NCRM text data workshop: Part 1

Hello, everyone, my name is Lewis Brace. I'm a lecturer at the University of Exeter, where I'm part of the Q step centre. And I am hosting this ncrm workshop on text data analysis.

This workshop is going to be broken down into three parts, as denoted by the three separate videos. Part One, which we've got on the screen here now is going to be a very, very quick, broad overview, Introduction to Python. Okay, it's going to be fairly quick, and we're only going to look at the elements of Python that we'll need in order to cover in order to do the tasks, we're going to look at parts two and three. Part Two, will then look at using Python to collect textual data, I help to scrape a website. And then in part three, we'll look at some basic introductory analysis techniques that you can use in order to analyse text data.

So each of the three videos will also be accompanied by what we call a Jupyter. notebook. Okay. In the broadest sense, you can think of Jupyter Notebooks as interactive textbooks. So, you'll see as we go along for today's session, I'll have these notebooks up on the screen. And you receive a big container a lot of text, but also in fairborn. Down, they have some interactive Python terminals. And these terminals here are interactive, so you can edit them on the fly if you want, and random and they will still run exactly like they would if you run in Python code and your ID.

If you look in the material supporting resources materials folder for this session, you'll find some in you'll find some PDFs in there that I've put together that will show you how to download install Python onto your computer. Those guides show you how to download the Anaconda distribution of Python, which is the distribution I recommend my students download and use. Now I've linked distributions as well, such as the umbrella distribution, and so on, which are perfectly adequate.

There's a new distribution called pycharm, which I heard is pretty good. I'm going to use it myself yet. But these guys you've got unlike what I tell my students do is download Anaconda. The reason I tell them to download the Anaconda distribution is because it contains a lot of the libraries and packages you'll be using in Python, which we'll cover towards the end this first session come pre-installed with Anaconda.

But the Anaconda distribution also contains the not only contains the spider ID to enable you to edit your and actually run Python code. But it also contains the software you need in order to be able to publish Jupyter notebooks. So, if you follow his guides, they should if you don't already have Python on your computer, if you follow this guide, they should be able to take you through a step the step by step process of downloading pi download Anaconda, installing it on your machine and opening Jupyter Notebooks as well. If you have any issues with O's, overdoing any of that my email address is on each of these Jupyter notebooks. There's also an over documentation stuff you'll see when you download the materials folder, so just feel free to drop me an email if you have any issues, and I shall help you to the best I can.

Okay, with all that said, let's start our introduction to Python. So, learning Python for the first time, I have no idea why this image exists. But it's often presented to snakes films, it's got to work with like learning a language for the first time. Okay, so as I said a moment ago, this first part, today's session is a very, very, very quick introduction to Python, we are only going to cover the bits that we need in order to be able to understand the code that we're using to collect Python data and analyse it later on. Okay. As such, there's going to be large parts of Python I'm going to just skip over because we don't have time to go over that to go over the language in that much detail today.

If you follow this link to the extra two step resources GitHub page on there, you'll find a lot of introductory Pay for materials I've developed previously. So, you'll find caught, you'll find workshop, previous workshops and courses up developed and down that have introduced around the concept of introduction of learning Python and using it for data analysis and stuff like that. And you'll find these workshops are a lot more in depth than what we're going to do today. And so, if you're interested in learning, more developing your knowledge and skills with this language, I suggest you click on that link and take a look at those resources. All of all resources contained on this page are open source, which means they're trained for you to use. And likewise, as you're going through those resources, if you have any questions, please just feel free to reach out to me, I'm happy to help.

Okay, so what is Python? The sort of official definition, if you will apply for it is as a high level interpretive, you know, programming language with a batteries included philosophy. Now, that's a very long, complicated sentence. But what does it actually mean? Well, as you can see from my slide here, there's four main components to this sentence that is a generic programming language that is high level, that its interpretive, and that it has this batteries included philosophy, philosophy, a general purpose, okay, so a general purpose programming language is a programming language that is designed to be used in the widest possible variety of contexts.

Okay. So, with languages like Python, Java, and C, you can use them for a wide range of different functions, you can use them for data analysis, you can use them to inject API's, you can use them to as the base of some kind of other software, and so on. If you, for those of you who are familiar with our, for example, I'll present a good contrast to Python and r is technically a formal scripting language, right? Because it's designed specifically to do statistical analysis. And only in recent years, it's been developed a little bit more than it can now do other stuff, such as text data, and some web scraping and so on. But at its core, it was designed to do statistical analysis. So that is not a generic programming language, because as this one sort of main aim in mind, were with backstamp, Python, Java, C, and so on can be used for a whole range of different tasks.

Second aspect of python is that it's what we call a high-level language. And what this means is that the syntax of the language, i.e. how the code looks when you have code looks when you type it into your terminals, or your user interfaces and so on. It looks more like a natural language, okay, i.e., it's more sort of English than it is in when you compare it to other languages. So, if you look at the code, syntax for C, for example, unless you're familiar with that language, it looks almost alien, it's hard to understand it's not immediately intuitive as to what the codes doing. Or say, if you compare a Python

code and look, you know, Python code is pretty obvious from the get-go, even if you're not familiar with it. And you'll start to see this as we go along during the first part of today's session.

It's an interpreted language. Okay? What does this mean? This means that your code isn't directly run by this hardware. Instead, you wait when you bought your Python code, and you run your Python code, your Python code is not automatically converted directly into machine code. Okay. And by machine code, I mean that your computer doesn't understand the languages you type. Okay. So, when you have Microsoft Word open, for example, you type your sentence, you know, a cat, the cat sat on the mat, right? But computers now understand the words the cat sat on the mat, it's not dealing with the words under the hood, right down at the bottom, all, each one of his characters has been converted into a series of zeros and ones, okay. And that's how a computer understands it. When you write your Python code, your code isn't automatically directly converted into zeros and ones, okay, it's sent to a special piece of software called interpreter. And the interpreter, interpreter then converts your Python code into machine code that your computer can understand and work with.

This is compared to see where you run your code. When you in order to run your code your code, you need something called a compiler which compiles your language together and converts it directly into machine code. And as we'll see in a minute, the fact that Python is interpretive means that it has a whole host of benefits, particularly where we're concerned for the kinds of tasks we want to do today, such as text analysis, and so on.

Finally, Python has this batteries included philosophy. This means that if you want to do a task, okay, the chances are there's something called a module out there, which already exists to do that task. Modules are prebuilt bits of Python code that people have built in order to a very specific task. So, for example, if you wanted to create some graphs, okay? If there was not a module for creating graph, you would have to, in your programming language, first create the canvas that you want your graph to be on when you need to create each of your individual axes, and then the axes labels and the data and so on.

In python, however because it has this sort of batteries included philosophy, which refers to these modules. Instead, you can say import my graph module and feed in the state you're in produce me a graph, it makes a lot easier for you. And our modules do a whole host of different things. So, for example, here, we're going to import anti-gravity. And if we import the package, anti-gravity, it just takes us to this little comic strip. And that's just a sort of very silly example of a Python package.

Okay. In reality, there are so many useful packages. That said, I've already mentioned the graph in one. One of the most fundamental packages you'll encounter is called NumPy, which stands for numerical Python, which has packages designed to do large scale numerical calculations. One of the other main packages which will come across later on today is pandas, which is designed to basically allow for easier data analysis and data wrangling in Python. For those of you who are familiar with our, this whole idea of modules is exactly the same as the concept of libraries.

Okay, so what are the advantages of Python? I was said a minute ago, Python is an interpreted language, which means it has a number of advantages. Firstly, it has automatic memory management. What does that mean? Well, it means that the computer works out how best to run your code for you.

So, to contrast Python with C blank programming language C, as in just the letter C, for example, when you're coding in C, you need to use these things called pointers are pointers are essentially a way that you basically will view telling the computer what part of the memory to use when it's executing a specific part of your code. Now, these can become this can become very tedious. As I'm sure you can tell from the anger in my voice, I've had to deal with this quite a lot. Python, however, says don't worry about that. I'll go all for you. And Piper has some very clever algorithms that operate under the hood that you don't have to interact with in order to do this kind of memory management for you. It is worth mentioning that you can edit these algorithms if you want to customise how the memory is used and how your code is run. However, I personally have never thought we need to do that. And I do some incredibly computationally heavy tasks. But I've so far found that these algorithms are efficient enough that I haven't had to alter them.

As I mentioned a moment ago, the syntax is in English. So, a natural language makes it nice and clear and easy to understand this. This makes programming a lot easier in Python, as compared to other languages, which in turn minimises development time. Okay, it's quicker to develop a script in Python and in other languages. And I will say Python as this emphasis on importing modules, which makes life so much easier, because it means you have to build everything from scratch all the time.

Okay. However, as much as I like Python, it's worth mentioning that there are some disadvantages to it. Firstly, interpreted languages such as Python tend to run a lot slower than compiled languages. So just as a rough estimate, if you develop the script in Python, and then develop the script to do the same thing in C, the C version of the script will run roughly eight times faster than that of Python and be able to script in Python. Okay. This means that there's a sort of trade off to be made. Okay. And as you can see, down here, I say which language is best, there's a whole bunch of factors you need to take into account okay. This graph is particularly pertinent to this sort of C Python example. Okay. So, for example, I could develop a set of script and see in the scripting pipe to do the same task. And one in C might run eight times faster. But because C is this, because it's so much harder to code in C, it might take me 20 times as long as actually build the script, and actually get to a point where it can run. So, I often have to make this tradeoff between efficiency in terms of CPU time, and how long it's going to take to run the task versus development time. Because I'm quite lazy, as most computer scientists are, I often choose the low development time just to get the script working, and I just never run in the background, while I go off and do other things.

And the modules we import are developed in a decentralised manner, which basically means that they're developed by nodes, four nodes, okay, and there's no sort of oversight body that does quality control with them. Okay. However, despite that, it is worth mentioning that the internet sort of sort of created this sort of natural selection, sort of mechanism for packages in that packages that are very good, and don't tend to get discussed much. Whereas packages that are really good get talked about by the online community along get shared a lot more. So, there's that kind of natural selection approach to even if there is not a centralised body monitoring the quality of them. And multithreading used to be difficult in Python. But it's got a lot better in recent years, it's even that it's fairly straightforward now.

Versions of Python. Okay, so it's worth mentioning quickly that when you look at Python, when you look at Python, those two main versions out Python two and Python three. And this refers to the way in

which Python is a relatively young programming language. Okay. And that means that when Python was released to the wider public in the form of Python two, as people started to adopt it and use it more and more, they had some issues with it, they needed correcting. As such, there was eventually this huge overhaul of the Python language, where live parts of it were sort of redeveloped tweaked a little bit. And this is released, released as Python three.

The syntax for two is slightly different. Okay. And this is the main reason I mentioned this is because if you google Python code examples, you might sometimes see little, little differences between code examples, if the code example you're looking at is 10 years old, or so. Okay, we're more recent examples will look slightly different. Okay, just do bear that in mind, there are some differences in the syntax. However, as I said, all of the modules that are used widely have been adopted have been out there for Python three now. And indeed, Python two, as of last year, isn't even supported anymore. So, everyone out there is using Python three now. Just be aware, you're on your websites like Stack Overflow, so on, and you're looking at particularly old posts, syntax might look a little bit different.

Okay, so background to Python. And it's not to look at it quickly. And how to actually code in this language. Okay, so variables. As those of you who are familiar with our, you'll perhaps be familiar with variables, they are pretty much work exactly the same way here as it would in our, for those of you haven't done any programming before. You can think of variables as being alphanumerical characters that stand for something. Okay? So, for example, if we look at this example, down here, okay, here's some Python code. Okay. And this line here says intvar equals five. And if I were to run this line, right? Actually, let's just run this line over that. Okay. And yes, it looks like nothing happened. But something did happen. What this did here was tell pop my python terminal, that whenever I type intvar, I want it to equal five. Okay? So if I say print into this trick word here is a function, which we'll get onto in a moment. But basically, as I'm sure you can tell from context here, this just says, print this variable to the screen here. That's all that line says. And if I run this, you'll see five comes out, because python now knows that the term intvar equals five. Right?

Now, by convention, it's common to have variable names start with lowercase and unsaved, uppercase letters, for other things, such as classes, and so on, which again, we'll go into later on. That's the convention that most people use. However, if you're coding just for yourself and not as part of a team, then you can pretty much do whatever you want. But I would recommend that at the very least, you remain consistent with yourself. Because if you start mixing up capital letters and lowercase letters, just as you're going through your code, it's only a matter of time before you trip yourself over.

Some keywords cannot be used as variable names, because they have built in functions in Python, for example, the thing, okay, so you can't have a function just called print. If I was to find do that, you know, I could type, if I was to run this line of code here, Python would throw an error messages out can't do it. And pipe is pretty good to tell you right? Now. Because the text is a black screen too. Fast pythons way of sort of telling you No, you can't use this because, you know, this is a function. And as a very specific role to play in Python, have examples of keywords, you can't use variable names or things such as an ID or, and you'll start hitting, you will start to notice these as you go along and start to use Python more and more. Like I said, Your Id always that you know, by showing a different colour anyway.

word variables, letters, stuff like that. So, you see here, we've got the string variable. And then we've got to speak, we've got our speech marks, and in there,we have a word if you will, which is food. It's worth noting, you can also use single speech marks, but you have to make sure you're consistent. So down to that one double speech markup start on one single one at the end isn't going to work, it will show you an error message. I recommend that when you're dealing with strangers use to this this allows you to then do things such as apostrophes with industry. It's also worth noting, because we're dealing with text data in this workshop, that if you're looking for matches in a string variable, then Python treats anything contained in these speech marks. Literally, and looked for an exact match. So, if I said for example, pipe and tell me if this sentence contains the word food, but over the word food with a lowercase F, when in the sentence is spelt with an uppercase f person who tell me that that word does not is not in that sentence, okay? Because it doesn't deal with capitalization is looking for an exact match. Okay, so it's looking for the word food with a lowercase f instead of a capitalised, capitalised, F.And then we have classes and stuff, which we'll get into later on. So here, do you see some examples. And if I run this, you can see I create three variables an integer variable, a float variable and a string variable. And I've used the print statement we looked at a moment ago just to print each one to the screen.

Going back to what I said a moment ago, about the differences between Python two of the Python three syntax when you're looking at examples online, a giveaway as to which syntax you're looking at, can actually be seen with a print variable, because in Python two syntax, the parentheses are not required. So, print statements look like this. Okay, whereas in Python three, we use the parentheses. So that's just a quick way to tell which kind of which version of the language you're looking at. And here is just an example of was 10, about the example the food and they're looking for an exact match. Along with some other examples here, type is another function.

Okay. And, as I'm sure you can guess, from its name type tells you what the type of the end what the type of the variable is. So here I've just said, Look, I've got my variable could intvar and I feed that into the type variable, and then I feed in turn, feed that into the click variable, okay, which sounds complicated, but all I'm saying here is like Print to the screen that type of the integer variable, do that. Do that and it thinks itself says class in case it's telling me that the variable ink bar is an integer, which is because it's we said assign it to number five.

Okay. And this example here is just sort of a further example, what we just looked at where you but whereby you can embed functions within one another. You can also convert variable types. So, if you look at this example here, if I run that, you can see we have our first variable, which is 295 dot 50, divided by two, which obviously creates a floating point number. And we can see that from both printing out the variable there and also print out the variable type, which comes out here.You know, on these print statements here, you'll see I've got a string here, where I have, in this case, one colon space two colon space, separated by a comma, all I'm telling Python to do here is to print this to the screen, and then print the variable. That's just so I get output like this, where I see 123 a master, so I can easily match up each of the lines of the statements to be quick print statement in my code, just so I can sort of match it for my own eye. And yeah, if we then take the variable, which at this point here is equal to 147,

point seven, five, and I feed that into the into arguments. So, the integer function is converted into an integer, okay, just the way of converting types.

That brings us to operators. So, operators, work in exactly the same way as any work as any of you've sort of done maps in school, or any form of wild programme forward understand. Okay, so you have your addition, subtraction, multiplication, here is one star division, which is forward slash and to the power of which is two stars. Okay, it's just an example here, we create a variable x, which is equal to 10, a variable, y, which is equal to two and then variables x, which is x divided by y. And then we ask it to print a Zed. And you see, he gives us five, which is the result of dividing 10 by two.

There's also something called the increment operator. So in both languages, if you have a variable x here, and you wanted to add one to it, to make it 11, you'd have to type x equals x plus one. Okay, which is intuitive, but, you know, typing that all the time is a bit lazy. So, it requires a bit of effort. And as I said, a moment ago, computer scientists are inherently lazy people. And I can say that because I'm one of them. And so they created this thing called the increment operator, which means if you want to add one to our variable, you can just type x plus equals one. And that will do exactly the same thing as if your x equals x plus one. Okay. And this increment operator works for all the other operators.

Well, that brings us to Boolean operators. And useful and quote using conditional statements, which we'll get into in a moment. And these include things such as and or not. So if you have a conditional statement, which again, we'll cover later on, so if you want to say, you know, if intvar and float varr equal this, do a thing, you use the and okay, but if you want to say equally, if you say you know if int_var equals five or seven, your thing, okay, so it's just a way of combining is sort of what we call conditional statements together, okay, and make conditional statements because they're saying, you know, we're only going to do this thing if this condition is met. But we'll cover that in a second. comparison operators. Pretty much self-explanatory.

You have a number of words, greater than which is an arrow point to the right, lesser than, which is an arrow pointing to the left., I've been coding Python for years and I still often confuse the two sometimes. We have an equal two which is the right arrow followed by an equals, lesser than, which is arrow and an equals and then there is is equals two, which is two equals, obviously, is equal to is two equals because one equals is how we create a variable. Okay, and if you try to use one equals here, it's such a in this example python would throw an error message.

Okay, so here's just an example. We have two variables int_var float_varAnd I say if int_var is greater than float_var print int_var is larger than float_var. And this is an example of a conditional By the way, okay? Because we're saying if this is true, do this. Okay, because int_var is five and float_var, 3.2, we think we get, we get this output because that condition is true, we say int_var is equal to five, int_var is equal to five, okay? If I say, if int_var is greater than five, you'll notice we no longer get that output. Okay?Now, just a quick note, to remember the difference between greater than or equals, to bet and or greater than or equal to variables. So functions, there is a difference there. And I know it sounds silly, but this is one of his silly little mistakes trips over even experienced programmes. It's just remember, you know, just make a difference from greater than or greater than or equal to two. So in this example, here, we have int_var in two, moving five, and if int_var is greater than int_var 2 to print this.

Okay? Which this obviously isn't true. So it doesn't print, when we say down here int_var is equal to or greater than int_var to print this, and we do get that printed out, because int_var and invalid two are equal to one another. Okay. So hopefully, that makes sense. I apologise. I'm rushing through this. But like I said, we don't really have much time to spend on basic Python here. And so if you want to go over any of this stuff in depth look at those resources I mentioned at the start of the session. Because there, there are workshops such as this, it will go into a lot more depth and take a lot more time to explain each of these sort of concepts.

It's very important to note. And this is especially true for those of you who have used our before that Python, like over all over generic programming language, as zero is a zero index based system. What that means is that the first element in any sequence has an index location of zero. The second element when has an index, location of one, and so on. So if we look at this example here, okay, we have a string, which is a pretty true statement, you know, dogs are better than cats, we then say, print the first element of the test string. And if we run that, you see we get the capital D, here, okay, which is the first element. It is also worth mentioning, so I forgot to mention it when we go there. We do indexing by putting the name of our sequence, and then the square bracket at the end, where we put in the number of the index of the element we want. Okay, so Likewise, if we want to do in dogs, which is the second element, but because Python index is zero, we go 012. We feed in one here, and you see we get the o. Okay, so remember that Python indexing starts at zero.

Just to confuse matters, if you wanted to go backwards for a sequence, then the numbers start at one. So let's say you wanted to get the last element in our string, which is the s in cats. You do square brackets when you put minus one and you get us. Likewise, if you wanted the T in cats, you run this and you get your T you put minus two Okay, and so on and so forth. You can get the middle, sort of middle sections, some string upside, of a sequences if you want. And you do that by adding your square bracket is normal, but then feeding in two numbers separated by a colon. And this will give you the elements within the range you specify. So here we got our re space b. So that's the re from ours space and then the b from better.

And I highly suggest that Oh, yeah, so also, if you just wanted the sort of first X number of elements from the sequence, or the last X number, you can get that by putting your square brackets putting in your index number and then putting a colon at the end of the fall, okay? colons are from a number will get all of the elements after location, five, when the colon before will get all of the elements before location five. If we run that, you can see the corresponding output there.

I highly suggest that if you decide you want to pay VAT, and if you decide you actually want to do some text data analysis using Python, I highly recommend checking out the resources I mentioned a moment ago, and spending some time with those resources and playing around and learning indexing. indexing is something that trips over everyone, even experienced programmers, such as myself, often make silly mistakes of indexing, and it causes us a lot of issues. There's one of those things it's worth playing with just to make sure you fully have equipment. I always wanted to want to mention commenting out code.

Quite often you'll write a bit of code that you don't want to run for a while for whatever reason whether it's just draft code, where each one of them that specific part of it, whatever you can run. And if you don't want to run this code, you can stop it from running just by putting a hashtag in front of that specific line of code. So, if we run this example, you'll see that Python runs all the code before the hashtag, but not after. Likewise, if I put the hashtag in front of here, you'll notice we get no output because Python is ignoring it. Okay. a handily little way broken up code.

That brings us to data containers.So Python has a number of what we call data containers. And you can think of these as sort of computerised Tupperware really. And that is just a way for us to store something, whether that's a sequence of characters of data, whatever. And, though different types of data containers in Python, okay, some you will use more than others, but each one has a very specific purpose. And it's worth being familiar with all the major ones before we do any code.

First Data container, we'll look at a dictionaries. Now dictionaries in Python, essentially work exactly the same way as a language dictionary does in the real world. Okay, ie, you have something you want to look up. And you have a definition. Okay? So, when you open up, you know, the Oxford English Dictionary, whatever, you know the word you want to look up, and let's say you want to look up the word bacon, you know, you, you're going to b b a and so on, you find the word bacon, and it gives you a definition of bacon. Okay?

In python terminology, the thing you want to look up is the word in our dictionary example you want to look up is called key. And this sort of definition, if you will, is known as the value. Okay? So, dictionaries are Python, often referred to as key value pairs. So, we have an example here, right? So here, I create my dictionary. And dictionaries are denoted by the little squiggly brackets here. And I'll create a new dictionary called prices, okay. And each entry into my dictionary is separated by a comma. Okay, so this is one entry here. This is another entry here, and so on. But each entry then contains two parts, it contains our key, are you the thing you want to look up? Here, my keys are strings, because they're obviously the names of foods. But equally, these could just be if you wanted to, these could equally just be integers as well. Okay? Or floats or whatever. Okay, we're here we've got our keys, and then we have a comma. Sorry we have a colon, and then our value, i.e. the thing that's returned, okay, so you can see here this is just a dictionary of some of my favourite food item. S,o my favourite food items and their corresponding prices. And if i ask pythod to print out that we see our dictionary has printed to the screen.

Okay, so what's the point of dictionaries? Well, you start to see, as we go through to the late sessions today, that if new things used to store data and access them, relative ease, okay, but just as a brief example, here, I asked quotes to print to the screen, the price of steak is calmer, I then put my prices variable, I then put the square brackets and put in the word stake. And you might notice that this looks exactly this works exactly the same principle we saw index and did a moment ago. And instead of using a number or index location, we're using the key from the dictionary. And if I ask you to print this out, you see that by printing the words the price of steak is and then space to the screen, and it goes right you want from the prices dictionary, you want the value that's associated with the key steak, so it goes to the key steak finds the value and insert it in for me. Okay, that's okay, I will be using dictionaries that are later on. In part two, so that brings us to tuples.

Tuples are a newer form of data container.Okay, so indexing, I want to cover indexing, because it's going to come up a lot in the rest of this workshop today. And it's a very important thing to understand. Okay. So, indices, enable us to access specific elements in a sequence. Okay, so here, a sequence can be anything, it can be a data storage list, which we'll look at in a moment, or it can just be a string. So no, you know, string, either consistent one word or sentence or whatever. And an element is a specific character within that sequence. So in this example, down here, elements are each of the letters and the whitespace. In the string, it was a list of numbers, and it'd be each individual number. Now, tuples are interesting, in the sense that once they are, they are immutable. Okay. And what that means is that once they are created, you cannot change the content that they have.

Okay. So, you create a tuple, as we do here, and tuples are denoted by the parentheses, what some of you may know, is just normal brackets. And you'll see here I'll create my tuple containing just two elements, five and 10. Okay. And I'll create my table. And once I've created, I can't alter the contents, okay? So, you can't add stuff to your tuple. You can't add new elements to a tuple. You can't remove them, you can't replace them. Okay. And so, I know right now, you're perhaps thinking, well, what's the point of these, this kind of container if you can't amend them. And tuples are actually really useful if you have a very specific sequence that you don't want to accidentally overwrite, which can sometimes happen when you're coding. So, for example, if you are building a computer simulation, and you have some initial seed numbers, that are random numbers you've generated, but you perhaps want to run the model a number of different times in various parameters, and you don't want to add to a and you want to a use the same seed numbers every time and be you don't accidentally override them, then you can store them in a tuple.

Okay, these are values, we won't be using them in this tuples in lush in this session, though, we'll be using a lot in this session our lists, and lists the perhaps the most frequently used type of data container in Python. And, unlike tuples, they can be amended, you can add elements to them remove elements, and switch elements around and so on. And they can also contain elements of different types. So, I've got to mention, two, I've got to mention when you're about to, when you're working tuples, every element in a tuple sequence has to be at the same type. Okay, so you can't have one integer, a float in a string, they either all have to be integers or have to be floats or have to be strings and so on. Lists however, that's not an issue. In lists, you can have elements of different types, you can have integers, floats, strings, all in the same sequence.

So, if you look at this example, here, we create a list called number list. Lists are denoted by square brackets. Okay, so be careful not to confuse them with indexing is perhaps hard to do, because it lists each element elysa separated by a comma, which is obviously not an indexing. So that should help you there. I then print off the list at the top of the list. And you can see here the list is printed containing the integers 123. And it's a type of list. And then in my second list, I create this new variable here called second list. And that is, again a list that contains one integer. Number one, it contains the list we just created called number list, and a string called Hello. And you can see from the printouts down here that python in is perfectly happy with that. It's perfectly happy to have one listen side never listen and so on. Okay.

Now why is tuples are immutable lists are mutable. So, like I said a minute ago, that means you can change the elements in them. And now a number of different ways to do that. Okay? So firstly, you can use indexing. So here, you see we've got in this first line with the printer here, you see, we've got our list that contains numbers 123. I then say, like Python, in the variable, list numbers, only to replace a second element. Even I'm typing one, because remember, python indexing starts at zero, okay? i wnat you to get the second element and I want you to convert to five, you can see from the second thing out, does that Okay, so python there's Okay, so 01, this, and this is the element you want, and it comes in and says, two, I'm going to set it to five. Okay. And that's fine. However, if however, you want to add a new element into sort of the middle of a list, if you will, but you don't want to overwrite any elements, then you can use the Insert function.

So, if you look at this example here and programme this, you see, again, we've got our list, which is just called numbers contains 123 is a numbers docking set. Okay, and here, we start to see the first example of something that's a little bit annoying in Python, in that not all the functions have the same syntax. And this is just, all in all programming languages have these little nuances. And it's largely due to how the programming languages sort of developed in a decentralised manner. Okay, so whereas with the print statement, for example, we see print, and it comes in a different colour, and then we put brackets over it. Here, instead, we have numbers dot insert, and dot insert the function, and we just say use dot insert, on this numbers variable within our parentheses, and we say, oh, and this parentheses, this Insert Function takes two what we call arguments.

Okay, the first argument is the index location where we want to insert something. The second argument is the element we want to insert into the sequence. So here we are saying insert the string obtains the word supplies into location to, okay, so Python looks at the list because 012 it was okay, you want to insert a surprise here. And it does that. And you see in the following thing out we have 012 and three there. If, however, you just want to add an element to the end of a list, you can do that using the append function. And append will just keep adding whatever you give it onto the end of the list in the last location. So, if we run this, you see you've got a list 123 and then go numbers dot append four numbers, dot append, six, and you see four and six are added to the list respectively. Equally, sometimes, you might want to remove elements from the from the list. And again, though, there are a couple of ways to do this. Okay. The first one is to use dot remove. Okay, so here you see we've got numbers, which is our variable of our list, there is a doctrine remove, okay, you then feed in the elements you wanted to remove from the sequence.

However, if we run this slide, you see that we have our original list here, which case numbers 123, and three, if I then use numbers dot remove, we get 123. Okay, that's because numbers, sorry, that's because the dot remove function will only remove the first element that matches the feet the effect in in the sequence, okay, so in this case, here, it's only removed the first three, it's encountered. Okay. So, you know, let's put another three here, for example. And now, you see that when it prints out, we still only got two threes left at the end, because Python still only removed the first occurrence of that elements. And so because of this issue, most of the time, when you remove a element from the list, people will delete it using indexing. Okay. And that's fairly straightforward. Right?

So here you can see what an example. Okay, we've got our list, which contains the numbers 1234. And we're down here we've got another function. You see, we've got del, which comes up in green because again, del is a reserve function words, which you can't use it as a variable name. And their works by obviously typing the word down the function del, and then you type your variable name, and then you feed in the index of the element that you want to remove. So here I'm saying remove element in location one, which is obviously, actually the second element because I finished the game zero index based. And if I run that, you can see down here, we have our own list of 1234. And then I save a movie element sequence two beside the movie element in location two. And it's done exactly that. It's removed thing, which Cosmos number two, which has now been deleted from our list? Okay. So that's lists. And again, looking at these sorts of examples here, obviously, containers, you can start to see why I say indexing is so important, and why I recommend you spend some time learning that

Forms. For loops, we'll be using quite a lot in a moment. So iterates over things, okay. And that's their main function is to iterate over a sequence, okay? So, when you create a record, a for loop essentially says, For all elements in a sequence, do something, okay? Whether that all elements in a list or elements in a string, whatever your sequences, okay? So as an example, we create a list of species, okay. And species, this contains dog, cat, shark falcan, deer, tyrannosaurus rex, because I couldn't think of any other animals to add in as an example. And I'm just going to print off this list, and it prints off just like we've seen before, not an issue within like, our code, okay, we then write our for loop. Now, the syntax of a for loop follows something like this. For I in species, do a thing. Okay? Now, the i here, is just a placeholder, okay? And it's worth noting, it could be anything, you could put tap the word hat, here, if you want, you could put the word right here, if you want, whatever, it could be anything. Often, when you look at Python code, you'll quite often see that I and j a quite often used by convention, there's no specific reason for that other than convention, much the same way as when you look at sort of mathematical equations, you see X and Y use the variable names a lot, much the same principle. So, a for loop as a syntax for I in species print I. Okay, so the thing we're doing is printing the elements on the screen. It's worth noting our for loop syntax, that indent here is very important. If you don't have that indent, Python will give you an error message. Okay, so that whitespace is very important. And if we were to run this bit of code, you'll see that Python iterates through each of the sequences.

Okay, so the first time it goes through this loop, because for i species, okay, and it says it says to itself, okay, this is the first time I've gone through it. So, I'm dealing with the first element, which is dog, so I'm going to print dog to the screen, second time it goes through this, okay, this is the second time I'm going through it. So, we're looking at the second element, which is cat, okay, and I have to print cat to the screen, and so on and so forth. Fairly straightforward.

You can also use it in this example here, which I won't bother talking you through and you can talk me through this in your own time, if you want. And you can see here we've got a list of numbers and this for loop basically just going through and as each of these numbers together until we get the total rate statements can be used in group to play for loops. We don't really have time to get into them here but look at the other materials I mentioned before we were talking about them in quite some depth while loops

While loops are a lot like for loops, but also going thing themselves. So ,a while loop, whereas a for loops says for all elements in a sequence, do a thing, while loop say while this condition is true, do a thing. Okay? So as an example, here, we've got our list of species again. And we've got a new variable by which we set to zero okay. And we say using a comparison operator, we say well i is less than three, print out the elements in the species list at location I. So obviously, I here is an integer variable, and then add one to I. Okay, so Okay, what's going on here? Well, first, the first time we go through the while loop is equal to zero. That's what we told it to be. So, python says Okay, so I is equal to 0 0 is less than three. So I'm going to go to the species list, and I'm going to print out the species application zero. And then I'm gonna add one to i, then it goes through again, it goes, Okay, so now i equals one. So, I'm going to print out the animal in the species element at location, index location one, which is the second element, and which is kept it thins out to the screen, and then adds it to one, and so on and so forth, until they get until I equals three, at which point it comes down and goes, Okay, well, i is less than three and goes, Oh, no, wait, I is equal to three, they're the same. So, I'm not going to do anything. And that's why we get dog cat shark, and then it stops,okay, because our index is either was increased, it's 1012. And then it's equal three. So as printed out anything after that. Hopefully that makes sense.

You can also create what's known as an infinite loop. infinite loops are just essentially where it's a bit of code where your while loop never terminates, and it runs forever. Okay, this is an example of a while loop here. And if you work, I've commented this code out, because I don't want it to move on, because it will run forever, and I'd have to terminate the console. And the reason this runs forever is because we never actually, this bit line of code here is faulty, it never actually adds any numbers to it. So, it is always zero, which means this condition is never met, which means it just keeps printing zero to the screen forever.

For loops versus while loops, you use probably use for loops more than while loops throughout your coding career. a for loop is a natural choice, when you want to cycle through a sequence, whether that be a string with numerous characters in it, a list, whatever. Whereas the world is a natural choice, if you want to keep doing an operation while a certain condition is true.

Okay, that brings us to condition conditionals. And we've seen conditionals briefly already, right? Those those if statements we looked at a minute ago, thelifose were four of conditionals. So there are three major conditionals that you can use, there's an if else and elif. And additional essentially says that, while this condition is true, do a thing. So, here's an example. Okay, we have a variable called work nights. And we set this to be equals to true to what we call Boolean variable. Okay, so, you know, true false, whatever. So we say we're not equals true. And then we say, if work night equals true, with two equals signs, because it's is equals to print, no beer else. Okay, I think you may have been so what essentially says is, if work nights is true, print beer, if work night equals anything else, print, you may have beer, if we run that, we find out that it's work night, so I can't have beer.

Okay, If however I set work night to false, then you see that we get a much happier result. And we can also use if, if you want to have a different sequence of true false statements. And there's an example of that here that you can play around with up to the session. But that brings us to functions. Functions are bits, really useful bits of code to create if you're going to have to do a task multiple times. Okay. And the reason is useful is because then you can crop the code out that chunk of code at once. And in the

future, you can just call it and refer to it you'd have to keep typing out individually. So for example, here we have our function Okay.

I've created a function called practice function. Functions are defined by first type in the word def for definition, and then a space event a practice them, so the name of your function, which is called practice and score function. parentheses, and then you put some placeholders isn't variables are never placeholders you put here should be equal to the number of variables. You think you know you're gonna have to feed into the function. And so I know, in this function I want to feed into variables. So I've put a and b there. There's a colon, and then this important. And then in this function, I just conduct the operation, okay? And I say right here, I want to create a new variable called answer, which is equal to a times b. Okay? We then end the function with a return answer. Okay, so what this does is create the function there.

And the third one down, I've created two variables x, which equals five and y equals four. Okay, down here, I've got another new variable, which is called calculated. And I say what this variable calculated is going to be equal to whatever output I get from this practice function when I feed in x and y. So apart from London, it takes x plus x, y, and it goes up to the function here and it goes, Well, x is the first sequence that in here, so now x is equal, is assigned to a, y is the second argument fed in here. So, y is not legal to b okay. And down here says okay, was a multiplied by b. In other words, what is x multiplied by? What is five multiplied by four? A complex the answer to that have no returns the answer? And B, this answer is then assigned to calculate it, we then ask you to click calculated, and we get the answer of 20. Okay. And this is useful, because it means later on, if you needed to do if you need to do this function repeatedly, you know, even setting up a function to say you've got other variables. This list I use l it's a lot like one on my screen. And I'll just say, can't type today. And if i just feed this in these variables in like that, when I get to print out perfect two, you'll see it gives us 12, this is fed in, h which is equal to 16, which is equal to and it's multiplied six by two, which has given us 12. Okay, so they're really, functions are really useful. Like I said, if you want to do an operation multiple times, and you'll see that when we carry on in a moment.

You can also have multiple returns. So, you can do multiple functions that return multiple things. From a function like that, okay. Okay, word on import, we covered import a lot already. This first was talking about the star were python as this sort of batteries included philosophy, right, where we say there's a module for every kind everything you need. Just as an example of that, I've imported a different module called date time. And I've created a variable says current time equals to the date time function. So call the date time function, form, the date, time package, and format, date, time function call the now sub function. All that does is basically print the current date and time.

As far as my computer is concerned, pandas, I'm just going to briefly mention pandas now because we're going to use it a lot in today's session. So pandas is another module you can import. And lastly, pandas is designed to be one of these packages as caters more towards people who use Python for data analysis. And as such, it provides a lot of useful tools that are useful to data analysis takes on data analysts such as easy to use data structures, and various modification tools and stuff like that. There are two main data structures and patterns that are useful series and data frames. Okay, we're going to briefly look at those just to end this first video. So, first things first, we're going to import pandas. You

can see I've put import pandas as PD, okay. You don't need to do that. But what this means is that later on whenever we call, whenever we call the pandas function, if I just imported it like this, whenever I need one, it's cool. So I'm from pandas, I'd have to type out the full pandas name like that. But because other computer scientists and as I said, we're lazy, which say, import pandas as PD, so later on, we can just type PD and pack and Python knows referring to this panda's package. So we do. Okay. So, like I said pandas says, too many days structures, a series and data frames. First is a series. And you can think of a series, a series is essentially a list, but with an inbuilt index. Okay, I'm not going to spend much time here looking at these, because the additional resources I mentioned before I present a lot of talk, it was pandas, the second data structure, which is more prevalent, and what we're actually going to use today, particularly in part three of today's workshop is a data frame. And you can just think of a data frame as a data table.

Okay, so any of you have used Excel or anything like that before, I think a frame is essentially the same sort of thing, right? In that you have your variables and your columns going down, and your observations in rows going across. Okay. And pandas has functions to read in data from a large number of different file formats. So you can read data from CSV files, from JSON files, which we're going to do in a moment, in part two and three of this from SQL files, and infant from XML files if you want. Python can sorry, pandas can also write out your data frames to various formats as well, such as CSV, so you can save them as a separate file. Okay. Just to end this first part of the video, it's worth spending some time looking at packages and so on to particularly NumPy and pandas, they have a lot of tools that will save you a lot of time. Okay, so that concludes the first video of this sort of free video workshop. Now, like I said, Of start, this has been a very, very speedy whistle stop tour of python. And I apologise I would have to go through so fast through it. But I want to get to the actual subject matter today.

Like I said, if you are interested in learning more about Python, which you will need to do if you want to do more of this sort of advanced textual analysis that are mentioned later on, check out this resources page here. Like I said, it contains a lot of very useful resources, such as so more in depth, Introduction to Python and Jupyter notebooks. So workshop materials, if you are social scientists use Python. How to use GIS and python into graphical information, so on. Okay, so check out that resource. And likewise, if you have any issues, you can contact me by email. My email address is l.brace@exeter.ac.uk Yeah, please drop me an email, always happy to help. Okay, that concludes part one. In part two now, which is video after this, we'll start looking at how to use what we've learned here, Python to actually build a web scraper so we can actually scrape the website in Python. Okay, I'll see you in part two in a moment.