[SQUEAKING]

[RUSTLING]

[CLICKING]

MICHAEL CUTHBERT: You'll remember-- probably it's all gone because it's like, what, a week ago-- p set four was mostly about the topic of equivalence classes, equivalence in music. What were some of the things that we were talking about back then, because this is just going to keep coming up all semester? What are some things that we could say, in some contexts, that two notes or two pitches are different, but in another context, we might say they're equivalent to each other? Yeah?

AUDIENCE: Octaves.

MICHAEL CUTHBERT: Octaves, great. Let's just shoot out.

AUDIENCE: Permutation.

MICHAEL CUTHBERT: Permutation.

AUDIENCE: Cardinality.

MICHAEL CUTHBERT: Cardinality.

AUDIENCE: Transposition.

MICHAEL CUTHBERT: Transposition.

AUDIENCE: Inversion.

MICHAEL CUTHBERT: Inversion. And the one that's not part of the optic?

AUDIENCE: Spelling.

MICHAEL CUTHBERT: Spelling, good. Just while I put some things on the board, shout out some more things we talked about in class that we didn't put on the problem set. What were some other things that we say things are the same, even though they might sound different?

AUDIENCE: Instrumentation?

MICHAEL CUTHBERT: Interpretation, yeah. We say, hey, that note's a C. And it's like, no, that note was on a piano and this one was on a trumpet. Good, other things?

AUDIENCE: Inharmonics?

MICHAEL CUTHBERT: Inharmonics, yep, that goes as a special class of spelling. It's good to use that term. Anything else? So I said, it doesn't matter how long the note is, it's still a C, right? Yeah, so duration, when we're talking about notes.

Good. I can't, apparently, lead a discussion and write things on the board. But I want to talk about two things that came up a lot. One of them, I don't need notation for and one of them I do. One of them is much better.

B sharp 4 to-- I don't know, it doesn't really matter, but it's F sharp 5. You were asked to write a program to automatically generate the interval. Sorry, you were asked to write a program to automatically generate the interval of anything.

This was one of my hidden test cases, that I think it's the only problem that over half, maybe 3/4 of the class, the problem came up with. Which one of these two notes is the problematic one? Take a guess.

AUDIENCE: B sharp?

MICHAEL CUTHBERT: B sharp, good. Now, B sharp is a problematic note the way E sharp is. Right? But it also has its own special problematic things. What's B sharp for? What note does that sound like?

AUDIENCE: C 5.

MICHAEL CUTHBERT: C 5, good. The issue that came up a lot is the only thing. What should this interval be? Let's think about that first. What should the interval be? Yeah?

AUDIENCE: Diminished fifth.

MICHAEL CUTHBERT: Diminished fifth. Let's think about the diminished fifth. Good. What would be the interval if it was C 5 to F sharp 5? Anybody?

AUDIENCE: Augmented.

MICHAEL CUTHBERT: Augmented?

AUDIENCE: Fourth.

MICHAEL CUTHBERT: Fourth, good. The 4 and the 5 are changed around, based on the number of distances in the step spacing, which you all did great on. But now, we have a different problem. And so, what came up a lot from various things, I saw a lot of--

[WRITES MULTIPLE D'S ON BOARD]

--is that the right number? 12th, I think that might not be enough Ds. Because the issue is that the B sharp is something that sounds in kind of octave 5, but it's not. So always, when you're thinking of how's your screwy prof going to try to screw you up, think about these things that cross over the boundaries between octaves, or sound like one thing, not the other.

A lot of times, what happened was you needed to do something mod 12 at a certain point, add an offset depending on how your algorithm works, and then, mod 12 it again, so this sort of double mod 12. At this point, we're going to be working for algorithms that get things right. We can work on efficient algorithms, more efficient ones, as kind of a project later.

Good, the second one that came up had to do with asking you to-- this was done by hand-- to come up with a pitch set that was equivalent to the one I gave you, using p permutation and t transposition, but without octave inversion or cardinality equivalence. And most of you are like, phew, I don't even want to think about inversional equivalence. So that's good. cardinality equivalence just means that I gave you three notes. How many notes should you have?

AUDIENCE: Three.

MICHAEL CUTHBERT: Three, good. The hard part, this was A 4, E 5, C sharp 4. What I'd like you to do, when you have a problem like this, is go back to do it the old-fashioned way first. Get out staff paper or use Finale music score or whatever you're using, and try to make sure that you're seeing what the notes are. So where's A4 go?

AUDIENCE: [INAUDIBLE]

MICHAEL CUTHBERT: Stop?

AUDIENCE: Two. Stop.

MICHAEL CUTHBERT: Stop, OK, good A4. What's the next one? E5.

AUDIENCE: [INAUDIBLE]

MICHAEL CUTHBERT: Stop, OK, good. Go slower, prof, right? And then, what's the next one? C sharp 4.

AUDIENCE: Stop.

MICHAEL CUTHBERT: Stop. What's the common name we would call to this, if this were a chord?

AUDIENCE: A major.

MICHAEL CUTHBERT: A major--

AUDIENCE: First inversion.

MICHAEL CUTHBERT: --in first inversion, yeah. Oops, I got to jostle this, otherwise this is blood red. Yes, first inversion. OK, so then, what you need to do-- you want to do is something that-- and it had to use both.

Now, I think I tried to say it in such a way that even an MIT student wouldn't be allowed to say, well, you know, not rearranging anything is a permutation and transposition by perfect unison is a transposition. But you needed to change them in some way. So one of the things you could do-- one of the things that happened a lot was that the octaves got changed.

And so, that ended up being about 40% of the errors that we saw. What you can do is, you could first re-permute things. Probably, the re-permutation that I would do first is, let's just put it in ascending order.

That's probably the permutation you'd like to have your computer do, anyhow. And then, let's just shift everything up. One way you can do it-- well, in this one, I was going to say you shift everything up a space or a line, but the E is the problem. So let's just transpose up a major second. What are we going to get here?

AUDIENCE: D sharp.

MICHAEL CUTHBERT: D sharp, B--

AUDIENCE: F sharp.

MICHAEL CUTHBERT: F sharp, good, not because-- well, no, this one, we had implicitly, because everything on this we had spelling-- that's an F, not a 5-- spelling equivalents. Yeah, so D sharp 4, B 4 and F sharp 5. When you get that back, that's there. Any other questions on problem set four? Hardest programming problem set so far?

AUDIENCE: No.

MICHAEL CUTHBERT: No?

AUDIENCE: Two was.

MICHAEL CUTHBERT: Two was harder for some people. Any would vote for one? No? OK. If one was the hardest, probably you're not here now. Sorry, I shouldn't-- yeah.

Problem set six, we'll be announcing today. It is also a slight programming one. It's designed to give you a little bit of a break. I'll talk about it in a bit. But it's more of the only participation grade type of problem set than trying to get the exact results. We're going to have a little bit of a break coming up for things.

Somebody want to volunteer? What electronic corpora did you find online? Adam?

AUDIENCE: Yeah, we used Musipedia.

MICHAEL CUTHBERT: Musipedia, great. What is that?

AUDIENCE: It's a dual database and search query data corpus.

MICHAEL CUTHBERT: Cool, great. Another group? Yeah?

AUDIENCE: We did Ultimate Guitar.

MICHAEL CUTHBERT: Ultimate Guitar, great. How's that encoded?

AUDIENCE: I mean, they started off with just like, text, ASCII guitar tab, but then, they expanded and incorporated more advanced software.

MICHAEL CUTHBERT: Oh, cool. Yeah, I remember when it was a lot of ASCII guitar tabs. I think everybody had their own parser for Ultimate Guitar at some point. Good, anybody else? Vanessa?

AUDIENCE: We did the Video Game Music Database.

MICHAEL CUTHBERT: Video Game Music Database, fantastic. Great repository. What were things mostly encoded for that?

AUDIENCE: Almost everything was in MIDI files.

MICHAEL CUTHBERT: Almost everything was in MIDI files, why do you think that was?

AUDIENCE: I think it's like, generally a standard and it's relatively small file sizes, so it's easier to mask out a whole bunch of different files.

MICHAEL CUTHBERT: It is. Also, that's probably how they were encoded in a lot of-- especially the early video games-- that were encoded, so when you're ripping from or something like that, but definitely not encoding sharps and flats. Good. One last one, Sruthi, your group?

AUDIENCE: Oh, we did Jazzomat. It was a jazz database.

MICHAEL CUTHBERT: A jazz database, fantastic. What were the encodings for that?

AUDIENCE: They had a variety. They had like, the transcribed sheet music for a lot of jazz solos, and then also a CSV file that just had every notes offset from the very beginning and its pitch as a number.

MICHAEL CUTHBERT: Cool. What you'll find is a lot of these data sets, they really have their own one-off, make your own format. I don't think that one's a standard, but I've seen lots of these, because that's how things tend to go.

Great, we're going to get to right back to representation in just a bit. We have special guest, Michael Good, who knows a little something about representations, being the creator of MusicXML, also from the dean's office. If you guys can stick around for 10 minutes, I'd like to-- before we do that-- let's get warmed up for probably, the thing that are on some people's mind, the downgraded midterm, which is now, I'm calling it a quiz, because I'd like to divide this into two smaller, lower stress things.

That will be on Wednesday. We'll start recording about 30 or 40 minutes in. I'm designing it to take 30 minutes. If we take a little bit longer, that's OK. It's not all about the rush.

But we will do one type of thing that is about recall. The quiz will be not about your programming skills, but about the reading. I want the quizzes to be more about the readings, so that you keep thinking about ideas that you're going to be doing in your projects.

As we keep going with further problem sets, each one of them will start having a little bit of what do we call it, assign yourself your own problem and start it. OK, we got some sound going. We don't really need that, but we can turn that down for a bit. Who has done Kahoot before?

OK, who knew that MIT, apparently, let their Kahoot subscription lapse sometime between the last time I'm in here? I'm only able to do 10 players. Can you all pair up and do in teams with the other people?

I know. I was trying to fill out the subscription form right behind class. Can you team up one, two, three-- three and three-- four, five, six, seven, eight? Actually, you can be your own team nine. Or do you want to do it with Michael?

OK, we might have one more thing to do it. Oh, yeah, use any nickname you want. Just make it family safe.

[STUDENTS CHATTER]

Oh, you can decide if you want to be--

AUDIENCE: How confident are you?

AUDIENCE: I'm not confident.

MICHAEL CUTHBERT: OK. This is just to judge how much time you feel like you might need to spend reviewing the things. OK, we have one, two, three, four. We have nickname, nickname two, great, four, five, six. OK, we'll do a little bit of the background music, but although, maybe--

[JAUNTY MUSIC]

We have eight. Anybody waiting on? Two more people want to be loners, go on their own?

OK, no. That's good. We're going to get started. Oh, we got winners. OK.

AUDIENCE: Ooh.

MICHAEL CUTHBERT: OK, great, super. Here we go. So about 20 or 30 seconds on each one. There's 11 questions, I think, 13 questions maybe.

[GONG]

Which of the following is a binary music representation? This is your time, if you haven't done this before. Lock in something. No points for wrong.

3, 2. What do we have? MIDI. MIDI is the only one that is typically represented in binary code and not in text. OK, I know. You would have thought I would have made the answer MusicXML to honor our guest, but we didn't.

AUDIENCE: We knew it wasn't MusicXML.

MICHAEL CUTHBERT: You knew it wasn't MusicXML, because you studied that well enough. Good, let's keep going. Creative nicknames are up top. According to Cook, what is ironic about the timing of the rise and fall of comparative and computational musicology?

[DIGITAL MUSIC]

Mostly, we got it. Comparative musicology, which is kind of designed well for computers, declined at the time computers came up.

[GONG]

That's Nicholas Cook's Comparative and Computational Musicology to review, if that one wasn't obvious. We'll keep going. We got much more on the board.

Who was not a composer whose music was discussed by Cook, who was not in there? This is a hard one. This is about as hard as it'll get.

[STIRRING MUSIC]

Three seconds. Oh, most people got it, Wagner. No, Ligeti was the one with all the graphs showing when things come in and out. And the Haydn one, Haydn was used as the example for the German nationalist song. He was the composer of the music, but not the lyrics.

OK, we got one fire. The music21 iterator-- talking about my own stuff-- that returns offsets from the start of the Part is--

[DIGITAL MUSIC]

You know. You too. Got one more person. Two seconds. Hey, almost everybody got flattened. I know.

You know that recurse-- somebody was trying to do the thought experiment that I recurse better. I keep telling you, use recurse. But in this case, flatten is the better one.

Whoops. OK. Who wrote about musical codes in the introduction to *Beyond MIDI?*

[LAUGHTER]

[BOUNCY MUSIC]

Oh, very nicely done. Nicely done. Michael Good does have an article in that book. But no, you don't know. That was an excellent virtual score.

MICHAEL GOOD: What I wrote before starting.

MICHAEL CUTHBERT: OK. Hey, getting close.

[STUDENTS CHATTER]

Yeah, Tymoczko criticized me for all of these things about music21 except what? This is hard.

[STACCATO MUSIC]

No, the one thing he didn't criticize me for is the worst feature on it. It is really hard to install. But somehow, he already got it installed, so he didn't care about that. OK, so that was one that shook things up, the snowman.

Here we go. We're over halfway there. This is an extra credit from an optional reading, so don't worry. But take a guess on this one. Donald Byrd created his *Extremes of Music Notation* thing to show that notation what?

AUDIENCE: It's got to be that one. It's got to be red. It's got to be red.

MICHAEL CUTHBERT: Yeah, good. We don't need to discuss that, even if you didn't do the reading. It's a really cool article if you haven't had a chance to look through it. Some weird things in there. Craig Sapp's name-- good guy-- appeared in this class for his creation of what?

OK, what did we get? Yes, the illustration of the same passage of music in dozens of encodings-- also known, allowing you to translate from one encoding to another, called the-- Rosetta Stone. He now runs the Humdrum thing. But David Huron here, nobody did this one and yeah. I don't know the answer to the red.

AUDIENCE: Does the yellow exist?

MICHAEL CUTHBERT: Yes, I believe so, somewhere. This is from the reading for today. We'll see if we got today's reading. What does David Lewin call his unifying rule for Species Counterpoint?

[SUSPENSEFUL MUSIC]

A lot of people, global rule. Those who said golden rule, by the way, I am going to misspeak and call it the golden rule a lot because I do that. We'll talk about it a little bit more at the end of class today. common or conventional music notation, also CWMN, consists of what?

[SUSPENSEFUL MUSIC]

Seven answers. Yeah, I know. All of the above is almost always the right answer when you take a guess. But no, in this case, it was just things that traditional western music notation, in some ways, so that you can write down non-western.

You can encode-- as you all know, you can encode many things into the computer that are not that. So great, we're doing well. Now, OK, three more.

AUDIENCE: Oh, my god.

MICHAEL CUTHBERT: Oh, wow, nice. We don't need to do that anymore.

[DIGITAL MUSIC]

[GONG]

Aarden and Huron showed how European Folksong varied across genres, times, locations, functions?

[JAUNTY MUSIC]

Hey, yeah, almost everyone's got that. This was the article that showed the geographic information systems back when that was very hard to do. We should be able to do this more easily with Google Maps and things.

OK, before we do the last one-- before we do the last one-- just to note, the quiz will not be a bunch of these little factoid things. That's just all I can figure out to do on Kahoot. It will be, actually, about conceptual issues and ability to apply these ideas. Let's finish this. Oh, it's about Sophia Sun and me showed connections between music and emotion using what types of musical scores?

[DIGITAL MUSIC]

And your answer? Oh, no. I know. I do too many Bach chorales and I do medieval music, but it was lead sheets. Oh, yeah, you read it. Great, so we have on the podium, third place, Nickname3.

Anybody want to claim their podium? Nicely done. Nickname2, yep. And on top--

[LAUGHTER]

I guess this shows that some people spend their time not thinking of creative names, but just reading the articles. Congrats, everybody.

So we've been talking a lot about encodings. And what you'll be doing over the course of the semester, sometimes, is working with these strange encodings. And maybe, as we're starting to get-- well, let me say.

You've had some problem sets where I've given you questions and you've given answers. Right? And now, you've given two problem sets, that encoding one and this corpora one, where you've been doing the research to find interesting problems, but I haven't been asking you to actually get an answer. You didn't need to make a corpus, a parser for guitar, Ultimate Guitar, in order to answer any questions.

We're getting to the end of the semester. We'll be bringing those two things, colliding together and doing more on how you can come up with your own problems and come up with your own solutions. Michael, would you like to come up and talk for just a little bit? I'd like to introduce Michael Good, who's an MIT grad and the inventor of MusicXML up here, who's visiting.

[APPLAUSE]

In here. You all have seen an interview with Michael, right? So just wanted to talk, the interview's a couple of years old now. I think we did it during COVID. Yeah, so what's been up and interesting in the world of anything with computer music, but especially music theory and coding, anything that you've been finding?

MICHAEL GOOD: I retired in January, so my mind has gone out of the music notation area.

MICHAEL CUTHBERT: I'll just stand close to the mike.

MICHAEL GOOD: I mean, some of the things that were in these projects were really cool. The body percussion encoding stuff, and it's like oh, well, that's pretty great.

MICHAEL CUTHBERT: Do you feel like now, you've got to come out of retirement and find a way to put all these things into MusicXML?

MICHAEL GOOD: Not yet.

MICHAEL CUTHBERT: OK.

AUDIENCE: Maybe in the future.

MICHAEL CUTHBERT: But what are some ways that somebody could get a particular musical project, if they're interested in non-western music or something else, and get such a standard musical encoding as you've created? What kind of advice would you give for that?

MICHAEL GOOD: Well, to develop a standard for a music representation, you need to have multiple people who are using it, right? And you need to have, I think, tools that support it. So that becomes-- if it's non-standard-- that becomes more of a less common repertoire, that becomes more of a chicken and an egg problem.

So look for something that has at least two applications that are doing that type of music, and see if you can do. Maybe, one's an application you wrote and one's an application somebody else wrote. Try to put something in between.

When I was starting MusicXML, a lot of people had tried to do a standard common western music notation format. I knew that if it didn't work with either Finale or Sibelius-- because those are the two programs people cared about then-- if it didn't work with at least one of those two programs, I was just going to give up, go home and find something else to do, because nobody would care about MusicXML if it didn't work with one of those programs. Nowadays, it would be MuseScore or Dorico or things like that.

If you're trying to do a standard music representation, then you need to have a community of people who are using it and applications that use it. Maybe, more interesting is just at this point, is to come up with cool things that could be represented, and do some analysis there, and try to figure out ways that these representations could be useful for the musicians who are doing these type of repertoires.

MICHAEL CUTHBERT: Michael, you did music when you were at MIT?

MICHAEL GOOD: Oh, yeah.

MICHAEL CUTHBERT: What kinds of things did you do?

MICHAEL GOOD: Well, I was a trumpet player. Nowadays, I sing. My wife and I met in the MIT Concert Band.

MICHAEL CUTHBERT: She's an electrical engineer, right?

MICHAEL GOOD: Yeah, she's one of the 40 top engineers at Analog Devices. She just was talking with Ray Stata at her company conference a couple days ago.

She's 6' 1". I'm 6' 3". I'm class of '79. She's class of '82. She played piccolo. I played trumpet. I played trumpet in the band, the orchestra, the Festival Jazz Ensemble, Chamber Music Society, musical theater, anything that needed trumpet.

MICHAEL CUTHBERT: Laptop ensemble?

MICHAEL GOOD: I played it. There was no--

MICHAEL CUTHBERT: There was no laptop.

MICHAEL GOOD: Laptops were not invented. The coolest thing, I did my thesis in the experimental music studio with Professor Barry Vercoe--

MICHAEL CUTHBERT: Which is now the Media Lab.

MICHAEL GOOD: --one of the groups that became part of the Media Lab. And the coolest thing was, we got to use VT 52 terminals instead of-- video terminals-- instead of paper terminals that I was using for all my other classes. I got to use the coolest computer technology that I had seen, at that point, in my computer music class. That was awesome.

MICHAEL CUTHBERT: Michael, here's a question I haven't-- I don't think I've ever asked you. You did music while you were at MIT and you did other things. And then, music wasn't your primary putting money in the bank for a bit.

You could ask a little bit about what that was. But what made you want to come back and do it? And is there anything for anybody who's not thinking of graduating this year or next year or whatever and doing music, but might follow a career like yours?

MICHAEL GOOD: Well, yeah, following a career like mine is kind of-- I mean, it's a whole different time now. When I was doing my bachelor's thesis at the experimental music studio, it was on one of these alphanumeric encodings, which is now called SCOT. It's actually still shipped, I think, as part of Csound. I was rewriting it from an earlier version, called SOGO.

But this was in 1979, so there was no MIDI, right? There was no computer music technology industry. OK, New England Digital was doing some claviers, but that was one tiny company.

There wasn't any industry to go to for a job, so I started to do usability work for my grad school. I got my master's here too. I did usability work until 2000, when you could see that everything was going to-- this is when the web was coming up-- and everything was going to be move this app over to the web from a GUI.

The first time I did that, it was a lot of fun. But doing it 1,000 times was going to be really boring, so I looked at-- there was an opportunity to do a standard music notation format that was downloadable sheet music starting to happen. It was a nascent thing.

There was, obviously, a demand for a standard computer music, because it was silly. You could download the music and all you could do was use it in the player or print it out. And it's like, you should be able to share it back and forth between programs like HTML and MIDI.

If you don't have a standard, it's nothing for that type of common thing. That was the motivation to do it, that I wanted to use downloadable music in any application. But I had the financial ability to do that, because my wife works for Analog Devices as an electrical engineer. And their company did really well.

And so, if I totally failed, it didn't matter, financially. So I mean, I was in a extremely lucky position there. Nowadays, though, you got each other all interested in music technology. There's all sorts-- I mean, to go through-- if I'd had a class like this, I wouldn't have had a class like this, because you didn't. I had to spend all my time like, building infrastructure.

Now, you can do the cool things on top of it, the cool problems on top of it. So find cool problems that people are interested in if you want to make a career. If you're just studying, find a cool problem that you're interested in. Well that's, I guess, the starting point anyway.

Find a cool problem that you're interested in, because if you're not interested in it, you won't be able to make a living at it. You'll just be terribly bored. And then, find if somebody else is interested in that or something adjacent.

MICHAEL CUTHBERT: One last question before I let you go, if that's OK. It's hard for us to think of that you went from the essentially teletype era, you don't see things on screen. You have to type it onto it and then, it all comes out on paper and stuff and what, 30 bytes per second modems and things like that? Until two months ago, when you retired, you were still totally on top of 99%-- as many percent of the new technologies as anybody can do and things.

Someday, it's hard for us to think, but these will seem as outrageous as a teletype or a horse and buggy and stuff. Not just for music people, but how do we keep up? How do we keep up, like you. With all the changing technology?

MICHAEL GOOD: If you're a tech person, I think you kind of do that. I live in Silicon Valley. You got to stay with what's going. I mean, I had this usability background, so I was always interested in user interface technologies.

I don't know anything about machine learning, outside of applications for it. In the music space, I'm excited to see what Soundslice is doing for using machine learning for optical music recognition, what AudioShake is doing for source separation of audio-- which everybody tried to do and nobody-- they're the first ones who are able to do it. They're very clear about which problems they can handle and which problems they can't, and why it's a good market and why it's supporting musicians.

MICHAEL CUTHBERT: We were talking a little bit earlier about with generative music things. To sum up, what you're saying is, make sure that there's a problem that you're trying to solve.

MICHAEL GOOD: Yeah, a problem that you're trying to solve that isn't putting musicians out of work would be great.

MICHAEL CUTHBERT: Well, thanks, Michael. I wish we could have had you earlier when we were specifically talking MusicXML, but you can't go six weeks into the semester without knowing MusicXML. Even knowing you're coming now, it's so great. So thanks again.

[APPLAUSE]

Feel free to stay. Feel free to move on. We're going to unlock intervals.

We're going to be moving on to the shortest topic of the semester, and that is a voice leading. You're going to get just a little bit from me today. You're going to have the world's shortest-- not the world's shortest, but a pretty short-- problem set. And then, we'll talk about it a little bit after the midterm on Wednesday and that will be it.

It's a very important topic. We're going to keep working on it. That's why it deserves to have its own number. When we're talking about voice leading, we're generally talking about how-- we just need one staff for this-- how multiple parts are moving at the same time.

This would be pretty bad voice leading, maybe very dissonant, but something. We're talking about not what's happening here and not what's happening-- that one's hard to do-- here, but the combination of the two. And why voice leading becomes a particularly interesting problem for computers is, there's two general ways of iterating. Both flatten and recurse are, in some ways-- well, no.

Sorry, recurse is you go through one part or one voice, and you go through everything, and you go through the other part and the other voice, and you go through everything. And then, you have something like flatten, which in the end-- especially if everything is just quarter notes-- you're going through one slice going to the next and going to the next slice.

Voice leading is a particular problem that requires you to be constantly moving back and forth between I'm moving according to voice or according to part, and I'm looking at what's happening at one time or another. One of the classic applications for starting off computational voice leading is to work on a topic called species counterpoint. Who has done species counterpoint in one of their classes?

Some instructors do it. Some don't. Not everybody has. I think it's invented-- it's an 18th century way of learning to write music like they wrote it in the 16th century. Not 100% relevant to music composition today, but still a fundamental skill, like you practicing your scales or something.

Mozart and Brahms practiced species counterpoint every day, sort of keep up their chops. The earliest, the first species that we tend to call it, is something is all in whole notes. Let me see if I can compose one while we're--

How many notes did I write? One, two, three, four, five, six, seven, eight, nine, good. Classic ones tend to be around nine notes long. And then, against this, in first species, you're going to write one whole note for every note with particular conditions.

Some of these conditions will be horizontally constraining, like for instance-- yeah, now that particular one. Don't do that. Why not?

AUDIENCE: Tritone.

MICHAEL CUTHBERT: Tritone, yep, this case augmented fourth or diminished fifth usually not allowed also, especially in this context. Good, so some of these things are going to be based on one case and some of them are going to be based on vertical things. This would not be allowed. Why not?

Sorry, I need two colors. What? Somebody said it. I just didn't hear.

AUDIENCE: [INAUDIBLE]

MICHAEL CUTHBERT: No, I think I heard it properly.

AUDIENCE: Seventh?

MICHAEL CUTHBERT: Seventh, yeah. In first species, the rule is all horizontal vertical intervals must be consonant and all vertical intervals must be consonant. But then, there are other strange little rules. This is a consonant interval.

This is a consonant interval. And this is a consonant-- I said I was going to do red. This is a constant interval. All these intervals are consonant. What's the problem here with these two, for anybody who's done this? Yeah, so parallel fifths.

So what's the interval G to D? Perfect fifth. E to B, perfect fifth. Two perfect fifths in a row. I just want to give there, for anybody who wants to, I don't have a formalized extra credit for this. But if you can, writing software that can solve any given what we call the-- this is the part that's given to you.

There's always one part that's given to you. It's called the cantus firmus, or the fixed song in Latin. And so, you fixed. You can't change it. One of the classic things is to write-- nowadays, in computers-- is to write software that can solve the other part, that can give you the other part there.

That's one of the ones, if you're interested in optimization and algorithmic things. This is one of the cases, the first cases in this class, where it is easy to write something that will do it properly, but for a 20-note or 40-note one won't be able to generate it in the time that the universe-- before the heat death of the universe. So it's something where algorithmic complexity comes in. You are not going to be given that assignment this year.

You all have worked pretty hard, right? This class is-- yeah, OK. So not doing that seems like a good thing. But instead, what you're going to be asked to do, between now and next Friday, is to come up and encode some of the basic features underlying how you might solve a problem like this, and specifically, how you might avoid things like this.

There's one or two things on this. This is not a get everything correct problem set, but looking quite down the road to the second-- probably the second to last problem set in this class, problem set nine-- this will be foundational knowledge on it. That one, problem set nine, you're going to be given a piece.

You're going to be asked to remove all the passing tones and all the ornaments and leave it just with the block chords. And so, being able to identify voice leading moments is pretty important.

OK, so let's unlock intervals. You've been working a lot on them. Unlocking interval, let's go ahead and from music21, import corpus-- which we'll use just at the very end, if we have time-- note, pitch, and the brand-new interval module. Is that big enough? Bigger, better?

We'll start with the worst variable name ever, i equals the interval module. It's lowercase. The interval class in there, P5. Let's start with a perfect fifth. These are all things that you've been doing.

Within the interval module, by the way, there are interval classes called chromatic interval, diatonic interval, generic interval, all these things. But the interval class will let you do, basically, almost anything in there. Great.

So now, you've created a P5 interval. You can get its name, which is very unhelpful. It just tells you what you just put in. But you can get its nice name with a capital N in the middle, perfect fifth.

That's just a little bit more helpful. Let's keep going. We can get its directed nice name. This is all in snake case or camel case, whatever you want to call it-- directedNice with a capital N, name with a capital N-- and get ascending perfect fifth.

Sometimes, we care about what direction the interval is and sometimes we don't. In melodies. We care or don't care? Take a guess. When you're talking about melodies, do we care which direction the interval is?

AUDIENCE: Yes.

MICHAEL CUTHBERT: Yes. When we're talking about the notes of a chord, do we care?

AUDIENCE: No.

MICHAEL CUTHBERT: No, not usually, because we're just talking about what things are. Great. So if we want to create a descending interval, we can create interval.Interval ('P-5') and get its directedNiceName.

You don't have to type all these things. I'll put this up later, if you're not a fast typist. But you can do that. Also, because I can never remember which one we prefer, you can do negative P5, because I don't know which one of those is actually better.

That's the most basic way of doing things. Let's try to get the intervals from pitches first off. So let's create two pitches. We'll create pitch.Pitch ('B3'), p2 pitch.Pitch ('C5'), so a little hard, but not too hard. You don't need to type the question marks where to get the interval, interval.Interval (p1, p2).

Instead of putting in a string, you can give it two pitches or something. I'm going to name this variable after what the name should be. The interval between B3 and C5, what should I call that, somebody who hasn't already gotten ahead in the computer? What is that interval? Always try to do things in your head the first time?

AUDIENCE: Minor ninth?

MICHAEL CUTHBERT: Minor ninth. I'll do lowercase m9. I do like naming things according to their interval, or according to what they're representing. It does hit my anal retentive like, everything must be in order streak, that when we do have something that should be a major third and it's capital M, but then, I go capital should only be used for classes.

But sometimes, we have to do that. OK. We have-- do we get that? Yep, [INDISTINCT]. Minor ninth, how many semitones in a minor ninth?

AUDIENCE: 15.

AUDIENCE: 13.

MICHAEL CUTHBERT: 13. So dot semitones will give you the number of semitones in an interval, which is often quite helpful. Obviously, m9.name will give you will give you m9, but m9.simpleName also in snake case. When you're dealing with compound intervals, there is something called simple name that will be very helpful for working through things.

Now, there's some slight difference. Let's come up with another interval. This one I'll call p8. I'll just show, so you can do some of these things here. We can also create an interval from notes, not just from pitches.

That's probably a mistake in my taxonomy, meronymy, whatever it is, but it's been there for a while. So p8 should be C 4 to C 5, good. The only difference why I'm creating an octave-- I should have just called it octave, right-- is that sometimes, that the octave simple name is perfect unison, p1. Right?

But sometimes, we care about the difference between octaves and unisons in a way that we don't care about seconds and ninths. So there is something called p8.semiSimpleName-- I wish I had spent some more time coming up with it-- which basically, is exactly the same as a simple name, except that things that are multiples of octaves get squished down to an octave.

A double octave also becomes an octave in a semi-simple name. Whereas, the ninths semi-simple name is minor second. I think there's also things, see if I still have all these in here, semiSimpleNiceName. Yeah, OK. There are all of these things in there.

Great, let's stop typing so much and let's create some aliases. Alias, let's just do capital I is the interval.Interval class. Now, we can call I and get new interval and P equals the pitch.Pitch class. Not a pitch class, the class pitch inside the module pitch.

That able to be seen? Great. So now, we can create a minor third just like that, just a capital I, parentheses n. Slow it down? OK? Slow down?

Slow down, yes, a little bit. OK, good. Great, pause for a second on this. So far, this is great, except for being able to get the interval from two pitches, which will probably be very helpful for you. There's not too much you can do once you have an interval object.

So let's start creating some things that you might like. You don't have to type all these things, but this is probably a nice one to do. The interval of a minor third is consonant.

Now, this is probably a mistaken term before I hit Enter, because all of these things are consonant in some of the world's traditions, and not in some of the other world's traditions. In fact, some of the early proposals of music21 said oh, and then, you'll be able to have a context objects, which will be able to affect how these things are.

If you're in 14th century Europe, this will be false. If you're in some part-- if you're giving it southern Indian context, it might be false or depending on what the time is. And I've never actually fully implemented that. Right now, unfortunately, this privileges the western classical music context. What should this be if it's in a western classical music context? Minor thirds are consonant or dissonant?

AUDIENCE: Consonant.

MICHAEL CUTHBERT: Consonant. Good, I programmed something right. Now, let's try something else. A major sixth, consonant or dissonant?

AUDIENCE: Consonant.

MICHAEL CUTHBERT: Consonant, good. Sorry, I'm just going to-- I'm typing a lot here. You can probably assign things variables. The number of semitones in a major sixth?

AUDIENCE: Nine.

MICHAEL CUTHBERT: Nine, good. Now, I'm going to create a diminished seventh, I('d7'). And the number of semitones in a diminished seventh?

AUDIENCE: 11.

MICHAEL CUTHBERT: So I got one guess, get another guess?

AUDIENCE: Is it nine?

MICHAEL CUTHBERT: 9, so diminished seventh is consonant?

AUDIENCE: False.

MICHAEL CUTHBERT: False. So one of the things built into the music21 ontology that's different from Dmitri Tymoczko's optic default ontology is that spelling matters in this case. A diminished seventh sounds the same as a major sixth in isolation, but it's not considered consonant. That's important to know.

I think last year, there was a month that went by without somebody reporting this as a bug to the music21 list. But most months, one to two people put that in, not see that so. It definitely means that I don't know.

Augmented seventh is not consonant either, even though an augmented seventh is another way of saying octave, perfect octave, great, super. Basically, anything in traditional theory, in traditional tonal theory, anything that is marked as diminished or augmented or doubly diminished or doubly augmented or whatever, is considered dissonant.

Now, the weird one that we can always debate is the perfect fourth. Who thinks it's consonance? Who thinks it's a dissonance? A lot of people not voting. I'm glad for either way people voted.

In this case, music21 regards it as not a consonance. Perfect fourths are generally-- they're the one weird interval. They're generally considered consonant when they are melodic, but dissonant when they're harmonic and above the bass. Take Medieval Music with me or Spiro. You can figure out the whole history of how this happened and why it's in the one weird double position, but it's the only perfect thing that is not consonant in some circumstances.

OK, what are some of the things you might do in the interval class? Well, let's start with-- let's create the perfect fifth one, p5. Hopefully, you've created that, good. And what we can do is, let's say c4 we'll just call this equals P('C4'). And now, p5 can transposePitch c4.

So you have a perfect fifth, creating that as an interval object, storing as p5. We can create a pitch object, just because of my aliases c4, start at c4. Then, p5 can transpose the c4, transpose pitch on c4 and gets a pitch object that's G4.

Now, once you've done that, if we look at C4, that object is unchanged. Transpose pitch creates a new pitch that is not the same pitch. If you wanted to do something useful with this, you would probably store it as G4 equals this.

And then, you can do something with your new pitch. Obviously, if you knew what the pitch was going to be coming in and what the pitch was going to be coming out, you wouldn't need to do this. You can just create your own damn G4. But this would work in general.

If you do want to transpose in place, I don't think we've seen this in music21, but you're going to start seeing it all over the place. There is a keyword only argument. That means, you must type this as a keyword, inPlace=True.

c4 will now be transposed and nothing will be returned. We run it. But now, c4 is a really terribly named variable, right, because c4 is now a G4.

If you're ever doing a demo, remember when we took two notes, we inserted two notes or removed them from the stream and then, everything was screwed up. I always-- I'm going to remember, learn from that experience and make c4 equal c4 right again. But sometimes, we want to just have a generic p equals here. This time, I'll write it out so that we can copy and paste this, pitch.Pitch ('C4').

And we still have p5. Let's just do this for i in range (13). We don't actually need i. Some people like to have underscore as a dummy variable. But we're just going to, for 13 times, let's take the pitch. And then, since we've just done it, p5.transposePitch(p, inPlace=True).

We create a c4. 13 times, we're going to go through and transpose it. So what is p now?

AUDIENCE: C 11. B sharp 10.

MICHAEL CUTHBERT: B sharp 10. Is that what it is?

AUDIENCE: F sharp 11?

MICHAEL CUTHBERT: Oops, not c. What do you call it, p? F double sharp 11, because if you notice, we were printing and then transposing is my little trick thing, just to make sure that we're all following things.

By the way, this is one of the weird reasons why I wanted you to work, early on, with weird octave numbers like 10 and 11 and negative 2 and things like that. Because quite often, it is simpler to do an operation like this, like let's all go around the circle of fifths and then, we'll just print the step of it and not care about the octave number.

And then, in the end, we can bring it back down a bunch of octaves later. But sometimes, you're going to end up in there. And sometimes, you're going to end up in double sharp territory, triple sharp territory, before then transposing down back the other way or doing some other path.

Almost never is F quadruple sharp 22 going to be a useful thing to think about musically. But sometimes, it's a useful point in the midpoint of your work. And by the way, that's how imaginary numbers came out first, that they were first used to solve, I believe, quintic equations in a way that made it very easy, even though they were in the intermediate stages, even though they were never the answer at the end.

Think of quadruple sharps and octave 22s and stuff as like that. OK, so a couple things. Is c4 still c4? Whew, yep, good. You can also transpose-- do the opposite way.

This is a real sign that Python allows and that other languages, I haven't been able to do it in. Instead of taking the interval and saying dot transpose pitch into pitch, you can take the pitch and say, transpose that by the interval. That's sometimes pretty helpful, so music21 object dot transpose by an interval class.

If you don't care about efficiency, you can always put in just a string and it will create an interval for you. And so, transposing up an augmented 223rd gets you G sharp 35. Is that right? Doesn't matter.

OK, looking at the time, one or two last things. If you ever want something when you're working with it, this is the interval object, again, perfect fifth. There are things like specifier in here. You'll be able to do a-- how to do dir on a what do you call it, on an object to see everything that's in it?

But these are probably the most useful things you'll see, so how many cents, how many hundredths of a semitone is it? Yes, you can do microtonal intervals if you care about such things. What's the chromatic interval?

What's the complement? If you have a major sixth the complement is a minor third. It's the thing that fills out the octave. Diatonic, is it a skip, is it a step? Can we reverse the interval? All these things, you'll find some of these things in here.

But the last thing I want to do is to think about this spelling, non-spelling thing, so implicit diatonic we'll say. Let's create the interval of four semitones. All I care is that it's an interval of four semitones.

Here's where, what do we call it? Polyvalent? Polygamous, whatever we call the interval, it will take-- that was a joke-- it will take an integer instead of a string.

We can say the int four. If I just look at this i4 right now, it looks like, hey, it's just a cheap way of saying major third. But it's actually a little bit different from that. If I take the pitch c double flat-- so pitch c double flat 4-- and I'd say, i4.transposePitch.

Take C double flat 4 and take it up a major third. I don't care about the accidental. But what's the step have to be if you're going up a third from C something?

AUDIENCE: E.

MICHAEL CUTHBERT: E. So it should be, what is that, E double flat 4. Instead, you get a D4. So there's something where, if you initiate an interval just by saying the number of semitones, you're telling that interval that the diatonic interval, the major third, is implicit.

That is to say, when you want to know if it's a major third, yeah, it will be a major third. But ifor will say no. Feel free to change this to anything that is four semitones up that is logical and easy to use. That's something that people will see in there.

Whereas, if we take the perfect fifth and C, which we created it with the p5 thing and see if it's an implicit diatonic, false. Perfect fifth will always take the one that we created with the string. P5 will always make sure that something is a p5.

I'm looking at the time, so I won't demonstrate the last thing, which is music21 still has a restriction that it won't go above quadruple sharps or quadruple flats. At that point, even the diatonic one will say, forget what I said. Let's make it work and round things.

Great, that's all the time that I have. If you transpose a whole score, you'll see that the key signatures transpose with you, which can be kind of helpful. I'll post my-- what do you call it-- the thing where I have that demo to the class. Great.