[SQUEAKING]

[RUSTLING]

[CLICKING]

MICHAEL CUTHBERT: Hi, everybody. Today we're going to spend the bulk of our time making up for lost lab time, lost time coding and doing-- implementing some of the ideas that we have, but first, we're going to step back to just a little bit on the quiz. Apologies for taking a while to get it back to you, but now you have it. And the quiz is the opposite of the practical side. It's more of the theoretical side. So I wanted to talk about a couple aspects of that.

Somebody's going to remind me to put the screen back down. Paul's not here today. So please do that. Some of the practical aspects on the quiz in space. First, why are authors and articles important? Why am I really sensing that not just you know what the three contexts are or what optic is, but really whose idea this is?

And there's really two reasons. One, the field is small. The field is small. And if you're working in it-- and it's young. Not young as y'all or-- it's even older than me, but young, like almost everyone who has come up with the great ideas that have changed the field are still living and people you can meet. But more importantly--

And so some of the guest speakers that we've had recently have appeared-- it's campus preview weekend, so we got to close this. Otherwise, it'll get loud. But some of the speakers that we've seen have actually already appeared in articles that we've been reading and things. So these are names that will keep popping up again. And when David Huron talks about Carol Krumhansl, who was after the quiz. But this is a name that, oh, I've already read that.

The other first reason is that-- I don't know if you saw on one of our guest speakers on music cognition. A lot of times, there were these nice, pretty pictures of-- I don't know. Can I draw a light bulb or something like that? And some idea and things.

And over in the corner, you would see Devin's 2019 or whatever. And those are just ways-- they're industry shortcuts for how you can learn more about a particular topic. So I don't tend to talk with footnotes. Maybe I should more-- Strunk and White, 2004-- but these are things that people do quite often. And it's because--

Here's the worst thing about almost every article you'll read, whether in science, engineering, humanities, arts that-- I'm telling you to read these articles and things, but then the articles aren't actually written for you to read. It's so ironic. Why do we write and publish research? For the most part, we're publishing research to talk to or to argue with or push back on or develop the theory of somebody else who's living.

So these scholars tend to talk to each other quite a bit. And so what they're doing when they're putting these names down is really-- there's a seat here today, if you'd like-- and what they're really doing is trying to argue against something that you might not already know. So when somebody is presenting an idea, they're really saying, hey, I've got a better idea than this other person. And so they're always going to be talking about names of other people. And so it's just kind of good to get that. So that's something there.

The other question I think that can come up in the context of these quizzes based primarily on the readings and these abstract things is, why are there so many darn old articles in this class? Why are you doing a lot of citations of things that are 10 years old, 25 years old? And then you got the Illiac Suite. What is that? 70-- 77 years old now? So what's the point of getting into it?

So I've been teaching for a while, and one of the-- there's one lecture that I love the topic, and I hope you'll love it too when we get to it, but I hate giving this lecture. It's the lecture on practical AI in music. And why is that? It's because the various techniques that I've been teaching have changed so often that they're not really relevant two years later.

So I'm going to be thinking about, oh, OK, this weekend, it's time to update everything from TensorFlow, which was so state of the art three years ago, and I wrote a whole lecture around it for those who know this, PyTorch, which I'm using now.

And it really means that there's a lot of things that we'll be spending time in that class. That if PyTorch ends up not being the super great thing that you're going to be using for the rest of your life because of GPT or whatever else that's coming out, those types of information will get stale pretty fast.

But the notion of things like equality and equivalence and how you can put equivalence classes, I hope, will stay with you even if you're not doing anything with music or even anything with computation. I hope there are things that will stay with you for the rest of your life, and that it'll change how you make an outlook on the world.

So that's a little bit why this class has a good amount of it. I figure if it's something that inspired me 15 years ago, and it's still inspiring me today, it has a good chance that it'll still be relevant inspirational in 10, 20, or 30 years from now.

For those who did the work on the global rule, somebody remind us what the global rule is and what-- first, where does it apply? Anyone want to take a guess? Matthew, do you want to take a guess or call on somebody else?

AUDIENCE: Counterpoint?

MICHAEL CUTHBERT: Counterpoint. Great. Super. So it's counterpoint rule, rule. And what's it trying to maximize in counterpoint? So I'm just trying to think. What's it trying to maximize in counterpoint? What's that? You want to try it?

AUDIENCE: I guess it's trying to maximize the musical notes or how good it sounds.

MICHAEL CUTHBERT: Max musicality. Absolutely. Great. Super. And it does this-- ironically, trying to maximize musicality came down to giving a formula or giving a set of formulas. And where do we begin with the global rule? [INAUDIBLE]?

AUDIENCE: From the last one?

MICHAEL CUTHBERT: Start from the end and go backwards and look at specifically, do you have what? Three possibilities for the last one? If you're going backwards? No, you have two possibilities because you can't repeat notes in this context. So you can be trying here and here against some other thing. David Lewin would actually not be happy with me. He always did his counterpoints in alto clef, but we'll do it in treble and bass.

So yeah, you're trying to figure out, OK, if we ended here and the previous note is here, what are the other notes that we can do? So we have traditional counterpoint rules, but we also have this notion that every note that appears in the line must at some point find what? You remember? Everybody quiet because they all know this so easily or because it's a little foggy? Easy? Foggy? OK, good. That's good. Good. I prefer silence when it's foggy because then I know what to do next.

So in order to get maximum musicality, you talked about smoo-- what begins with smoo--? He wants what kind of lines?

AUDIENCE: Smooth

MICHAEL CUTHBERT: Smooth lines. So we want smooth lines. So if at some point, you have a d here, a low d-- and we're going to get to c-- what are the notes that we need to see after this? Oops, just turned red. What are the notes that we need to see after this? What's a note we need to see after d?

AUDIENCE: E?

MICHAEL CUTHBERT: What's that?

AUDIENCE: E?

MICHAEL CUTHBERT: E. Good. Good. So if we have an e, then what happens if we have another d after that? What are we going to neec after the d at some point? Go ahead.

AUDIENCE: Another e.

MICHAEL CUTHBERT: Another e. Good. So basically, whenever you have a note, you have to have after it every note between here-- every diatonic note. Not every chromatic note. You're not going to need a d sharp at some point. Every diatonic note between here and the final cadence note. And that's why it was a little bit easier to work backwards.

Now, why was this rule important for this class? This one might be a silent because it's pretty easy, because almost everything in this class. Who can do it? [INAUDIBLE]? What kind of thing can apply this rule?

AUDIENCE: Computer.

MICHAEL CUTHBERT: Computer. So you can have an-- it can all be done. The ironic thing, we're maximizing musicality through algorithms. Great. So that's about all I want on that. And people whose chose that problem, those tended to do pretty well. Now let's get to second, equality and equivalence. And almost everybody chose to do this one. What's the name associated with this, without looking down? Dmitri is the first name--

AUDIENCE: Tymoczko.

MICHAEL CUTHBERT: Tymoczko. Let's try to spell it together. Give me a--

AUDIENCE: T--

MICHAEL CUTHBERT: T-- give me a--

AUDIENCE: Y--

MICHAEL CUTHBERT: Y-- give me a--

AUDIENCE: M--

MICHAEL CUTHBERT: M-- give me a--

AUDIENCE: O--

MICHAEL CUTHBERT: O--

AUDIENCE: C-Z-K--

MICHAEL CUTHBERT: K--

AUDIENCE: Q--

MICHAEL CUTHBERT: Q, Tymoczkq. He'd like that. No, Tymoczko. So this one's a little bit hard, and I would not-- whatever. Basically, one of the things I try to do is I put in optic and whatever you put and see if Google could find the author on the first thing so that-- on the first page. And just remember that, if you put in Wolfgang Mustard, it finds Mozart. So Google is pretty good at this, but-- OK, so Dmitri Tymoczko's optic. Almost everybody got optic-- octave-- P-T-I-C--

AUDIENCE: Cardinality.

MICHAEL CUTHBERT: Cardinality. Good. I know this back of my hand. I just don't know how to spell the word optic. And then most people, thank you for including my little contribution of spelling on the list, a lot of people. Great. So people did really, really great on that.

Here's some of the things that, though, were a little bit harder. I think I might have thrown some people for a loop, and I didn't try to grade too hard on this, because I realized, ooh, I didn't do the greatest job on this in lecture. So it was the difference between equality and equivalence. When we talk about sameness, which one do you think we're most likely talking about? Votes for equality? Same, equivalence. Oh, we have more on equivalence.

I might not have done as well on this. I think of sameness as having more to do with equality in some sense and equivalence meaning can be grouped together. And we could say it can be grouped together in some logical connection or can be grouped together, connected in some logical way.

And the other thing that I think probably falls a little bit on me because didn't come out as clearly is that, when we're talking about same, some people-- and you got full credit if you did this-- talked about the big picture. This note is same in every conceivable way. So are this piece is the same as this other piece in every conceivable way? And that's one way that we can think of a grand scale equality of problem, is that you all are really smart and you all know how to conceive of things that can break that.

I hit Play on this recording and then I hit Play on this recording again, but one of them happened 10 minutes later, and one of them was played through on speakers that had 10 minutes more of wear and tear on them or something. So you can always conceive of the way it's different. So maybe that's not as good, but we might think of equality of aspects of something, equality of aspects of-- that--

I'm trying to think. You all are sitting down, so I'm trying to think. Maybe Jake is. That we are not exactly the same, but I think we're approximately the same height. Maybe we're the same height. Let's say that we are. So we are same in heightness or something like that.

So two notes could be same in frequency, equal in frequency. So that's tend to use equality for some sort of attribute of something that is identical or not cannot be distinguished from each other. Whereas equivalence, we tend to think of as things that we can group together.

None of us are the same in any real, deep, great way, but we all could be lumped under the equivalence class of MIT 21M.383 students. And it kind of tells you also that equivalence classes might be overlapping, because some of you might be also MIT 6.042 students together, or you could be in something else. So they could be overlapping.

Then the hardest part, I think, on this problem that snagged a bunch of people is where-- in what contexts equivalence classes might be used and might not. That we might think of [PLAYING INSTRUMENT] as one chord, arpeggiated, and [PLAYING INSTRUMENT] as the same chord or-- I guess I just use same in an equivalent sense.

So maybe I'm less dogmatic there-- as an equivalent chord, even though one of them was going up and the other one was going down. So we might think that the permutation doesn't matter in that particular class when we're talking about, what chord did I arpeggiate? And then later, we might say, well, direction is another equivalence class. Did it go up or did it go down?

And contexts for when you might use one and when you might not use one. A lot of times the context where equivalence classes come in, it's a little kind of too obvious in a class called computational music theory and analysis. The equivalence classes tend to not come in very-- as often in performance than in analysis.

So I think more than one person wrote, well, that you can think of a performance where cardinality doesn't matter. And I might have written something like, well, if you play that measure three times because that's what's written-- and I only play it once-- it's going to matter for a performance class play, but we might not think of the cardinality as mattering.

If we're trying to do a search on, does this piece have any diminished triads? We don't really care how many major triads and how many minor, but in that case, we might just be lumping everything under the type of triad equivalence class and not counting how many. A particular one, but we use these all the time in real life.

So the example that popped into my head today is, if you're going to Taylor Swift-- if you're going to be analyzing a Taylor Swift song and you're going to look at the harmonic structure of it, does it matter, presumably, if you're listening to Taylor Swift's recording or if you're listening to a cover band, one that's trying to be faithful for harmonic structure? Not really. You hope a good cover band is trying to sound like that, trying to use the same harmonies.

So we could come up with a cover band equivalence class cover. Maybe we just call it the cover equivalence class, which applies if you're doing harmonic analysis. Does it matter whether you're seeing Taylor Swift or a cover band if you just paid $900 for front row tickets? Yeah, it does. So we can think of live performance as a place where cover equivalence doesn't exist.

Now, we mostly talked about pitch equivalence. So that's a big thing there. So that we might be thinking too much about pitch and not enough about duration equivalence, not enough about bigger things.

The other thing just to talk about is that, often, when we're talking about these pitch equivalences, like octave or cardinality or things, we're not really thinking about the sort of-- I don't know-- the scale of the piece.

We're not really thinking, does this piece have the same number of notes as this piece if we apply cardinality equivalence or something? But we might be thinking about, does this chord have the same number of unique pitch classes if we're applying octave, permutation, transposition, and cardinality? So that level comes up quite a bit.

I think there's one place where we do tend to think of equivalence classes on a piece level sometimes. That's when you have-- I don't know-- Beethoven's third symphony in-- I don't know-- E-flat major. Sorry, of course, this isn't a music history class on the history of Beethoven, but that we do tend to group pieces in classical music by key.

The other place that immediately comes to head, where there might be equivalence classes of pieces is a DJ might have a beats per minute equivalence class, where you want to group together all songs with the same BPM so you can switch between them. So think on that.

Any thoughts or questions on this? And thank you to the person who gave me the opportunity to explore the permutation equivalence class on happy birthday.

OK last ones to talk about, the three contexts. This one was a little bit harder. Some people got. It was by Eleanor Selfridge-Field. And some people thought it was the decently similar about the same time in the semester article by Nicholas Cook. And if you wrote that, you are getting some credit because you were really in the right ballpark. So she talks about the three main contexts talks about are-- you can read them off or better sing them out. First one?

AUDIENCE: Symbolic?

MICHAEL CUTHBERT: So symbolic is-- well, we'll get-- is there but not one of her terms. And that ends up being the problematic one. But yeah, we'll keep that one. Else? What's that?

AUDIENCE: Graphical?

MICHAEL CUTHBERT: Graphical. Yep. Graphical. And what's the opposite-- what do we contrast graphical things with?

AUDIENCE: Sound.

MICHAEL CUTHBERT: Sound. Good. And then the logical is what she has. And so symbolic will be in there, things. One of the things that came up a lot in the quiz was that people were immediately going to a higher level than what she was intending. So when we think about graphical, this is a graphical context for representing music, but in her mind, what is the graphical thing I just created here? Anyone want to take a guess? If you don't go to high--

AUDIENCE: Circle and a line?

MICHAEL CUTHBERT: Circle and a line. Going to say-- great, a circle and a line. So by the time you move that to, Professor, you drew a quarter note with a stem down, stem down, on the third space with this context, you draw a C, you've already been moving to the logical domain.

Similarly, when you hear a sound-- I can only represent sound right now. Well, duh, I can do it in the sound domain but in the graphic domain of sound. You don't necessarily yet hear, oh, I hear a C. Maybe you can say-- or I hear an A. Maybe you can say-- if you're an oscillator in your head, you can say, I hear something oscillating at 4:40 times per second. But then to move that to an A4, octave 4, is a motion to the logical domain.

So this, I don't think I was clear enough on and maybe Eleanor Selfridge-Field wasn't enough or we just need to read it a little bit more closely. And similarly, even things like instrument, I hear an instrument, you're already moving to the logical domain.

You might, if you're a signal processor, be saying, oh, I'm hearing a waveform where only the odd partial harmonics are present, and then your memory bank is saying, oh, I'm hearing a clarinet. So those are the odd [INAUDIBLE] I don't teach that class. I think that's right.

So because elements of the logical domain, a lot of our answers were transferring to the sound or the graphical domain, the logical domain got sort of tangled up with the next higher domain, here, the next, more abstract thing, which in this case, I'm going to call it the meaning domain. We can call it the her-- I'm going to actually try to write this without looking down-- --menutic domain, which is the study of the construction of meaning or the conceptualization of meaning.

It's actually a pretty cool word if you don't know it, but I don't like it too much because my whole life, I've had blowhards, if I say that word. Who use hermenutic and throw it around everywhere? I'm having a hermeneutic construction of meaning. It's like, well, no, that's just what that means. If you [INAUDIBLE] up, learn your hermenutics [INAUDIBLE]. So I'll often just call it the meaning domain.

And so you might be that your logical sound domain, you've heard [MAKING TONGUE CLICKING SOUNDS] And your graphical domain, you saw a whole bunch of things on page and your logical domain said, oh, there's a whole bunch of randomly spaced staccato 8th notes, or even randomly spaced might be too far down. So at this level, you might say, oh, the title of the piece is fireflies, and this is a depiction of how the fireflies are coming in and out. So there is a next level of domain to have there.

Difficulty is that-- there's this word semantic that came up a lot, semantic-- sorry, I'm writing over it, but I'm writing there for a particular reason-- that Eleanor Selfridge-Field uses to imply this level of understanding. That that's the greater meaning of things. So it's a fourth context there. But over time, it's kind of been migrated up in music information retrieval to be closer to the logical domain.

So people will say, oh, I don't want to-- I don't want to know exactly how you're going to draw it on the page. I want to know it's semantically a quarter note C5 or something. And so this term, I'm going to try to avoid too much because it's being used in two very different contexts here. Questions on this or more discussion?

I know we want to get to some coding and get to some other things, but thanks for sticking with all of this. Oh, and there was one thing that came up and a really good answer that made me realize that I definitely didn't say this in class. That she says, most notation formats focus on one of these-- or music representations focus on one of these things. So yeah, definitely the CD focused on sound and the PDF of music focuses on graphics and things like that.

But the one way that that article has aged is that nowadays, we have so many formats that try to do-- encode two or more of these things together. So that you might have MusicXML, which is encoding some pixel placements of notes also and also encoding-- you can do links to the waveform of something. And even going from CD to MP3 in modern formats, yes, it encodes the sound but also encodes some of the metadata, the year it was proposed and things like that. And nowadays, they can even be putting recommendations to other songs or something like that. Cool.

Let's come back because want to make some things. We've been in theory mode for too long. Last class, we ended by doing some algorithmic composition with random walks. Here's a random walk composition that I wrote in 10 minutes once, and I just want to say one or two things on it. Maybe you hate it. Maybe you like it.

[MUSIC PLAYING]

So one of the things I did in my aesthetic on the piece, just to say a few little things on it, is that I gave a large amount of gravity toward whatever note, which is a kind of transpose to make it more dissonant, whatever note something started on. When it went away, it was really, really likely to get pulled back to where it came from. So that's why you'll see just a few things that get perturbed out of way.

And then, as the piece goes on within each section, the gravity got weaker and weaker. And so that kind of let it have-- I don't know-- kind of the opposite of a Lewinian global rule that things got a little bit farther out as it went.

The other thing that I couldn't believe when I first saw, I was like, oh, I think this will be a good idea, write the code or the code, generate put, and it sounded just a mess, totally. Maybe it still sounded like a mess to you. But putting in these staccato marks made all the difference for making it more of a percussive piece. So you can do things like that if you want your pieces to be a little bit more interesting. Yeah.

AUDIENCE: Is there some form of articulation label that you could use to avoid having to staccato every note?

MICHAEL CUTHBERT: You could-- in modern thing you would say-- you dot, dot, dot, dot, and then you probably say staccat sim staccato-- similarly, staccato throughout. Or probably you would just tell a performer, if you're talking to actual humans, all notes smaller than whole notes are staccatissimo or something.

AUDIENCE: Is it possible to implement that in this graphical software or does the MIDI play not recognize?

MICHAEL CUTHBERT: Yeah, it doesn't recognize. When you start talking, you think, ever more expressive as it goes, the MIDI just ignores that, for the most part. I mean, maybe that's the great large language model thing too.

The first thing that takes a Mahler score is MIDI and give it all of his rehearsal marks, the [SPEAKING GERMAN], all that stuff. It's like, oh, I must turn myself into fire and perform this in this way. And yeah, maybe that would be kind of interesting. I hadn't thought of that. Thanks for the idea.

Yeah, so we don't have that type of thing. Besides, when you're doing computation things, one of the things is it's pretty cheap to, in a loop, put staccatos on every single thing. Yeah, great. So anyhow, I want to spend time on that, just a little amuse-bouche before getting to go ahead and open up Jupyter. I'll give you a second or two.

We have a bunch of little things that we haven't gotten to yet that I want to make sure that we can get to. Reminder that the problem set is a Russian nesting doll of unlocking tools, so that please pay very close attention on each of the questions on what tools have been unlocked. But if I'm telling you, well, you now have access to chord, and now you have access to Roman numeral, I should at least introduce these to you.

I can't remember if I introduced chord very briefly, or if we didn't. Maybe-- I don't think we did. I probably need some other things. I'm going to import chord note pitch for now, so go ahead and do that. Did I put the screen down even? It's all good.

OK, so the hardest way that you can create a chord is, let's say, C equals note.Note C. I'm giving you the hardest way, so maybe you don't want to do this because I'm going to give you the easier way in a bit, or if you just get behind. Note G-- oh, I should probably give them octaves, shouldn't I? C4, E4, G4.

And then we'll say ch for chord, "cha." Remember that chord is a bad name for a variable-- chord.pitch-- chord.Chord. Lowercase chord indicates it is a--

AUDIENCE: Module.

MICHAEL CUTHBERT: Module. Uppercase means--

AUDIENCE: Class.

MICHAEL CUTHBERT: Class, good, and so we're getting the object to that. So we could do c-- oops. Apparently there's a button for Google Search-- c, e, g. And do that. The other thing that you can do, and you can actually thank Dmitri Tymoczko for bugging me on, Mike, why can't I just do this? OK.

That's more often how you're going to be doing things when you're coding here. But obviously, when you're doing something algorithmically, quite often you're going to have notes. You can also start with pitches here.

Some of the things to say-- chords, you can iterate over them. For thing in ch, print thing. And you get the notes and that each of the nodes are there. You can change the duration of one note in a chord and not the other, but that doesn't work too, too, too well.

Let's create a second chord. Weird equals chord.Chord-- oh, I don't know-- E3, G5. What are we going to do? Let's put in D7, C-sharp 8. That's a pretty weird chord. I don't know. I'll put in a B9 in there, just because I know exactly what that will put in.

This is the function that you're going to wish you had access to on one of them, the root function. We all get that. This one might not have a name-- commonName. Diminished-minor ninth chord-- is that correct? Weird.show.

Woo. That one went too high. OK, let's-- oh, because it's an E3. No, that actually might be correct because it has to fit it all into one staff when you do this. Maybe instead of E3, let's say E4, 5, 6-- octaves don't matter-- 5, 5, 6, 6. It'll sort it for us.

Yeah, that looks a little bit better. Easier, though, is close_and_weird-- I guess I'll call it-- equals weird.closedPosition. Oops. OK. So who remembers what closed position is from 301 or something like that? Yeah?

AUDIENCE: It's in one octave.

MICHAEL CUTHBERT: Yeah, it's all within one octave. There's actually two different things. Some definitions allow the bass note to be in a different octave, but the top nodes are all as close together as you can get. Yeah, and that can be pretty useful for an analysis type of task.

So we'll choose a different Bach piece today. I'm sorry I'm not keeping up my promise to do a bit more than just Bach. But it does make things go a little bit faster. And this will say, bach equals corpus-- so I'm importing corpus. Sorry. I should be talking around-- corpus bwv269.

This is going to be one of our favorite ones coming up for a bit. I think it's 269. Do I have the right template? Yep, great. [SPEAKING GERMAN]

Great. And so one of the things we can do is we can get-- so what did I call it? Bach-- let's do bach.chordify.show. I think we've encountered this, but we haven't really talked about it very much.

And I'm going to need to zoom out for a second, or you can do this on your own. Or maybe I'll just do this-- bach.measures-- so we remember this. It begins with pickup? Yep-- 0 through 4 .show. Chordify is slow enough so you probably shouldn't call it every single time. bach.chord-- bach... Might as well make the part that's faster go first. bach.measures, 0 to 4, .chordify.show.

You don't have to-- am I going too fast today? Yes, I am? Good. Thank you for having the courage to nod when you do it. So we're going to do the same thing, and I just want to make sure that we can get-- I can't really tell if that's a G, but it does look about an octave below that note, and I do know that that's a G. And so yep.

So it's taking at every moment what's sometimes called a salami slice, which is to say, every time something changes, there is a new chord created. So we can see here that there's nothing that changes here in between here and here. So this is a chord. This is a chord.

But here, this will need to be made into two chords in order to deal with it. And we can see here that this works. It's rather hard to figure out what chords these things are. So what did we just learn? Yeah?

AUDIENCE: What's MXL?

MICHAEL CUTHBERT: It's a MusicXML format that is compressed MusicXML. So it's kind of nice that it's MusicXML all compressed. So MusicXML takes up a lot for notation format and not a lot for video format or something-- not a lot for something in the sound domain, but a lot for something in the logical domain of things.

But it compresses incredibly well. Anybody who works on compression algorithms might find that interesting. So a lot of people compress their MusicXML. Good question. OK, so what's something we just learned that we can do to help make this a little bit easier to read, to figure out what chords are happening? Yeah, John?

AUDIENCE: After [INAUDIBLE] to compress.

MICHAEL CUTHBERT: Yeah. Yep. Did somebody want to-- what was it that we used to compress the chords? We just learned--

AUDIENCE: Closed position.

MICHAEL CUTHBERT: Closed position, good. So let's say chf, or chordified, is bach.-- here, we'll just do a little excerpt-- 0 through 4 .chordify. And I'll make sure that that's the same as before, that I did type it right. Yep.

OK, so how am I going to do this? Actually, before I do that, let's just look at it in text-- show.text. Instrument 3, 4. Yep. And we have all these chords within measure.

So we want to put all of these chords into closed position. What's our favorite three-letter word that we're going to start with?

AUDIENCE: Four.

MICHAEL CUTHBERT: Four. Great. So let's do this. For my_chord in-- what do we call that? We call that chf or chf.-something. Whisper to your neighbor what the answer should be, and then we'll all shout it out.

[SIDE CONVERSATION]

Who votes for the next thing being a colon? No, nobody? Who votes for recurse? Who votes for flatten? Who votes for something else? OK, something else?

AUDIENCE: A bracket in the chord class.

MICHAEL CUTHBERT: Ah, yeah-- bracket on the chord class, which is equivalent to-- anyone remember?

AUDIENCE: Recurse.

MICHAEL CUTHBERT: Recurse and?

AUDIENCE: Get elements...

MICHAEL CUTHBERT: Get elements by class. So yeah, that would be really, really great. The other thing-- by the way, this is a case where flatten isn't going to be too bad either because we don't really care what order we're getting them. We're going to put them all into closed position.

So for my_chord chord in this thing, we can do something like this. We say, closedPosition. Oh, shoot. But that returns a new thing, and we don't have a stream to put it in. A lot of these things that change streams and change chords and stuff like that has an in-place option.

It also has something-- forceOctave, and we'll say forceOctave equals 4. And that's something that's pretty useful. That says, now you're shrinking everything to be within one octave. Let's not have you do your own transpose function or something after that. Let's just put it all into four. So hopefully this works so I don't--

AUDIENCE: Should be running closedPosition on my_chord, not c?

MICHAEL CUTHBERT: Oh, thank you. Thank you. This class is like a running example of how two people coding code faster than two people coding together code more code faster than two people coding separately-- maybe not 17. That's probably too many. But yeah.

So I haven't ever figured out the time to fix the tie algorithm when you do closed position because obviously it's kind of cool that it finds that, oh, there's a tie that was opened, but then this got moved to here or something. So great. So now we have a certain number of chords that are happening here. What key do we think the piece is in now that we can see it more easily?

AUDIENCE: G major.

MICHAEL CUTHBERT: G major, good, good. And so now we can start putting Roman numerals on things. What do we call this in G major?

AUDIENCE: The first...

MICHAEL CUTHBERT: I.

AUDIENCE: I.

AUDIENCE: IV.

AUDIENCE: IV.

MICHAEL CUTHBERT: IV. What--

AUDIENCE: IV6.

MICHAEL CUTHBERT: IV6-- IV in first inversion?

AUDIENCE: VI.

MICHAEL CUTHBERT: VI, yep.

AUDIENCE: V.

AUDIENCE: VI-- or first inversion.

MICHAEL CUTHBERT: V first inversion. Now, then there's certain things that we have to be like, ooh, but was that-- we go all the way back up to here-- no. OK, good. What's hard to tell is if that was a passing tone or if that was-- which one of these two was the passing tone in that case-- can you see my pointer? Yeah-- which one of those two was the passing tone there.

Great. So we can go there, and we can do this here. We'll get to how we can have the computer do all of this for us in just a bit. Questions or anything? I should go a little bit more slowly. Yeah, John?

AUDIENCE: If we force octave to be four, why do we have some notes in the fifth octave?

MICHAEL CUTHBERT: Why do we have some notes in the fifth octave? What it says is that forced the lowest note to be an octave 4, and so since it can go up to an octave-- yeah, so that's the case where it's the highest here. It does mean, if you do compositions where you're putting chords in closed positions and jumping them around, it can sound very jumpy.

So that's one example here. Great. Anything else? I should have been going with this. Let me-- OK. Let's dir the chord class for a second-- chord.Chord. Oops. Actually, I should be just diring one of the chords.

So some of the things are things that you've already seen because chords are music21 objects. Obviously, they have a duration. You know how to deal with it. They don't have a .pitch attribute. They have a .pitches, which gives you all the pitches there.

But there's some kind of things-- canBeTonic. Can this chord be a tonic in some key? What chords can be tonics? What kinds of chords? What equivalence class of chords?

AUDIENCE: Major chords.

MICHAEL CUTHBERT: Major chords and--

AUDIENCE: Minor.

MICHAEL CUTHBERT: Minor chords, great. Super. So that's some of the things. Does the chord contain a seventh? I think weird does. .containsSeventh-- True. But do we still have that first one? Yep. ch.containsSeventh. So you'll find that there's all these slightly useful things in here.

I don't know what C-- is it an augmented triad? Is it an augmented sixth? Is it a French augmented sixth? It's a half-diminished augmented sixth? A lot of these things here-- is it transpositionally symmetrical? These things come up quite a bit.

And you can put the same pitch more than once into a chord. So sometimes you want to remove the redundant pitches, redundant pitch classes, and so on. Sometimes you might have weird chords that you want to respell and so on. So these are some of the things that are of useful in here.

And if you want to talk about ontology errors here, ch.third gives you the third. ch.fifth gives you the fifth. And ch.base gives you a bound method of something. So that's something I wish-- I wish I could figure out how to change that these were done differently early on, and it's a little too late sometimes to change things, just like it's a little too late to change how certain words in English are spelled.

Cool. So that's the end of the introduction to chords, unless people want to do something with this.

AUDIENCE: Roman numerals?

MICHAEL CUTHBERT: What's that?

AUDIENCE: Roman numerals.

MICHAEL CUTHBERT: Roman numerals-- we'll get to Roman numerals in just a second. So great. So we're definitely going to get to Roman numerals. But first, a small leftover from our work on cognition and keys and the probe tone method by-- anyone know, if you know the author?

AUDIENCE: Carol--

MICHAEL CUTHBERT: What?

AUDIENCE: Carol Krumhansl.

MICHAEL CUTHBERT: Carol Krumhansl. Great! well done. Yep. Who was one of the people who referenced this. Yeah, the probe tone message by Krumhansl, Carol Krumhansl. And so what we're going to do is I'm going to get us to two pieces here.

You don't have to do this part if you don't-- actually, at this point in the semester, you don't have to type anything that I'm typing unless it's helpful for you. Maybe in the first part, it's good. So this is a particular little jig. We're going to be working with a lot of jigs in the next time that we do some coding together. So that's a nice little 9/8 jig, a little bit unusual. What key does it look like it's in?

AUDIENCE: D major.

MICHAEL CUTHBERT: Yeah. You always have to-- you can let people give us a chance to do it. Yep, D major here. And then we'll get a piece by the 20th century composer Arnold Schoenberg, Schoenberg opus. S-C-H-O-E-N-B-E-R-G slash O-P-U-S 19/movement6-- all one word.

I know it's a little bit hard. I wish I had another title that we can do. It's a little piano piece. Great. And Adam, what key is it in?

AUDIENCE: No one knows.

MICHAEL CUTHBERT: OK. Yeah, so this is a rather ambiguous piece. Just so you can get a tiny bit in your ear, just the first couple chords.

[PIANO CHORDS]

So something wrong on-- there's a cleft change missing.

[PIANO MELODY]

Yeah, I know there's something a little bit wrong in this encoding. Anyhow, a little bit ambiguous piece. So what I want to do is let's get into unlocking some key analysis things.

So what we can do is-- we called it a jig, right? Yep. k1-- our first key will be jig.analyze key. So that's pretty good.

Now, in addition to analyzing key, there's specific-- a number of people have implemented their own analysis things. So you'll have to get this-- so we're going to run through a bunch of different methods-- the standard key method and Simple. So key is something I put a guarantee to people using music21 that that will be whatever I think is giving the best results. And I'll try to keep improving it and get that.

Krumhansl is exactly Carol Krumhansl's probe tone method with the exact same things, which was learned-- we learned the different weights, the different amounts, how important the tonic is, how important the third is, how important the fifth is in the key. How did she learn that?

AUDIENCE: ProTone.

MICHAEL CUTHBERT: ProTone. So she was asking people, just getting people's reaction to how far and out. So it's based on cognition. The Essen one was based on labeled keys for the entire Essen folk song repertory, which we've used a couple times on things, and it's come up in this class quite a bit.

And so people use that database on the ones that were labeled in the key to learn not from people, but from what number of notes were in a piece that was labeled as in F, what's the number of things were labeled as in E? And all of these weights, by the way, have these weird numbers, like the tonic 6.238-- don't make it too much like pi, 2 pi or something-- so these really long, complex numbers on things like that.

Where Craig Sapp, who has come up in this class a couple of times for making the Rosetta Stone and things-- I think it was Craig Sapp who came up with a very simple-- he's like, no, we can just use 1, 0, 1, 0 1, 0.5. And I think everything in his weighting is either 0 or 0.5. And he's like, it's much easier to remember, much easier to program, and it works, he thought, even better. So we have all these in here.

So we'll do this here. You don't need to know Python f-string formats, but I'm going to give 20 letters for the method. And then we'll analyze using that method here. Actually, we'll just do this-- key_out. I'll make this a little bit easier to read. And so we'll call key_out equals jig.analyze using the method. Oops.

AUDIENCE: You put method in a string.

MICHAEL CUTHBERT: Method-- thank you. I put method in a string, a string of Methodists. So when you put the jig into there, everybody agrees it's D major. That makes pretty good sense, right? You would not trust any algorithm that came up with a result other than D major for this entire piece. This is the whole piece.

But let's see what happens when we give it Schoenberg. So I'm going to just copy and paste, the copy and paste method for coding quickly in class. And did I call it sch, I think? And here, we end up with quite a bit of distinctions. Maybe E minor and A minor aren't too far apart, but we even have a major key in there.

So depending on how you do your weighting, different things could be better. There's always the goal of trying to come up with a better key analysis routine. And the other main unsolved task in this domain is, pieces aren't always just in one key.

And I'm not talking about Charles Ives with multi-tonality and things. I'm talking about pieces change keys throughout the piece. And sometimes it's easy. They change the key signature at that moment. But quite often, pieces change keys without giving you a key signature.

Can you have a method that says, well, it begins in E, and then it goes to A, then it goes to C major, and then it goes some other place? Can you detect where all those key changes are? So that's something that-- people have worked on it a little bit, but it's still an unsolved general problem.

Any questions on these types of things? By the way, you can-- there's a bunch of other things you can use on analyzing. You can get the pitch range and stuff like that. But that works.

Oh, and I'll just say, on this last one, key_out.alternative-- what is it? I can never-- alternateInterpretations interpretations gives you a list of-- so this is one, C major. It gives you a list from the next most likely to by far the least likely. In fact, let's do that on the jig. We'll run the jig one more time.

What did we call it, k1? Yeah. So k1 is the jig's key, and k1.alternateInterpretations. Yep. So if it's not in D major, it might be in B minor. Yeah, it might be, depending on what the second section-- it's very unlikely to be in A-flat major.

Great. With the last 10 minutes-- but I'm always happy to slow down-- questions, come back, talk. No? OK, sure. Great. So let's get what we all want, which is Roman numerals. So go ahead and, from music21, import roman.

And I'm trying to find where I have this part broken up. Here we go. OK, within roman, we can say rn equals roman.RomanNumeral. What are the two parts of a Roman numeral the computer is going to need in order to figure out what pitches there are? Sorry, I--

AUDIENCE: [INAUDIBLE]

MICHAEL CUTHBERT: Yep?

AUDIENCE: [INAUDIBLE]

MICHAEL CUTHBERT: Yep. We'll call that the figure, the string representing things like I, V things and the key. So what we can do is, if you had a key object-- I think we still have one, k1-- we can pass that in and get this IV in D major. The other thing-- what we'll often do when we're coding in class is I'll just give it a capital D for D major. Lowercase d would be D minor. And that works just as well.

If you're actually writing a program, unfortunately one of the least-optimized things is creating a new key object. It's about an order of magnitude slower than it should be in an idealized world. So it is helpful to reuse your key object if you're going to keep using it over and over again.

OK, so we have Roman numeral IV in D major. Somebody this side of the room, what pitches are we going to be expecting with Roman numeral IV in D major? Yep, go ahead.

AUDIENCE: G, B, D.

MICHAEL CUTHBERT: G, B, D. Were you going to that?

AUDIENCE: Yeah.

MICHAEL CUTHBERT: Good, good. So what we can do is G, B, D. And what octave should they be in? Do that more gesturally so that we can get that. You went--

AUDIENCE: It doesn't matter.

MICHAEL CUTHBERT: It doesn't matter. It doesn't matter, right? So there's this weird thing that happens that, when we want to put a pitch onto a staff or something, we need to give it an octave. We need to reify it at that moment. But it doesn't matter.

So here's the first question on this is, what kinds of-- when we say that we have the same Roman numeral here as the same Roman numeral here, what kinds of equivalence classes are we talking about if I say that Roman numeral IV equals Roman numeral IV? I want to throw some equivalence classes on pitches or notes. Somebody throw out an easy one because we just said one.

AUDIENCE: Octave.

MICHAEL CUTHBERT: Octave, good. So that's definitely one. Yeah, go ahead.

AUDIENCE: Permutation.

MICHAEL CUTHBERT: Permutation. Yeah, it doesn't really matter what order they are. I'll come back with octave on just a little bit because there's a caveat, but later. So everyone should try to figure out the caveat on that. What else?

AUDIENCE: Cardinality.

MICHAEL CUTHBERT: Cardinality, right? I mean, is-- I almost wrote on that. Yeah, because this is I in C major, but so is this, and so is this, and so is this. It doesn't really matter how many notes we have. Good. Now, when we're just talking about a Roman numeral like IV, what's the other thing that we're in there for? What do you think about? Yeah?

AUDIENCE: Whether it's major or minor.

MICHAEL CUTHBERT: Whether it's major or minor, yep. So we have something on-- let's try to think some of these things. And do we have the same blanket octave equivalence, the one that we've been thinking of that you've already coded? What's my caveat on octave? Yeah?

AUDIENCE: I would think the second chord inversion.

MICHAEL CUTHBERT: Chord inversion. Yeah, we have this different kind of inversion where this is I, and this is I first inversion. What is it? Ib if you're British, I6 if you're American. Yep.

Yeah, so we have this unusual thing, which is very usual for us as musicians or as people who have taken traditional class, but a little bit different for-- well, the octave of this doesn't matter in itself. We can move this up to-- sorry, this is all over in the corner-- G, E, G, C, and it's still I6. So there's something about the octave that doesn't matter.

But we don't have, we might say, a bass pitch equivalence. So we're always going to be creating new equivalence classes on things. And so there's a very unusual equivalence class. We also have a kind of limited transposition equivalence class that I can write C, E, G and say it's I.

But I can also write D, F-sharp, A and call it I, as long as I say that I is I in C major, and I is I in D major, so that there's this abstraction out of-- that transposition, it's measured according to distance from what note? Not a note like C or something, a conceptual note.

AUDIENCE: Tonic.

MICHAEL CUTHBERT: Tonic, yeah. So we want to have things that are the difference from the tonic. So I think this might actually be different on yours than on mine because I'm running a beta version that I haven't still haven't got the bugs out. So you can see, Roman numeral comes from some abstract class called Harmony, which also represents chord symbols, which I will not get to today.

And it comes from Chord. I think you're missing ChordBase probably, for most people. Or maybe-- no, it's there? It's there? OK, great. I couldn't remember what version got that in. Great.

So with our Roman numeral IV, we can get back, its figure is IV. Well, that's not telling you too much. Let's get another one. We'll call it rn1 equals roman.RomanNumeral flat II6 in C major. What's another word for flat II6?

Anyone remember-- flat major II6? Don't worry if you don't know this because this is something that some people may know, but it's the next semester beyond the prereq. Yep?

AUDIENCE: [INAUDIBLE] augmented sixth chords.

MICHAEL CUTHBERT: It's traditionally taught the same week as augmented sixth chord, so everybody-- or the week before it. That's the type of stuff-- it's named after an emperor-- or no it's not. It's named after a city-- a Neapolitan chord. So it's named-- not Napoleon chord, sorry-- named after the city of Naples. Very, very cool chord. If we had two more minutes, I would play it. So here's something here. It has a VI at the end, so what do we call it? What's the inversion?

AUDIENCE: First.

MICHAEL CUTHBERT: First inversion. So you can call inversion and put it as I. You can see it's a little bit strange that it has-- inversion takes some time to calculate on very complex chords. So it's method. But we can get that its romanNumeralAlone-- this is a little bit wordy. I should have come up with another thing-- is II and its frontAlterationAccidental-- woo-- is a flat. So it's a II with a flat in front of it.

So one of the things we can do, just looking at the last moment, what you'll probably be wanting is roman.romanNumeralFromChord. So given a chord and a key, what's the Roman numeral? So we still have weird-- no, let's start with ch in C major. My CEG is still there. Great.

What would that be in-- here, let's just do this-- for s in ABCDEFG? We'll print this. Print the Roman numeral in the-- I'm using s for step. Does that makes sense?

Some odd, odd chords in there. But you can figure out these types of things here. And so we can even do roman.RomanNumeralFromChord. I almost always, when I'm using this a lot, abbreviate this. So rnfc equals that, so I can just do that. romanNumeralFromChord, weird.

Oops. I should give it a key. There we go. Sharp I, half-diminished flat VII, VI, flat V, flat III. So music21 will try to find all these things for you.

I've kept you over, so I'm not going to show you how you can run this on every chord in a Bach piece, put it as the dot lyric, and then reshow the piece. We'll leave that as an exercise for take home and might come into a class later. So thanks, everybody.