Principles of Sonification (Nonvisual Data Science Workshop Series)

(0:03 - 1:38)

Hi everyone, my name is Sarah Kane. I'm a PhD student in astronomy at the University of Cambridge and I am one of the co-instructors for this non-visual data science tutorial series alongside Patrick Smith who is a blind programmer and programming instructor and consultant with the Iota School. This is the fourth of the five tutorial series.

Today we'll begin talking about data sonification and the next tutorial, next week, will also cover the same topic. Again these tutorials take place Tuesdays from 6 to 8 p.m. UK time, that's my time zone, 1 to 3 p.m. Eastern Time in the US or whatever that corresponds to in your respective time zone. These tutorials are being very generously supported by Pandas and Numfocus which makes them free and available to all of you and all of the recording you're watching right now and all of the written curricula are online and available to you.

I really encourage you to check out the curriculum, it's got all of the code we'll go over as well as some more details and you know written instructions for everything we're doing today if you want to revisit it in written form and of course Patrick and I are always happy to answer your questions over email. We also have office hours on Thursdays from 6 to 8 p.m. UK time, 1 to 3 p.m. Eastern Time in the US and so on and so forth over the same zoom link that this live tutorial occurs. Now those of you who are watching this right now might be able to tell that this is not a live zoom course happening.

(1:39 - 5:59)

We're currently, well I, this is not the Royal We, but I am currently re-recording this tutorial. We had some unfortunate sound difficulties during the live zoom so we wanted to make sure there was a nice clean version with no sound difficulties available to everyone for posterity so to speak. So I'm excited to go over this curriculum again with all of you and again if you have questions please feel free to reach out.

Now without further ado, especially because there's no one else here with me, I think we're just going to jump right into things. So as you have before I encourage you now to open up Anaconda prompt. I'm already on the screen, I'm going to turn on my NVDA speech again, didn't want it talking over me during the intro.

Speech mode beeps, speech mode talk. Excellent and hopefully you can hear that okay. Okay so open up Anaconda prompt as we have in the previous three tutorials.

If you are not already there like I am you can do that by hitting the Windows key and typing in Anaconda prompt to open it up. Again this is Anaconda prompt not Anaconda

navigator and what I want you to do right now is not to start IPython. Again please do not start IPython yet which is what we do usually after opening Anaconda prompt because that's the sort of interactive workspace we have to write code and test it.

But what we're actually about to do right now is to do something called installing a package. Now we've used packages in the previous tutorials. We've used pandas and NumPy.

These are like libraries of code that other people have written that we can now make use of and those packages because they're really popular and widely used are available with the Anaconda installation that you already have. But some less common packages like Astronify which is the sonification package that we're going to use today is not already pre-installed with Anaconda. So we're gonna have to install it ourselves.

No worries it's a pretty simple. It's as simple as one line of code actually. But before we get into that I want to just go over a few caveats.

First of all unfortunately to our knowledge we really only expect the Astronify installation to work properly on Windows computers and also maybe on older MacBooks. If you have a MacBook with one of the newer Mac chips so not an Intel chip but a Mac chip so that's the M1, M2, M3 Mac we have had no luck getting Astronify to work on those new Macs. Likewise if you are a Linux person using Linux unfortunately we also have not had luck getting Astronify to work on Linux.

This alongside our desire to use NVDA as the screen reader we're teaching with is kind of one of the motivations for us encouraging all of you to use Windows for this tutorial series. Also if you have other packages already installed so say you already program a bunch and have other things installed on your Anaconda it is possible that this might not work. It's really hard to predict how different package dependencies might not get along with each other, might not work out properly.

So if you have some sort of error message because you have other packages already installed please do let Patrick and I know either in office hours or over email and we will try to help you out. But for the vast majority of you who have installed Anaconda nice and fresh for this course this should be quite simple. We hope anyway.

Cross your fingers. So installing Astronify and I've uninstalled it actually so that I can install it with you now is as simple as one line that we're going to type again not in IPython just in the Anaconda prompt. We are going to type PIP.

PIP is our package manager here. It's the one managing all these libraries of code. PIP is the thing that's going to install this package for us.

There's another package manager out there called Conda. Some things are available via Conda others via PIP. We won't get into that.

(5:59 - 8:54)

This one we're going to use PIP and if you're curious check out the curriculum online where I talk about it a little bit more. So it's PIP P-I-P space space install "I-N-S-T-A-L-L" PIP space install space Astronify "space A-S-T-R-O-N R-O-N-I-F-Y" PIP install Astronify A-S-T-R-O-N-I-F-Y and hit enter. "Collecting Astronify.

Using cached Astronify 0.1 pi 3 none any dot WHL. Requirement already satisfied. Astropy in C. Users."

Alright so I'm going to stop that now because there's a lot of text happening there. So I don't want you to hear all of it but I wanted some of it to play because I don't want you to feel like alarmed when NVDA starts reading out just this absolute barrage of text. It should mostly be along the lines of like requirement already satisfied you know installing XYZ.

If you start getting some angry sounding messages like you know couldn't do this or whatnot that's a good time to start maybe doing some googling or contacting Patrick and I but hopefully it'll all be nice things that are like requirement already satisfied and it should end with successfully installed Astronify and that should be all you need to do. Occasionally I think I've seen some people get a message where it says proceed question mark yes why slash no and you have to hit type Y for yes and then hit enter. I don't think most of you should get that but if you do not worry.

And with that congratulations the Astronify package is installed cool done. First module or first section of this curriculum or this tutorial is checked off the list. What I'm going to do now is I'm going to actually step away from the terminal step away from some code and give you a little bit more theory about what sonification is and how it works.

Now I've just mentioned sonification a bunch of times here without actually saying anything about what it is so we're gonna get into the theory we're gonna get into the examples and then we'll get back to the code yeah. On that note I'm going to turn off the speech for NVDA so it doesn't decide it wants to talk over me or talk over any of the sonification examples which we did have a little bit of in the original zoom so you know this is one a sound issue we will dodge. Speech mode off.

(8:55 - 19:13)

All right and with that we will get into what is this sonification thing. So in the simplest terms sonification is the representation of data via sound. This is directly analogous to data visualization where data is represented via some visual medium.

These are your line charts your bar charts your pie charts your histograms the list goes on there are a ton of data visualizations out there. Importantly the data does not literally look this way it does not literally look like a line graph it doesn't look like a bar chart this

is how we choose often to represent it through some visual medium. In fact there are a lot of arbitrary choices involved in these in these visualizations for instance you have a line chart how are you going to scale the axes how large will you make the points these are all various different choices that you can make in these data visualizations.

It is a representation of the data a way we choose often to picture the data and oftentimes they're quite useful they let us understand trends and things like that but there are issues with data visualizations and not the least of which which will be of course great interest to me and you is that they're not accessible to people with visual impairments like me. So data sonification where we represent the data as sound sort of corrects or steps around that one that sort of problem. Now before we get more into data sonification and what exactly a sonification could be besides the sound representation of data I want to break down some common misconceptions about sonification.

First of all sonification generally doesn't involve words so it's like non-speech audio so for instance me standing here and telling you about data for instance when Patrick told you about the Airbnb data set that is not a sonification it's just him telling you about the about the data so it's not typically what we'd consider a sonification a sound representation of the data it's like a word representation of the data something along the lines of that I don't have a more elegant word for that but it's not generally what we'd consider sonification. Sonification also is not what we consider like natural sounds or or just the sounds themselves. What do I mean by this? Well for instance imagine you have someone walking and you are listening to their footsteps to understand to measure how quickly they are walking.

In this case the sound of their footsteps is not a sonification. The sound of their footsteps is the data so the sound is not a representation of the data the sound in that case is the data. Scott Fleming in his talk at the Space Telescope Science Institute's Day of Accessibility had another great example bird songs not a sonification bird songs would be your data if you were going out and observing bird songs same sort of idea here.

So it is in the same way that a data visualization is not literally like what the data looks like a data sonification is not literally what the data sounds like. Alright so we've broken down roughly what sonification is it's a sound representation of the data and it's not words and it's not sounds from nature. Okay we've got the big picture here the only way to concretize this sort of idea of what sonification is is going to be to get into how we actually sonify things.

Today we're going to go over two of the most common sonification techniques this is going to be modification and parameter mapping what they are what their differences are and we'll also talk about some or we'll also show some examples of each. Alright let's begin by talking about modification. You'll often hear modification described as the most direct mapping the most direct representation of data as sound.

What do we mean by this? Well suppose we have two variables here X and Y and in this case Y is our dependent variable which means that as X changes Y changes in response. Now for our purposes here I'm going to say X is time so as time goes on Y changes with time and so if Y is varying with time going up and down as time goes by you can kind of imagine Y moving like in a regular wave not like a smooth wave pattern but it's going up and down so Y the height of the wave is changing with time. Now we call the height of a wave the amplitude and just as this height this amplitude of Y is changing with time we can imagine the amplitude of a sound wave changing with time at the same sort of way.

So basically if Y goes up the amplitude the height of our sound wave goes up and if Y goes down the amplitude or height of our sound wave goes down. So we are mapping the height of Y to the height of the sound wave. This is what we mean by direct mapping we're treating Y like it's a wave and then just making the sound wave that represents Y behave in the exact same way as Y in terms of how it goes up and down.

Now our ears perceive amplitude the height of the sound wave and again sound is a wave in the air so our ears perceive the amplitude of a sound wave as the volume it has to do with like this air pressure essentially you'll hear it as changes in volume and so essentially what is happening is by going up and down Y is controlling the value ofhsorry the volume of the sound wave we hear representing the data. Alright so we've covered how the volume is controlled by Y. What about pitch that's another really important part of sound how high or low of note is. Well this comes down to something we call data sampling.

We don't measure data continuously rather we have some sampling rate which is often how we measure the data. So for instance I'm going to play you some sounds from the accessible oceans project some modification from the accessible oceans project and the times are measured each hour this is our data sampling rate so we have like 24 measurements in a day one measurement each hour. Likewise we don't play the sound continuously but rather we play it at some data sampling rate where we play each note at an individual moment.

So we have our data measurement rate or a data sampling rate and we have an audio sampling rate where we play the rate at which we play the sounds. Now the sampling rate or the frequency at which we measure and then play the modification of data is analogous to sound frequency which our ears perceive as pitch. Thus the frequency of the sound playback influences the pitch at which we hear the modification.

Often we cannot play the modification of the data with the same frequency as the original data sampling rate because the then the frequency or the pitch we'd be playing would just be outside the audible hearing range for humans. Thus in audification we often have to do something that I've seen called translating or shifting the sound representation of the data how the sampling rate at which we play notes just so that the

pitch is in the audible pitch range. Now what I'm going to do is I'm going to play you a sonification of the tides from the accessible oceans project at two different sampling rates.

It might be a little bit difficult to hear over the recording like from my computer to your computer but there are two different sampling rates and you can hear that there's a pitch difference between the two. If you can't hear it through the recording I do encourage you to go from the tutorial curriculum page to the accessible oceans page which is linked there and has the examples on it. So if you can't hear the pitch difference here definitely encourage you to go check that out.

I'm going to go play that now so it'll be two sonifications. All right that was pretty quick so I'm going to play that one more time. All right hopefully you could hear the noise happening there.

That is the tide cycle you're hearing. So tides go in and out you know every 24 hours that's at least as much as I know about oceanography tells you what I know and so what you're hearing is that actually every 24 samples not that we can hear the individual nodes because they're so close together you hear that as the tides going up and down control the volume of the modification going up and down that each of those cycles is that 24 hour cycle of the tides going up and down. Now I'm going to play this one more time the same modification before I play the next one that's at a different sampling rate so you can hopefully hear the two pitches side-by-side.

(19:22 - 1:50:38)

All right hopefully you can hear that on my end I can definitely hear the pitch difference I'm hoping it comes through on the recording fingers crossed so what you might have been able to tell on those two is first that whoop whoop whoop is still there because we're still measuring that same tide or we're still representing that same tide data of it going up and down the pitch is lower on the second round of data and also it's a little bit slower it's a little bit slower between each whoop whoop whoop and the modification lasts a little bit longer because the data sampling the frequency at which we play those notes is is a little bit slower which lowers that pitch and also lengthens the time of the modification. So again hopefully you could hear that difference and if you couldn't definitely recommend checking it out on your own computer. All right so if modification is so straightforward we're treating the data like a wave and then actually one second before I keep going I think the computer is about to die rather than suffer another technical difficulty which should make me quite sad I'm just going to plug this in sorry folks.

All right we're back at business. Okay so if modification is so straightforward where we're really just treating the data like a wave and then representing it as an analogous sound wave why don't we do it for everything? Well there are a lot of reasons. First it's pretty

limited and what we can or cannot sonify.

For instance how would you audify make an audification or a sonic representation using audification of an image of some 2d data you know with an X and a Y colors there are too many different parameters going on there to put into this amplitude so this volume and then the time. So it's really better suited for a very specific type of data a simpler type of data. Also there might be other reasons we don't want to use audification for instance maybe there's a more like aesthetically pleasing a nicer sound that we can use rather than these amplitude changes in the sound wave.

An aesthetic difference is a really valid reason to have a different representation of the data. I mean I can say that I know a lot of scientists that spend a lot of time trying to make their data visualizations look nice look pretty so that matters as a choice too and it's a totally valid choice to make. Again we're not literally showing what the data sounds like we're making choices of how to represent it and depending on our audience and our goals we might make different choices.

So let's get into what one of those different choices might be and that might be to use a very popular and very flexible sonification technique called parameter mapping. Parameter mapping is a more flexible as I mentioned form of sonification that has become popular in the recent years and in parameter mapping in essence we map or connect different aspects or dimensions of the data to different parameters of the sound representation such as pitch or volume and rhythm. Like modification we could use parameter mapping to sonify data consisting of say one independent variable that's what I called Y and one dependent variable that's what was time for us here but one of the strengths of parameter mapping is that we can actually map much represent with sound much more complex data than we can with modification because there are many more aspects parameters to the sound that we can use that's pitch and rhythm and volume different instruments and timber I'm not a musician but the list goes on there are a lot of different knobs we can turn in our sound to represent different dimensions different aspects of our data.

All right let's clarify the concept of parameter mapping using an example suppose you have measurements of number of sales of something I'm gonna go with ice cream because I like ice cream I can totally go for some right now and net profits from those sales of ice cream over a period of 50 years we choose to map the number of sales to pitch such that more sales are represented by higher pitches and to map net profits to volume such that more profits are represented by a louder note so essentially we let pitch be mapped to sales and volume be mapped to profits from those sales and now we're letting time that independent variable what we'd have along you know the x-axis if this were a visualization for any visualization folks we let time represent be represented by time in the sonification such that each note is a measurement in time so that you know if you're listening later in the sonification you know you're hearing later in time so

the first note represents 50 years ago and the last note of the sonification represents the data measured now suppose as we listen to the sonification that we hear a period in which the pitch of notes drops so the pitch goes down we can tell from the mapping decisions that we made that this means the number of ice cream sales in this period has dropped relative to the other measurements perhaps it's wintertime or a recession and people aren't buying as much ice cream now suppose that over the course of the entire data sonification we hear the volume of notes generally increase but the pitch remains relatively constant this means that profits have gone up but the number of sales has gone has remained relatively constant so the cost per ice cream sale the profit per ice cream sale must have gone up that's what I'd call inflation finally suppose that the amount of time between notes decreases towards the end of the sonification this might mean that the data were collected more frequently in later times so that there's less time between each data point note here that there is no one correct way to map our data to different parameters we could just as easily have mapped sales to volume and profits to pitch thus we make choices generally through testing and experimentation to see which representations of our data most effectively and most accurately capture the importance of information and communicate of the information and communicate it preferably in a way that's both understandable comprehensible and aesthetically pleasing this is truly just no different from a data visualization where graphics will go through many iterations of designs to clearly represent the data data representations whether sonic or visual always represent choice so one of the things I hear most often about sonifications is that they're arbitrary is that you know this isn't you know there are so many different ways you could represent the same data how do you ever know anything because that you could just represent the data so many different ways but this is just how representations of data in general go it involves choice it's kind of a fun thing all right let's actually listen to a sonification or to a parameter mapping example now because my background is in astronomy we're going to listen to an astronomy example this is from the NASA and Chandra X-ray Center's universe of sound website and what we're going to do is we're going to listen to the sonification but it also comes with a description which tells us how the break down the parameter mapping so what aspect of the data was mapped to which a parameter of the sound and then we'll listen to the sonification all right it's so we're going to listen to a sonification of the galactic center or of data from the galactic center this was taken from the Chandra X-ray telescope I believe explore the center of our very own Milky Way galaxy the translation begins on the left side of the image and moves to the right with the sounds representing the position and brightness of the sources all right we've already got our first parameter mapping the translation begins on the left side of the image and moves to the right they don't say this directly but what this means is that time within the sonification is mapped to how far we are moving from left to right in the image so you know if you hear in something earlier in the sonification it's closer to the left you hear something right at the end of the sonification later in time you know it's representing something on the right of the image the light of objects located towards the top of the image are heard as higher

pitches while the intensity of the light controls the volume all right that's two more parameter mappings there so pitch is controlled by how far we are up and down the image with higher pitches representing objects higher up in the image so closer to the top where lower notes are going to represent objects lower down in the image we also have a second parameter mapping in the sentence the intensity of the light controls the volume what this means is that brighter objects are going to be represented by louder noises and dimmer objects in the image are going to be represented by softer noises stars and compact sources are converted to individual notes while extended clouds of gas and dust produce an evolving drone what does this mean well smaller objects are going to be little notes larger objects longer sounds so the size of the object has to do with the duration of the note the crescendo happens when we reach the bright region to the lower right of the image this is where the 4 million solar mass supermassive black hole at the center of our galaxy known as Sagittarius a star resides and where the clouds of gas and dust are the brightest all right I'm gonna hit play hopefully right I love that sonification that's one of my favorites the universe of sound project has lots of sonification so I definitely recommend checking it out you can get a super long crash course into parameter mapping I think they're lovely really lovely to listen to I'm biased because I like space but there you have it I don't know about you but I think for me the easiest thing to pick out is that note duration thing I could definitely hear those little chimes that are representing stars and then the longer notes especially towards the end where they told us that supermassive black hole is I can hear that sort of longer stretch of notes really lovely I recommend if you want to go and listen to that maybe several more times see if you can hear those individual parameter mappings the pitch representing how high we are along the image up or down the volume representing how bright something is I definitely recommend that I think it also probably would be easier to hear over your own computer again rather than through this recording I'm hoping fingers crossed that the sound records much better here than it did for everyone else sorry if you hear some background noise my guide dog is standing up and shaking but I'm hoping it records much better than it did during the original zoom all right so that is modification and parameter mapping the two I would say most common sonification techniques you will encounter that's also a really quick crash course into what sonification is just keep in mind again sonification is a sound representation of the data now if you're sitting here and thinking Sarah it's super pretty I think it's cool but how in the world do you ever learn to interpret that very understandable and especially it's understandable given that we think of data visualizations graphs and charts and line graphs and so on and so forth as intuitive and easy to interpret but that is not at all true children in schools spend years learning to read graphs in the US where I'm from there's a standardized test for college like to get into university called on the ACT and a whole section of the thing is basically reading graphs and kids study for months for that so clearly we need to learn to read and interpret data visualizations and there are studies suggesting the same should be expected to be true of data sonifications that we should expect to need to learn how to understand and interpret them this kind of makes sense

no one's born learning to read a data visualization no one's born learning to understand like already knowing how to understand a data sonification so our goal for the rest of this tutorial recording is going to be to get some basic shapes in Python and then to sonify those different shapes with that astronify package that we just installed in such a way that we can sonify those different shapes and understand how they sound so you can hear this is how a straight line sounds different from a curve which sounds different from something that's moving in sort of a wave like shape the goal here is to develop some intuition for you so things sort of make sense a little bit more on a natural level so that next week when we get into sonifying some more realistic data something that's not just shapes and lines you'll have this intuition or at least the beginning of an intuition already built so that's the goal for the rest of today so now we're going to return to Anaconda prompt or maybe you're already sitting there and I'm going to turn NVDA back on "speech mode beeps speech mode talk" and now we can start IPython. So just as we have before I'm going to start IPython by typing I P Y T H O N IPython no spaces and hitting enter. "Python 3.1 1.5 packaged by Anaconda Incorporated Main September."

Alright I'm not gonna let it play that whole thing through because I think you've heard it before but essentially you'll know you're an IPython because it'll tell you your Python version and eventually you'll get to it if you let it finish it'll tell you we're in in one we're in the first input line. Now we're going to start by making some of what I'm going to now refer to as synthetic or simulated data what this basically means is not like real data from like observations or measurements this is basically like fake data we've made to follow various shapes or if you're math savvy and remember like your high school algebra classes we're actually going to just be making some simple algebraic equations here we're not actually going to go through making all of them we're going to go through making several of them so you get the idea of like how this math is going to work and then there's a CSV file that has everything already made that you can load in from the URL I encourage you to go to the curriculum it's under data preparation under the final subheading called just in case quote onquote basically it gives you the CSV file in the URL and just gives you the code to paste it in yes so I encourage you to do that but if you're interested for how exactly all of that synthetic so again all of that not real measured data was made in the CSV file the curriculum goes into painstaking detail about how every column was made and exactly the equations that's going into it so we'll kind of do a brief intro to that just so you have a rough idea of how this CSV was made but if you want all the nitty-gritty details the curriculum has everything all right so those of you who remember algebra classes will remember that like functions equations have two different variables what I've called independent or X and dependent Y so Y is a function of X. Y is going to change when X changes so what I want to do first is I want to make an array of numbers that's going to be our X values so imagine these all of these values so I want for in our case I want 100 values evenly spaced from 0 to 10 so that at each value of X going from 0 to 0.1 0.2 0.3 whatever the the even spacing will be will have Y change a different value of Y at each of those values and that Y will change

differently depending on what function we do now as was evidenced by the fact that I had to already sit and think about like the first three numbers in those numbers ranging from 0 to 10 I really don't actually want to sit here and write out a list or an array of a hundred numbers evenly spaced from 0 to 10 I hope you don't want to do that either I think we can all agree that that sounds really quite painful so the good news is NumPy has a built-in function that will do this for us so let's get started I'm going to start by importing our favorite packages here so I'm going to do import NumPy import space NumPy and "N-U-M-P-Y as NP" it's suggesting the way I always type NumPy but we're not going to do that at the moment into all right and you can hear I hit enter and now it's saying you're on the second line I'm gonna do import pandas "import NumPy" it wants to repeat import NumPy we're not going to do that but we're not gonna need pandas at the moment but we will need it very shortly so let's just get it in here "I-M-P-O-R-T space P-A-N-D-A-S" import pandas in three whoo good news all set and ready to go so what I've just done is brought the NumPy and pandas packages into this coding session all right and next what we're going to do is we're going to use something called the linspace function the linspace function is a built-in function in NumPy that will basically give us an evenly spaced array of numbers starting at a specified start number and ending at a specified stop number I think this will be easiest to conceptualize if we actually do an example so let's start by doing NumPy so "N-U-M-P-Y" so from NumPy so NumPy dot dot NumPy dot linspace that's L-I-N-S-P-A-C-E so basically from the NumPy package pull the linspace function open parenthesis I'm gonna do 0 comma 5 comma 5 close parenthesis. That's NumPy dot linspace open parenthesis 0 comma 5 comma 5 that first 0 that's our start value for the array the second number the 5 is the stop value for our array and then that third number 5 is the number of values I want so basically what I'm saying is linspace give me an array of numbers from 0 to 5 evenly spaced and I want 5 of them and when I hit enter it's going to return to me an array of those numbers "out 3 array 0 1.25 2.5 3.75 5" yep there we go and it has spit out an array that goes from 0 to 5 5 numbers evenly spaced let's try this again we can change the parameters let's try NumPy dot linspace "N-U-M-P-Y" dot linspace dot "L-I-N-S-P-A-C-E" "open parenthesis "left paren" I'm gonna go 0 comma 10 comma 5 what is this saying it's saying our start our start value is 0 our stop value is 10 and I want 5 numbers so NumPy dot linspace open parenthesis 0 comma 10 comma 5 close parenthesis give me 5 numbers evenly spaced from 0 to 10 "out 4 array 0 2.5 5 7.5 10" and now you can hear we get an array of numbers going all the way up to 10 this time now this is almost what I said I wanted I wanted an array of numbers from 0 to 10 but I wanted a hundred of them so this time we're just going to increase that third value from 5 to a hundred so instead give me a hundred numbers evenly spaced from 0 to 10 and also I don't want the computer to just spit it out at us I actually want to save it so just as Patrick taught us to save variables this time instead of just doing NumPy dot linspace I'm going to do X equals NumPy dot linspace because I'm gonna say hey make this array of numbers from 0 to 10 100 of them and call it X. "X space equals space n u m p y dot l i n s p a c e left paren 0 comma 0 comma 1 0 0 right paren" X equals NumPy dot linspace open parenthesis 0 comma 10 comma 100 close

parenthesis in 6 and there's no output this time because we've just saved X but if we type X and hit enter X "X equals NumPy dot" it wants to suggest that I repeat that same line from before I don't want to repeat the same line I just want to hit X and hit enter "out 6 array 0 0.1 0 1 0 1 0 1 0.2 0 2 0 2 0 2 0.3 0" all right I'm not gonna let it go through the whole thing but as you can hear it's spitting out a rather long array of numbers that are relatively close together and it'll keep going all the way up to 10 a hundred of them we don't want to listen to all of that or at least I don't you're welcome to listen in your own time if you really want to hear all hundred of those numbers but now we have checked that indeed it's saved as X we have an array of numbers evenly spaced from 0 to 10 and a hundred of them so rather small spaces now I'm going to do something I'm going to make one of these equations as an example so that you can see how roughly I've made these sort of simple shapes we have going and then we'll load the CSV so that we just have all of the data so what I'm going to do right now is I'm going to make something called a sine function a sine of X so what I mean by this is a sine function is a trig function a trigonometric function basically it's a wave so as X increases Y responds to X by moving up and down in an even wave pattern it just goes up and down at regular intervals as X goes up so essentially what we are doing right now is we are making a wave shape and I just want to call this Y because this is how we often do this in math classes so I'm going to actually I'm going to call it Y underscore sine to specify that we are making a like a sine function it's always good to like name your variables in such a way that you know what they mean because here why if I just call Y it could be like any function if I call Y underscore sine we know it's a sine like a sine wave so we are making a wave right now one Y any line so Y underscore S I N E that's how we spell sine for for you know in math it's not like a sign like a street sign yes. so y_sine "space equals space" um and fortunately NumPy so sine is like non easy math thing to do like a multiplication it's something you generally want a calculator for fortunately NumPy has a built-in sine function so all we have to do is do NumPy dot it's actually sin so it's not a full sign it's NumPy dot s i n and that's NumPy's sine function and you ... that was a typo it should be NumPy dot sign not NumPy parenthesis dot s i n left for n right now we want that left for n so Y underscore s i n e equals NumPy dot s i n open parenthesis 2 times X that's going to be "2 star X right paren" so we are saying that Y sine is equal to the sine of 2 times X again this is a really powerful thing about NumPy arrays that you can do math with them like this you can multiply them you can take the sine of them all sorts of things like this and we can hit enter and if we print out Y sine now so if I just type Y underscore sine "Y line s i n e and hit enter out a array 0 0.20064886 0.39313661 0.56963411 0.72296256 0.84688556 0.93636273 0.98775469 0.99897117 0.969555" sorry to play so many of those numbers for you but I wanted you to hear that it goes up and then back down if I let it play even longer we would have heard it go all the way down below zero and then back up again which I encourage you to do because it'll give you an intuition for what's going on with this function but essentially what I want you to understand is that a sine of X is a wave so it's going to go up and then down and then up again and then down again all right cool let's make

maybe one more really quite simple equation here I in fact it's going to be the simplest equation it's going to be a linear equation we're going to make Y equals X which basically means that for every step X makes to get larger Y gets larger by the same amount and I'm going to call this Y underscore linear Y Y line linear Y underscore L I N E A R "L I N E A R space equals space" and it's just Y equals X so Y underscore linear equals X and that's basically saying it's like a straight line just slanted upwards every step X gets bigger Y gets bigger by the same amount in 10 and if we output Y underscore linear "1 L I N E A R out 10 array 0 0.1010101 0.2020202 0.3030" all right I'm going to stop it there but maybe you remember from when we printed out X that this is exactly the same so basically we've just gone and we've made a an array that makes the exact same that is exactly the same as X so every step X gets bigger Y stays the same amount so it gets bigger by the same amount it is a straight line now again if you go to the curriculum it goes through in absolute detail how to make every single one of the equations that we're going to load into the CSV and so if you're mathematically minded or you just want to learn more about how to do math in Python I definitely encourage you to check that out it covers quadratic equations all sorts of things like that but we're not going to get into all of that right now just because I know it's a little bit tedious and everything's already loaded into a CSV so if you're more on the data analysis side of things you don't want to do all the math then we don't have to get into all of that maybe yeah so I encourage you to check out the curriculum either at this moment to see the rest of the math happen or to load the CSV which I'm about to do I actually with the benefit of hindsight I would love to show you guys how to do a quadratic equation as well just because I want to show you guys how to do an exponent maybe this is the math person in me showing so I'm just going to show you how to do a quadratic really quickly now this isn't actually the exact quadratic that I have in the CSV I've sort of shifted and compressed that a little bit just to make it a nicer shape to listen to and again the curriculum goes into painstaking detail about how exactly that goes but I just want to show you how to do an exponent so a parabola is the shape of a quadratic equation it's basically a curve that goes up up up really steep as you get to the turning point it sort of starts to level out it turns around and then gets steeper and steeper as it goes back down or up again depending on whether or not you flipped it so it's almost like a wave but it's not like the sign because the sign goes up and down repeatedly the parabola is just a curve that turns over once and it's steepest at the sides and flattest at the top and it's the simplest form of a quadratic is y equals x squared I'm going to save this as an equation that or as a variable that I'm going to call y underscore parabola because that is the shape of a quadratic so it's going to be y underscore "y underscore p a r a b o l a" parabola space equals space x squared and the the main reason that I am taking the time to do this right now even though it's all in that csv is that I want you guys to know how to do exponents in I want to recap how to do exponents in Python just in case because I think it's good to know so I always when I started doing Python expected an exponent to be a caret it is not it is two asterisks so x squared is x asterisk asterisk square two "x star star two" so y parabola equals x asterisk asterisk two in 12 and if we

print out y parabola just by typing it and hitting enter "y line p a r a b o l a out 12 array 0.000000000 e plus 00 1.02030405 times" we had some crazy numbers I don't I don't find that easy to interpret I don't know about you I think they're all in a scientific notation that's what that e means that it's put it you know in the form of so for instance 9e2 would be 9 times 10 to the power of 2 which is 200 not 200 is 100 sorry about that so 9 times 100 or 900 it's what we do for like really large or small numbers in science and Python loves to print things out like that sometimes if it gets a little unwieldy I don't know about you but I find that a little bit difficult to interpret just listening to it scientific notation is a little bit hard to interpret just hearing it or looking at it I would say but the thing to keep in mind again is that a parabola a quadratic equation is that curve shape going up and then down just once and if you want to see the exact way I did this to make it in the CSV then again check out the curriculum it goes into absolute detail about how I made this this CSV so you can have all the nitty-gritty you can know the exact equation I used for everything and again if you're curious about that feel free to stop by office hours or shoot me an email all right I'm going to pop out of um out of ipython and into uh into google chrome ... ... and because I am low vision I am well accustomed to squinting at my computer so you might notice me navigating in a way that is a little bit more sighted than maybe um just fully using NVDA that's also...me not being a full NVDA desktop I usually use mac so one little caveat if you are working from uh the curriculum as it is now there's one little thing we're going to need to do before we copy and paste that little snippet of code again this is under that subheading just in case in the module preparing data for sonification now I have done a little shorthand in this code snippet that in retrospect I shouldn't know so I might go back and fix it but basically I am very accustomed to shortening pandas as pd you'll see people do this a lot it's just because it's shorter to type out than pandas but we have to tell our computers like hey I'm going to call pandas pd from now on so we've already imported pandas but we're actually going to import it again and tell the computer actually from now on I'm going to call it pd so that's just import "I M P O R T "import pandas "space p a n d a s" as pd space "a s pd" import pandas as pd so bring in pandas and from now on we're calling it pd again we've already brought in pandas we're into this ipython session but now we need to remind the computer tell the computer hi I'm calling it something different now we only need to do this just because if you copy and paste straight from the code I put in the curriculum I call pandas pd there which was an oversight on my part so I might go back and fix that I probably ought to so I'm going to hit enter importing all right we've imported pandas as pd and now I am just going to paste in that line of code or those three lines of code actually um from uh oh actually there's one other thing we need to cover before we paste in those lines of code and that's going to be uh something called um astropy tables now astronify the sonification package that we are using today is a package that was made by astronomers to be used by astronomers and we'll get into more detail about what exactly it is and how it works in a second but because it was made by astronomers for astronomers um it uses a specific type of data a object type in python that astronomers like to use this is something called an astropy table astropy is astropython

basically it is a python package that has a ton of astronomy functionalities things like astronomy units like things like light years and parsecs it also has coordinate system transforms all sorts of things like that are handled within astropy it also has a data type actually there's multiple but in this case it has this uh this type um of object called a table for our purposes here an astropy table is exceptionally similar to an astro uh to um a panda's data frame and in fact they're so similar that astropy has a built in function to uh convert a panda's data frame directly to an astropy table um and the reason we're going to do this is because um the reason we want to do this is because uh um astronify only takes as input astropy tables it really just wants astropy tables because it was made for astronomy data so we have to convert our panda's data frame to an astropy table in order to give it to um in order to give it to astronify to sonify so what i'm going to do here is i'm going to import table from astropy so basically say from this astropy package bring in this table things we need it so i'm going to type from "f r o m" so from astropy that space that's "a s t r o p y" dot table "dot t a b l e" so from the subsection of astropy called table space import "i m p o r t" so from astropy dot table import table except here table is capitalized "space T a b l e" so from astropy table import table and again the table that second table so not astropy dot table but that table after import that's got to be a capital t there "in 15" all right and it's imported we've brought the table thing in so if i bring in these lines of code i'm just going to copy and paste those three lines of code in from under just in case um i'm sorry to all of you guys watching on the recording that i'm not reading this out it is just a horrendously long url um maybe should i read it out oh no i i'm gonna i'm gonna really encourage you guys to go and copy and paste this from the uh from the curriculum or maybe we can put it in the description of this youtube video um because i think me reading it out loud right now is just going to lead to some suffering and also to uh just mistyping it i don't think it's going to be very helpful but there are three lines in this code the first line that we're going to copy and paste in is url equals that really long url it's basically hey the url is called this save it as this thing called url the second one is a function we've seen before it's df equals pd dot read csv url so it's basically saying make a new data frame called df and then pd remember that's now pandas pandas dot read csv so read the csv at that url after url it's comma index underscore call equals zero that means set the index of our new data frame as the zeroth column and then the third and final line of code is tbl equals table so capital t table that's the table we just imported from astropy dot from pandas open parenthesis df close parenthesis what that's saying is make a new table call it tbl and make it from the pandas data frame that we've named df all right now i'll finally stop yammering and copy and paste that in well i've already copied it warning dialogue warning you are about to paste text that command prompt terminal ipython i trust uh it was asking whether or not i trusted that copy and paste that was putting in and and i do trust it because that's my own url this is the moment we cross our fingers that uh it's gonna work um again first line is the url "e slash sonification e selected main slash prepared data dot csv raw equals true selected slash url equal" that's it reading that part of the url that's that first line "df equals selected space selected pd dot read csv url index co l equals zero

selected" all right and that's it reading the second line df equals pd dot read underscore csv open parenthesis url that's the url from the first line comma comma index underscore call equals zero and then finally that last line "tbl equals t selected i selected ...that wasn't very helpful that that final line is tbl equals table capital t dot from pandas open parenthesis df close parenthesis so make a make a table from the panda's data frame df and call it tbl i'm gonna hit enter and cross our fingers that it works "in 16." all right no error messages read out that means it worked success all right our data is prepared it is in an astropy table let's take a moment to explore this astropy table so first what we're going to do is we're going to print out the column names the way we do this for an astropy table is by doing tbl so that's the name of our table t b l and then we're going to do dot call dot ... it's already trying to suggest it dot c o l n a m e s tbl dot col names so from table take the column names and if we hit enter it'll just print them out "out 16 x linear n e g linear parabola abs sine in 17."

all right so in this table there are six columns there is x linear neg underscore linear parabola abs and sine these are the x actually the exact same x that we made with that lin space function above and then five other functions of x so variables that are following a function of x linear is that y equals x that we made above so it's just every step that x makes to get bigger y gets bigger by the same amount so it's a straight line going up neg underscore linear is y equals negative x so it's still a straight line except every time that x gets bigger y gets smaller so it's a straight line slanted downwards parabola that's the quadratic equation we were talking about now in this case it's not exactly y equals x squared it's a little bit different um i believe it's something like if i recall correctly i want to say it's let me take a look y equals negative times the quantity x minus five squared plus one um if any of you are algebra people if not don't worry about it essentially what we care about here is not so much the math as the shape so we understand the shape we're going to represent with sound and remember this is a curve that goes up up up up new notification no no no no sorry um someone's notification um okay um what we care about here is that parabola is a curve that goes up really steep then as it starts to turn over it gets a little bit flatter stays flat turns over gets steeper again going down absolute value what i've called abs abs that is a function that goes up up up and then very sharply turns back down so it's a straight line slanted upwards kind of like our y equals x and then it very sharply turns around and goes slanted downward in a straight line like our y equals negative x and then finally sine that is our wave so y equals sine of x so that y goes up and down at regular intervals in a wave as x gets bigger all right so those are our six columns of the table next let's just take a look at what's inside one of those columns and the way we're going to call one of the columns uh of the table is a little bit different than the syntax you've been using with patrick to call the columns of the panda's data frames i believe we're going to do tbl tbl so from table what we've called our tbl that's the name of our table open square bracket left bracket x it's already trying to suggest uh what we're going to do open bracket we're going to do quotation mark x quotation mark and then close square bracket so it's tbl open square bracket uh

quotation mark x quotation mark close square bracket so basically um in x or in table we're calling the column called x which is saved in quotation marks uh sort of like how we do for pandas data frames they've they've got quotation marks around because they're a string they're they're a word and then if we hit enter "out 17 less column name equals x type equals float 64 length equals 100 greater 0.0 0.1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0.2 0 2 0 2 0 2 0 2 0 2" all right i'm gonna stop that there um i don't think we need to listen to the whole thing but you might have noticed it was a little different at the beginning from how a panda's data frame prints out um a column uh but it tells you there's a column the column name is called x it tells you that it's a float which means it's decimal numbers tells you the length of the column i actually don't i guess i wasn't paying enough attention hopefully it said 100 there because there should be 100 numbers and then again if you remember from before those are the same numbers that were in our x array that we got from numpy dot linspace so phew this astropy table has preserved the data that we made and again the um and then that we put in that csv and again the curriculum goes through in detail every step of how we made this synthetic so again this not real data to represent shapes put it in a data frame and then get it into this astropy table that we now have awesome so we know which columns are in our pen or in our astropy table and we've checked on one of the columns to make sure that it's a new notification from oh this is new note of sorry about that folks i am uh i'm borrowing a friend's laptop to do this since i am a mac user primarily um and in fact i don't even know how to put a windows laptop on do not disturb so we might have some notifications coming in um hopefully not too many um okay moving onwards um so we've checked on this astropy table we've made sure it doesn't look fishy kind of is behaving for our purposes an awful lot like a panda's data frame that's good news that is all we really need to care about and with that we are going to uh check off that data preparation stage and celebrate because we are about to get into actually sonifying things with one caveat first i want us to talk about what astronify is and how it works all right so astronify is a python package we've installed that is developed by scott fleming at the space telescope science institute clara brosser at space telescope science institute and now at the university of saint andrews um jen kotler at space telescope and kate meredith at glass education and it is a package designed to sonify or represent a sound a very specific type of astronomical data called a light curve light curves represent the observed amount of light from an object typically a star as we see it on earth so it observe it measures the amount of light we see um generally from a telescope so how much light we are collecting from a telescope from a specific star over time astronomers like to call that observed amount of light how much light we receive from an object here on earth we like to call that flux so we are observing the flux the amount of light seen from a star over time thus light curves allow astronomers to observe variability in the amount of light a telescope records coming from a star or to record how bright a star basically appears to us on earth observing this variability can give insight into all sorts of incredible phenomena including explosive stellar activity called flares so that would be a sudden increase in brightness at a period of time and

small periodic dips in the amount of light observed from a star caused by a planet orbiting that star and blocking some of its light as it passes through our line of sight and orbits the star um as it passes through our line of sight with the star we call this latter phenomenon a an exoplanet transit and we've used it to discover thousands of planets outside our solar system orbiting other stars what's essentially happening is that at regular intervals as the planet orbits the star it gets in between our telescope and the star and blocks a tiny fraction of the light and it does that over and over and over again so you know if you see a star that looks like it's getting dimmer by a tiny regular amount at regular intervals you can be like there might be a planet getting in the way there and boom you found a planet. Astronify works by mapping the observed flux again remember that's the observed amount of light we see from the star that's our dependent variable that's changing with time to pitch such that higher fluxes are represented by higher pitches so if you have a moment where you observe more light that data is going to be represented by a higher pitch each observation of the star's brightness is thus represented by a note where the pitch is controlled by the value of that observation it is worth noting here that although it's the default to have brighter like higher fluxes brighter moments brighter data points mapped to higher pitches the user can set a parameter within Astronify to invert the pitches such that higher fluxes more light are mapped to lower pitches there might be you know personal preference reasons you might do that. We might get into discussing parameters involved in Astronify sonifications next week and this will hopefully make some more sense with some more examples so I'm going to once again going to turn off NVDA's speech mode so it doesn't talk over the sonifications speech mode off speech mode beeps nope I don't want it on speech mode off all right and I'm going to go to Astronify's example this is straight from their web page I highly recommend that you check out Astronify's web page it's again linked on the curriculum um it's a really incredible project I think it's great so I highly recommend you check it out and I'm going to play something this is a sonification of a light curve containing a stellar flare did you hear that I'm going to play that one more time so the sound stays relatively constant until boom there's this moment of where it gets way higher pitched and we know from the mapping we've just discussed that that higher pitch is representing a moment where the star gets a lot brighter as we see it and that's the flare a flare of light and activity from the star that we can really clearly hear as a change in the pitch of the sound representation of the data all right cool now as a concept check moving uh thinking back to our discussion of modification and parameter mapping do you think that Astronify is parameter mapping or modification I'll wait for a second it's awkward to like wait and and give you a chance to think about it when there's like no one on like a zoom call with me or anything I'm just sort of like yes I sit in uh I sit in silence and I uh I let people hopefully online think about it okay that's probably enough time um so Astronify is parameter mapping because in this case the flux or how high y goes up or down our independent or our dependent variable it goes up or down is mapped to pitch so it's not controlling the amplitude of the sound wave which remember we perceive generally as volume it's controlling the pitch so that is a parameter mapping

where we have mapped the flux to the parameter pitch in our sonification all right now here we've discussed Astronify within the context of its intended use sonifying light curves however light curves are just the type of 1d data or what I would call 1d data in astronomy so we have two variables time and flux x and y I suppose mathematicians might tell me that they're actually two dimensions 2d because it's time and flux those are two dimensions I'm going to call it 1d variable um hopefully there are no mathematicians who are going to like hunt me down for this but there are all kinds of 1d data out there including maybe some data that you've uh you've dealt with before so we can realistically present other 1d data using Astronify beyond just light curves but before we get to all that complicated stuff let's go back to that sonification education point we were getting back before and this was inspired by a talk that Scott Fleming one of the developers of Astronify gave at the Space Telescope Science Institute's Day of Accessibility wherein he talked about about sonification education and gave some examples where we sonify um where we where he sonified with Astronify some of these simple shapes very similar to what we're about to do so now we're going to make our own sonifications you're going to get a sense for what Astronify does all right I'm going to turn my volume down a little bit so I don't blast our ears out with uh with NVDA "speech mode speech mode talk" I'm going to turn the volume up uh I'm going to turn the volume up hopefully not too much hopefully that's a reasonable amount of volume we'll hope for the best okay we've installed Astronify but we haven't imported it into this Python session so what I'm going to do is I'm going to write from "F-R-O-M" from Astronify that's "A-S-T-R-O-N-I-F-Y.series" it wants to finish for me but I've like finished the sentence or finished the line for me but I'm not gonna let it from Astronify A-S-T-R-O-N-I-F-Y dot series "dot s-e-r-i-e-s" so from the subsection of Astronify series import so space "space i-m-p-o-r-t" import and we're going to import a function called Soni series that is capital s so capital S-o-n-i-series capital S-e-r-i-e-s so the s in sauna and the s beginning series are capitalized from Astronify series from dot series import sauna series "wx python is not found for the current python version pyo will use a minimal gui toolkit written with tkinter if available this toolkit has limited functionalities and is no more maintained or updated if you want to use all of pyos" all right so I just let it play some of a warning that came out I think a lot of people have been getting this warning I've been getting this warning I think Patrick did as well I heard from some other people from the live class that they got this warning nothing to worry about it's just like a little warning maybe bad advice but I often ignore warnings if it's not an actual error message so maybe don't take that to heart but I just wanted you to hear it so that if you hear it on your computer you don't worry it's no biggie and now we're going to get started with the sonification now this is going to involve inherently some kind of repetitive four lines of code I'm going to go through the first time or two relatively slowly but then I encourage you and then I'm going to get faster just to you know so we don't take too much time it's again all of these lines of code are on the online curriculum so if you get lost if I start moving too quickly at some point just go and copy and paste it because it's all a little bit repetitive it's like the same thing with like one word changed so there's no shame in like

copying and pasting as long as you understand what's going on and couldn't do it yourself afterwards but most programmers copy and paste so what we're going to do first is make an instance of a soni series so the sauna series is a class we're going to make a soni series object I'm going to call it soni underscore linear "s o n i" underscore linear " l i n e a r" soni underscore linear and I'm going to do equals space equals and I'm going to do um and the reason I'm calling it soni linear is because this is going to be the sonification object the sauny series object for a linear equation so remember that's our straight line going up "space s o n i s e r i e s" remember that's all one word with the s in soni and the s in series both capitalized then open parenthesis tbl so make a soni series object from our table which we've called tbl now there are two extra things we need to do here in this function and that is we need to say hey astronify I know you're made for astronomers sonifying light curves but um actually so I know that you expect these time columns and value columns to be like time and flux but actually we don't have time and flux we have like x and linear that's the name of our columns so we're just going to tell soni series hey expect time call the time column to be x and val call to be linear the value column so that's tbl comma space time "t i m e" underscore c o l "line c o l" equals "equals" open quotation mark x close quotation mark so say the time call is our x column that's the name that we printed out above when we checked the column names in tbl comma "comma" space "space" and then the value column which is val col "v a l" underscore "line c o l" c o l equals "equals" quotation mark linear "l i n e a r"close quotation mark close parenthesis soni underscore linear equals soni series with the s's capitalized open parenthesis tbl comma time underscore c o l equals open quotation mark x quote close quotation mark comma v a l underscore c o l equals open quotation mark linear close quotation mark close parenthesis here's the next line well we haven't actually played anything we've just made the object there's another little step I want to take I want to say hey sonification um make the notes kind of a wider spacing and this is something I've played around with already um and a choice that I've made again remember how data representations involve a lot of choice I'm increasing the note spacing here from the default just so it's a little bit slower and easier to hear so the way we're going to do it is we're going to do soni_linear the thing we just made dot note underscore spacing dot n o t e that's note n o t e underscore space equals "space equals" space and I'm going to set it to 0.05 this is something that I have in advance tested and decided is a good note spacing for us again choices we can make "0.05" all right sonny underscore linear dot note spacing equals 0.05 in 21 all right we've changed the note spacing now we're going to do soni underscore linear "s o n i" underscore linear "line l i n e a r" dot sonify that means make the sound representation of this thing "s o n i... one" open parenthesis close parenthesis because it's a function but we don't have any parameters we want to put in so sonny underscore linear dot sonify make the sonification make the sound representation "in 22" and the last thing we need to do is soni underscore linear dot play open parenthesis close parenthesis all right and this is the moment where I will once again turn off NVDA so it doesn't talk at us um you might get a warning that says like port MIDI closed that's what NVDA is going to try to read for

me if you get that warning uh don't worry about it um but I have uh "speech mode" off I've turned off speech mode I'm going to turn up my volume and I'm going to hit enter and it will play our sonification before we start again remember what we're sonifying we're sonifying the shape of a straight line slanted upwards so as time as x gets bigger we should expect y to get bigger and remember y is mapped to pitch I'm going to play that again I'm just typing that same line again exact same thing just so I can play it a second time okay can you hear that the pitch gets higher and higher in a sort of straight line as we move along in time which is the same as moving along along that value you might hear this sort of this sort of wobble that's just because there's like a little bit of a spacing again that's actually the spacing I've done between the notes where I've spread them out where there's a little bit of space but some overlap between the notes and your ears hear that as like kind of a wobble in the noise but the pitch in general is going up in a straight line representing that straight line speech mode speech mode talk all right NVDA is back on turn the volume back down so we don't hurt my ears here okay we've got plenty of time so that's good let's sonify our negative linear equation so that's the straight line slanted downwards and I'm going to this is the exact same lines of code except instead of calling it sonny linear I'm going to call it sonny underscore neg underscore linear and instead of the value column being linear it's going to be neg underscore linear which is the name of that negative line column in the table so let's repeat those same four lines of code "s o n i line n e g line l i n e a r space equals" so soni underscore s o n i underscore neg underscore linear equals space soni series s o n i again remembering those capital s's s e r i e s soni series again take open quote or open parenthesis tbl make a soni series from our table tbl "comma time call space i m e line c o l" sorry that should be time call that k was an accident time call so remember it expects the x-axis the independent variable to be called time we've called it "equals tick x tick" we've called it x "comma" and our val col space a "l line c o l" our value column equals in this case is neg linear tick "n e g line l i n e a r tick" right paren sonny underscore neg underscore linear equals sauna series open parenthesis tbl comma time underscore col equals open quotation mark x close quotation mark comma val underscore call equals open quotation mark neg underscore linear close quotation mark close parenthesis make us on a series object from the table call it soni neg linear where the time column is x and the value column is neg linear "in 25." all right we're going to once again change the note spacing "s o n i line n e g line l i n e a r" sonny underscore neg underscore linear dot notice underscore spacing "dot n o t e line s p a c i n g space" so soni underscore neg underscore linear dot note underscore spacing equals "equals space" 0.05 "0.05" that's the same as we did before "in 26" and then remember those last two lines "s o n i" soni underscore neg underscore linear "line s s n e g line l i n e a r" soni underscore neg underscore linear dot um sonify "s o n i f y left and right paren" open and close parenthesis we don't have anything in them sonny underscore neg underscore linear dot sonify parentheses in 27 and then the last line is that sonny underscore neg underscore linear dot play open quote or open parenthesis close parenthesis "s o n i line n e g line l i n e a r dot p l a y left right paren" and i'm going to

turn off nvda again "speech mode off" and i'm going to turn the volume up and you can hear the pitch starts high and goes low let's play that again i keep expecting to hear nvda like read the line because i'm typing the same line again right now to play it again um and i expected to speak to me but of course i just turned speech off so um jokes on me i guess that's my own fault soni underscore neg underscore linear to play it again excellent all right i'm going to do one more thing you can hear again it goes much in the same way that sonny linear went up in a straight line here the pitch goes down in a linear fashion it goes down in a straight line with time because this is representing a line slanted downwards all right let's um i'm gonna do one more thing i'm gonna do soni underscore linear dot play so you can hear that just sort of side by side so this is that straight line slanted up sonny underscore linear dot play just to play that straight line slanted upwards this time cool can you hear that yeah it's going up the pitch goes up this time instead of down i feel like i can almost like feel it going up um maybe that's sort of a weird thing to say maybe i've just listened to a lot of astronify uh sounds um it's kind of fun and spacey i like these they kind of feel like aliens in a fun way i like it um all right i'm going to turn speech mode back and i'm going to remember to turn the volume down so we don't like pain anyone all right and what i'm going to do next is i'm going to sonify the sine wave function remember that is um our wave that goes up and down y equals sine of x it's a wave that goes up and down now we are again this is the same four lines of code where we make the sauna series object we change the note spacing we sonify it and we hit play and uh just with those couple of little differences but because it's so similar i'm going to start moving a little quicker again all of this code is on the curriculum online feel free to copy and paste if you you know feel the need um i understand but also make sure you understand what's going on so this time i'm going to call it soni underscore sine

"s-i-n-e s-o-n-y line l-i-n-e" l've called it sonny line i actually want to call it soni sine "s-i-n-e" all right soni underscore s-i-n-e "equals space" equals Soni Series with those capital s's open parenthesis tbl comma space time call underscore equals x in quotation marks x comma val col a underscore c o l equals sine because that's the name of the column with the sign function in our table and again if you don't remember you can always just do tbl dot col names and hit enter to print out the list of column names in your table if you need to check "s-i-n-e tick" and again remember sign is in quotation marks and it's spelled s-i-n-e and close the parenthesis or it'll throw an angry "in 31" changing the note spacing soni dot sine "line" soni underscore sine sorry not soni dot sine dot note spacing underscore spacing "s-p-a-c-i-n-g" space equals 0.05 "0.05" soni underscore sine dot note underscore spacing equals 0.05 just move the notes a little further apart "in 32" soni underscore sine dot sonify "s-o-n-i-line s-i-n-e dot s-o-n-i-f-y" son underscore sign dot sonify "left paren right paren" make those parenthesis "in 33" hit enter sonny underscore sign s-o-n-i-line s-i-n-e dot play and we will do the same business of increasing the volume and turning off nvda "speech mode off" and hitting enter oh that one's a lot more fun right i think that one's a lot more exciting than the straight lines you

can hear the as the as that wave goes up and down so as y gets higher and then lower higher and then lower the pitch goes up and down and up and down at regular intervals i'm gonna play that one again actually because i really quite like that one so i'm just typing that same line sonny underscore sign dot play cool that's our sine wave we've played two different lines one going up one going down and we've also played our sine wave let's turn the volume back down "speech mode talk" all right there are just two more functions to sonify here they are our quadratic or our parabola so again that's the curve that starts steep turns over once a little bit smoother at the top and then uh turns back downwards and gets really steep again and then also our absolute value function which um goes up in a straight line very sharply turns back around and goes right back down the goal here is i want to sonify these two i'm going to play them actually side by side so i'm going to make both sonifications wait to play them until they're side by side because i want you guys to learn to hear the difference between a sharp turnaround and a more gradual turnaround so a more gradual turnaround in our parabola which is a smoother curve and then a really sharp turnaround in the absolute value and again we're going to move a little quickly through the code here so feel free to copy and paste or to check the curriculum if you need to so that's so i'm going to call this soni underscore parabola space equals space and then it's Soni Series not forgetting those capital s's open the parenthesis table time underscore col equals, equals x with x in quotation marks another comma space val_col = here it is called 'parabola' is the name of the column "in 36" all right we've made our parabola sauna series we're gonna change the note spacing "s o n i line p a r a b o l a dot n o t e line s p a c i n g" soni underscore parabola dot note underscore spacing space equals space 0.05 "0.05" same as before just doing that because that's what i tested and i think sounds best feel free to change that note spacing make it bigger or smaller and see what you think "in 37" again that's tweaking with uh with the parameters of the sonification to see what you prefer again full of choices then soni underscore parabola "s o n i line p a r a b o l a" dot sonify so make the sound representations and i'm gonna hit enter but i'm not gonna do um soni underscore parabola dot play just yet because again i want to sonify the absolute value thing at the thing function at the same time so let's do that really quick i'm going to call that sonny underscore abs abs because sonny underscore absolute underscore value sounds really painful to type out so abs soni_abs = SoniSeries with the capital s space "s o n i s e r i e s" soni series open the parenthesis tbl comma time_col='x' remembering our quotation marks val_col made a typo there v a l underscore c o l i feel like that's kind of easy to make a typo on equals abs because that's also what the name of that column is because i guess i never wanted to type absolute value so i just called everything abs so soni underscore abs equals sauna series open parenthesis tbl comma time underscore call equals open tick or open quotation mark um x close quotation mark comma val underscore call equals open quotation mark abs close quotation mark close parenthesis "in 39" soni underscore abs dot note spacing we're changing the note spacing s o n i dot a b s nope i did i did soni i did soni dot abs i meant to do soni underscore abs nope i did space i'm gonna do soni underscore abs dot note spacing sorry my brain is apparently

leaving me "dot n o t e line s p a c i n g" soni underscore abs dot note underscore spacing equals 0.05 all right importing and then sonny underscore abs dot sonify open and close the parenthesis and now we can play the parabola so that smooth curve going up and then turning around going down next to the absolute value which is that sharp turnaround like the top of a triangle so first i'm going to do the uh i'm going to do the parabola that's soni underscore parabola dot play soni underscore parabola dot play open and close in the parenthesis and again turning off nvda "speech mode off" turning up the volume and hitting enter oh no we have an error "speech mode beeps speech mode talk" this is terrible "line a b s dot p l a left right paren pio warning port midi closed in 43" interesting um i got an error when i tried to play the parabola but the uh the absolute value seems to have played let me try the uh let me try the parabola again... ...that's soni underscore parabola dot play open parenthesis close parenthesis i'm going to turn off nvda and hope it works this time "speech mode off" all right i guess it will work this time i guess it was just angry before i'm going to turn nvda back on so i can play the absolute value again "speech mode beeps speech mode talk" that's going to be soni underscore abs "speech mode off" so that's sonny underscore abs dot play you're going up up up up up and then suddenly changing and going back down at a constant rate let me play the parabola one last time so you can hear them side by side again again that's sonny underscore parabola dot play open and close the parenthesis it's a much slower change to the pitch turning and going back down we are literally hearing the difference in those shapes a sharp versus a slow turnaround i encourage you to play that often on your end just until you get that sound for the feeling for it and again play all of the sonifications again and again i'm going to play the sign again sonny underscore sign dot play we can hear that wave pattern going up and down that we don't have in that absolute value or the parabola the there's more turnarounds in the pitch going up and down than we have in those other two shapes so again the goal here is to develop an intuition for what astronomy is doing and how different shapes how different behaviors of our data because again this is all about a representation of the data it's what is this y doing as x changes which we have represented as what is the pitch doing as time changes it's developing this intuition for what is going to happen to our data representation for different types of data behavior that is the goal here that is all about what this education process is about this is what we teach people to do with data visualization for years and what we're just developing an intuition for now this is where i'm going to conclude the fourth tutorial from this non-visual data science tutorial series this is the first of two sonification tutorials and the fourth of five total tutorials so next week we'll pick up with more data sonification using astronomy getting with some more realistic data i'm really excited to see you there i'm excited to do more sonification with you again we are very grateful to pandas and num focus for generously supporting this tutorial series and i encourage you if you have questions to reach out to me or patrick via email or to come to our office hours same zoom link as the tutorials 6 to 8 p.m uk time on zoom and also to check out our curriculum online where all of this code is hosted yeah thank you so much for coming i'm excited to talk more sonification with you all

next week hopefully in a live recording if the sound permits all right thanks all i'm just gonna wait a few seconds because i'm worried the recording is gonna like cut off the last few seconds um and i don't want it to cut off like the actual end so if this is still in the video sorry for this like awkward few extra seconds where i'll like wave at the camera and sort of ramble on rambling rambling rambling also crossing my fingers that the recording worked because if that didn't work i'll be quite sad and i guess sit here for two more hours which would make me quite sad um so fingers crossed everyone i suppose if you're seeing me still rambling here then it's worked all right i think that's probably enough time i'm gonna go stop