.

Try with an example. Start with this in your terminal window:

```
play :c4
sleep 1
play :c
```



```
play :c4
sleep 1
play :c
```

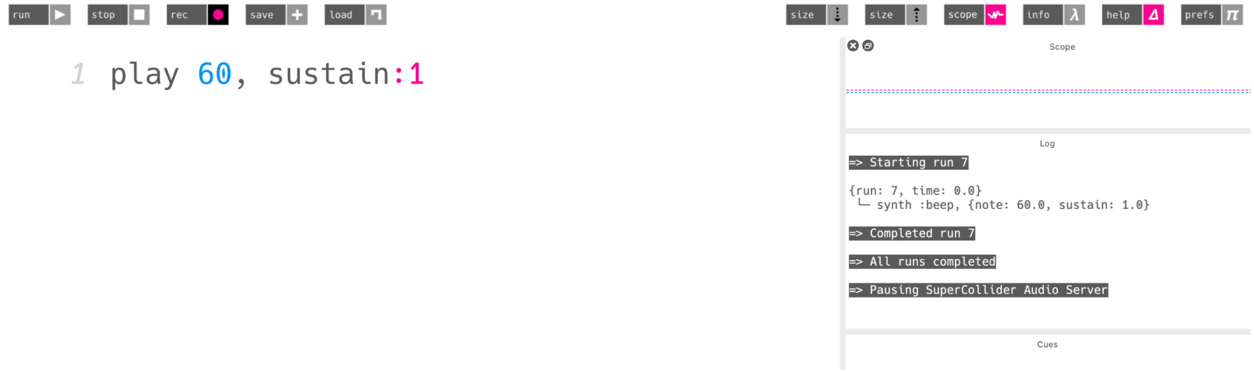
```
=> Stopping all runs...
=> Stopping run 1
=> Completed run 1
=> All runs completed
=> Pausing SuperCollider Audio Server
```

Notice how the notes are the same? Now try changing the number after c in the first line to see how the sound changes.

Sustaining Notes

Additionally, we can change how long the note is played. This can be done by adding the command sustain after the play command, with the number of seconds you want the note to be held. Here is an example:

```
play 60, sustain:1
```



```
play 60, sustain:1
```

```
=> Starting run 7

{run: 7, time: 0.0}
 └ synth :beep, {note: 60.0, sustain: 1.0}

=> Completed run 7
=> All runs completed
=> Pausing SuperCollider Audio Server
```

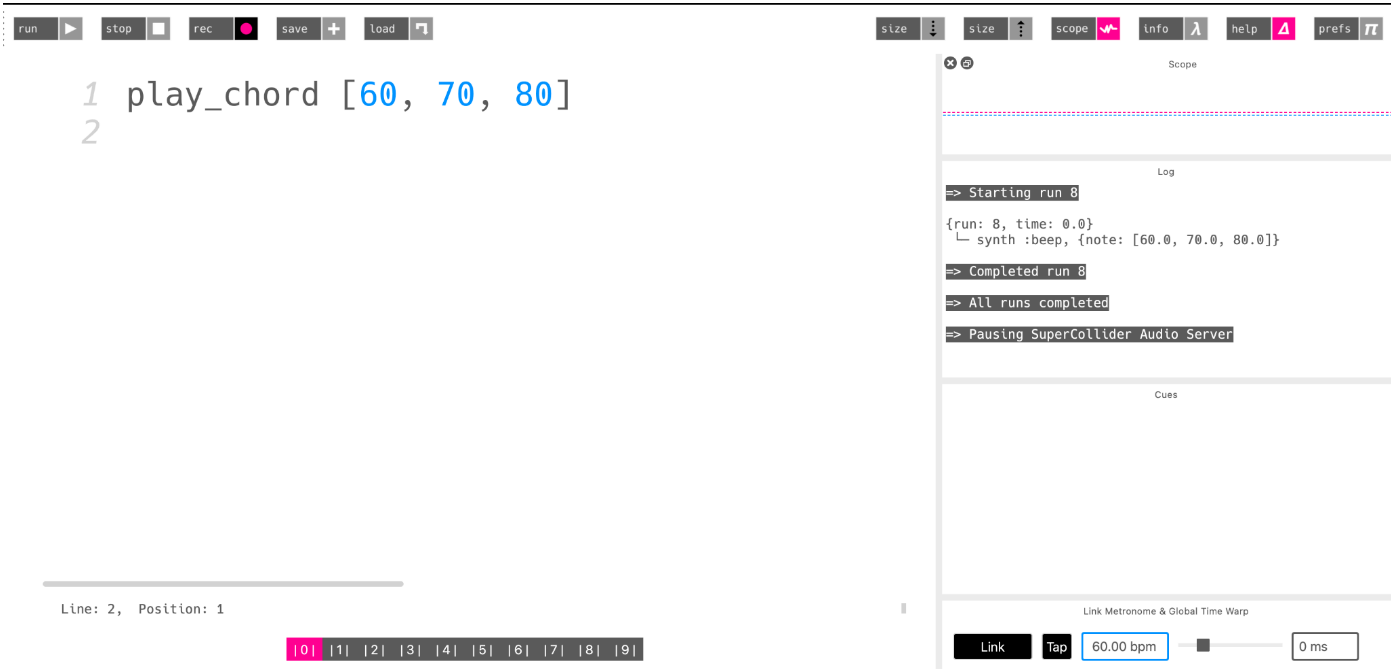
If you want to change how long the note is played for, simply change the number after sustain.

Play_chord and play_pattern

Now that we know how to play different kinds of notes, let's learn the Sonic Pi command for how to play them while saving some lines. If you want to play notes at the same time, you must use the play_chord command, like so:

Copy and paste this code into Sonic Pi!

```
play_chord [60, 70, 80]
```



```
play_chord [60, 70, 80]
```

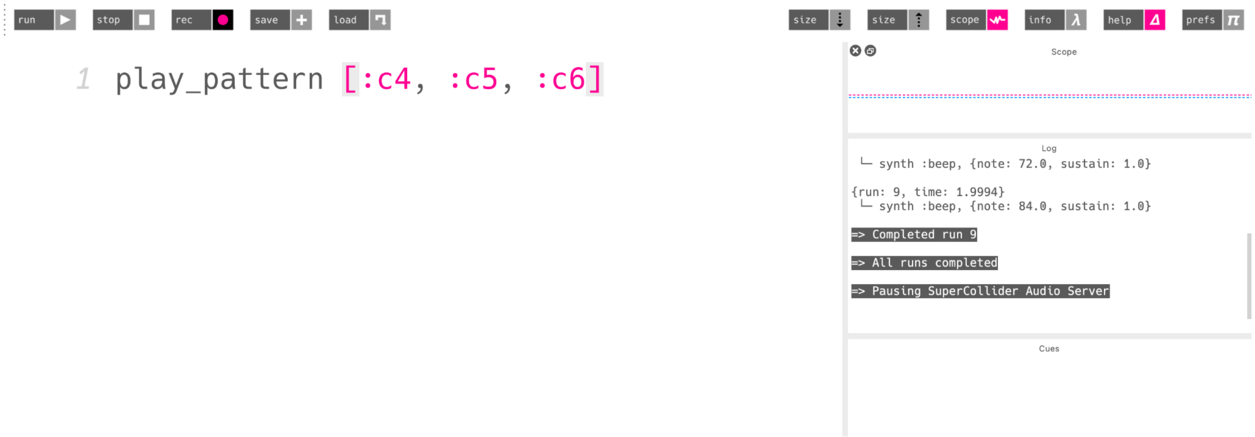
```
Line: 2,  Position: 1
```

```
=> Starting run 8

{run: 8, time: 0.0}
 └ synth :beep, {note: [60.0, 70.0, 80.0]}

=> Completed run 8

=> All runs completed

=> Pausing SuperCollider Audio Server
```

If you want to play these notes in a sequence, use the play_pattern command, like so:

```
play_pattern  [:c4, :c5, :c6]
```

Next, copy and paste it into Sonic Pi:



```
play_pattern [:c4, :c5, :c6]
```

```
 └─ synth :beep, {note: 72.0, sustain: 1.0}

{run: 9, time: 1.9994}
 └─ synth :beep, {note: 84.0, sustain: 1.0}

=> Completed run 9

=> All runs completed

=> Pausing SuperCollider Audio Server
```

Do you feel you’re picking up skills to be part of The Cloud Hoppers band? Lets learn some more to save the concert.

Step 3: Advanced Techniques

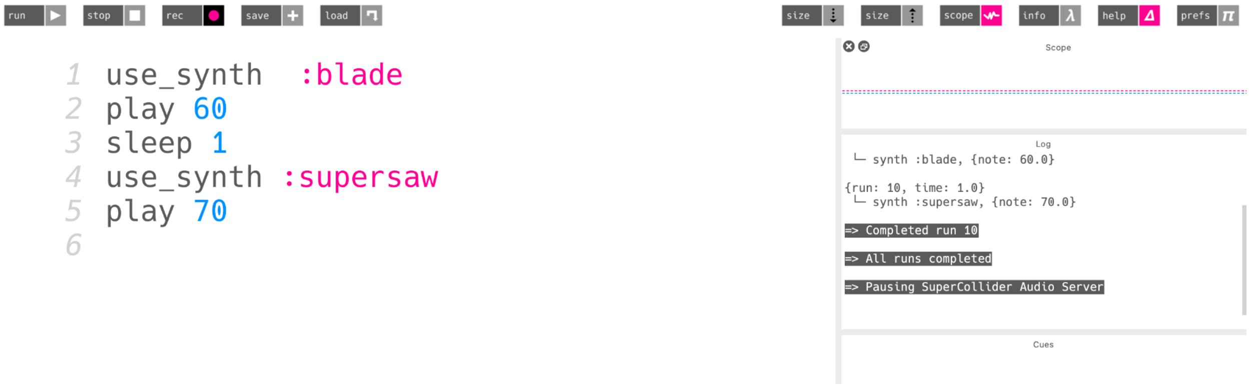
Now that we know the basics of Sonic Pi, let's move on to more advanced techniques. A music concert has many different instruments. Lets learn how to change instruments using Sonic Pi.

Using Synthesizers

By adding the use_synth command, we can change the instrument used to play the sound. Make sure to be aware of where you add the use_synth command, as it will modify all of the code in the proceeding lines unless another use_synth command is used after it. Here is an example:

```
use_synth :blade
play 60
sleep 1
use_synth :supersaw
play 70
```

Copy and paste this code into Sonic Pi!

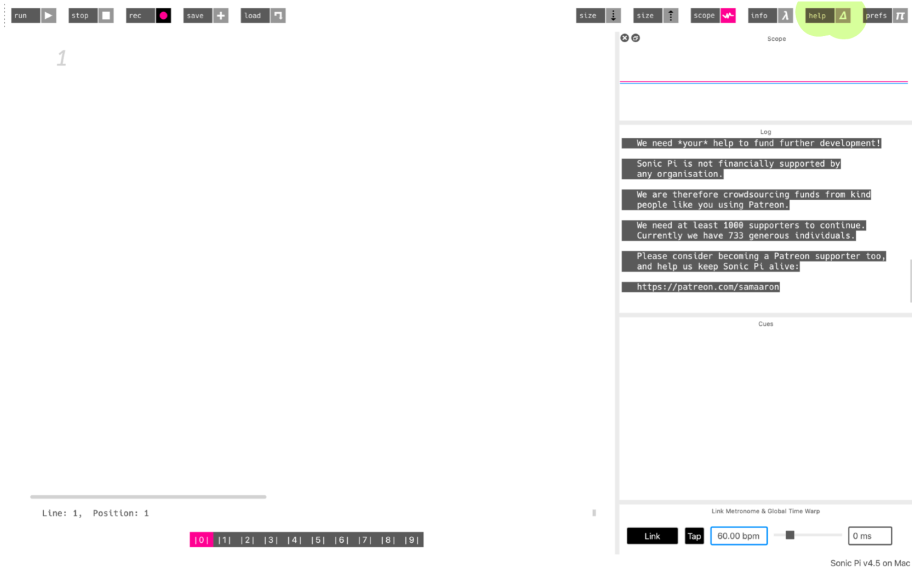


```
use_synth  :blade
play 60
sleep 1
use_synth :supersaw
play 70
```

```
 └─ synth :blade, {note: 60.0}

{run: 10, time: 1.0}
 └─ synth :supersaw, {note: 70.0}

=> Completed run 10

=> All runs completed

=> Pausing SuperCollider Audio Server
```

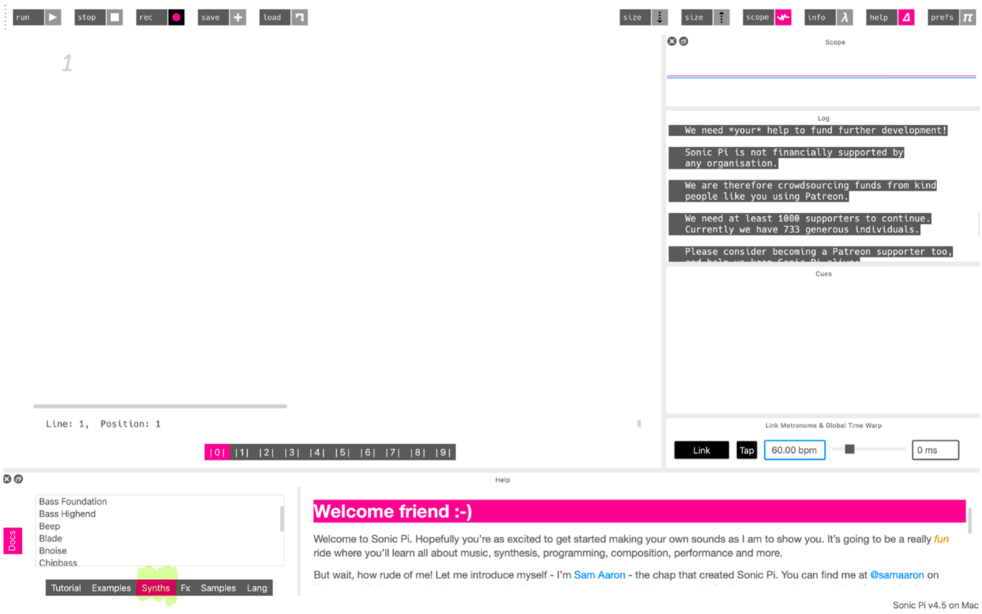
You may be wondering, “What synthesizers can I use?” To access the full Sonic Pi library of synthesizers, follow the guide below:

1. First, click the help button on the top right of your screen.



```
We need *your* help to fund further development!

Sonic Pi is not financially supported by
any organisation.

We are therefore crowdsourcing funds from kind
people like you using Patreon.

We need at least 1000 supporters to continue.
Currently we have 733 generous individuals.

Please consider becoming a Patreon supporter too,
and help us keep Sonic Pi alive:

https://patreon.com/samaaron
```

2. Then, click the Synths button at the bottom of the screen to access Sonic Pi’s full library of synthesizers. If you click on one, it will show you the proper spelling and the way to type it.



```
We need *your* help to fund further development!

Sonic Pi is not financially supported by
any organisation.

We are therefore crowdsourcing funds from kind
people like you using Patreon.

We need at least 1000 supporters to continue.
Currently we have 733 generous individuals.

Please consider becoming a Patreon supporter too,
and help us keep Sonic Pi alive!
```

Welcome friend :-)

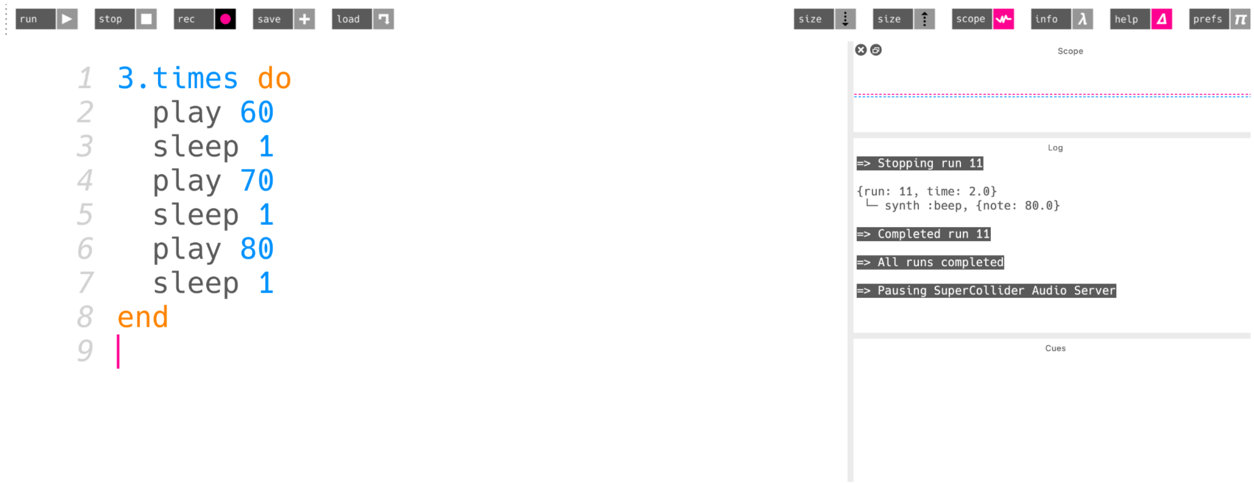
Welcome to Sonic Pi. Hopefully you're as excited to get started making your own sounds as I am to show you. It's going to be a really *fun* ride where you'll learn all about music, synthesis, programming, composition, performance and more.

But wait, how rude of me! Let me introduce myself - I'm Sam Aaron - the chap that created Sonic Pi. You can find me at @samaaron on

Loops

Now that you've learned various ways to program in Sonic Pi, your code might start to grow longer. But what would you do if you wanted to repeat some of your code? Would you just copy and paste it? While you can do that, it would be very inefficient. Also, you'll have duplicate code all over the place. Instead, you can use various functions that are more effective for repeating code instructions. In this case, we can use loops. In Ruby, we need to specify the number of times we want to loop our code, and which code we want to loop. Here is an example:

```
3.times do
      play 60
      sleep 1
      play 70
      sleep 1
      play 80
      sleep 1
end
```



```
3.times do
  play 60
  sleep 1
  play 70
  sleep 1
  play 80
  sleep 1
end
```

```
=> Stopping run 11

{run: 11, time: 2.0}
 └─ synth :beep, {note: 80.0}

=> Completed run 11

=> All runs completed

=> Pausing SuperCollider Audio Server
```

Notice that on the top, the code has the number 3. This specifies the number of times the code needs to be looped. The indented code is the code that needs to be looped and concludes with an end statement.

Exercise:

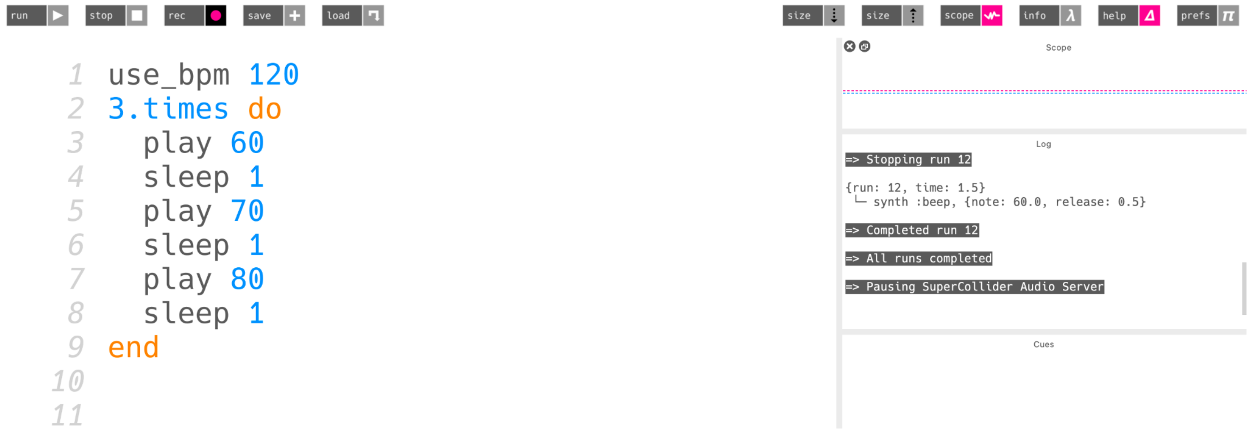
1. Change the number of times the code is looped.
2. Use the code learned so far to change the code that is looped

Speeding Up and Slowing Down

One thing that distinguishes various types of music is the speed at which it is played. Classical music might be played slowly, while pop music might be played at a much faster pace. Let's learn how to incorporate that into our code.

To speed up or slow down your code, use the use_bpm command like so:

```
use_bpm 120
3.times do
      play 60
      sleep 1
      play 70
      sleep 1
      play 80
      sleep 1
end
```



```
use_bpm 120
3.times do
  play 60
  sleep 1
  play 70
  sleep 1
  play 80
  sleep 1
end
```

```
=> Stopping run 12

{run: 12, time: 1.5}
 └─ synth :beep, {note: 60.0, release: 0.5}

=> Completed run 12

=> All runs completed

=> Pausing SuperCollider Audio Server
```

Note that the number after use_bpm in the first line specifies the bpm or the beats per minute. The higher the number, the faster your music becomes.

Can you see how this technique will be useful when you try to save The Cloud Hoppers concert?

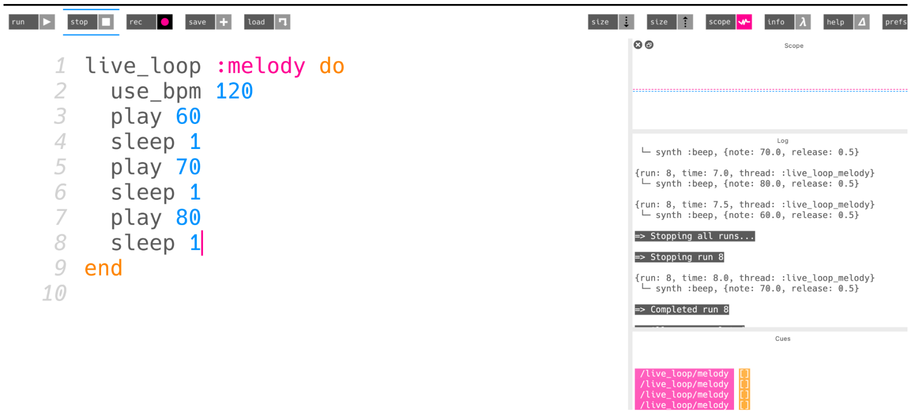
Live Loops

In Sonic Pi, we can use live loops to put our code in an infinite loop. Why would you do this you ask? The reason behind this is so we can hear the music that we have coded playing and think of improvements. If you want to change anything in the music while it is playing, simply just make your edits and press play again! You don’t need to press stop.

Here is an example of using live loops:

```
live_loop :melody do
use_bpm 120
3.times do
      play 60
      sleep 1
      play 70
      sleep 1
      play 80
      sleep 1
end
```

Then, copy and paste it into Sonic Pi:



```
 1 live_loop :melody do
 2   use_bpm 120
 3   play 60
 4   sleep 1
 5   play 70
 6   sleep 1
 7   play 80
 8   sleep 1
 9 end
10
```

```
 └─ synth :beep, {note: 70.0, release: 0.5}

{run: 8, time: 7.0, thread: :live_loop_melody}
 └─ synth :beep, {note: 80.0, release: 0.5}

{run: 8, time: 7.5, thread: :live_loop_melody}
 └─ synth :beep, {note: 60.0, release: 0.5}

=> Stopping all runs...

=> Stopping run 8

{run: 8, time: 8.0, thread: :live_loop_melody}
 └─ synth :beep, {note: 70.0, release: 0.5}

=> Completed run 8
```

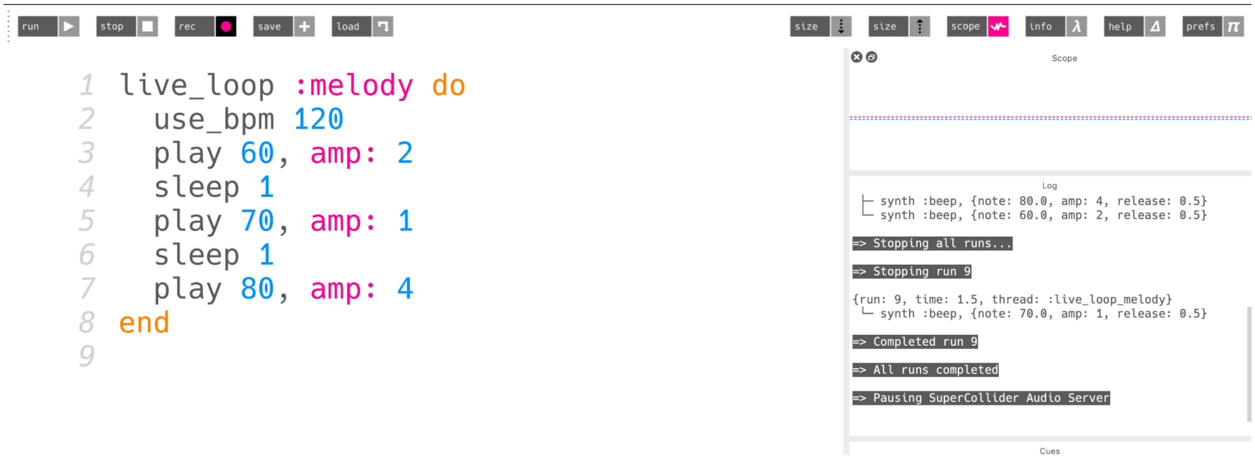
```
/live_loop/melody  []
/live_loop/melody  []
/live_loop/melody  []
/live_loop/melody  []
```

The live loop is like a normal loop where you still need a beginning and an end. However, the syntax is a bit different. In line 1, the parts live_loop and do are required. Each live loop has a name and our name is melody. You can change it to whatever you want.

Changing the Volume

In Sonic Pi, we can also change the volume of individual notes. This can be done by using the amp: command. Here is an example:

```
live_loop :melody do
      use_bpm 120
      play 60, amp: 2
      sleep 1
      play 70, amp: 1
      sleep 1
      play 80, amp: 4
end
```



```
live_loop :melody do
  use_bpm 120
  play 60, amp: 2
  sleep 1
  play 70, amp: 1
  sleep 1
  play 80, amp: 4
end
```

```
 ├─ synth :beep, {note: 80.0, amp: 4, release: 0.5}
 └─ synth :beep, {note: 60.0, amp: 2, release: 0.5}

=> Stopping all runs...

=> Stopping run 9

{run: 9, time: 1.5, thread: :live_loop_melody}
 └─ synth :beep, {note: 70.0, amp: 1, release: 0.5}

=> Completed run 9

=> All runs completed

=> Pausing SuperCollider Audio Server
```

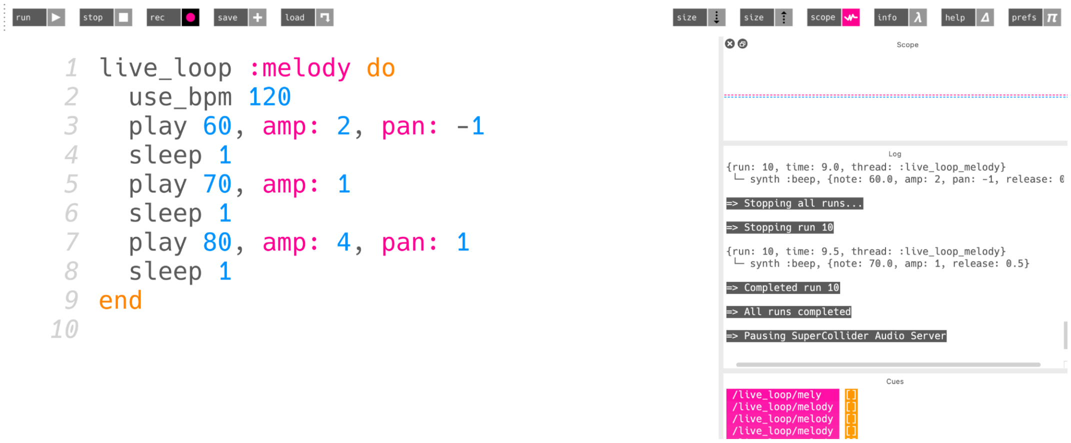
Just like sustains, changing the amplitude of a note requires putting a comma after the note, command amp:, and a number that indicates volume of the note.

Panning

Sonic Pi has a unique feature that allows you to control which of your computer's speakers outputs the audio that you create. This is called panning. You can do this by adding the command pan: after the play command. Once you add pan, use -1 to make the audio come from the left speaker or 1 to make the audio come from the right speaker. If you want the audio to come from both speakers, there is no need for the pan command.

Here is an example:

```
live_loop :melody do
      use_bpm 120
      play 60, amp: 2, pan: -1
      sleep 1
      play 70, amp: 1
      sleep 1
      play 80, amp: 4, pan: 1
end
```



```
live_loop :melody do
  use_bpm 120
  play 60, amp: 2, pan: -1
  sleep 1
  play 70, amp: 1
  sleep 1
  play 80, amp: 4, pan: 1
  sleep 1
end
```

```
{run: 10, time: 9.0, thread: :live_loop_melody}
 └─ synth :beep, {note: 60.0, amp: 2, pan: -1, release: 0

=> Stopping all runs...

=> Stopping run 10

{run: 10, time: 9.5, thread: :live_loop_melody}
 └─ synth :beep, {note: 70.0, amp: 1, release: 0.5}

=> Completed run 10

=> All runs completed

=> Pausing SuperCollider Audio Server
```

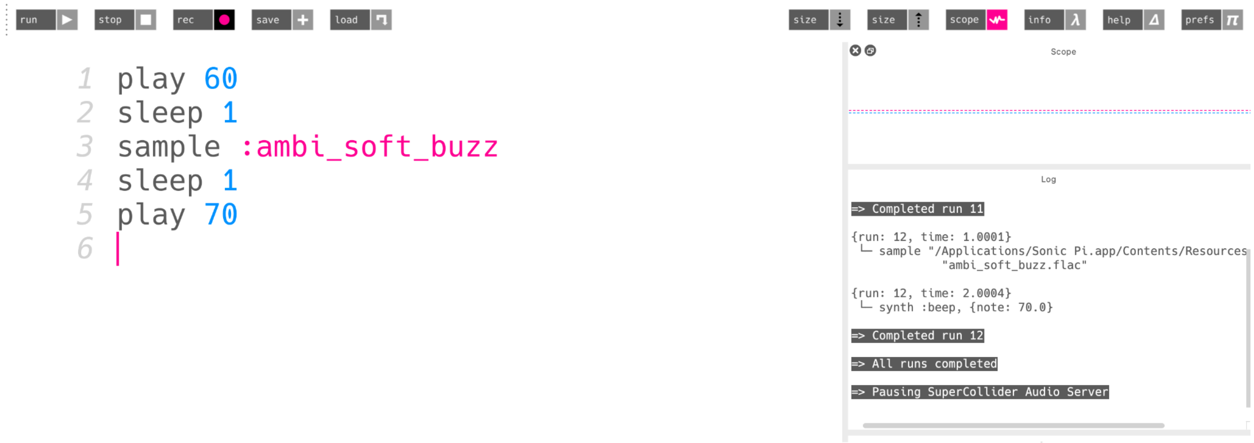
```
/live_loop/mely     []
/live_loop/melody   []
/live_loop/melody   []
/live_loop/melody   []
```

Samples

Like synthesizers, Sonic Pi has a library of samples available for access. Samples are essential parts of other sounds or music recorded for you to use in your music. You can use these samples with the other skills we have learned to cool combinations.

Here is an example of how to use samples in your code:

```
play 60
sleep 1
sample :ambi_soft_buzz
sleep 1
play 70
```

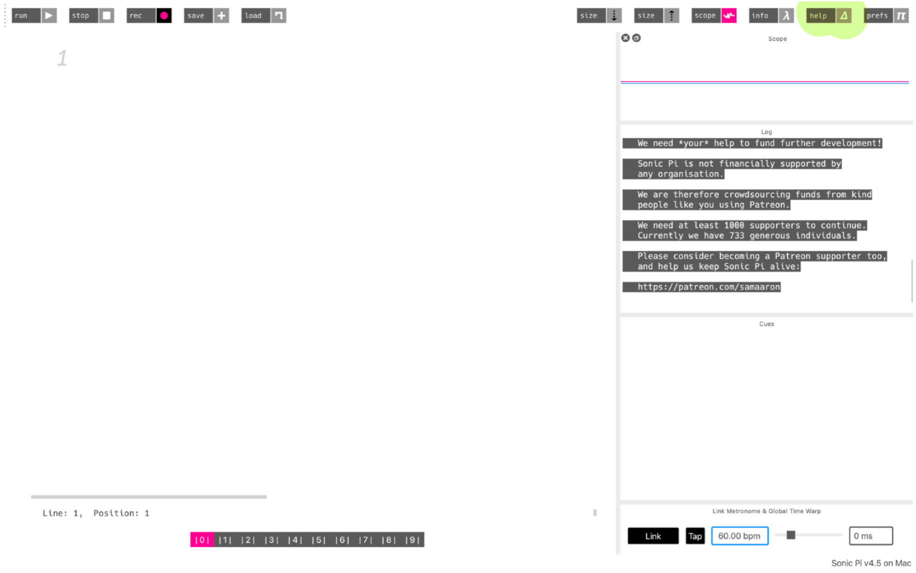


```
play 60
sleep 1
sample :ambi_soft_buzz
sleep 1
play 70
```

```
=> Completed run 11

{run: 12, time: 1.0001}
 └─ sample "/Applications/Sonic Pi.app/Contents/Resources
            "ambi_soft_buzz.flac"

{run: 12, time: 2.0004}
 └─ synth :beep, {note: 70.0}

=> Completed run 12

=> All runs completed

=> Pausing SuperCollider Audio Server
```

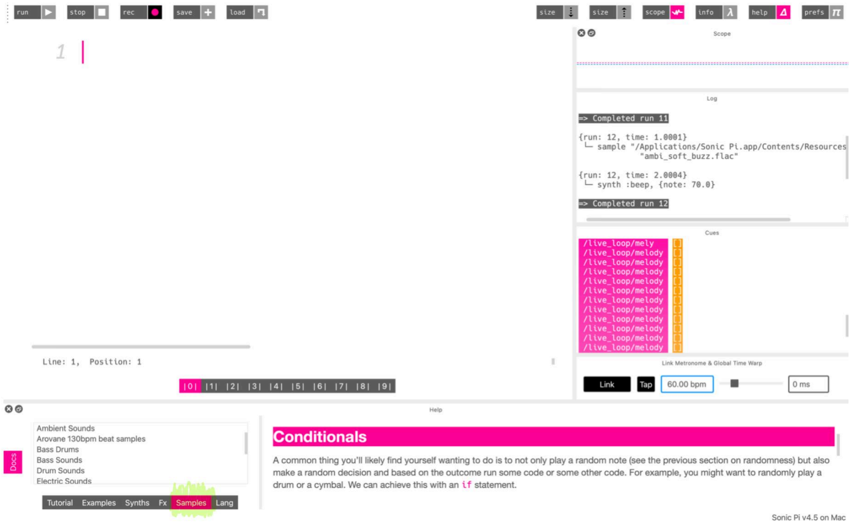
To access the full Sonic Pi library of samples follow the guide below:

Click the help button on the top right of your screen if you haven't already.



```
We need *your* help to fund further development!

Sonic Pi is not financially supported by
any organisation.

We are therefore crowdsourcing funds from kind
people like you using Patreon.

We need at least 1000 supporters to continue.
Currently we have 733 generous individuals.

Please consider becoming a Patreon supporter too,
and help us keep Sonic Pi alive:

https://patreon.com/samaaron
```

Click the Samples button on the bottom left and try out a few samples.



```
=> Completed run 11

{run: 12, time: 1.0001}
 └─ sample "/Applications/Sonic Pi.app/Contents/Resources
             "ambi_soft_buzz.flac"

{run: 12, time: 2.0004}
 └─ synth :beep, {note: 70.0}

=> Completed run 12
```

```
/live_loop/mely      []
/live_loop/melody    []
/live_loop/melody    []
/live_loop/melody    []
/live_loop/melody    []
/live_loop/melody    []
/live_loop/melody    []
/live_loop/melody    []
/live_loop/melody    []
/live_loop/melody    []
/live_loop/melody    []
/live_loop/melody    []
```

60.00 bpm

0 ms

```
Line: 1,  Position: 1
```

|0| |1| |2| |3| |4| |5| |6| |7| |8| |9|

Ambient Sounds
Arovane 130bpm beat samples
Bass Drums
Bass Sounds
Drum Sounds
Electric Sounds

Conditionals

A common thing you'll likely find yourself wanting to do is to not only play a random note (see the previous section on randomness) but also make a random decision and based on the outcome run some code or some other code. For example, you might want to randomly play a drum or a cymbal. We can achieve this with an `if` statement.

Congratulations! Now that you know how to use Sonic Pi, compose something that will help Phippy and Friends save the Cloud Hoppers' concert! Once you have done so, go back to phippysonicband.com for the next steps!