[SQUEAKING]

[RUSTLING]

[CLICKING]

MICHAEL SCOTT ASATO CUTHBERT: So go ahead and open up Jupyter Notebook, and we'll get started working with streams. What's very different from this topic and many other topics we'll have is that the introduction to the topic will also be the unlocking video for the topic.

You've certainly done enough with representation in topic two-- representation-- that you deserve to have some way of representing music to start. So we can go ahead and, from music21, import stream-- the stream module.

You're also going to get a second unlock now, and that is from music21, import corpus. And corpus is a collection of pieces that you've either created yourself or what we'll be using-- a collection of pieces that come with music21.

So the corpus module has one method that we'll use a lot, and that's parse. And so you load something from the disk. You can look online in the music21 user's guide to see the list of pieces that come together-- come with music21 corpus.

But we happen to know one pretty well from our last look, and that's Bach, "BWV 66.6." And we'll call that bach_score-- second. No, we'll just call it Bach. bach. Bach equals "BWV 66.6." And we see what Bach is. We see, hey, it's a type of stream called a score.

What we'll find out is that the music21 basic way of representing music starts, usually, with a score. There are some things bigger than the score called, opus, and stuff like that. But usually, you're going to start with a score.

And like any music21 object, you can show it. And so we can show the entire score. It usually takes a little bit longer the first time. But then it comes up, and we can see it has four parts-soprano, alto, tenor, bass. It's in 4/4, abbreviated as common time. It begins with a pickup. That's always going to be hard.

And one of the things we'll see-- though music21 will call this measure zero, one, two, and so on, your notation software, such as MuseScore, which is what I'm using to automatically generate this, might not know how to work with pickup measures as generated from music21. We're generating, automatically, to MusicXML, and then loading that into MuseScore and then generating a PNG file and putting it back into Jupyter Notebook. Lots happening on the fly when we say bach.show().

In any case, for some reason the measure numbers don't translate, so watch out for this. music21 thinks of this as measure three, and MuseScore thinks it is measure four. So anyhow, we have four parts. Pretty short score-- all fits over here. And I've already asked you to think about what the key is, what are some interesting weird moments in various parts and so on. We can also show this as a MIDI file, though that sounds not nearly as good as the recording you listened to.

[BACH, "BWV 66.6"]

So that's useful. If I made my screen bigger, then the thing looks pretty right. So we have a score, and a score is a type of stream, and a stream is a type of container. And so it works like a list. You can see exactly what is in the score. And we can see that there's a metadata object and then four part objects and something called a staff group. And that makes it so that all the parts look right together.

So within this, we can get the soprano part by getting the second element from it, which-- we can call that p0 for the first part because we can get that as the first thing. And we can see that p0 is another type of stream called a part. So parts are contained within scores. Think about whether this is time-wise or part-wise. Well, it had part, so it's part-wise.

So we can also take this and show this particular part. And I reset for a second, so it takes a little bit longer. There we go. Just one part going there. And we can see everything that is in part 0. Since I've said that this is a part-wise format, think about what you would expect to see next.

OK, have you thought about it? That's right. It's mostly a bunch of measures. There's an instrument object that says it's a soprano instrument, but then there's just 10 measures. Measure 0 is numbered that way because it's a pickup measure, and so on.

So we can get the pickup measure. We'll call it, pickup. And what do we want to do? It's the second element in there, so we get 1. And we can just show the pickup. That's not too exciting. Let's get the first real measure. Yep, great.

And we can see that m1 is also a string. It's a measure object. And it reports that its number is 1. Yep. And we can see-- and it has something called an offset. And as we said, in music21, we measure things in quarter lengths, for the most part. And so measure 1 begins one quarter note from the beginning of the piece itself.

And if you look again at this part, yeah, you can see there's quarter note 0, quarter note 0.5, quarter note 1. So measure 2 begins where? At offset 5. And then 9, 13. In this particular score, they're all off by four. Great.

So let's see what's inside measure 1. And here, we have the notes, finally. There could be-- if there were multiple voices in the stream, there could be voices at this point, but there aren't. This is just a single part.

So now that you can see these elements of a stream, let me show one other important thing that you can do. You've done the show, which by default, puts it out as MusicXML. When you're putting it in a script or in the Jupyter Notebook, it puts it out as MusicXML, opens up MuseScore if you have it, turns it into a PNG, and puts it right back.

But there's another thing you can do. You can show it as a text file. And that shows you not only what elements are inside the measure, but also what their offsets are. So every element, every music21 object, every note, every clef has an offset within its current stream-- within its current container, and it also has a duration. And in this case, each of them is quarter notes, so their durations are 1.0.

I forgot when I was making this video, first, there's something really important here. Notice that this first note has an offset of 0, but it's not the first note in the piece, it's the second or third note in the piece but at the second quarter note. So why is its offset not 1?

Well, it turns out all of the offsets are measured from the start of the container. So this is a measure that begins at offset 1. And therefore, its first note is at offset 0 within the container. So if you're going to figure out what the offset of a particular element is within the piece as a whole, you're going to need to add up all of the offsets of previous containers back to previous [INAUDIBLE].

So this is how you can load up a score that already exists and get to particular notes. So in fact, let's get that a-- say, 0. And we can see that a is a note, and we can get its full name and all the other things that we worked with before-- pitch, duration, duration.type, and so on.

So this is how you can load a preexisting stream from a MusicXML file, like "BWV 66.6" or from a MIDI file, or from an ABC file, or any of the number of music representation formats that music21 supports.

But now, let's go the opposite way. Let's create and build up our own stream. So we'll build it up from scratch. Let's import notes which have already been unlocked, and let's create a note.

When we show this note, music21 is actually doing something for us. It's creating a whole set of streams around it for us. But we can do it our own way. So let's create a voice, which is the type of stream. And we will insert that note at position 0. Oops. We will insert that note at position 0 into the stream. You'd only think that working with music21 for 15 years and having created it, I'd remember what goes first the offset or the element, but I sometimes forget.

Great. We'll unlock another module for you right now. We'll unlock the meter module. Yay. Unlocking. And we'll create a meter and a time signature of 1/4. And we can see that something like this. Yes. Things like that.

Now, let's create a measure for it. And we will insert, at the beginning of the measure, the time signature, and also insert, at the beginning of the measure, the voice that contains the note. We can put more than one thing at the beginning because time signatures don't have any duration.

Now, we can create a part. And we can insert the part at the beginning of the measure at the beginning of the part, or we can put the measure at the end of the part. And since there's nothing in the part right now, append will be the same as inserting a 0. But you don't have to remember how many things are in there yet. Now, we'll create a score, and we'll put the part at the beginning of the score.

One of the mistakes that people make with music21, quite often, is that they get so used to appending things that they'll append the part to the score, which-- the first part-- that's fine. But then they'll append the second part to the score. And if you think about it, parts don't follow each other. They're all at position 0. So you want to make sure to always insert parts at position 0.

Now, we have a score that looks basically the same as when we created the n note. If we modify some parts of the-- of the original note objects, something we can see-- that the score will be updated as we do.

Now, that doesn't match the time signature because we explicitly put a time signature in there so it doesn't create one for us. So let's create a rest. And then in that voice, let's insert, at 0.5-- now, let's make it easier to append the rest to there and now show all this.

If we do a show of text on the score, we can see that it includes the part, the measure, the time signature, the voice, and then the note and the rest. Well, that's enough for streams for today. Thanks for paying attention.