# Creating Sonifications with Astronify (Nonvisual Data Science Workshop Series)

(0:00 - 5:25)

All right, well, welcome all to the last workshop in the non-visual data science workshop series. My name is Patrick Smyth. I am the chief learner at IOTA, and so this is the last workshop in our non-visual data science workshop series, so workshop number five.

It's our second workshop on sonification, and Sarah Kane is going to be, you know, leading us in a little further exploration of making data representations with sound rather than sight, and I just want to go through a few administrative items, and then we'll get started. So we will, I'm in the final stages here. I'm waiting just to hear about something about the server setup, but we're going to have a mailing list set up for people getting started with data science with a visual impairment, so I'm going to send out an email when that's all set up.

You'll see that in an email that will go out. Hopefully, I'll have it done by the recording email that will go out tomorrow, most likely, but if not, it'll be later in the week. It'll be its own separate email.

We also have a couple of events coming up that I'd like to kind of highlight to you, so I'd recommend keeping an eye on the email following this, but, you know, one, for example, is Tony Fast, who's been coming to these workshops and is doing an event with Space Telescope Science Institute related to, you know, basically the topics that we're doing here. He's developing a accessible version of Jupyter Notebooks and may be an interesting time to, you know, if you're interested in that, it sounds good. He may need some testers and so on in the coming weeks and months to try out his prototype, so, and, you know, the usual, I'd just like to thank a couple of people.

We will, our helpers are in the room, so we have Elizabeth and Alex in the room today. I will also be here helping out, so if you have any questions, just drop them in the chat and one of us will respond to you, and you can either message myself, Patrick Smith, Alex Ogden, or Elizabeth Sola privately if you prefer, or you can just ask your question publicly and we'll manage it that way. I'd also like to thank Pandas and NumFOCUS for hosting these workshops and, you know, and especially Patrick Hofler, who's a Pandas core developer, who sort of helped us put this series together.

And finally, there's one small sort of technical item that a couple of people have emailed me about one specific thing. So, in the second and third workshops, we learned how to, maybe even as far back as the first, we learned how to use the IPython magic command save, so %save and %load, and so people were not really having so much trouble with the save command, but then when they were loading their files, there was some difficulty with the behavior that is experienced when you load. So, the part that I think I

didn't explain, and I'm going to update the tutorial, I have an issue open, so I will not forget that, but the extra step that we sort of left out is that when you do the %load and the file name to load in your code, then once that successfully executes, what it basically does is it kind of like almost like types in all the code, so it's in a giant block in IPython, and in order to run that giant block, so you may hear different things, you may hear the code that you imported, you may hear dot, dot, dot, that's what some people were hearing, but basically in order to sort of complete the loading process, you need to hit the enter button twice, okay, and once that happens, it basically is as if you typed in all of that code yourself, and it runs each line separately, and once that's done, if you want, you can clear the terminal with control L, and then you should have a nice clear session with all the variables and so on available to you that you had before, and there's also, some people were asking about how to store specific variables and so on for next time, so for example, if you want to store a data frame, and for that, I would take a look at the %store magic command, so essentially, you could do a %store and the name of a variable, and it will save it for you to load it in in a future session, and I think it's %storer-r or something like that will allow you to load variables, so that's something people seem to have a lot of interest in, and a few people were emailing about it, so I thought I'd cover it.

(5:27 - 7:32)

All right, so I think that's everything administratively, and of course, you know, I guess I'll maybe spill a little bit now since, you know, I think people are most engaged right at the beginning, and so I'll just say, you know, I'll also say this at the end, but these workshops, you know, it's been a real privilege to put together these workshops. We will be creating a website specifically for the curriculum, so it won't always be on GitHub. Once it kind of stabilizes, once it's all finished, we'll move it to its own website, and I would ideally like to, you know, turn it into a larger resource with some of the suggestions and so on that have been provided, and of course, this is, you know, we have 10 hours with you.

We covered a certain amount of material, but obviously, there's a lot to data science. There's a lot more to possibly cover, and so, you know, I hope to do more with these materials, but there will be, at the very least, what we've done so far will be available in a sort of a website resource. When that is available, I will send out an email about that, and I mean, I would also say if you, you know, we're all part of the blindness community.

We're all part of the VI community. If you are, you know, in touch with specific blindness organizations, I mean, I'm in touch in conversation with some people as well, but if you think that, you know, you have contact with someone in an organization that you've enjoyed these sessions and that you would like to see more of this kind of work happen, either in terms of creating resources or in terms of, you know, teaching workshops and so on, then, you know, feel free to reach out to them and say, hey, this was really good.

We would like to see more of it, and then, you know, and hopefully, I would love to see the major blindness organizations put some more resources specifically behind these kinds of more, you know, more modern skills that I think are important for the blindness community.

(7:34 - 9:27)

Another, you know, and another thing that we are, I mean, some conversation with some organizations is, you know, maybe having, like, some small grants or small prizes for infrastructure development and so on, so that would, you know, I really think that some of the big organizations should get behind some of this work. If you have contacts in those, just reach out to me. Drop me a line or put a word in someone's ear, you know, and have them reach out to me, so I, you know, and I also just want to thank all of you participants.

I mean, you guys really come out in large numbers, you know, and there's a huge number of people who are following along asynchronously as well and reaching out by email every week and everything like that, so there's actually kind of a hidden group of blind folks who are really interested in learning data science that is even larger than we're seeing in these rooms, which is, and there are pretty big crowds in these rooms, too, so thank you. I just want to thank you all for coming out, for learning. I know it's challenging, but it's been great to connect and to see this, you know, hopefully it's a start of something cool, so all right, that's enough spieling.

Sarah, do you want to, I'll introduce Sarah Kane. She is a PhD student in astronomy and astrophysics at Cambridge and a Marshall fellow, and she's going to be leading our second workshop right now on sonification, so Sarah, it's enough talking from me. Thanks, Patrick.

All right. Hi, everyone. Excited to get started again.

I'll share my screen. I have share my screen. Hello.

All righty. Make sure share sound is on, and we are good to go. Before we begin, I just want to mention off my camera, because someone had the suggestion that that might help with some of the difficulties we had last week, so let's hope for the best.

(9:28 - 11:03)

Apologies if anyone is low vision or sighted here. If you don't appreciate me being, you know, just a square, sorry, but we're going to give this a try. On the topic of audio difficulties, I know we had rather extensive audio difficulties last week.

I just want to mention that there is a nice fresh recording. I re-recorded the tutorial from last week, so if you found last week a little difficult to follow because of some of the

audio difficulties, I highly recommend you go check out the new recording, especially right at the end when we listen to all of those sonified shapes that we made. I think that's the most informative part of the tutorial, so I highly recommend going to check that out if you are interested.

I'll also mention that we still seem to be having some audio difficulties where sounds from the terminal do not seem to like to play through Zoom that we can tell, so I'll just tell you in advance that all the sonifications for this week I have pre-prepared as audio files, so, you know, when I'll have you guys typing sonification.play or whatever we name it to get things going, I will click away to an audio file and play the sonification for you from there, and hopefully we will be on our merry way. All right, now I'm really done spieling. Let's see.

I'm going to turn NVDA speech back on. "Speech mode B. Speech mode talk." And I just want to confirm, can you hear that okay? Yes, no? Yes.

Yes, we can. Thank you. All right.

(11:03 - 11:18)

Oh, and I don't hear that active speaker alert, so maybe the video off is the answer. All righty. So, as we always do, I'm going to ask you to begin, please, by opening Anaconda Prompt.

(11:18 - 14:43)

You can do that by hitting the Windows key and then typing Anaconda Prompt, and it should be the first result that comes up after that. And once you are in Anaconda Prompt, we will start IPython by typing, as we always have, IPython, all lowercase, all one word. "I-P-Y-T-H-O-N.

Python 3.11.5 packaged by Anaconda Incorporated." All right. I think we've heard that a bunch of times.

I wanted, you know, us to hear the Python version one more time, but we probably don't need to listen to that whole thing. All right. So, now we are in our interactive IPython environment, and we are ready to get started.

The exciting news is, today, we are going to sonify some real data. What I have prepared for you is a CSV file of temperature data. This is real data, real temperatures from cities around the world, and also some rainfall data.

We'll look at that data in detail before we sonify it, but in order to actually import that data, we are going to need to import our favorite packages that we've used all along. We're going to import NumPy and import Pandas. Hopefully, this is a familiar procedure

by now.

I'm going to start in that in one line by typing import NumPy. "I-M-P-O-R-T." So, that's import space NumPy.

"N-U-M-P-Y." All right. And it's telling me I'm on the next line already.

We don't expect any output from importing something, and I'm going to type import Pandas like the bear. "I-M-P-O-R-T space P-A-N-D-A-S in three." All right.

I've typed import Pandas. Hit enter. And this is the moment where I'm going to ask one of the helpers.

I think Alex or Elizabeth has it at the ready to copy and paste two lines of code in. This is going to be the URL that we're using, and then it's going to be df equals Pandas.readcsv, that URL, and the index column equals zero. Basically, this is saving us that step of typing out the URL.

So, I highly recommend you go and copy and paste that from the chat. It's rather a long URL link. And then it's just our very familiar Pandas.readcsv. I myself am going to copy and paste it in as well because it's a long URL.

All right. It just read the very last bit of what I just copy and pasted in. But the important thing is, again, to remember that we're this df equals Pandas.readcsv, that URL.

So, what we're doing is we're making, as we always do, a Pandas CSV from the data in that or not a Pandas CSV, a Pandas data frame from the data that's saved at that URL, and we're calling it df. And I'm just going to hit enter. It might take a second.

About a second exactly. You can hear it says in four. We're on to our next line.

(14:44 - 15:02)

Now, the data is all already there. Unlike last week, we're not going to sort of craft any of it ourselves. But I am very strongly of the opinion that we should not be sonifying or visualizing, representing our data in any way until we actually take a look at that data.

(15:02 - 16:29)

And all you really know about this data is that it's something to do with the weather. Like, I've mentioned something about temperature. You don't really know what's in here.

So, I don't know about you, but my very first question is, what is actually in this data? What are the columns? And so, you might recall from Patrick's lessons that we can check on the columns in the data frame by doing df. That's what we've named that data frame and those lines I had you copy and paste. So, we'll do df.

And then dot. Whoops. It wants to suggest something I've typed before.

I don't want to do that. I want to do df.columns. "C-O-L-U-M-N-S." That's df.columns and hit enter.

I'm going to tell you now that this is going to be a rather long output because there are 13 columns in this. So, we're going to listen to them all the way through because, again, we want to know all the columns in our data frame if we're using the thing. "Out four.

Index. Year. Month.

Day. New York City temp. Philadelphia temp.

Pittsburgh temp. Orlando temp. Austin temp.

Seattle temp. Delhi temp. Delhi rain inches.

Delhi temp model one. Delhi temp model two. Dtype equals object.

In five." All right. So, what columns do we have here? We have three columns that specify the date.

(16:29 - 18:32)

That is the year, month, and day at which we have our data. Then we have a whole slew of columns with the temperature, presumably daily temperature, in a bunch of different cities. New York City, Philadelphia, Pittsburgh, Delhi, India, blah, blah, blah.

We're not going to use all of those columns. But I wanted to, A, give you a larger data frame to handle because this is often how data comes. And, B, because if you want to play around with more data, then it's available for you after this tutorial.

You already have something you can sort of test and mess around with. So, plenty of data available for you. If you're interested in where I got this data from, it's all linked on the curriculum online.

So, if you're curious, feel free to check it out. It is from two different sources that I curated and put together. There was a little bit of, like, processing that I did to sort of get it in a format that would be roughly convenient for us to use.

We have that Delhi rain inches column. That's the daily rain in Delhi, India, in inches, as the name says. And then those last two columns, the Delhi temp model one and temp model two, those are actually columns I created.

And we'll get into those a little bit later if we have the time. But essentially, they are predictions for the temperatures in Delhi, India. And one of them is intended to be a good prediction, and one of them is intentionally quite a bad prediction.

But we will get into that later. All right. So, now we know roughly what is in this data frame.

My next question is, like, how big is it? I've told you there are 13 columns, but I just want to print out the whole shape, you know? I mean, if I weren't here to tell you that I know there are 13 columns, I don't know about you, but I wouldn't want to sit and, like, count the columns. And I also want to know how many rows there are, how many days of data are there. So, I'm going to do, as Patrick taught us, df.shape. And I'll hit enter.

(18:37 - 18:57)

All right. So, there are 9,265 rows of data. That is each of those I'll tell you is one day of data.

And 13 columns. I can tell you right now that because I've handled this data already, that's about 15 years' worth of weather data in this CSV. So, it's rather a lot.

(18:58 - 19:55)

But my next question, after knowing, well, this is a rather large file, is, all right, but, like, what dates does it actually run from? And the way I can find out is by indexing the first and last columns. I'm going to assume this data runs in chronological order. First, because shuffling up the dates would be chaos.

And I have yet to see a data file that does that. But B, because I have already, you know, handled this data. So, I'll tell you it runs in chronological order.

So, if we check the first day or the first row of data, that should be the first day of data. So, we can get the first date. And if we check the last row of our data frame, that will be the last date of data.

So, we can know when our data starts and ends. And again, Patrick taught us how to get the how to index by row number. That's going to be DF.

(19:55 - 21:10)

DF.IL.ILOC. So, ILOC, LOC for location, and I, index, because we're indexing by number. Then it's going to be open square bracket. Zero. Right bracket. Not parentheses, brackets. Why is it a zero? Well, Python likes to number starting from zero.

So, if we do DF.ILOC brackets one, we're actually going to get the second row. All right, let's hit enter. Out six.

Year 1995.000000. Month 1.000000. Day 1.000000. I'm reading all of the values in that row. I only care about the first three, the year, month, and day, because that's the date that I care about. Annoyingly, they're in decimal format, just because that's how this file

is read in.

It's read in as floats, you'll recall, we call them. But I can tell that the year is 1999, the month is one, and the day is one. So, our data starts on January 1st, 1999.

(21:10 - 21:25)

All right. When does our data end? So, what is the range of dates for our data? Well, then I'm going to do DF.ILOC. Open bracket. So, the same thing as the line before so far.

(21:26 - 22:54)

Left bracket. Now, I could do what? What was the length of the data frame? 9265. I could type, because I know how many rows there are for the last one.

But that requires me remembering how many rows there are in our data frame. So, a nice little shortcut is that if you want to index the last row of the data frame, you can just type negative 1. Dash 1. Dash 1, negative 1. So, it's DF.ILOC, open bracket, negative 1, close bracket. Right bracket.

And the fun thing is, if you do, for instance, negative 2, that'll give you the second to last row, negative 3, third to last row, so on and so forth. This is another sort of Pythonic numbering thing. Maybe other programming languages do it as well.

I don't know. Don't quote me on that. I only know Python.

"Out 7. Year 2020.000000. Month 5.000000. Day 13.000000." All right. And I'll stop it again after the day, because I'm not interested in hearing how hot it was in Austin, Texas, on, as we now know, May 13th, I believe it was, of 2020. So, we have just about 15 years of data in here.

So, that's great. Now we know how large our data frame is. We know the time range.

(22:54 - 24:26)

We know roughly what sort of data is in it. Cool. We're getting a good picture of things in this data frame.

And then the last thing I'm going to want to do, before I sodify things, is actually take a look at one of the columns. I'm going to do Philadelphia, because that's where I went to undergrad, and I love Philadelphia, and I know roughly what the weather should be like in there, like in the city of Philadelphia. And I just want to check on the data in the Philadelphia column first, to make sure nothing seems funky.

Second, you know, I don't even know right now whether this is in degrees Celsius or degrees Fahrenheit, which is, you know, a little problematic because there are no labels

here, like no units. But if I take a look at the data, I can probably guess based on the numbers we get out whether it's in degrees Celsius or Fahrenheit, and I would like to know that. All right.

So, the way I can do this is, if you recall from the list of column names, and no worries if you don't because I'll tell you. The temperature in Philadelphia, that column name is called Philadelphia underscore temp. And so following the syntax that Patrick taught us, if we want to say check the average temperature in Philadelphia, then I can do df dot Philadelphia That's going to be P-H-I-L-A-D-E-L-P-H-I-A.

(24:27 - 24:44)

You know, before I did university in Philadelphia, I actually had a really hard time spelling that city. All right, so that's df.philadelphia underscore temp, T-E-M-P. So this is just calling the Philadelphia temp column from df.

(24:44 - 24:56)

And then I just want the mean attribute, so dot mean, M-E-A-N. "Dot M-E-A-N." And then open parentheses, close parentheses, just like Patrick taught us.

(24:56 - 25:15)

"Left paren, right paren." All right, and let's hit enter. Out 8, 56.41700485698867, in 9. All right, so the average temperature in that Philadelphia temperature column is about 56 degrees.

(25:15 - 25:28)

That tells me for two things. First of all, having spent time in Philadelphia, sounds about right to me. It sounds about right if it's in degrees Fahrenheit, because if that's in degrees Celsius, Philadelphia is on fire.

(25:28 - 25:37)

Which I certainly hope it isn't, because I would like to go back to visit. So I know two things. First of all, from a rough analysis, that average seems correct.

(25:38 - 25:52)

And also, I now know our units are in degrees Fahrenheit. I can also, and for the sake of thoroughness, I could also check the max. That's going to be the same line we just typed, except instead of dot mean, it'll be dot max.

(25:52 - 26:26)

So we can do df, dot Philadelphia, Philadelphia, underscore temp, dot max, "left paren,

right paren, out 9, 92.9, in 10." All right, so the maximum temperature we have recorded for Philadelphia is 92.9 degrees Fahrenheit. I'll tell you for certain that it definitely gets hotter than that in Philadelphia.

(26:26 - 26:44)

So I would be scratching my head a little bit about this and feeling some concern. Except for the fact that I believe that this is actually the average temperature over the whole day. So if you consider the fact that it's like cooler in the night, yeah, I can imagine that the hottest day in Philadelphia is on average 92.9 degrees.

(26:45 - 27:11)

For the sake of completeness, let's look at the minimum. "df, dot P-H-I-L-A-D-E-L-P-H-I-A, line, T-E-M-P, dot M-I-N, left paren, right paren." So that is df, dot Philadelphia, underscore temp, dot min, open parenthesis, close parenthesis.

(27:12 - 27:32)

That is the same line we have been typing except now instead of min or max, it is min. And let's just see what we get. "Out 10, 9.4, in 11." 9.4 degrees Fahrenheit. Yeah, that sounds right to me. For you Europeans, that's going to be, I don't know, degrees Celsius.

(27:32 - 27:37)

I'm going to bet on like negative 10 Celsius, something like that. Miserable. Miserable is the temperature.

(27:38 - 27:42)

All right. We've explored our data frame. We know what's in it.

(27:42 - 27:48)

We know like the columns. We know the date ranges. We know actually the type of data we have in here.

(27:48 - 28:00)

We've checked the column we're going to sonify first, Philadelphia, and made sure that everything seems to be in order there. I think we're good to go. Let's get into actually sonifying.

(28:00 - 28:22)

Now, inherently before we visualize something, or visualize or sonify, so before we represent data, there's always going to be some legwork we have to do. The legwork

we're going to have to do here is two steps. First, what we're going to want to sonify is the Philadelphia temperature versus time or versus our date.

(28:23 - 28:33)

However, right now our date is spread across three columns. It is year, month, day. That is going to be really difficult.

(28:34 - 28:41)

Actually, that's going to be impossible for a sonify to parse. Like those three columns, that's not going to work. We don't want that.

(28:41 - 28:56)

Why did I leave it this way? Well, first of all, this is a format you'll often see in data. It's the way the data originally came, so I want us to be able to work around it. What are we going to do is we're going to make a new column in our data frame.

(28:56 - 29:13)

I'm going to call this time step, and this is going to basically be the number of days since our data began. So the first day, January 1st of 1995, will be zero. January 2nd will be one.

(29:13 - 29:30)

January 3rd will be two, and so on and so forth. This will now act in our sonification as the time column because those nice just numbers will be easy for a sonify to parse. And in the sonification, a sonify doesn't really care about the value of the time.

(29:30 - 29:45)

It cares about the distance between times, so what I'll call the time step. That's why I'm calling this column the time step column. It's because say we took, you know, the data every day, and then suddenly we took it, oh, a week later, so there's a larger gap.

(29:46 - 30:10)

That could be a problem, but here the data is taken at really regular intervals, just once a day for about 15 years. So we can make sort of a fake time column, and a sonify will treat it just the same, and it will work out very well for us. So, Patrick already taught us how to make a column like this, and I will refresh your memory if you don't recall.

(30:10 - 30:24)

First we're going to call the column, though we haven't made it yet. That's going to be

df, open bracket. It's trying to complete things for me.

(30:24 - 30:27)

I'm not there yet. Give me time. All right.

(30:28 - 30:48)

Df, open bracket, quotation mark, or like little tick mark. Then it's time step, all one word, so time step, T-I-M-E-S-T-E-P. Time step, close quotation mark, close bracket.

(30:49 - 31:08)

So this is a slightly different syntax from the one you were using with Patrick to call a column. Here we're using brackets and quotation marks instead of the dot format, which is, I think, maybe a little quicker to type. This format, in my experience, just works better for creating a new column, so we are making a new column of the data frame.

(31:09 - 31:18)

There isn't already a column called time step. We're creating a new one. I found when you do df dot time step to make a new column, it throws an error.

(31:18 - 31:45)

So this one time we'll use this different syntax here to call the column. So that's df, open square bracket, open quotation mark, time step, close quotation mark, close square bracket, space, equals, space. And then the function we're going to use here is the one that Patrick taught us, pandas dot range index, P-A-N-D-A-S dot.

(31:45 - 32:23)

So that's pandas dot. The R in range is capital R. So that was a capital R, A-N-G-E, then index, capital I. So capital R in range, capital I in index, "N-D-E-X." And it's all one words, so pandas dot range index, open parenthesis. Then it's going to be start equals zero. "S-T-A-R-T equals zero." So we've opened quotation marks, not open quotation marks, open parenthesis, start equals zero.

(32:23 - 32:37)

That means the start of our steps will be zero. So remember that's going to mean that January 1st of 1995 is time step zero, comma, then space. Then I'm going to do stop.

(32:37 - 32:49)

"S-T-O-P equals." Stop equals, I'm going to type L-E-N. "L-E-N." Then I'm going to do open parenthesis. "Left paren." D-F.

(32:49 - 33:08)

"D-F." Close parenthesis. "Right paren." That means stop at the length of our data frame. In this case, that'll be, I think, what was it? 9,625, 9,265. Clearly, I can't remember off the top of my head exactly the number of rows in our data frame.

(33:08 - 33:27)

So the L-E-N, open parenthesis, close parenthesis, with D-F in the middle, will just automatically give us the number of rows in data frame. And that's because I want that last date, which was what? May 13th of 2020. Ah, 2020.

(33:28 - 33:48)

What a year. May 13th of 2020, I want that to be 9,265 or whatever the last row index is in our time step column. So that's going to be stop equals L-E-N, open parenthesis, D-F, close parenthesis, comma.

(33:49 - 33:58)

Comma. And then I'm going to type space. "Space." Step equals one. "S-T-E-P equals one." Step equals one.

(33:59 - 34:07)

So just increase in steps of one. So January 2nd should be day one, January 3rd, day two, so on and so forth. Again, counting from zero, as Python likes.

(34:09 - 34:17)

All right, hit enter. All set. If we want, we can check on that new time step column.

(34:17 - 34:25)

So maybe I'm going to do D-F dot time step. Again, it's trying to guess what I want. It's wrong.

(34:25 - 34:43)

D-F dot time step, and I'm going back to this more familiar syntax of D-F dot to call the column. T-I-M-E-S-T-E-P. And then I'm going to do dot head, and that will return the first five rows of this time step column, just to check that everything seems okay.

(34:43 - 35:21)

"Dot H-E-A-D, left paren, right paren." So that's D-F dot time step dot head, open parenthesis, close parenthesis, hit enter. "Out 12, 0 0, 1 1, 2 2, 3 3, 4 4, name, time

step, type, in 64, in 13." All right, so you might have heard 0 0, 1 1, that's basically saying the 0th row is 0, the 1th, the 1st, sorry, long day. The 1st row is 1, the 2nd row is 2, etc., etc. That all sounds good.

(35:21 - 35:37)

That's exactly what we wanted. All right, I said there was two steps we needed to get this sonification going. And the second one is you might remember that Astronify, the sonification package we use, wants its data in an AstroPy table.

(35:38 - 35:46)

I'm not going to dig into that again. If you don't remember that, I recommend you go check the curriculum or the recording from last week. I explained about AstroPy tables.

(35:46 - 36:23)

For now, we're just going to import table from AstroPy and then turn our data frame into a table. All right, so that's going to be to first import the necessary class. That's going to be from AstroPy.table, so that's from space, A-S-T-R-O-P-Y.table, dot T-A-B-L-E, space, import, space.

(36:23 - 36:43)

And then this is going to be table again, but with a capital T. So that's capital T, table, "T-A-B-L-E," from AstroPy.table, import table, "in 14." All right, that went through no problem. Hopefully it did for you as well.

(36:43 - 37:00)

And then this is the exact same thing we ran last week. I'm going to use the AstroPy table built-in function from underscore pandas. That will take a data frame, our data frame D-F, and return a table, which I'm going to call T-B-L.

(37:00 - 37:21)

So that's going to be T-B-L, space equals, space. And again, these spaces just make it nicer if you're going to pass this iPython session on to someone sighted who's reading it. Or I'm low vision, so I'm reading this now, and I find it easier if things are separated.

(37:21 - 37:33)

So I'm doing it for my own sake as well. But if you don't want to include the spaces, you don't have to. So that's T-B-L equals, space equals, space, capital T, table.

(37:34 - 38:00)

Sorry, I accidentally typed in capital A. I don't want a capital A, just capital T. Capital T, table, dot from underscore pandas, "dot F-R-O-M, line P-A-N-D-A-S." Open parenthesis. Close parenthesis.

(38:01 - 38:17)

So that's T-B-L equals capital T, table, dot from underscore pandas, open parenthesis, D-F, close parenthesis. Make an astropy table from the pandas data frame called D-F. That's our data frame, and call it T-B-L.

(38:18 - 38:31)

Hit enter. And if we just want to double check on that table like we did last week, we can type T-B-L dot call names. That is C-O-L names, not call like a phone call.

(38:31 - 38:47)

C-O-L like the beginning of columns. So there it goes, trying to again make it this time it's right. T-B-L dot C-O-L, N-A-M-E-S.

(38:47 - 39:04)

C-O-L names, and you'll notice this is slightly different from how we get columns in pandas. Astropy tables are like similar, but not the same. Just different enough from pandas data frames to make you have to Google every time when you want to do something.

(39:05 - 39:33)

But this will give us the columns in table T-B-L, just so we can check that they're the same as the ones in the pandas data frame. "Out 15, year, month, day, New York City temp, Philadelphia temp, Pittsburgh temp, Orlando temp, Austin temp, Seattle temp, Delhi temp, Delhi rain inches, Delhi temp model 1, Delhi temp model 2, time step in 16." All right, amazing.

(39:33 - 39:51)

We have all the same columns, including that new time step column we made. Fancy, fancy. All right, I am about to sonify this, but before I do, if there are any persistent questions before we get sonifying, please do let me know.

(39:52 - 40:17)

Now is a good time for that. I am hearing silence. I'm hoping that is because there are no questions and not because my audio has gone out.

(40:18 - 40:34)

But I am monitoring. We can hear you. Ah, excellent.

All right, all right, that's the that's the better option. Sometimes I feel like I'm like, I'm sitting alone in a little room I feel like I'm speaking into the void, you know. All righty.

(40:35 - 40:49)

If I don't hear any questions, I'm going to keep on moving on but feel free to ask questions in the chat, as always, we're going to get into the sonification. You might notice it took us quite a few minutes to get to this point. That is deliberate.

(40:50 - 40:58)

Data representations are only useful if you know what's going on in your data. So it shouldn't be. I think I see people do this a lot.

(40:59 - 41:07)

Heck, I'm in the habit of doing it, too. You just sort of throw either a visualization or a sonification at your data. And that can be useful.

(41:07 - 41:16)

But, you know, I also think it's useful to have a sense of what's going on first. So now we know what's going on in this data. That'll make the sonification a lot more useful.

(41:17 - 41:30)

All right, let's get into it. We're going to import Sona series from Astronify, just like we did last week. So that's going to be from Astronify.

(41:30 - 41:48)

So from space. A-S-T-R-O-N-I-F-Y. Astronify.series. dot S-E-R-I-E-S.

(41:49 - 41:56)

From Astronify.series, space. Import. Space.

(41:57 - 42:06)

Sona series. That's capital S in SONI, capital S in series. So that's capital S-O-N-I.

(42:06 - 42:14)

S-O-N-I. Capital S. ... And then the rest of series. E-R-I-E-S.

(42:14 - 42:39)

From Astronify.series, import Sona series. "WxPython is not found for the current Python version. Pyo will use a minimal GUI toolkit written with Teak." All right, I'm not going to play that the whole way through. I think basically everyone I've spoken to has gotten this sort of warning. I wanted to play a little bit of it again, just so you know not to worry.

(42:40 - 42:58)

I think there's this like audio Python package called Pyo that Astronify is built on that is going through some changes. Oh no, don't be back. Oh my gosh, we had all this time of silence and then I get that speaker alert again.

(42:58 - 43:12)

... All right, we're gonna hope that that doesn't come back for good. I enjoyed that not happening like it did last week so everyone please cross your fingers for me.

(43:13 - 43:23)

Um, okie dokie. Okay, so don't worry about that. I will say also from, you know, some testing I've done from preparing for this tutorial.

(43:24 - 43:34)

I don't necessarily know whether it's Astronify or Pyo, which Astronify is built on. It has some technical difficulties. So you might encounter those now.

(43:34 - 43:53)

So one of the most common problems I've noticed is a persistent clicking noise during the sonifications when you play them from the terminal. I have saved the audio as a file which is how I'll play it for you because it's from the terminal doesn't seem to play very nicely over Zoom. That does not have that persistent clicking noise.

(43:54 - 44:06)

I don't know why it's in the terminal. I assume it's some sort of bug. I'll also say that sometimes there will be a persistent clicking noise for a couple of seconds before the sonification actually begins.

(44:06 - 44:25)

Don't panic. And then occasionally, especially if you didn't have headphones plugged into your computer when you started the terminal session and then you go and plug the headphones in. When you try to actually play a sonification, it'll throw a little bit of a fit because it can't find the right audio output anymore.

(44:26 - 44:39)

None of these are insurmountable issues. Sometimes the worst case scenario for any of these bugs is clicking noise, which is a little bit of a nuisance. Or hopefully none of you have to do this because this would make it very difficult.

(44:39 - 44:50)

You'd lose your place. But restarting the terminal session. I don't want you to have to do that right now, but if you encounter these bugs in your own time, don't panic.

(44:50 - 44:59)

It's not an insurmountable problem. It's just a little bit of bugginess just because this is a little bit of a smaller package. So just want to give you a heads up.

(45:00 - 45:10)

Hopefully the helpers can help you in the chat. If any of these problems come up, you might see them crop up in my own terminal. But as I said, we have the sonifications all already saved.

(45:10 - 45:20)

So one way or another, you will hear them. On that note, let's get sonifying. So I'm going to make an instance of the sauna series class.

(45:20 - 45:43)

That's going to be our sonification object. And because I am sonifying Philadelphia temperatures, I'm going to call this SONI underscore Philadelphia. "S-O-N-I line." Philadelphia. "P-H-I-L-A-D-E-L-P-H-I-A." SONI underscore Philadelphia equals.

(45:43 - 45:53)

"Space equals." And then I'm going to do SONI series. Again, remembering that the S's in SONI and series are capitalized.

(45:53 - 46:05)

"S-O-N-I-S-E-R-I-E-S." So that's SONI series, capital S-O-N-I, capital S-E-R-I-E-S. Open parenthesis.

(46:06 - 46:26)

T-B-L. So make a SONI series object from the data in our table, T-B-L, comma. Remember again that we need to tell the SONI series class what our time and value columns are called.

(46:26 - 46:38)

In our case, we've just made the time column. That's going to be our time step column. And then our value column, the thing that's going to change over time is going to be Philadelphia temp.

(46:38 - 46:55)

So let's tell SONI series that now. So we're going to do time underscore C-O-L. "T-I-M-E, line C-O-L." T-I-M-E underscore C-O-L, time call equals. "Equals." Open quotation mark.

(46:55 - 47:06)

"Tick." Time step, all one word, because that's what we called that column. "T-I-M-E-S-T-E-P." Close quotation marks. Then add a comma. Comma.

(47:07 - 47:10)

Space. "Space." Val underscore call, V-A-L.

(47:10 - 47:17)

"V-A-L, line C-O-L." That's for value column. Space equals.

(47:17 - 47:22)

"Equals." Or I guess not space. We can do space here.

(47:22 - 47:29)

"Space equals space." That's probably the proper form. And then that column is called Philadelphia.

(47:29 - 47:31)

Don't forget your quotation marks. "Tick." Philadelphia.

(47:31 - 47:44)

"P-H-I-L-A-D-E-L-P-H-I-A." Underscore temp. "Line T-E-M-P." For temperature, close your quotation marks. "Tick." Close your parenthesis.

(47:44 - 48:09)

Right paren. And hit enter. My goodness, I'm getting a Zoom audio alert now? "Settings window. Security. Close closes the window. Desktop windows." Can you hear me? Yes, we can. Indeed. Oh, my goodness.

(48:10 - 48:13)

Wow. We really can't win with the audio issues here. All right.

(48:13 - 48:21)

We will plow forward. I got an audio alert saying that my audio device wasn't detected anymore. I didn't even try to play anything yet.

(48:21 - 48:26)

All right. This is a rather long line of code. So maybe if one of the helpers wants to paste that into the chat.

(48:26 - 48:37)

This is a good line of code to become familiar with, though, because we're going to use it over and over. Basically just changing the name of the sonification. So we won't sonify Philadelphia all the time.

(48:38 - 48:48)

And then also changing the name of the value column from Philadelphia temp to some other column that we might be interested in. All right. Next I'm going to sonify this thing.

(48:48 - 49:05)

S-O-N-I line P-H-I-L-A-D-E-L-P-H-I-A. So what I've done here is I've typed soni underscore Philadelphia again. That's S-O-N-I underscore Philadelphia to call that object again.

(49:05 - 49:11)

Then I'm going to hit dot. It wants to do something else. I don't want it to do that.

(49:11 - 49:22)

And I'm going to do dot sonify. That's S-O-N-I-F-Y. I'm going to do open parenthesis, close parenthesis.

(49:23 - 49:29)

And then that's it. I'm going to hit enter. So it's S-O-N-I underscore Philadelphia dot sonify.

(49:29 - 49:42)

So make the sonification out of the SONI Philadelphia SONI series instance. So make the sonification. This isn't going to play the sound quite yet.

(49:43 - 49:48)

So just hit enter. It's now sort of done all of its pitch mapping. It's ready to go.

(49:48 - 49:58)

It's found how it's going to represent the sounds. The last step is to play the thing. I am not even going to, well, I'll try once to play it from terminal.

(49:59 - 50:17)

I don't have the highest hopes, but we'll cross our fingers. We have the audio file ready. And I want you to keep in mind from last week that, remember, Astronify maps pitch to the value such that, by default, higher values are mapped to higher pitches.

(50:18 - 50:33)

So what you should hear here is higher temperatures represented as higher pitches and lower temperatures represented as lower pitches. I think the sonification is pretty fun. My office mates tell me that they think it sounds like wind, which is not a bad thing.

(50:34 - 50:42)

And the way we are going to play the sonification, you might recall from last week, is SONI, S-O-N-I. S-O-N-I. Underscore Philadelphia.

(50:43 - 50:54)

"Line P-H-I-L-A-D-E-L-P-H-I-A." Dot play. Open parenthesis, close parenthesis.

(50:56 - 51:05)

SONI underscore Philadelphia dot play. If anyone wants to take bets over whether this is going to actually work on Zoom, please feel free. Now is your chance.

(51:05 - 51:18)

Hopefully it will work on your computer. It works on my computer when I'm not on Zoom. There just seems to be an ongoing disagreement going between Zoom and whatever terminal audio situation is going here.

(51:18 - 51:41)

So I don't have the highest hopes. Desktop. ... "in 21." All right. I don't believe that that worked.

(51:41 - 51:45)

Hopefully you can hear me again. It did. Oh, it did? It worked.

(51:45 - 51:52)

It worked. Can you redo it without your screen reader on? Yeah. Oh, my goodness.

(51:52 - 51:56)

I am absolutely astonished. All right. I'm going to turn off NVDA.

(51:56 - 52:03)

I'm going to type that SONI Philadelphia again, dot play again. I stopped it early. So we didn't hear the whole thing.

(52:04 - 52:08)

And we'll cross our fingers that it works a second time. Wow. I would have just lost that bet.

(52:08 - 52:24)

"Speech mode off." Do you know about the up button? You mean to the up button? Yeah. You don't have to retype this stuff if you've already retyped, if you've already typed something.

(52:24 - 52:27)

Yeah. Yeah. I could, but I want to, you know, I don't want to.

(52:28 - 52:30)

Sorry. Sorry to interrupt you. Gotcha.

(52:30 - 52:33)

Gotcha. But, yeah. Yeah.

(52:33 - 52:36)

Okay. Here we go. Let's cross our fingers again.

(54:16 - 54:22)

Okie dokie. I'm hoping that went through. And you might notice that's a rather long sonification.

(54:22 - 54:33)

That's because we have a lot of data points there. So it takes a long time. Last week, actually, you might remember me changing the note spacing by doing, like, the dot note spacing.

(54:33 - 54:40)

There I was slowing it down because we had relatively few data points. There we had, like, 100. Here we have over 9,000, I believe.

(54:40 - 54:48)

We don't want to slow this down anymore. This is already, I think, over a minute of sonified data. So we definitely don't want to slow it down.

(54:48 - 54:55)

You can play around with the note spacing and speed it up if you want. I didn't think it worked very well sped up. I think it was a little hard to interpret.

(54:55 - 55:10)

But, you know, feel free to putter around with that if you would like. The default note spacing is, I think, .01 seconds between each note. So feel free to go wild and change that, just like we did last week, if you want to.

(55:10 - 55:27)

There are a couple of things I want you to think about. First, notice how you could hear those up and down cycles of the temperature going up and down and up and down at sort of regular intervals. You are literally hearing the turn of the seasons.

(55:27 - 55:43)

You're hearing years go by in the data as those moments of higher pitches are the miserably swampy Philadelphia summers, and the moments of lower pitches are the miserably cold Philadelphia winters. I do actually really love Philadelphia. Don't get me wrong.

(55:44 - 55:59)

But so you're hearing the cycle of the seasons in the sonification. That is one of the fastest non-visual overviews of the data to get that sort of sense of the shape that we can do. Just think about it.

(55:59 - 56:35)

If you didn't know Philadelphia was sort of a temperate place, saying you don't know

whether it's the desert or Antarctica, where there's maybe not as much of a cycle of a season, all you have to do is play that, and you know immediately, ah, Philadelphia has seasons. Another thing that you might want to notice from the data, something that might be adding to sort of that wind-like effect that we have going there, it's a little what I'm going to call wobbly. It is not like a perfect wave-like pattern where it goes up and down and up and down perfectly like that sine wave that we made last tutorial.

(56:35 - 56:53)

That's because this is real data, so you could get like an anomalously hot day in the winter and a nice cool day in the summer. At least you could dream of a nice cool day in the Philadelphia summer. So the data is not following a perfect trend, and I think that sort of contributes to, yeah, that spooky wind vibe.

(56:54 - 57:22)

All right, so that's our first sonification of the day and your first sonification with real data. And also, I'm not going to play it again now because it's rather long, but I totally encourage you if you want to play it again later, you can listen, maybe see if you can count all 15 years of our data in there, things like that. But I think one of the best ways to learn more about this sonification is going to be to sonify something else and compare it.

(57:22 - 58:16)

So let's dive right into our next sonification. We are going to sonify the daily temperature in Delhi, India. So let's make a new instance of the Soni Series class. Actually, before I break into that, I just want to see do I have any questions because we've finally burst into the sonification scene. So if there are questions, I'll address those now. When you just played that sound and it repeated, does it repeat indefinitely or do you have control over that? Yeah, that's an excellent question.

(58:16 - 58:26)

So it sounded like it was repeating, but actually those were just multiple years of data we have. So it won't repeat indefinitely. It will repeat for those 15 years of data we have.

(58:26 - 58:37)

So there were 15 cycles there. Now, I know this is rather a long sonification. So if I didn't want to listen to it until the end, which we did this time, we listened to it until it stopped.

(58:37 - 58:51)

So that stopped on its own. That wasn't going to go on forever. If I want to stop it early, I can type soni underscore Philadelphia dot stop, open parenthesis, close parenthesis, and

the same way we did dot play.

(58:51 - 58:59)

So stop and play are sort of parallel to each other. I might do that later. I should probably even add that to the curriculum because I don't think I wrote it down there.

(58:59 - 59:14)

But yes, it is not indefinite. It goes basically the time is determined by how many data points you have and then the duration of each note and then the spacing between each note. We're using the default duration and spacing.

(59:14 - 59:22)

So of the notes that Astronify gives us. But again, you can play around with those. Those will affect how long the sonification is.

(59:24 - 59:45)

So yes, this is a rather long one, but it's not indefinite. All right. I hear some silence.

(59:45 - 59:56)

So I'm going to turn NVDA's speech back on. "Speech mode beeps. Speech mode talk." Excellent. We are back in business. And I'm going to make that sonification of the Delhi, India temperature.

(59:57 - 1:00:13)

So I'm going to do Soni underscore Delhi. "S. S-O-N-I." And Delhi has, I guess I'm probably butchering the pronunciation, but it's like Delhi, so an H after the L. So Soni, S-O-N-I underscore.

(1:00:13 - 1:00:26)

"Line D-E-L-H-O dot play." That O was a typo. So that's S-O-N-I underscore D-E-L-H-I.

(1:00:26 - 1:00:52)

Space equals. "Space equals. Space." And we're going to make this, again, an instance of our Soni series class. So that's capital S-O-N-I. "O-N-I." Series. Capital S again. "S-E-R-I-E-S." So Soni underscore Delhi equals Soni series with capital S's. Open parenthesis. T-B-L.

(1:00:53 - 1:01:19)

"T-B-L." Comma. "Comma." So again, make a Sona series object from the table T-B-L.

Space. "Space." Time underscore call. So call C-O-L like column. "T-I-M-E." Underscore call like column. "Line C-O-L." So time underscore call equals time step in quotation marks just like last time.

(1:01:19 - 1:01:30)

So our time column is again that column that we made that we called time step. "Equals T-I-M-E-S-T-E-P." Quotation mark.

(1:01:30 - 1:01:33)

Tick. Or tick. Comma.

(1:01:33 - 1:01:44)

"Comma." Space. "Space." Val call for the value column. V-A-L. "V-A-L." Underscore call C-O-L. "Line C-O-L." Equals.

(1:01:44 - 1:01:55)

"Equals." Open quotation. "Tick." And then the Delhi temperature column is called Delhi underscore temp. "D-E-L-H-I." Underscore temp.

(1:01:55 - 1:02:02)

"Line T-E-M-P." And then close your quotation marks. Close your parenthesis.

(1:02:02 - 1:02:07)

"Right paren." So to summarize, that is SONI with an I. S-O-N-I. Underscore Delhi.

(1:02:08 - 1:02:12)

Equals. SONI series with those capital S's. Open parenthesis.

(1:02:12 - 1:02:18)

T-B-L. Comma. Time underscore C-O-L equals quotation marks time step.

(1:02:19 - 1:02:26)

Close quotation marks. Comma. Val underscore call equals open quotation marks Delhi underscore temp.

(1:02:26 - 1:02:31)

Close quotation marks. Close parenthesis. So make a SONI series a SONIFICATION object.

(1:02:31 - 1:02:42)

Call it SONI Delhi. From the table T-B-L, call the time column time step. And the value column is the daily temperature in Delhi, India.

(1:02:42 - 1:02:45)

Hit enter. "In 23." Good to go.

(1:02:45 - 1:02:56)

And then these next two lines should hopefully be familiar. It's going to be SONI underscore Delhi. "S-O-N-I." Underscore Delhi. "Line D-E-L-H-I." Dot SONIFY.

(1:02:57 - 1:03:06)

"Dot S-O-N-I-G-Y." Ooh, not G. Y-G-F-Y. SONIFY, not SONIGUY.

(1:03:08 - 1:03:12)

That's not it. "Left right paren." And don't forget those parenthesis at the end.

(1:03:12 - 1:03:29)

So again, this is just going to actually SONIFY our data. So map things to pitch, you know, map the values to the pitches the way we want. "In 24." And then SONI underscore Delhi dot play. "S-O-N-I. Line D-E-L-H-I.

(1:03:30 - 1:03:46)

Dot P-L-A-Y." And again, don't forget those parenthesis, open and close. "Left right paren" Because this is, you know, a function we're calling, but we're not entering any parameters. So we just leave some empty parenthesis there. And we will again cross our fingers that this works.

(1:03:47 - 1:03:51)

And we'll hit go. And again, this is going to be a rather long one. Pile warning.

(1:03:52 - 1:05:33)

Speech mode off. All righty. Pretty cool, huh? So I noticed a couple of things about that Delhi temperature sonification.

(1:05:33 - 1:05:42)

First, I noticed that there was, like, less noisiness. It seemed, what I mean by that, like there was less wavering. So it seemed more of a smooth trend.

(1:05:42 - 1:06:05)

So maybe Delhi is less likely to have that random, like, hot winter day that Philadelphia could have. It also sounded to me like there is more variation between, you know, the hottest and coldest days. But one of the things I noticed is that, like, on average, the pitch, like the average pitch of that sonification was pretty similar to the average pitch of the Philadelphia sonification.

(1:06:06 - 1:06:36)

And I don't, I've never been to India, but I kind of expected Delhi, India to be a lot hotter on average than Philadelphia. And thus I would expect, naively, the average pitch of our sonification, knowing that higher temperatures are represented by higher pitches, I would expect the pitch of our sonification to be higher on average. I'm going to check that because, you know, I don't know for certain whether maybe Delhi is actually really cold.

(1:06:36 - 1:06:47)

So we've already done this for Philadelphia. And this is one of the reasons why I recommended you do this before you sonify something. Now I have to go back and check the average temperature in Delhi, India.

(1:06:49 - 1:07:06)

And I'm going to check this using the data frame. So our data frame DF still exists in this IPython session, even though we've been using TBL, or tables, since. That fromPandas function, not going to destroy our nice data frame, fortunately.

(1:07:06 - 1:07:28)

And as I mentioned before, the syntax, like the way we access AstroPy tables is like just different enough from Pandas data frames to be inconvenient. And frankly, unless you intend to be an astronomer, you don't need to learn how to deal with AstroPy tables. I think sometimes even astronomers don't know how to deal with AstroPy tables.

(1:07:28 - 1:07:36)

So let's work. And the data in our data frame is the same as the one in our AstroPy table. So let's work with the data frame.

(1:07:37 - 1:07:47)

So I want to test the average temperature in Delhi, India. So that's going to be DF. Oh, I got to turn my speech back on.

(1:07:47 - 1:07:50)

"Speech mode beep. Speech mode talk." Got to remember to do that.

(1:07:50 - 1:08:03)

Okay, so that's DF. Dot Delhi underscore temp. "Dot D-E-L-H-I line T-E-M-P." Dot. "Dot." Mean for average.

(1:08:03 - 1:10:40)

"M-E-A-N left paren, right paren." That's DF dot Delhi underscore temp dot mean, open and close parenthesis. "Out 25, 76.96521316783594 in 26." All right, so that's like what almost 77 degrees Fahrenheit on average and I recall that the average temperature in Philadelphia was like 56 degrees Fahrenheit so this is about 20 degrees hotter on average in Philadelphia so it's sure like on average a lot hotter. Let's hear maybe like what the minimum temperature in that Delhi, India column temperature column is so that's going to be DF dot Delhi underscore temp dot mean, open and close parenthesis. "F-G."

Whoops, that's not correct. "G-F-D-F" Dot Delhi underscore temp.

"Dot D-E-L-H-I line." Okay, sorry, checking on the guide dog heard a suspicious noise didn't like it. "T-E-M-P." DF dot Delhi underscore temp dot min. Dot M-I-N. Open and close parenthesis. "Left parent right paren out 26 43.9 in 27. "All right, so the coldest temperature we have recorded for Delhi is 43.9 degrees Fahrenheit. And I think it was what like 12 ish for Philadelphia so it is never getting as cold in Delhi as it gets in.

Well, I mean, some days it'll, but like it never gets as cold as the coldest temperature in Philadelphia, and that is not immediately obvious from the sonification. What gives? Well, what is happening is we are creating individual soni series objects for each of our sonifications. And in order to make a sonification that sounds nice, Astronafy maps and scales.

Each of these sonifications individually. What do I mean by this? Well, they map the values of our temperatures to pitches independently of each other. So a note that we hear in the Delhi sonification is not even if it's the same like pitch, it's not representing the same temperature that we hear in the Philadelphia sonification, because the pitch mapping is different, because the scaling is different by default.

(1:10:40 - 1:11:11)

So again, these are the default settings and they do this to make it sound nice. And in fact this is not a unique thing to Astronafy or sonifications. The most common Python visualization package matplotlib is going to automatically scale our visualization axes based on the data points you give it as well.

So this is not a sonification unique thing. But, because these are the default settings we do. Sorry, guide dog.

(1:11:12 - 1:11:56)

Having a moment. Okay, because these are the default settings, we can fiddle around with them. And the one setting we're going to want to change the pitch mapping parameter will want to change in particular in Astronify is something that they call the zero point. Basically, they pick some central pitch of the data. By default, they will pick the median value of our data and map that to a central pitch of 440 hertz. I can't like conceptualize in my mind exactly what 440 hertz sounds like, but that's like sort of I take it the center of the pitch range of the sonification.

(1:11:57 - 1:13:33)

And so every time what Astronafy does is it calculates the median of the data and says, all right, we're going to represent anything like this median value as 440 hertz in the sonification. But because Delhi and Philadelphia are such different temperatures, they have very different median temperatures that are being represented by that same pitch. Fortunately, we can change again what Astronify calls the zero point so we can change the value that it's mapping to that central pitch. So what I'm going to do is I'm going to change the, I'm going to change the central or not the central pitch, the zero point. Sorry, I'm going to change the zero point of the Delhi sonification to be equal to the median value of the Philadelphia temperatures because we know that the Philadelphia sonification is going to be scaled to its median by default. So I'll just set the Delhi sonification to be scaled to that number two.

I could scale them both to some arbitrary number I could set their zero point for both of them to be like I don't know 70 degrees. But at this point, you know, this way we only have to change one of the sonifications, we can just muddle around with one of them. So, the first thing I need to know is hey, what's the median value of the Philadelphia daily temperature, we've printed out the mean the average but I want to know the median because that's what Astronafy by default is using.

(1:13:34 - 1:15:02)

So to do that, I'm going to do, df, no, sorry lots of typos today, df.philadelphia underscore temp dot median, m-e-d-i-a-n "dot m-e-d-i-a-n," and don't forget those parentheses, "left paren, right paren, out 27, 57.0, in 28." Alright, so the median temperature in that Philadelphia temperature column is 57 degrees. Now, this line of code that I'm going to use is basically letting me go into the what Astronafy calls the pitch map arguments.

So the sort of parameters it uses to map the values of our data to pitches and the

sonification. I'm going to dig into those, and I'm going to change the zero point pitch map argument to be 57 of the Delhi sonification, because that'll make it the same as the Philadelphia zero point because by default, again, the Philadelphia zero point will be 57. It's median value.

This is a rather long line of code. I'll go through it here. Maybe one of the helpers can sort of fling it into the chat as well just because it's kind of got like a lot of dots and so on and so forth.

(1:15:02 - 1:15:19)

Alright, let's do SONI underscore Delhi. So I'm calling that object again, that's SONI underscore Delhi object. And then I want to go pitch underscore mapper.

(1:15:19 - 1:15:43)

"P-I-T-C-H line M-A-P-P-E-R." So SONI underscore Delhi dot pitch P-I-T-C-H underscore mapper M-A-P-P-E-R. So call the pitch mapper, call the way that the Delhi temperature data is being mapped to pitches.

(1:15:43 - 1:18:41)

And then we want to call the pitch map args, the pitch map argument. So sort of the parameters of the way these pitches are being mapped. So dot pitch "P-I-T-C-H" underscore map "line M-A-P"underscore args A-R-G-S line "A-R-G-S." So just to summarize so far, that's SONI underscore Delhi dot pitch underscore mapper dot pitch underscore map underscore args A-R-G-S. Open square bracket, open quotation marks, then zero underscore point, underscore point, close quotation marks, close brackets. So basically what have I done here? I've called the pitch map argument zero point.

So that thing we've been discussing, that point in the data that will be represented every time is 440 hertz. I'm calling that from this Delhi sonification. And I'm going to set it equal to 57 because that is the median value of our Philadelphia temperatures.

I'm just going to hit enter. All right. That bit is done.

The only two things we have left to do, we have to, in order for it to play properly, we have to re-sonify the data. So basically we have to remap the pitches, the values of our data to the pitches. "S-O-N-I line."

So that's SONI underscore Delhi D-E-L-A-I dot sonify, as we're familiar with, dot S-O-N-I "left right paren in 30." And before I play this new Delhi sonification, I'm going to first replay some of the Philadelphia sonification just so we have that basis of comparison. I probably won't let it play through the end.

I'll play it for a little bit. So at some point I'll call that like dot stop function, but I'm going

to turn off my, I'll type out that SONI Philadelphia dot play with NVDA speech on and then I'll turn it off so we can listen. "S-O-N-I line P-H-I-L-A-D-E-L-P-H-I-A dot P-L-A-Y left right paren." So that's SONI underscore, oh my goodness, Alana, you are going to knock my chair over. I'm so sorry. Okay, SONI underscore Philadelphia dot play open and close parentheses.

I'm going to turn off NVDA speech. And hit play, or hit enter I suppose. That audio sounds rather weak.

(1:19:03 - 1:19:14)

Give me give me a verdict in the room, folks. Can we hear that well enough or should I play it from the recording? It sounded a little weak on my end. It sounded pretty okay.

(1:19:16 - 1:19:29)

Maybe it's that. All righty then. All right, so just, I didn't let that play the whole way through because it's pretty long, but just as our basis of comparison for the Delhi sonification, which will now play I'm going to turn NVDA back on.

(1:19:29 - 1:21:25)

"Speech mode beeps, speech mode talk." All righty, and then I'm just going to do SONI underscore Delhi dot play "S-O-N-I line D-E-L-H-I dot P-L-A-Y left paren right paren" and we'll do the play "speech mode off." All righty.

(1:21:25 - 1:22:01)

So hopefully you could tell that the pitch in general was way higher with that new Delhi sonification than with the original and then with the Philadelphia temperature sonification. Now the pitch mapping still isn't, I don't believe, identical just based on the way that Astronify maps pitch, but this gives us a little bit more of a comparison. Now playing those two sonifications side by side, we can definitely, definitely hear that the average temperature, like in general, the temperature in Delhi is higher than the temperature in Philadelphia because the pitch is so much higher in that sonification.

(1:22:01 - 1:22:12)

Now I personally think that this sonification with the higher pitches doesn't sound as good. It's not as informative either. You can't hear like the changes between seasons as clearly.

(1:22:13 - 1:22:46)

There is quite simply a reason that Astronify, the Astronify team chose to map the pitches in the way they did, that those defaults are set the way they are, and it's

because it generally makes a pretty good sound. I might be a little biased because I was involved in the usability testing for Astronify, but there is a reason that those defaults are the way they are. So I'm not necessarily telling you that when you have to change the pitch scaling and like the zero point every time, I'm not telling you to do that.

(1:22:46 - 1:23:19)

I'm telling you to be aware of the fact that if you are comparing two sonifications that the scaling might be different, and that is no different from a visualization, wherein, you know, for instance, a plotting sort of software like Matplotlib will scale the axes of your visualization automatically based on the points. It's something to be aware of. We need to be a little bit more aware of it in a sonification than a visualization because visualizations generally have like axis labels where they'll have like, you know, the upper and lower limits of the axes like written on there.

(1:23:19 - 1:23:57)

We don't have like a marker in the sonification for like what the upper and lower values are. You have to sort of go and check the data yourself to do it, which is why I was sort of harping on about how we really ought to do that with our data. And, you know, it's just something really to be aware of, more or less.

I'm really interested in this problem of sort of axis labeling in sonification. How do we give people that information. So if you have thoughts about it, feel free to pop by office hours or to, you know, shoot me an email.

I'm always happy to chat about that. It's something I'm very interested in. But for now, I just want to make you aware of it.

(1:23:57 - 1:24:44)

All right. Sonification scaling. Check.

Any questions in the room? In practice, how often do you have to do this, like scaling, tuning? Is this like something that you like listen to the sound from like multiple scales? Or what's practice look like for you? Um, so for me, personally, like when in my daily life, the honest answer is that the computer I use daily won't install Astronify. That's the honest answer. That is the most honest answer.

(1:24:45 - 1:25:30)

In terms of what I expect back from, you know, when I had a computer that ran Astronify, you might have noticed like when I opened the terminal, it says Rob, because I'm borrowing my friend Rob's computer. Thank you, Rob, hero of these tutorials. Um, that sounded like sarcasm, but I mean it seriously.

Anyway, back when I had a computer that ran, that ran Astronify properly, you know, it's going to depend on a couple of things. I mean, realistically, I personally probably wouldn't fiddle with them too much, in part because I think Astronify, the way it maps things by default does give you the best sound. And regardless, you're going to have to check sort of your maximum and minimum values anyway to sort of know where you're sitting in terms of data values.

(1:25:30 - 1:25:43)

It's going to kind of depend on your goal as well. Like if your goal really is to compare two things, then yeah, you might want to fiddle around with this. But if not, you know, it's really going to depend on what your goals, what you want to do.

(1:25:44 - 1:26:32)

There might be other parameters in the sonification that are more typically changed. So in our next little subsection right now we're going to talk about an invert pitch map argument where we like flip the pitches. I'll get into that in a second.

But I know that I think other usability testers found that like quite useful just regularly to sort of flip the pitches, based on the use cases that Astronify is built for, which is, if you'll recall from last week, these light curves, this type of astronomical data. So, it's going to be a mix of like, what are your goals and what are your preferences, realistically. Thank you.

(1:26:32 - 1:26:53)

All right, I hear silence in the room. So, let's get cracking on the next bit. The only thing to do is to sonify something else.

(1:26:53 - 1:27:06)

"Speech mode beeps. Speech mode talk." Thank you.

All right. Next, let's sonify a very famous city's temperature. Let's sonify New York City's daily temperature.

(1:27:06 - 1:27:20)

I am going to call this one Soni underscore NYC because I do not want to type New York City every time. Too many letters. And now Patrick can tell us whether these values are reasonable because Patrick knows.

(1:27:21 - 1:27:48)

So, I've been to New York before. All right, let's do Sonny underscore NYC. S-O-N-I-N-Y-C.

Sonny underscore NYC, space equals space. Space equals space. Hopefully this is starting to get familiar.

We're going to call the sauna series class again. We're making a new instance of it with the New York data. So, remember those capital S's in sauna series.

(1:27:48 - 1:28:01)

"S-O-N-I-S-E-R-I-E-S." Open your parentheses. "Left paren." Type T-B-L for table. "T-B-L." Hit comma.

(1:28:01 - 1:28:25)

"Comma." Space. "Space." We want to specify our time column or time underscore call equals time step. Again, remembering those quotation marks around time step. "T-I-M-E line C-O-L equals tick T-I-M-E-S-T-E-P." Close your quotation marks. Tick. Hit a comma.

(1:28:25 - 1:28:46)

"Comma." Hit space. "Space." And now we want to specify our value column or val underscore call. "C." Nope, not C. "Val C-O-L equals N." Now it's trying to finish for me. Nope.

Me first. Val, V-A-L. "V-A-L." Underscore call. "Line C-O-L." Equals.

(1:28:46 - 1:28:51)

"Equals." Quotation mark. New York City Temp.

(1:28:51 - 1:29:00)

That's New underscore York underscore City underscore Temp. Close quotation marks. You can see why I wanted to abbreviate it to NYC.

(1:29:00 - 1:29:21)

"Tick. N-E-W line Y-O-W." Nope, not W. "W-R-K." New York. New underscore York. "Line C-I-T-Y line T-E-M-P." Tick. Right. All right.

(1:29:21 - 1:29:29)

That's SONI underscore NYC equals SONI series with those capital S's. Open parentheses. T-B-L.

(1:29:30 - 1:29:36)

Comma. Time underscore call equals in quotation marks time step. Close quotation marks.

(1:29:37 - 1:29:40)

Comma. Space. Val underscore call.

(1:29:40 - 1:29:44)

C-O-L. Not call like phone call. Call like column.

(1:29:44 - 1:29:54)

Val underscore call equals open quotation marks. New underscore York underscore City underscore Temp. Close your quotation marks.

(1:29:54 - 1:30:03)

Close your parentheses and hit enter. "In 34." So we've just made a new instance of the SONNY series object for the New York City temperature.

(1:30:04 - 1:30:24)

All right. Next, this should be familiar. SONI underscore NYC. "S-O-N-I." Underscore NYC. "Line NYC." Dot SONIFY. Map our values to pitches. Dot S-O-N-I-F-Y. "Star." Nope, not star. Open and close parentheses.

(1:30:24 - 1:30:38)

"Star. Left paren. Right paren." SONNY underscore NYC dot SONNIFY. Open and close parentheses. Again, remember we've abbreviated New York City as NYC here because I could not be bothered to type that every time.

(1:30:39 - 1:30:53)

"In 35." And then SONI underscore NYC. S-O-N-I-line-N-Y-C. Dot slay. Dot P-L-A-Y. Open and close the parentheses.

(1:30:53 - 1:30:57)

"Left paren. Right paren." And I'll turn off NVDA speech.

(1:30:57 - 1:32:17)

"Speech mode off." And we'll hope it works again. All right.

(1:32:17 - 1:32:28)

I'll stop that a little bit early. I just did that dot stop instead of dot play. But I'm wondering if anyone could hear -- I'm hoping it came through on Zoom because I could hear it on my end, though it's a little subtle.

(1:32:28 - 1:32:50)

Could anyone hear in there that there is this odd sort of low note, this sort of bah, bah, bah, really low-pitched note that just sort of showed up again and again sort of at irregular intervals? Could people hear that over Zoom? We heard it. I was wondering if that was a really cold day or something, like a sub zero or something like that. Yeah, yeah.

(1:32:50 - 1:33:05)

So it sounds like this really low value that's like coming up again and again and it's like kind of suspicious. Like it's the same pitch every time that I could tell it wasn't just showing up in the winter. So it has to be like sub zero in the summer.

(1:33:06 - 1:33:25)

That sounds a little questionable. So let's check to see whether that was just some like weird sonification effect because I know we've had some crackling noises, you know, things have been a little strange. So what am I going to do? I'm going to pull out that invert pitch thing that I actually just mentioned to Tony.

(1:33:25 - 1:34:04)

So built into Astronify in that pitch map args thing that we used before is another option called invert, which basically flips the pitches such that now lower temperatures will be represented by high pitches and high temperatures will be represented by low pitches. Now, this made a lot of sense given the use case for Astronify, what they were trying to do with it. But it's going to be really useful for us here because I think that those really low notes are going to be easier to hear as a high little like chime basically than as that sort of like sort of in the background of our sonification.

(1:34:04 - 1:34:20)

So let's invert our pitches. This line of code is going to look really similar to what we did to switch the zero point except instead of calling zero point here, we're going to call invert. So it's going to be Sonny underscore NYC.

(1:34:21 - 1:36:23)

Oh, got to turn speech back on. "Speech mode be speech mode talk." All right, Soni underscore NYC. Oh, and underscore NYC "line NYC" dot pitch mapper again we're calling that pitch mapper the way we bring you know the values of our data to pitch. "P. I. T. C. H. Line" dot pitch underscore mapper "M. A. P. P. E. R". dot pitch mapargs A. R. G. S. "A. R. G. S." So that's Soni underscore NYC dot pitch underscore mapper M. A. P. P. E. R. dot pitch underscore map underscore args. Then open our brackets open our

quotation mark. And in this case we're calling the pitch map argument invert. That's I. N. V. E. R. T. Take right bracket open and close in your bracket and your parentheses not in that order. Wait what no close your quotation mark in your parentheses.

I am struggling with these today, and we're going to set this equal to space equals. True with a capital T. So set Soni underscore NYC dot pitch mapper dot pitch map args in brackets and quotation marks invert equals true with a capital T capital T. R. U. E. E. Good grief T. R. U. E. Set the pitch map invert argument equals true. So basically by default invert is false.

So that means low values are low pitches. That's kind of intuitive to me here. We're just going to flip it around.

(1:36:24 - 1:43:56)

"In 38" and hit enter. And then we're going to signify Soni underscore NYC S. O. N. I. Underscore NYC line and N. Y. C. Dot signify dot S. O. N. I. F. Y. Open and close your brackets or your parentheses not your brackets. Left parent right parent.

So Sonny underscore NYC dot sonify open and close those brackets and hit enter "in 39." So now we've remapped our values to pitches with that new invert parameter and then Soni underscore NYC "S. O. N. I. Dot. N.Y." Sorry not dot. Not dot Soni underscore NYC "line and N. Y. C." Dot play "dot P. L. A. Y. left paren right paren" open and close those parentheses. And then you can just hit enter.

I'm going to turn off NVDA again speech mode off and I might stop and you know I might not make us listen to this whole thing I want us to hear the notes again and remember we flipped the pitches so I would now expect those low notes if they're not some like audio effect from the sonification to show up as rather obvious high notes. All right, I'll stop that early because I think we've heard quite a few of them those little Those high notes that are in fact the same high note. So it sounds like the same pitch kind of at random intervals to me.

So again, remember that high note is is is low values because we've inverted the pitches. What gives why is that in there? What is it like? Are there a bunch of like negative 10 degree days in New York City? Well, my first question is again like we learned last time. We didn't really check the scaling of this thing.

Like maybe the whole New York City temperature column is totally just like a mess. So let's check the average temperature in New York City first to see whether that looks roughly acceptable "speech mode beeps speech mode talk." Hopefully this is sort of familiar.

This is going to be df dot new underscore York line. Okay. Nope.

df.new_york_city_temp.mean() open and close parentheses "out 41, 55.6406152185 56449 in 42." Right. So the average temperature in New York City is 55.6 degrees.

That's kind of "selected." Whoops. Oh no. "Unselected" the so so that's what like a degree ish lower than the average temperature in Philadelphia so that that doesn't seem too questionable. So let me check. We know that those high notes we were hearing are low values because we flipped the pitches.

So let's check the minimum temperature in this column and see what's going on. Oh, so that's going to be df dot new underscore York underscore city underscore temp again because in our data frame this column is called New York City temp and then it's going to be dot min open and close parentheses line. "df.new_york_city_temp.mean() out 42, minus 99.0, in 43."

Whoa, minus 99 degrees Patrick does it ever get down to minus 99 degrees in New York City. Yeah, you see the bricks exploding. So what's going on why are there negative 99 in our data.

Well, what turns out is happening is that there are some days where we don't have data for the temperature in New York City, and in this particular data set they said well we got to put something there we have to put something well, well let's just put a value that no one will think it actually gets to it in New York City yeah negative 99 degrees no one's gonna think it gets to negative 99 degrees that's fine. Just fill all the empty spaces with negative 99, but a stratify has no way of knowing that the people in this that made this data set we're like missing value, negative 99. So it just goes on its merry way and solidifies it, which is why we're hearing this sort of repeated identical tone.

It's because every time there's a missing data point is solidifying it as if it's real data. Now in the spirit of complete transparency. All of these columns originally had missing data.

And for the columns we have already solidified here so that we have something nice and clean to work with. I basically replaced those missing values with a method that we call linear interpolation. I'm not going to get into that right now.

If you're curious about how I sort of pre process the data to make it usable for us here. Feel free to reach out to me and I'll sort of just, you know, fling the dot py scripts I use that you and you can take a look at it, but all of the, you'll often find in real data that we have missing values. This is sort of like a fact of life when you work with data that there's going to be missing values.

Sometimes it'll be filled with some, you know, some, you know, impossible numbers so that you can easily identify all that's not real data, as they've done here, but sometimes what you'll see in data science, especially in Python, is you'll see missing values filled in

as something we call an NAN, or you'll sometimes hear people call it NONs or NANs. I call them NONs, my office mates make fun of me because they call them NANs, but they're NANs, or more accurately a NumPy NAN. The pandas has its own version too.

What is this NAN thing? It stands for not a number. And so often if you have like a blank space in a data frame, it will just get parsed as a NAN, a NAN, as a not a number, there's not a number there. Now, a lot of functions, especially like pandas and NumPy functions are sort of accustomed to these NAN things, these missing value placeholders, and will sort of ignore them in the way you'd hope.

(1:43:56 - 1:47:03)

But sometimes they will wreak havoc on some functions, and then you'll spend forever like hunting through your data, where's the missing value, where is it, oh my goodness. Fortunately, there's a really quick and easy way to check whether or not our data has an NAN, a not a number, this like not an actual number value in it. And that is with the NumPy is non or is NAN, is N-A-N function.

So if we do NumPy, N-U-M-P-Y, dot is, dot I-S, N-A-N, N-A-N. So all one word, NumPy, dot is N-A-N, open your parentheses, "left paren", D-F, "D-F", dot New York City, D-F dot new underscore York underscore city underscore temp, so calling that column of D-F with, with, you know the New York City temperatures in it. "Dot, N-E-W, line, Y-O-R-K, line, C-I-T-Y, line, T-E-M-P, right paren." So that should read NumPy dot is N-A-N, open parentheses, D-F dot new underscore York underscore city underscore temp, close parentheses. So basically, please NumPy, check if there are nons in this column of the data frame, and hit enter. All right, I'm not going to let it finish.

But what it gives, that's a lot of falses. Well, what it's doing is it's giving us an array, like a, you know, like a big old chunk of values, what we remember learning from Patrick are Booleans, so true or false values. It has the exact same shape as this column.

So we said, hey, check if this column has NANs, and it's returning a Boolean array of the same shape as the column we asked it to check, where it says false at any location where there is not a NAN, so there's not a number. And if there is a position where there is a NAN in that data frame or that column we had it check, it will return true. Okay, but I don't know about you, but I do not want to read 9,265 rows of falses to check whether there's a single true in there.

That sounds horrendous. What will we do instead? We're going to call the .any function. So if we type that exact same line that we just did, and then tack a .any, open and close parentheses at the end, that'll tell us whether there is a single true value in that array.

(1:47:03 - 1:48:15)

So basically, even if 9,264 of those values are false and one of them is true, then .any

will return true. So basically say, are there literally any NAN values in here? So that's going to be numpy.isnan(df.new_york_city_temp) And it's trying to tell me, yes, you want to do .any, and it's right this time. numpy.isnan, open parentheses, df.new_york_city_temp, close parentheses, and then add at the end .any. And call the open and close those parentheses, leave them empty, and hit enter.

(1:48:16 - 1:49:18)

Out, false. There is not a single true value in the return of that isNAN function, which means there are no NANs, no not a numbers in this array. That's a good thing, because if there were missing values and they were at random filling some with negative 99.0 and others with NAN, I would be upset.

That would be messy. But when dealing with real data, it's good to know the different formats you could get these sort of missing values as. So we've checked now, there are no NANs in our data, good to know, but we have these missing values, these negative 99.0s. What if I do not want those little chimes in my data? What if I say, that's very annoying, please get rid of them? Well, fortunately, actually, before, this is going to be the last bit we cover in this tutorial.

(1:49:18 - 1:50:52)

So before I do this last bit, would anyone like to ask questions? I get ahead of myself. Any questions repeatedly coming up or otherwise we will hurdle through the last section of the tutorial. I'm hearing silence.

So let us get a move on. All right, let's do this last portion. All right.

So as I said, what if we don't want those little chimes for those negative 99.0s in the data that are just sort of there? What do we do? Well, fortunately, another pitch map argument that they call min-max percent allows us to sort of clip outermost values from the values we're sonifying from the sonification. So, for instance, suppose I set the min-max percent equal to 5,95. That means that anything, the 5% lowest values in the data are going to be excluded from the sonification.

(1:50:52 - 1:51:09)

And the 5% highest percent of the data, so the 5% greatest values, are also going to be chopped out of the sonification. Do not include them. Get rid of those.

We do not want them here. Cut them out. They are extraneous values.

(1:51:10 - 1:51:23)

So we can just chop out, you know, we know these are really low values. Chop out, you know, the lowest some percent of our data. Get rid of them.

Don't include them in the sonification. Please, Astronify, get rid of them. And then re-sonify the thing.

(1:51:25 - 1:51:35)

So let's get into that. It's going to look, again, very similar to when we change that invert equals true. It's also going to look very similar to when we change that zero point.

(1:51:35 - 2:22:12)

It's going to be soni underscore nyc dot pitchmapper dot pitch_map_args['minmax_percent'] ... ... close your quotation marks tick close your bracket right bracket so that is soniunderscore nyc dot pitch underscore mapper dot pitch underscore map underscore args a r g s open your brackets open your quotation marks min m i n max m a x underscore percent close your quotation marks close your brackets hit space space hit equals space and then this is going to be equal to a list type object so we're going to make a list it's going to have two numbers in it our lower bound and our upper bound now i'm going to make our lower and i'm going to put this in brackets because it's a list left bracket so open our brackets i'm going to make our lower bound five "five" what does that mean it means chop out the five percent like the lowest five percent of our data because i think that should chop out those negative 99s for us now there is also a min max value where you could cut out at exact values um there's a in the in the curriculum there's a description of the astronomy parameters that talks about this in detail i tested it i the way min max percent sounded a little better so we're gonna stick with that um so that's going to be equals open bracket five to cut out the five percent lowest data five comma comma space because we want to specify the upper limit and i'm not concerned about the upper limit here because uh because the missing values are negative 99s um we probably for the sake of completeness here really should have checked the maximum value in uh the new york city temperature column uh you know to make sure there was nothing really funky going on there um but uh for now take my word for it the missing values are those low numbers also again it would be chaos if they had missing values and filled them with like negative 99 and like a thousand that would be incredibly rude of them if you ever make a data set don't don't do that that's horrible um okay so then i'll do five comma space 100 1 0 0 so basically that means go up to the hundredth percentile go up to the maximum value of the data don't cut off anything at the upper end "right bracket in 46" all right so we've changed the min max percent again that's going to be soni underscore nyc dot pitch mapper dot pitch map args with a pitch underscore map underscore open your brackets open your quotation marks min max percent with the um with an underscore between min max and percent close quotations close brackets equals open brackets five comma space 100 close your brackets hit enter again we're just chopping off the five percent lowest data out of our sonification hit enter all right again we need to remap the data to pitch so that's uh soni underscore nyc dot sonify s o n i line sonny underscore nyc and n y c dot

sonify dot s o n i f y open and close your parentheses "left paren right paren in 47" and then soni nyc dot play "s o n i line n y c dot p l a y left paren right paren" and remember before i hit play here before i hit enter on sonny underscore nyc dot play recall that we have still inverted the pitches you haven't uh changed invert equals false um so just keep in mind here that uh that you know higher pitches are still representing lower temperatures that's okay here because i think what we're interested in is hearing whether those notes are sort of cut out and i think they're at least for me they're easier to hear as higher pitches maybe for you they're easier to hear as lower pitches but just keep that in mind that we haven't changed that argument so it's still there all right let's hit play oh let me turn off npda speech speech mode off all right here we go i've stopped that but mostly even though i hear some like sort of high pitch like almost kind of squeaky noises that aren't super pleasant they're not those individual notes that are like those so i'm not hearing anymore those really high pitched single notes the same pitch at sort of random intervals sounds better to me though again feel free to play around with the min max percent like where you put that cut off if you're interested um that concludes our discussion on missing values and also looking at the time it includes our discussion our tutorial for today because it's 8 p.m i will say there's even more material available online for the curriculum some model testing um so if you're curious about that i highly encourage you to go check that out there's also discussion of astronafi's different um arguments i just figured i'd rather provide more material online than we could get through because um you know i'd rather you guys have more to work with than less so if you're curious to go through a little more there's more online on the curriculum which was sent in the email today um and i'm happy to stick around and answer questions anything like that um i will uh i'll leave my screen sharing for the moment in case there are any coding questions before we wrap up all right i'm gonna stop sharing screen then because i'm not hearing coding questions i'm hearing silence and then i'll i'll turn my camera back on wow i'm here again all right well i'd just like to say thank you sarah for walking us through both last week and this week and for introducing us to this idea of sonification um i guess i do have a question for you you know and i'll first just you know say you know we're out of time so if anyone wants to uh to hop off the call maybe you have another uh meeting or something like that uh maybe you want to eat dinner then um then you know thank you i'll just say a thank you for joining us and thank you for uh for spending time learning some of these non-visual data science techniques um but sarah i did have a question which i guess is maybe you could give us the your um your opinion on the state of sonification or making sonifications and like if you think things are going to um get a you know a little more stable or a little more interesting in the next couple years like what's next for sonification yeah okay this is this is a really interesting question and it's a tough one to answer i think at least within astronomy which is where i'm most familiar there's a lot of interest in sonification um there's kind of been like an explosion of sonification sort of outreach things and and softwares to do things and inherently with interest comes improvement um however another necessarily necessary thing for improvement is like community cohesion and money um and so there has to be

i think you've talked about like these project grants that you would love to see from other you know organizations like there needs to be like some money put into these things so that people have like the funding to sort of make things more stable and also make things more standardized i've taught you how to use astronify here but a different sonification software might sonify things entirely differently so i would say in terms of there i i think of sonifications in roughly two aspects sort of these like beautiful outreach sonifications that's kind of like the universe of sound sonification that i played for you last week where it was like that galactic center sonification i think those are in really good shape people really like those um people's responses to those are really high they're really great for outreach and community engagement including among the blind low vision community things look good for outreach for the more technical sonifications such as astronify which are meant for doing research doing science i think that's when things get a little more shaky i think things get um yeah a little more experimental but again i think with more interest in these things comes more improvement i've talked to um i've talked to people who are interested in and working on these things i think part of the problem is again a lot of the funding for sonification is still in this outreach realm which helps get that moving quicker just in what i've seen and and you know i i don't have all the answers i think things are going to get better inherently just because there are people interested how quickly i don't know um that said i do know and you also know the people at space telescope science institute who make astronify um and they are awesome cannot say enough good things about them so if there are problems with astronify in particular do like you know reach out to them uh submit something on github their email is on um uh like there's an email on the astronify website which is linked in the curriculum like get in touch with them they're they're wonderful people um realistically i've actually probably should have reached out to them to ask about the whole mac install issue um which maybe i still could do and update everyone but um on the astronify in particular front they're uh they're wonderful so yeah yeah scott is very nice thank you yeah um and i would say just to add to that i mean i think that it's kind of the non-fun parts of making these libraries is kind of the issue you know which is making sure that the sound is compatible across every os and then also maybe that there's like fallbacks if you know if it's not creating you know so then that's not really the fun kind of programming that's kind of the kind of programming programmers didn't exist so i think that does definitely requires to pay somebody to sit and test and run it on a lot of different os's and even getting your hands on a bunch of different computers kind of costs money so um but yeah i would i'm hoping that it does uh we do get a nice stable library um or maybe it's going to be a strong fire um cool anyone else have any questions for sarah i think maybe that's a hand raised so you want to hop on the mic you can't tell who it is but is that rosanna rosanna it is i had asked this in the chat uh but i had to leave so i didn't get the answer if it was given but uh can you remind me when these uh office hours are and if they're the same link as what i got on as what this was the same link and they are going to be on now on thursday at the same time of day as this tutorial so that's 1 p.m eastern time 6 p.m gmt um thursday and then that will be

the last office hours um sadly but you know we will always be available by email i can't because i want to try it uh i don't know why but when i did it uh with last week following your instructions for last week it did not play sound for me did not did it play this week this not this week either well i didn't i wasn't able to follow along this week i did it after the fact last week all right well are you on a windows computer yes all right let's go through yeah do you try to come to office hours on thursday and we can troubleshoot together or send me an email and we'll take a look together um yeah sometimes the terminal is a little laggy with astronify um that's my first guess that you know it might be like this like turn it off and on again sort of issue which is annoying but that would be my first guess but yeah we can we can diagnose it together another one that i'd like to try is i've been on a weight loss program and i've been keeping track of my weight you know every week i want to try to make a sound application with that that would be this would be a perfect thing for that actually yeah because it's that sort of value versus time um that works really well for for astronify so yeah give it a try you just sort of have to dump it in like a well you have to dump it in a for our purposes a data frame first and then make that into an astropy table to give to astronify um but yeah should be relatively straightforward yeah this type of thing is just down my alley i love perfect stuff thanks a lot thank you for coming thank you rosanna does anyone else have any questions for sarah while we have her a lot of music hi i do have a question hello this is laura hi laura hi go ahead oh i'm just gonna ask what i can what what what your question is no i was just asking are you gonna have any more classes um coming up any more series well i will say um that you know we we're not really a school we're a consultancy right so i know maybe there's an opportunity to talk a little bit about iota just briefly at the end of everything again um you know we are a small consultancy um where we work on technology you know technology including web technologies um curriculum development and uh and very kinds of writing so really you know what what i like to do is partner with an organization um in this case i kind of brought i thought that this should exist so i went to pandas i thought they would be a good partner and i asked them you know could you would you work with us to create something for this community and they said yes which was really amazing but you know it really it's very helpful to have advocates out there because you know i'm all um you know i do work with cool people like sarah um and a bunch of others a couple of other blind folks like krishna and people like tony um you know who are my colleagues but you know really what we need is a a little more surface area to um if you thought this was a good um you know we do a lot of i love to make curriculum i love to teach people stuff if you thought this was useful you know send an email to um you know whatever blindness organization you're closest to um or you know other organizations that you think might have resources to kind of put behind this and the reality is um you know i i um i i used to do a lot of uh this kind of work and then i just put it on the internet and do it for free for people and that you know i that's what i always try to do but i also have a little baby at home here now and i do need to try to uh to partner with people to bring in some some funding to do this kind of work and it does it's very time consuming uh to put together these kinds of workshops

um and uh so yeah so yeah if you if you have any ideas about organizations to to put me in touch with or you know to tell them to reach out to me or you could propose it yourself and just say loop me in later you know that's what we do so and and i guess the answer is maybe hopefully and we will try thank you very much cheers laura thank you and i reached out also so maybe excellent i think yeah i think i have an email from you and i just haven't responded yet no problem no problem remember lots of emails come again it's a very kind email and your email was a little longer so i wanted to do a more substantial reply and i just haven't got no problem i totally understand it's been very useful especially for me coming from data science i wish i knew this before like i had this series before i went to data science especially since i'm blind it's just made everything so much easier you did a master's correct i think i'm remembering your yeah i know it's very appreciated and i think you had a common experience which is experience i had doing my phd too which is if you don't mind me saying i'm sorry uh that you we bind people we blind and vi folks were often uh wind up spending more time on accessibility than on the thing that we're we're actually doing um which is fine and then everyone calls us advocates and and and so on but then really sometimes we actually just want to do the work that we signed up to do so it can be a little frustrating right um and things could take a long time you know it took it took me a long time to finish my phd um and that's reality you know but i guess hopefully at dinner you know i was just talking to another blind um uh uh uh you know technologist uh josh melee um and i think he's at amazon now but he uh he basically said well when i was younger there was no such thing as a screen reader so you know what i mean so i think things are it doesn't almost feel like it but but maybe things are improving it's just a little slower than we want thank you yeah thank you and also i will say you know um if if you know i've been talking about this um with uh some others but basically if if you are the only person if you're a person in your community you've joined these workshops and you feel like you've gotten something out of it maybe you've learned a few things and you're practicing on your own feel free to use these materials to teach your own workshops um i have to talk to the pandas folks about what license they want to release these under but it's an open source project i have to imagine if it's remotely possible i will release these under a creative commons license and i encourage you to to use them to teach if you want to lead workshops for your own community on these same materials i very much encourage you to you don't have to ask for permission though i you know i would love to hear about any work you do with these materials um so and you can build on them too if if we do manage to put the creative commons license on it um which you'll see an update on that in the repository if we manage to do that so um okay i mean anyone else have something for sarah a lot of music oh i i have this is tim and i have just a it's a more general question these classes have been really really good um if i want to learn more about data science pandas astronify are there particularly good sources to go look at to learn more to dig deeper into these topics that sounds like maybe a sarah question sonification specific or data science in general a lot of both actually well i'll give my answer and then maybe sarah will be able to to jump in i would say it's a little bit

different difficult to answer that question for data science in particular just because i think there are limited materials specifically for the visually impaired i would say something one avenue is to try to get good at python and the and whatever environment you choose to use whether it be vs code ipython um so i would recommend joining some blindness related mailing lists um i i have a issue for this but i will be putting in a resources page on the um in the curriculum i haven't actually finished it i started it but i haven't finished it but there are there are good mailing lists like python viz and program l i think is what it's called which are very active mailing lists for python and via visual impaired programming in general um and i am going to you know i've just paid for it or whatever and the server got set up and everything so we should have in the next day or two a new mailing list for data science unfortunately it does feel like very 1990s with these technical blindness communities you know in terms of it's very mailing list driven i do think we there's a there's space here for that we could we should have more resources online centralized and and books and so on but it feels like very early days so i'm hoping we'll have more of that soon i'm hoping to put this online as a resource and i would love to turn it into something like something resembling a book in the coming months and years but we'll see um but yeah there's not a lot out there just for us vi folks um and i would i learned a lot of this mainly by um first i learned python with a book called learn python the hard way this is a long time ago maybe nine or ten years ago or maybe even more and then a few years ago i got more into data science and i really just sat with a lot of blogs and tutorials and stuff and there i didn't find that there were that many books that i found super helpful um for the data science specifically so it was kind of a lot of just trying to do things and then looking up how to do them um sarah maybe you have a more encouraging answer no i mean i think i think uh this is very much like the researcher way to learn to code like i took one coding class in undergrad um that almost fortunately went on zoom part way through because that made it much easier for you to follow because everything was online instead of like them typing on a screen um so it was almost easier that way but um and then and then it was really just like oh i need to do this thing for research well i better google and figure it out um and that's that's more or less how i learned to code just google and discover um which is not super helpful here i know but um yeah maybe if we do put together a resource list as far as learning sonification there are some things i can recommend um first of all for more like outreachy talk um the astronomer Wanda Diaz Merced um who's a blind astronomer gave a Ted Talk um about sonification i think if you google like blind astronomer like that'll like blind astronomer ted talk that'll come up um and i think that was one of my first introductions to sonification i've also linked on the curriculum i should have linked um both the astronify web page which is all their documentation and then also dr scott fleming from the space telescope science institute he's the project lead on astronify his talk about astronify at the space telescope science institute day of accessibility that's another great thing to check out um if you send me an email and want to know more i'm happy to dig up some more resources for you and then if you really want to dive like into the deep end of sonification there is a sonification world chat um every used to be every

month now it's maybe every couple months ish um that's on zoom with like sonification researchers from all over the world um and i could if you reach out to me i could see about getting you added to that uh to that group as well if you're really interested though um that one tends to be a little bit more technical on the research side as opposed to like the using sonification side so um that's my little spiel there and with that tim thank you tim if i'm getting your name correct um and let me just give a little i'm gonna um probably end the recording now in a minute but i just want to give people kind of a few like words of encouragement i think for here um and basically i just want to say first of all you know the usual that we're vi folks we're blind via folks it this is definitely an area where you know the tooling could be you know that you know we've used some tools here that are definitely usable and and you can get a lot done and i would say you can do most things but it takes it is harder we it takes a lot of us more time to learn it and also we don't have access to some of the same approaches that our side of side of colleagues would um with that said i think that you know we're having a little bit of a moment for this stuff that people are really working on this there are organizations that are becoming more interested in it um that and i'd also say if you can manage to to develop a little bit of your python skills if you can manage to develop a little bit of your data science skills then it is a useful moment to to you could you could really have make a difference in terms of the tools that are available to people their curriculum and the resources that are available to people um or even if you wind up teaching using other using some of the resources we've created and so on then then that could really make a difference for other people so it's you know it's a typical area but it's i do think things are moving i think it's a good time to get involved i also think it's an area that makes sense for blind folks there's no reason we can't do data science um it's a very it's it's a very non-visual medium um when you actually get down to it and when you actually learn how things work um so i'll say that and um i'll also say you know i'm i'm trying i'm i'm trying to activate some folks who you know maybe they have resources maybe their organizations and so on and also to create some resources so i'll be creating this mailing list i have a website up called blind coders.com it basically has nothing on it but we will you know put this on it i'm going to put this up there if i can or at least link it if pandas wants to host it i'm going to link to other projects that i know about um if people if the mailing list is active if people get involved i will try to build that out i'll try to maybe maybe we'll add forums maybe we'll add other resources maybe we'll add you know whatever maybe we'll try to host other people to to write some curriculum and do some workshops like this um and um and i'll say you know stick with it like hopefully this provides you some basis for learning um check out some the book once i send out the resources check out the other resources for learning um and and um and you know i think that you you know you you can really actually you know uh it may be discouraging you know you do a few days of it and you're like this is really hard but if you stick with it for a year you stick with it for two years people always overestimate what they think they can do in a day but they underestimate what they can do in a year in a year if you stick with stuff like this you um you will you'll be actually be a programmer you know

what i mean and people get hired and so on after doing only a year or two of learning how to program and i've already seen some of you like people like nickel um and joy you know they've been doing contributions they did nickel did his first github contribute open source contribution on github that i saw during this and um and that's very encouraging to see and there's a lot of meetings and so on you can jump on um there's the the um jupiter accessibility meetings that meet month i think monthly or something like that there are the sonification world chat um these these are open source as a culture where you should just speak up jump in create issues create poll requests are basically suggestions for how things should be and and also open meetings so and you you you're allowed to speak up you're allowed to participate um you know i think we're often discouraged we're often encouraged to be in the shadows and be um you know not speak up as blind folk but um but i think you know we need to so okay so thank you very much i really appreciate you all making this such an amazing energetic workshops we had over 350 people sign up for these workshops we had some of the workshops they had set you know 50 70 80 people in them there's tons of people following on home they're sending people are sending updates people are translating the curriculum i'm feeling a lot of energy with this and i hope to keep it up i hope to create more resources for you all so thank you very much