Beat Making Activity

Start by opening Thonny, and creating a new file in the directory where you keep your files for this course. Call it something like "beat_activity.py", and make sure that the soundfont you downloaded ("Emu_Planet_Phatt_Hip_Hop.sf2") is in the same folder.

1. Start by writing the code to import everything from the SCAMP library. Then, create a new Session object and assign it to the variable `s`. When creating the Session `s`, include a keyword argument `default_soundfont` and set it to the name of the soundfont, "Emu_Planet_Phatt_Hip_Hop.sf2", so that we are using the soundfont with the drum kit sounds. Help, Solution
2. Add a line that sets the tempo of the new Session object `s` to something around 70 - 130. You can experiment with it later. Help, Solution
3. Add a line of code to ask `s` to print the available instrument sounds. (Remember, you can start typing `s.print` and then hit tab to have it give you suggestions.) Try running the code, so that it prints out the possible sounds. Solution
4. Add a line to ask `s` to create a new drumkit part, and store this new instrument in a variable. If you want, you can create a few more instruments using other presets, maybe another drum kit or a record scratch. Once you're done, can comment out (add a hashtag in front of) the line that prints the instrument sounds, so that it doesn't print every time. Help, More help
5. Ask the instruments to play some notes using the `play_note` function! Put a bunch of `play_note` calls one after the other, and try to adjust the pitch, volume and duration arguments until you get something that sounds awesome. Remember that for the drum kit presets, different pitches will give you different samples, whereas for some of the other presets (like the scratch presets) different pitches will transpose the same sample up and down in pitch. Help, More Help
6. Try adding the `blocking=False` keyword argument to some of the `play_note` calls. You might also add some calls of the `wait` function. Can you make it more interesting? Help
7. Take your whole beat and wrap it in a while loop to make it loop forever! Help, More Help
8. Try changing the while loop so that it only happens a certain number of times. You can do this by creating a variable to count how many

times you've gone through the loop, adding 1 to it every time at the end of the while loop, and then changing the **condition** of the while loop to check that that variable is below a certain number. A different approach would be to have a variable keep track of how many iterations are *left*, subtract from that number, and check that it's greater than zero. Help, More help

9. Try adding a second beat after the first one is done repeating. You can copy and paste the entire while loop and place it after the first one. Then change the beat inside of the second while loop to do something different. Maybe it also can repeat a different number of times? Help
10. Add one more big while loop to loop both beats forever. Help

At this point, you might be starting to get some interesting ideas: What if you added even more while loops? Or made the outer loop happen only a certain number of times? Or even nested the **outer** loop inside of yet **another** while loop?

You could also experiment with using the loop count variable inside of `play_note`. What happens if you do something like `drum_kit.play_note(60 + count, 0.7, 1 / count)`?

It's also totally okay to keep it simple and make the beat sound as interesting as you can!

Once you're done, you can save your beat to a wave file by putting the line `playback_settings.recording_file_path = "my_beat.wav"` (or whatever you want to name the file) right after the first line in which you import SCAMP.

Help 1a

Importing things

To import something from a library in Python, you use the following syntax:

```
from [library] import [thing we want to use]
```

So, for instance, if we have a library called "pets", and we want to import the Dog class, we might write:

```
from pets import Dog
```

To import *everything* from a library, we can use an asterisk. So to import all sorts of pets from the pets library, we might write:

```
from pets import *
```

What line of code would you use to import everything from the scamp library?

Creating an object

To create an object of type `Dog`, we would use the syntax `Dog()`. Remember that this is like calling a function, but because dog is uppercase, it means that we're creating a new object.

To **assign** this new dog to the **variable** my_dog, we would write:

```
my_dog = Dog()
```

Finally, if we wanted to use the **keyword argument** "ears" to specify that the ears should be floppy, we might write:

```
my_dog = Dog(ears="floppy")
```

Notice that, since it's surrounded by quotes, the word "floppy" is a **string**.

So how would you write a new line of code to create a new `Session` object, setting the `default_soundfont` keyword argument to "Emu_Planet_Phatt_Hip_Hop.sf2", and assigning it to the variable `s`?

Back to instructions

Help 1b

You can import the scamp libraries with the line:

```
from scamp import *
```

The `*` means everything. One of the things we are importing is the `Session` class. To create a new Session, you can use the line:

```
s = Session()
```

To make it use the "Emu_Planet_Phatt_Hip_Hop.sf2" soundfont, which has the drumkit sounds, add a keyword argument:

```
s = Session(default_soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
```

So at this point, your code would read:

```
from scamp import *
s = Session(default_soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
```

Back to instructions

Help 2a

If you have an object `my_dog` (of type/class Dog, say), you could set its name to "rocky" by writing:

```
my_dog.name = "rocky"
```

So what could you write to set the tempo of the Session `s` to, say, 100?

Back to instructions

Help 2b

To set the session tempo, add a line like:

```
s.tempo = 90
```

So at this point, your code would read something like:

```
from scamp import *

s = Session(default_soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
s.tempo = 90
```

Back to instructions

Help 3

Simply add the line:

```
s.print_default_soundfont_presets()
```

At this point, your code should read:

```
from scamp import *

s = Session(default_soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
s.tempo = 90

s.print_default_soundfont_presets()
```

When you run it, you should get a printout like:

```
PRESETS FOR Emu_Planet_Phatt_Hip_Hop.sf2
   Preset[001:116] E-mu Systems  1997 1 bag(s) from #0
   Preset[128:009] Kit 10 1 bag(s) from #1
   Preset[128:008] Kit 9 1 bag(s) from #2
   Preset[128:007] Kit 8 1 bag(s) from #3
   Preset[128:006] Kit 7 1 bag(s) from #4
   Preset[128:005] Kit 6 1 bag(s) from #5
   Preset[128:004] Kit 5 1 bag(s) from #6
   Preset[128:003] Kit 4 1 bag(s) from #7
   Preset[128:002] Kit 3 1 bag(s) from #8
   Preset[128:001] Kit 2 1 bag(s) from #9
   Preset[128:000] Kit 1 1 bag(s) from #10
   Preset[000:001] SE Sub 1            1 bag(s) from #11
   Preset[000:002] SE Sub 2            1 bag(s) from #12
   ...etc...
   Preset[001:114] Scratch 16 1 bag(s) from #251
   Preset[001:115] Scratch 17 1 bag(s) from #252
   Preset EOP
```

Back to instructions

Help 4a

Remember in the first script we looked at, there was the line:

```
clar = s.new_part("Clarinet")
```

Can you add some modified versions of this line to your code?

Back to instructions

Help 4b

To add a new instrument, use a line like:

```
drum_kit = s.new_part("Kit 3")
```

Add a couple different instruments in this way. At this point, your code should look something like:

```
from scamp import *

s = Session(default_soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
s.tempo = 90

# s.print_default_soundfont_presets()

drum_kit = s.new_part("Kit 1")
record_scratch = s.new_part("Scratch 1")
```

But use whichever kits or presets you want to experiment with!

Back to instructions

Help 5a

If you've added a drum kit instrument and stored it in the variable `drum_kit`, you can play a note by writing:

```
drum_kit.play_note(60, 0.7, 1.0)
```

Remember that the **arguments** go in the order pitch, volume, length. So you could make it louder by changing the 0.7 to an 0.9, and you could make it shorter by changing the 1.0 to an 0.5.

Can you add several of these lines to your code, changing the instrument, pitches, volumes, and lengths to get an interesting beat?

Back to instructions

Help 5b

For example, add some lines like:

```
drum_kit.play_note(60, 0.4, 0.5)
drum_kit.play_note(60, 0.4, 0.5)
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(63, 0.6, 0.25)
drum_kit.play_note(63, 0.6, 0.25)
drum_kit.play_note(56, 0.8, 0.25)
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)
```

So the full code at this point might look like:

```
from scamp import *

s = Session(default_soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
s.tempo = 90

# s.print_default_soundfont_presets()

drum_kit = s.new_part("Kit 1")
record_scratch = s.new_part("Scratch 1")

drum_kit.play_note(60, 0.4, 0.5)
drum_kit.play_note(60, 0.4, 0.5)
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(63, 0.6, 0.25)
drum_kit.play_note(63, 0.6, 0.25)
drum_kit.play_note(56, 0.8, 0.25)
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)
```

Back to instructions

Help 6

To make a play_note call go immediately to the next line, add the `blocking=False` keyword argument, like this:

```
record_scratch.play_note(53, 1.0, 2.0, blocking=False)
```

The note still plays for its full length; we just don't have to wait for it to finish before moving on to the next line. This means that something this will cause the record scratch and the kick drum hit to happen at the same time:

```
record_scratch.play_note(53, 1.0, 1.0, blocking=False)
drum_kit.play_note(60, 0.4, 0.5)
```

To wait for a certain number of beats add a function call like this:

```
wait(0.5)
```

At this point, the full code might look something like this:

```
from scamp import *

s = Session(default_soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
s.tempo = 90

# s.print_default_soundfont_presets()

drum_kit = s.new_part("Kit 1")
record_scratch = s.new_part("Scratch 1")

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)
```

Back to instructions

Help 7a

Remember that the syntax for a while loop is the word "while", followed by a test (or **condition**), followed by a colon. You then indent all of the lines that you want to loop.

If we want to loop forever, we need a condition that is always true. So we can just use the value `True`:

```
while True:
    # put some code here that you want to loop
```

Back to instructions

Help 7b

To make your beat loop forever, put the line `while True:` before all of the play note calls, and indent all of the lines you want to loop. So the full code at this point might look like:

```
from scamp import *

s = Session(default_soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
s.tempo = 90

# s.print_default_soundfont_presets()

drum_kit = s.new_part("Kit 1")
record_scratch = s.new_part("Scratch 1")

while True:
    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)
```

Back to instructions

Help 8a

To count the number of times that you have gone through a loop, you could start by defining a variable called count:

```
count = 0
```

. . . and then inside of the loop, add one to count each time:

```
count = 0
while True:
    # put some code here to play some music
    count += 1
```

This would still go on forever, though, since the **condition** is always `True`. What could you change the condition to, so that is stops when count goes above a certain number? (More details in help 2.)

Back to instructions

Help 8b

To have the code in the while loop happen, say, 3 times, add the **condition** `count < 3`:

```
count = 0
while count < 3:
      # put some code here to play some music
      count += 1
```

This way, the loop happens when count is 0, 1, and 2, but when it gets to 3 it fails the test, since 3 is not less than 3. Sometime it can be helpful to print the value of `count`, so that you can keep track, like this:

```
count = 0
while count < 3:
        print(count)
        # put some code here to play some music
        count += 1
```

An alternative approach is to count **down**, using a variable to keep track of how many loops are remaining, like this:

```
loops_remaining = 3
while loops_remaining > 0:
        print(loops_remaining)
        # put some code here to play some music
        loops_remaining -= 1
```

By the way, it doesn't matter what we name the variable. It could be called `x`, or even `count_chocula`:

```
count_chocula = 3
while count_chocula > 0:
        print(count_chocula)
        # put some code here to play some music
        count_chocula -= 1
```

But, better in general to name it something that would make sense to someone else looking at your code for the first time.

At this point, our full code might look like this:

```
from scamp import *

s = Session(default_soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
s.tempo = 90

# s.print_default_soundfont_presets()

drum_kit = s.new_part("Kit 1")
record_scratch = s.new_part("Scratch 1")

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
```

Back to instructions

Help 9

Copy the whole section of code, including where you define the counting variable, then change the second loop to sound different from the first. At this point, the full code might look like this:

```
from scamp import *

s = Session(default_soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
s.tempo = 90

# s.print_default_soundfont_presets()

drum_kit = s.new_part("Kit 1")
record_scratch = s.new_part("Scratch 1")

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
```

Notice – this is important! – that we reset `count` to zero after the first loop. otherwise it will keep counting from where wherever it left off (in this case 3).

Back to instructions

Help 10

To make several lines of code happen over and over forever, all you need to do is place `while True:` before them, and then indent them:

```
while True:
    # put some code here that you want
    # to repeat forever
```

In our case, we would take *all* of that code with our two while loops and indent the whole thing!

My final code looked like this:

```
from scamp import *

s = Session(default_soundfont="Emu_Planet_Phatt_Hip_Hop.sf2")
s.tempo = 90

# s.print_default_soundfont_presets()

drum_kit = s.new_part("Kit 1")
record_scratch = s.new_part("Scratch 1")

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
```

Back to instructions