Putting it all together!

In this last guided activity, we're going to put together everything that we've done so far by making the beat and melody scripts that you've created happen at the same time. To do this, we're going to need to use functions!

1. To start, open up your beat script, your synth script, and then create a blank new file in which to make a combined version. To this new file, add a few lines that:

   - Import everything from the scamp libraries
   - Create a new `Session` object and assign it to the variable `s`. Note: this time, there is no need to use the `default_soundfont` keyword argument. This is because we will be using sounds from both of the soundfonts we've been using, so we will specify the soundfont we want to use every time we create a new instrument.
   - Set the tempo of `s` to something reasonable (probably the same as what you set it to in the beats activity)

   Help

2. Open up your beat script, and copy all of the lines in which you create new instruments into your new script. Since we're not specifying a `default_soundfont` for the session, we now need to add a new keyword argument to each of the `new_part` function calls specifying `soundfont="Emu_Planet_Phatt_Hip_Hop.sf2"`.

   The idea here is this: all of the instruments we create belong to the session `s`. If we set the `default_soundfont` for the session, then any instrument we create will use that soundfont. Alternatively, when we create an instrument, we can use the `soundfont` keyword argument at that time to tell the computer which soundfont to use. Help

3. Next, open up your synth script, and copy all of the lines in which you create new instruments into your new script. (There may be only one.) For these instruments, set the value of the `soundfont` keyword argument to "Synths.sf2". Help

4. Now, define a new function, and place the main body of the beat script inside of it (making sure it's all indented one degree further). Add a line after the function definition that calls the function. You should be able to hear the beat play back. Help, More Help

5. Do the same for the synth part. After the beat music function, create another new function and copy the main body of your synth script into it. Call that function at the end of your script instead of the beat function. You should hear the synth part play.

   Now what happens if you call the beat function and then the synth function? The synth function and then the beat function? What's going on here? Help

6. Now, use the `fork` function to make one of the parts play on a parallel time stream. This way you can hear both at the same time! Help, More Help

7. From here, you might spend some time adjusting the parts so that they fit together in a more interesting way, or perhaps expanding them. Once you're happy with the result, insert the line `playback_settings.recording_file_path = "alltogether.wav"` to make a recording!

Help 1

All of these lines together should look something like this:

```
from scamp import *

s = Session()
s.tempo = 90
```

If you want, you can actually do this instead:

```
from scamp import *

s = Session(tempo=90)
```

When you use the keyword argument `tempo=90`, the Session's tempo attribute gets automatically set at its moment of creation.

Back to instructions

Help 2

We can specify that we want to use the beats soundfont like this:

```
drum_kit = s.new_part("Kit 1", soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
```

So in my case, the code at this point looks like this:

```
from scamp import *

s = Session()
s.tempo = 90

drum_kit = s.new_part("Kit 1", soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
record_scratch = s.new_part("Scratch 1", soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
```

Back to instructions

Help 3

At this point, my code looks like this:

```
from scamp import *

s = Session()
s.tempo = 90

drum_kit = s.new_part("Kit 1", soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
record_scratch = s.new_part("Scratch 1", soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
synth = s.new_part("CandyBee", soundfont="Synths.sf2")
```

Back to instructions

Help 4a

Remember that you can define a new function by writing:

```
def my_function():
    # write whatever you want the function to do here
```

To then call the function, you would write:

```
my_function()
```

Can you give the function an appropriate name, and copy the beat part in?

Back to instructions

Help 4b

In my case, after all of the setup code and instrument creation, my original beat script looked like this:

```
while True:
    count = 0
    while count < 3:
        record_scratch.play_note(53, 1.0, 1.0, blocking=False)
        drum_kit.play_note(60, 0.4, 0.5)
        wait(0.25)
        drum_kit.play_note(60, 0.4, 0.25)
        drum_kit.play_note(63, 0.6, 0.25)
        drum_kit.play_note(56, 0.4, 0.5)
        record_scratch.play_note(62, 0.6, 0.5)
        drum_kit.play_note(61, 1.0, 0.25, blocking=False)
        drum_kit.play_note(63, 0.6, 0.25)
        drum_kit.play_note(63, 0.6, 0.25)
        drum_kit.play_note(59, 1.0, 0.25, blocking=False)
        drum_kit.play_note(56, 0.8, 0.25)
        drum_kit.play_note(61, 1.0, 0.25, blocking=False)
        record_scratch.play_note(56, 0.8, 0.5)
        drum_kit.play_note(63, 0.6, 0.1)
        drum_kit.play_note(63, 0.8, 0.1)
        drum_kit.play_note(63, 1.0, 0.1)
        count += 1

    count = 0
    while count < 4:
        record_scratch.play_note(51, 1.0, 0.5, blocking=False)
        drum_kit.play_note(60, 0.4, 0.3)
        drum_kit.play_note(60, 0.6, 0.1)
        drum_kit.play_note(55, 0.8, 0.1)
        count += 1
```

You might name the new function `beat_music`, or something like that. So, copying the code above into the new file and placing it inside of the function definition, and then calling the function afterward, I got the following:

```
from scamp import *

s = Session()
s.tempo = 90

drum_kit = s.new_part("Kit 1", soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
record_scratch = s.new_part("Scratch 1", soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
```

```
synth = s.new_part("CandyBee", soundfont="Synths.sf2")


def beat_music():
    while True:
        count = 0
        while count < 3:
            record_scratch.play_note(53, 1.0, 1.0, blocking=False)
            drum_kit.play_note(60, 0.4, 0.5)
            wait(0.25)
            drum_kit.play_note(60, 0.4, 0.25)
            drum_kit.play_note(63, 0.6, 0.25)
            drum_kit.play_note(56, 0.4, 0.5)
            record_scratch.play_note(62, 0.6, 0.5)
            drum_kit.play_note(61, 1.0, 0.25, blocking=False)
            drum_kit.play_note(63, 0.6, 0.25)
            drum_kit.play_note(63, 0.6, 0.25)
            drum_kit.play_note(59, 1.0, 0.25, blocking=False)
            drum_kit.play_note(56, 0.8, 0.25)
            drum_kit.play_note(61, 1.0, 0.25, blocking=False)
            record_scratch.play_note(56, 0.8, 0.5)
            drum_kit.play_note(63, 0.6, 0.1)
            drum_kit.play_note(63, 0.8, 0.1)
            drum_kit.play_note(63, 1.0, 0.1)
            count += 1

        count = 0
        while count < 4:
            record_scratch.play_note(51, 1.0, 0.5, blocking=False)
            drum_kit.play_note(60, 0.4, 0.3)
            drum_kit.play_note(60, 0.6, 0.1)
            drum_kit.play_note(55, 0.8, 0.1)
            count += 1

beat_music()
```

Notice that all of the indentations created by the while loops are shifted over by one, since they are inside of the function definition. Also notice that the call to the `beat_music()` at the end is not indented, since it is part of the *main* script, not the function definition.

Of course, it goes without saying that your code should look fairly different, since your beat is different!

Back to instructions

Help 5

Define all of the functions first, and then call the synth part, like this:

```
def beat_music():
    # all of your beat code goes here


def synth_music():
    # all of your synth code goes here


synth_music()
```

In my case, this turned out like this:

```
from scamp import *

s = Session()
s.tempo = 90

drum_kit = s.new_part("Kit 1", soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
record_scratch = s.new_part("Scratch 1", soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
synth = s.new_part("CandyBee", soundfont="Synths.sf2")


def beat_music():
    while True:
        count = 0
        while count < 3:
            record_scratch.play_note(53, 1.0, 1.0, blocking=False)
            drum_kit.play_note(60, 0.4, 0.5)
            wait(0.25)
            drum_kit.play_note(60, 0.4, 0.25)
            drum_kit.play_note(63, 0.6, 0.25)
            drum_kit.play_note(56, 0.4, 0.5)
            record_scratch.play_note(62, 0.6, 0.5)
            drum_kit.play_note(61, 1.0, 0.25, blocking=False)
            drum_kit.play_note(63, 0.6, 0.25)
            drum_kit.play_note(63, 0.6, 0.25)
            drum_kit.play_note(59, 1.0, 0.25, blocking=False)
            drum_kit.play_note(56, 0.8, 0.25)
            drum_kit.play_note(61, 1.0, 0.25, blocking=False)
            record_scratch.play_note(56, 0.8, 0.5)
            drum_kit.play_note(63, 0.6, 0.1)
            drum_kit.play_note(63, 0.8, 0.1)
```

```
            drum_kit.play_note(63, 1.0, 0.1)
            count += 1

        count = 0
        while count < 4:
            record_scratch.play_note(51, 1.0, 0.5, blocking=False)
            drum_kit.play_note(60, 0.4, 0.3)
            drum_kit.play_note(60, 0.6, 0.1)
            drum_kit.play_note(55, 0.8, 0.1)
            count += 1


def synth_music():
    while True:
        for pitch in [56, 70, 64]:
            synth.play_note(pitch, 1, 0.25)
            synth.play_note(pitch + 4, 1, 0.25)

        synth.play_note(50, 1, 2)

        for pitch in range(52, 62):
            if pitch % 2 == 0:
                synth.play_note(pitch, 0.8, 0.125)
                synth.play_note(pitch + 4, 0.8, 0.25)
                synth.play_note(pitch - 3, 0.8, 0.125)
            else:
                synth.play_note(pitch, 0.8, 0.125)
                synth.play_note(pitch + 2, 0.8, 0.125)

        synth.play_note(51, 1, 0.125)
        synth.play_note(53, 1, 0.125)
        synth.play_note(54, 1, 0.75)


synth_music()
```

Now what happens if you write at the end:

```
beat_music()
synth_music()
```

Or what about:

```
synth_music()
beat_music()
```

Why can't they happen at the same time?

The reason is that programs generally work *sequentially*, executing lines of code one after the other. In this case, since each part contains a `while True` loop, we never get to the other part.

What we want is for the two parts to happen *simultaneously*. In computer science, when two things happen at the same time, it is called *parallelism* (imagine two parallel lines, going simultaneously in the same direction). We'll address this in the next step.

Back to instructions

Help 6a

In order for both parts to happen at the same time, we need to use *parallelism*. The syntax to start a parallel process in scamp is to call "fork" on one of the parts, like this:

```
fork(beats_music)
```

This starts the `beats_music` function going on a parallel time stream, and then immediately moves on to the next line of code in the main script.

Important: Do not put the open-close parentheses after `beats_music` here! We aren't *calling* the function in a traditional sense. Instead, we are *passing* the function, as an *argument* to the `fork` function. We are saying "Here's the function `beats_music`, go ahead and fork it."

To run both functions at the same time, one of them needs to be forked, and one can be called in the normal way. You can choose to fork the `beats_music` function and then call the `synth_music` function, or you can choose to fork the `synth_music` function and then call the `beats_music` function.

You can also fork both functions, but be careful: this will start *two* parallel child processes, but then the script will end immediately, causing those processes to stop before they even get going. To stop this from happening, you can call afterward `wait_for_children_to_finish()`.

Back to instructions

Help 6b

To fork the beats part and call the synth part, write:

```
fork(beat_music)
synth_music()
```

To fork the synth part and call the beats part, write:

```
fork(synth_music)
beat_music()
```

Both should cause the two parts to run simultaneously. One final possibility is to write:

```
fork(beat_music)
fork(synth_music)
s.wait_for_children_to_finish()
```

This forks both the beats and the synths part, and then keeps the main script alive so that they can play.

My final code ended up looking like this:

```
from scamp import *

s = Session()
s.tempo = 90

drum_kit = s.new_part("Kit 1", soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
record_scratch = s.new_part("Scratch 1", soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
synth = s.new_part("CandyBee", soundfont="Synths.sf2")


def beat_music():
    while True:
        count = 0
        while count < 3:
            record_scratch.play_note(53, 1.0, 1.0, blocking=False)
            drum_kit.play_note(60, 0.4, 0.5)
            wait(0.25)
            drum_kit.play_note(60, 0.4, 0.25)
            drum_kit.play_note(63, 0.6, 0.25)
            drum_kit.play_note(56, 0.4, 0.5)
            record_scratch.play_note(62, 0.6, 0.5)
            drum_kit.play_note(61, 1.0, 0.25, blocking=False)
```

```
            drum_kit.play_note(63, 0.6, 0.25)
            drum_kit.play_note(63, 0.6, 0.25)
            drum_kit.play_note(59, 1.0, 0.25, blocking=False)
            drum_kit.play_note(56, 0.8, 0.25)
            drum_kit.play_note(61, 1.0, 0.25, blocking=False)
            record_scratch.play_note(56, 0.8, 0.5)
            drum_kit.play_note(63, 0.6, 0.1)
            drum_kit.play_note(63, 0.8, 0.1)
            drum_kit.play_note(63, 1.0, 0.1)
            count += 1

        count = 0
        while count < 4:
            record_scratch.play_note(51, 1.0, 0.5, blocking=False)
            drum_kit.play_note(60, 0.4, 0.3)
            drum_kit.play_note(60, 0.6, 0.1)
            drum_kit.play_note(55, 0.8, 0.1)
            count += 1


def synth_music():
    while True:
        for pitch in [56, 70, 64]:
            synth.play_note(pitch, 1, 0.25)
            synth.play_note(pitch + 4, 1, 0.25)

        synth.play_note(50, 1, 2)

        for pitch in range(52, 62):
            if pitch % 2 == 0:
                synth.play_note(pitch, 0.8, 0.125)
                synth.play_note(pitch + 4, 0.8, 0.25)
                synth.play_note(pitch - 3, 0.8, 0.125)
            else:
                synth.play_note(pitch, 0.8, 0.125)
                synth.play_note(pitch + 2, 0.8, 0.125)

        synth.play_note(51, 1, 0.125)
        synth.play_note(53, 1, 0.125)
        synth.play_note(54, 1, 0.75)


s.fork(beat_music)
synth_music()
```

Back to instructions