Business Analytics And Text Mining Modeling Using python
Prof. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology Roorkee

Lecture-22
Numerical Python - Part IV

Welcome to the course business analytics and text mining modeling using python. So in last few lectures we have been focusing on the numerical python package NumPy package in the python environment . Now before I start you know further discussion in NumPy I would like to you know touch upon few aspects about this Jupyter network interface and here for example till now we have been using scripts.

And you must have seen the outputs you know from the previous one. So you might have figured out on your own how to get rid of these outputs. However for once I will also you know demonstrate how to perform this. So you can see a number of lines of code written there in 1 4 5 in 1 4 6 in 1 4 7 and associated outputs also you can see in this you know script and what we can do is we can go to the cell tab.

(Video Starts: 01:29)

And you can see all output at the end of this you know cell tab you know menu and you can see at the end you can see clear .So in one go we would be able to clear all the output. So we will have to run the whole script again if we happen to be using some of the you know objects which have been defined at the early part of the script. So let me clear this for once. So you can see I have just pressed clear.

And what will happen is that all the numbers in these you know indexing in and within the brackets the numbering is gone and the output is also gone. So this is just to in just in case you have not figured out this aspect of Jupyter network, then just in that case I wanted to you know demonstrate this. Now if you want to because you know we have been using we have been accessing some of the earlier defined objects in this script.

So we would refer to run many lines of code before we start our discussion on the next part. So

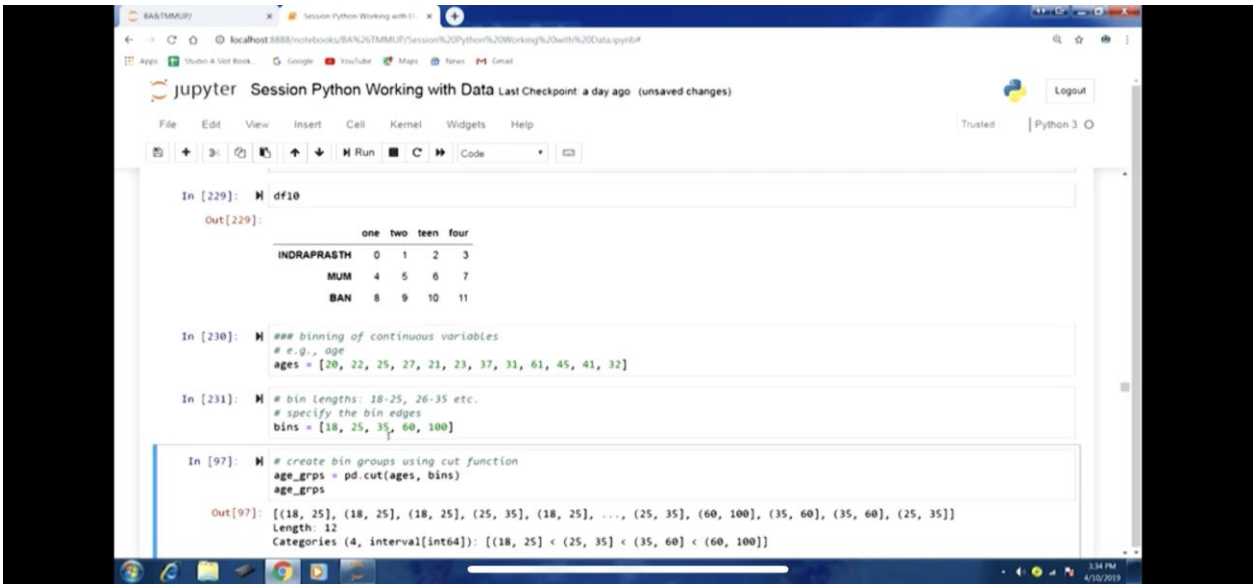
that we can do we can again go to the cell tab and you can see run all and run all above. So in this case suppose you know we want to start from the you know transpose method. So let us see if we want to start from this part. So you can see I have clicked at that particular line and now within the cell tab I will say run all above.

So all the previous lines of code they would be run. So let me click this and the previous lines would be run and use you would see that you know processing is going on you can notice the same here in python 3 here and this processing is going on. So let it run and then you would see the new output would be generated in this fashion you can always work with any python file in python script.

So if I scroll above you would start seeing some of the output there, in case you know for demonstration purpose I might have displayed you know for some particular line of code I might have discussed the error and other things. So the output will is stop at that point, you can see the output is stopped at this point 1 1 2 and you can see we ran into an error that was part of our discussion. So now we will have to you know process from this point again.

So you know again I will click here and I can go here and then I can run all below. So from this point I can start running this script. Let me go back again, however in this fashion whenever we have these kind of lines you know we will be regenerating the output in a sense anytime we can clear the existing output and we can go back and rerun those lines of code again. So let me go back to the point we want to start our discussion in this lecture, let me go back.

(Refer Slide Time: 03:00)



So in the previous lecture we stopped at this point fancy indexing .So this part we were able to cover. Now we will start our discussion on transposing an array. So just like in matrices you know here you know we can use a 2d array which is essentially very similar to a matrix structure and you know some of the matrix operations, some of the linear algebra operations that are performed you know using matrices, they can also be performed using 2d arrays.

So let us go through some of these examples. So first thing is transposing an array just like in mattresses. So we can you know transposing is essentially about swiping axes or dimensions. So dimensions for aspect we have already discussed you know when we started our discussion on array. Now let us take this example. So in this line 146 we are you know initializing this array 2 dimensional arrays using a range.

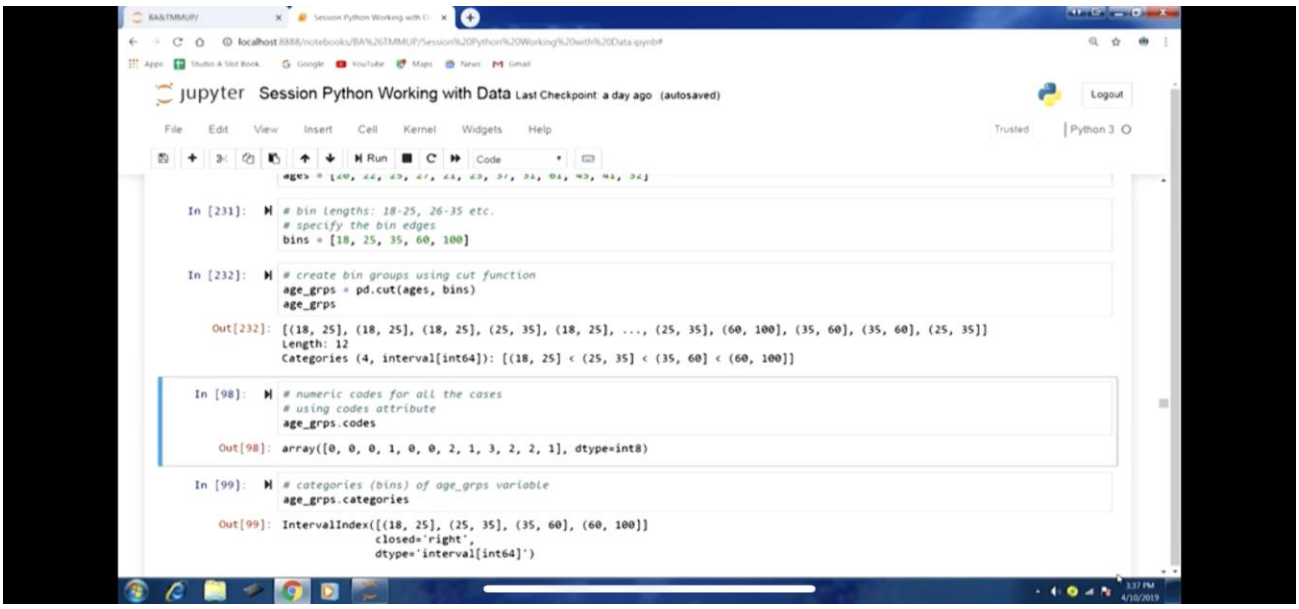
And we are also using the D shape function which will essentially give us the is where the argument that we are passing is essentially giving the details about the dimensions. So first dimension will have you know 3 elements, second dimension will have 5 elements and we are using a range function to initialize those elements. So let me run this and we can see array 2d 4 output 3 0 8 here you can see that this you know 3 cross 5 array has been produced here.

So you can see the structure of this 2d array or any 2d array for that matter is quite similar to

what we must have learned in matrices in linear algebra. So here we can perform a transpose operation. So for this we have this capital T attribute. So for any 2d array and array 2d 4 and .T in this fashion we would be able to transpose this array. So essentially it is about swapping the axis. So let me run this and you can see then in the output 309.

And you compare it with the output 308. So output 3 0 8 over 3 cross 5 and what was there what are the values that are there in the first row now have become first column in output 309 and the second row in 3 0 8 has become second column in 309. So this is effectively you know transposing a matrix here essentially you know when we talk about arrays we can express it like swapping axes. So from arrays perspective better words would be swapping axes or dimensions. So in that sense we can understand it from the area's perspective as well as from the matrices perspective. Now we also have a part from T attribute we also have a transpose you know method here which can be used. So again for array 2d 4 again will you know run this and you can see we will get the same output you can see output 3 0 9 and 3 1 0 are same

(Refer Slide Time:05:08)



So the transpose method can be useful you know useful in certain other situations as well. So for example if we are passing the axes or dimensions as arguments essentially when as I said the matrix terminology you know transpose has a certain different sense, when we talk about the

arrays it is slightly generalized sense that we are talking about, the generalizations talks about this you know swapping out axes.

Now transposing a you know matrix is a special kind of swapping and when we talk about arrays we can do many other types of swapping as well. So those other types of swapping will also involved you know different axes which might not be in a particular order you know as we do in transpose. So will need to you know pass those axes as arguments if we want to do any swapping.

So for that this transpose method can be really useful in those situations. So if axes or dimension are passed as arguments, then their order is used for you know permuting. So permuting essentially we are going to perform the swapping of you know values along those dimensions. So let us take this 2 values for 2d array to indicate order of axes or dimensions . So here you can see added 2d 4 dot transpose method and 0, 1.

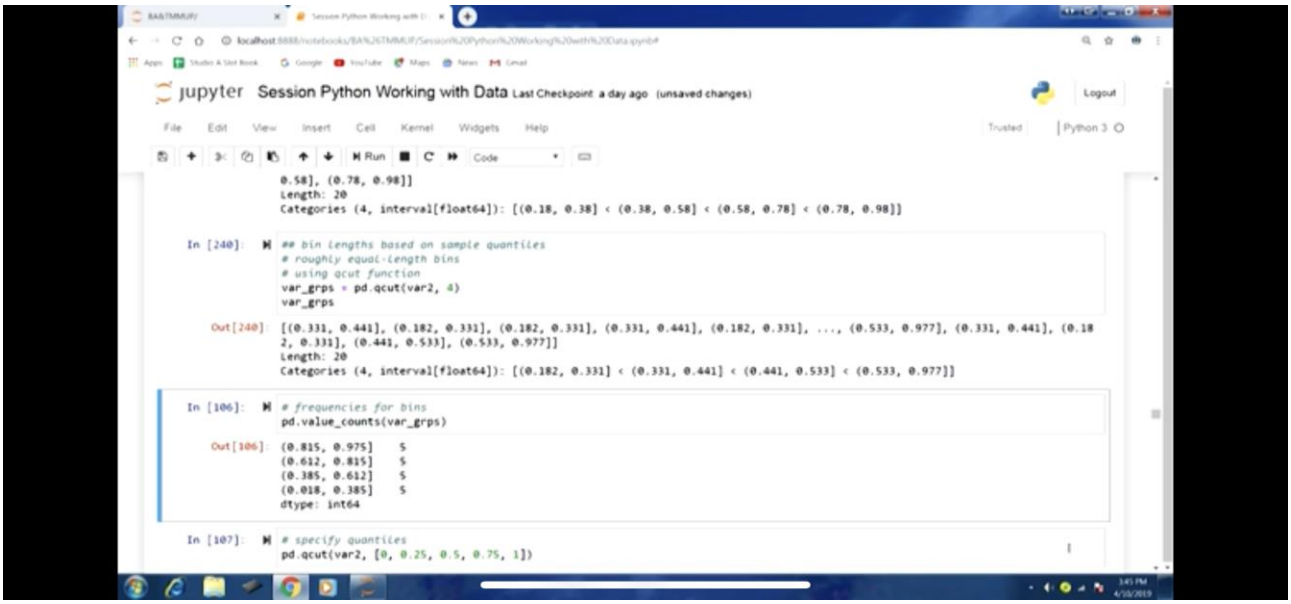
So here we are passing the arguments we are telling the method the axes which are to be permuted which are to be you know swept here. So you can see we have indicated order 0 and 1. So you know array 2d 4. So this is the order that we want to keep in the output. So 0 and 1 is the order of the this particular array 2d 4 itself. So therefore if we run this transpose method by passing 0, 1 in the output will get the same array.

So because it is actually you know the order that is has been passed for this permuting is the same as the array which we want to permute. So let me run this and you can see output 3 1 1 is a nothing but array 2d 4. Now if we want to simulate what we do in a you know a matrix transpose that we can do in the next line , that is where we are swiping second axis you know this rap is about second axis would come first and the first axis second.

So if you look at the arguments that we are passing here in the transpose method 1, 0. So the you know second axes now will be coming fast and the you know first axis will come later .So permutation will happen as you know as per this order and this is effectively the transpose that we typically do in matrix. So if I type this array 2d 4.transpose and 1,0 essentially it is similar to

simulation of matrix transpose for this 2d array.

(Refer Slide Time: 11:16)



So if I run this you can see the output 3 1 2 and the previous output that we had obtained using the you know using the transpose method without passing an arguments and also .T attribute capital T attribute , both you know all 3 outputs are same you can see output 3 1 0 and 3 1 2 and as we have seen that output 3 0 9 and 3 1 0 and 3 1 2. All these 3 outputs are same. So essentially nd arrays the concept of transpose that we are familiar with in the matrix contest it has been extended right.

So this is about you know the transpose method. Now let us move forward now we are also familiar with the inner matrix product you know matrix multiplication that we typically perform. Now that is you know we can use 2 dimensional array you know to simulate that however again here in the case of in the arrays context even this concept can be generalized. So for this we will use this method np.n this particular function np..function.

So let us take example of this array array 2d 5. So here we are using this random.random function 6, 3 is the dimension for you know this array. So let me initialize this. So this is the 2d array that we would like to use, now let us look at the shape attribute for this and you can see 6, 3. Now if I take a transpose using the capital T attribute and then take shape of that then you

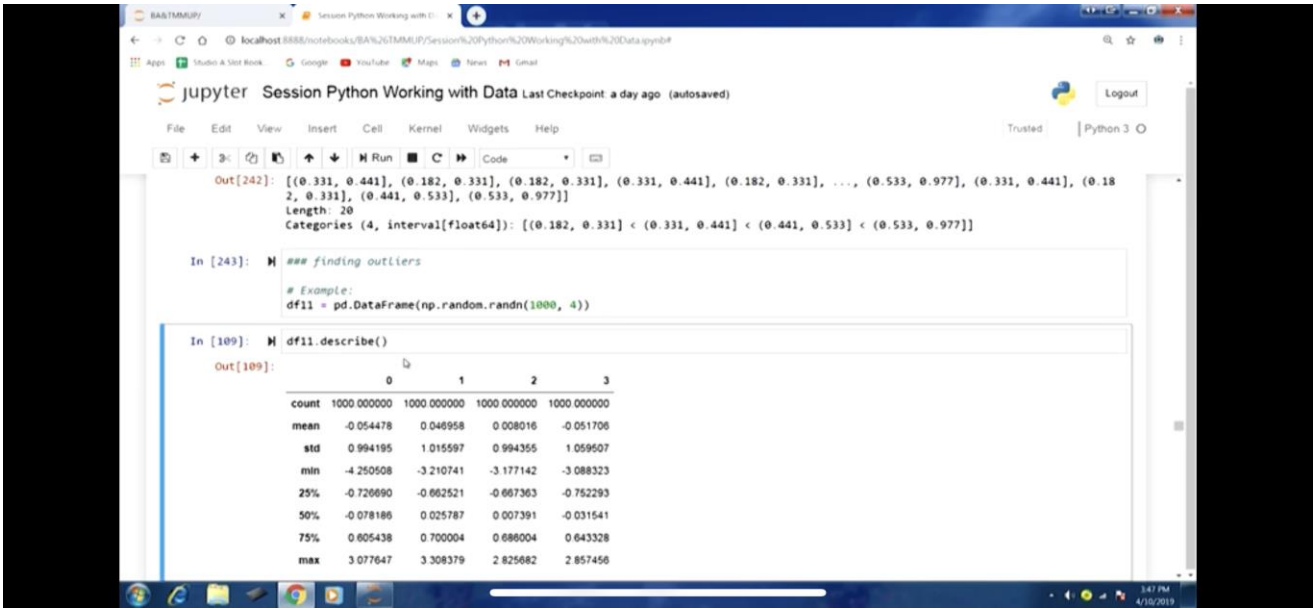
would see that will get 3,6.

So you can see typically when we do matrix multiplication we have to check whether the matrix multiplication the inner matrix product is feasible or not. So you can see in this case, if I multiply the matrix with its transpose then of course it is going to be feasible. So 6 cross 3 the first one and the second one 3 cross 6. So the other way around also it is feasible. So essentially in the next example what we are going to do is we will multiply 3 cross 6 matrix and 6 cross 3 matrix which are you know transpose matrix of the original matrix.

So result would be a 3 cross 3 matrix as we understand from the you know matrix algebra. So if I call this function np.. and within the parentheses I will have to pass these you know these arrays as arguments array 2d 5.t that is the transpose of this and then array 2d 5 the original one and the this matrix multiplication can be performed and you can see the same in the output 3 1 6.

Now for nd arrays we can really extend this concept this kind of you know multiplication concept that we are familiar in the case of matrix .So this can be extended. So let us take 3 dimensional array. So here you can see I am going to create an array 3d 1 and we are using a range function to initialize the values and we are using the reshape here to you know fix the dimension 2, 2, 4

(Refer Slide Time: 13:03)



So this is a 3d array. So first dimension will have 2 element the second dimension will have 2 elements and the third dimension will have 4 element. So if you multiply these 2*2*4 then you would see that we require 16 values and that is why in the a range function we have passed 16 as the argument. Now this if I run this I will get a 3d array you can recognize it looking at 3 brackets as well and now I can create the T attribute here to actually you know obtain a transpose of it.

So array 3d 1.T, if I run this and you will see the output 3 1 8. So this is the you know transpose of this array 3d 1 and you know because we have used you know T. So the typical the special you know order that is used in a matrix transpose that is going to be used here, the same you can recognize in the output itself right. So if we compare 3s among 7 and 3 1 8 here you can see 0 1 2 3 4 there and you can see that you know the shape of first thing is the shape of this transpose will reverse.

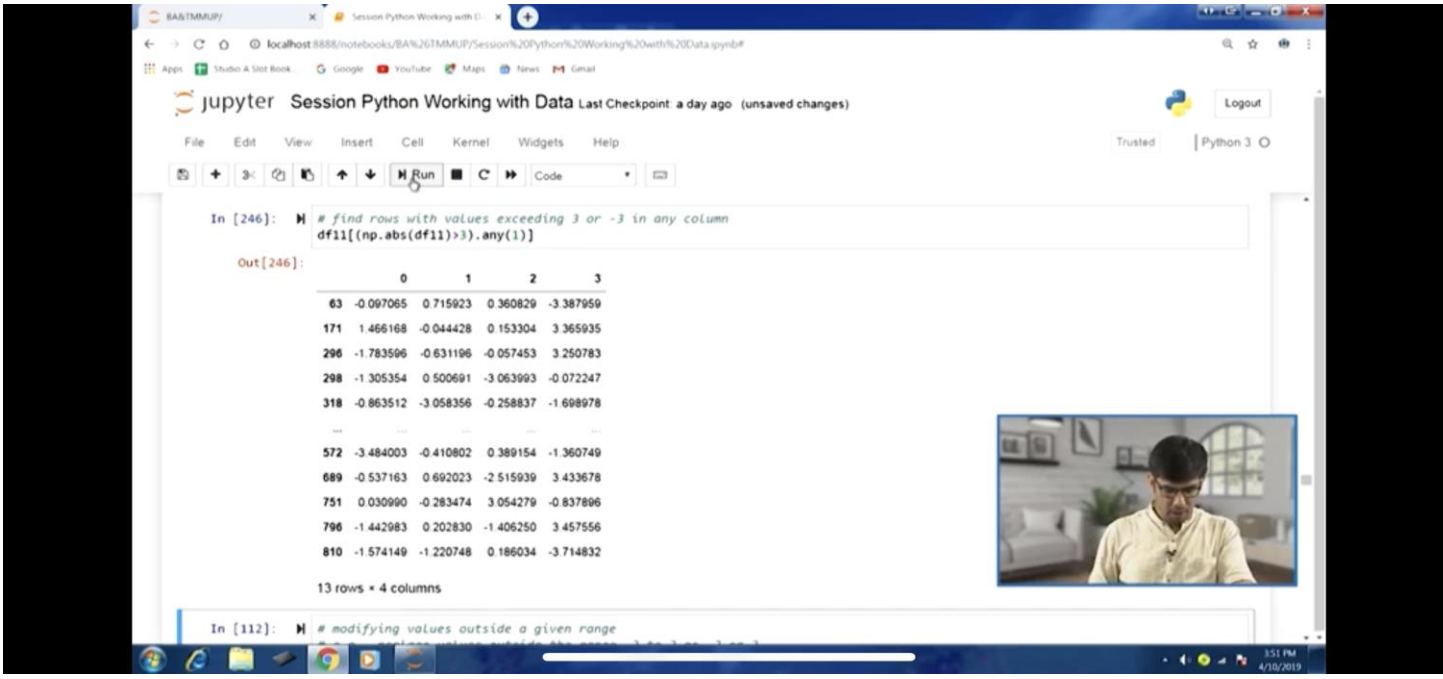
So you can see the next line array 3d 1.T.shape, if I run this you can see 4, 2, 2. So the first thing that you should notice that the shape would be something like this, the transpose output 3 1 8 you can see now the first dimension will have 4 elements and then second dimension 2 and then third dimension 2 elements. So you can see in the array 3 1 array the first 1 0 then 1 then 2, then 3 and then you can see that you know 4 5 6 7 have been you know they are in the first dimension in separate list.

And then you can see that 8 9 10 11 which were in the separate dimension there and that they have been clubbed with 0 you can see. So you can see how 2 cross 2 cross 4 has been converted into 4 cross 2 cross 2. So just like in 2d it is the extension of that 2d version of transpose in 3d. So very you know just comparing the output you would be able to understand what is happening here, now the kind of permutation that is happening.

So the last text is being permeated with the first texts in that sense, this particular transpose is being performed. Now let us take a few more examples for any 3d array we would require 3 values to indicate any order of axes or dimensions for this kind of you know transpose that we

would like to perform on a 3d array. So let us say see for any given 3d array the default order is going to be the original order is going to be 0 1 2.

(Refer Slide Time: 15:24)



That means first axis you know indicating 0 first comes first then the second axis indicated by 1 comes second, the third axis indicated by value 2 comes third as you can see in the comment section original. Now the desired is the one where we would like to make certain you know changes .So you can see the desired is 1 that means second axis is coming first then 0, the first axis is coming second and the third axis you know it is and position no change there.

So the swap that we want to perform is that second axis first, then first axis second and the last axis unchanged. So for this we will have typed like this array 3d 1.transpose and whatever is the desired order that we want to perform we can indicate it like this in a tuple. So you can see if I run this you can see the output has been accordingly produced here and you can compare it with the original one and you can see how the changes have happened.

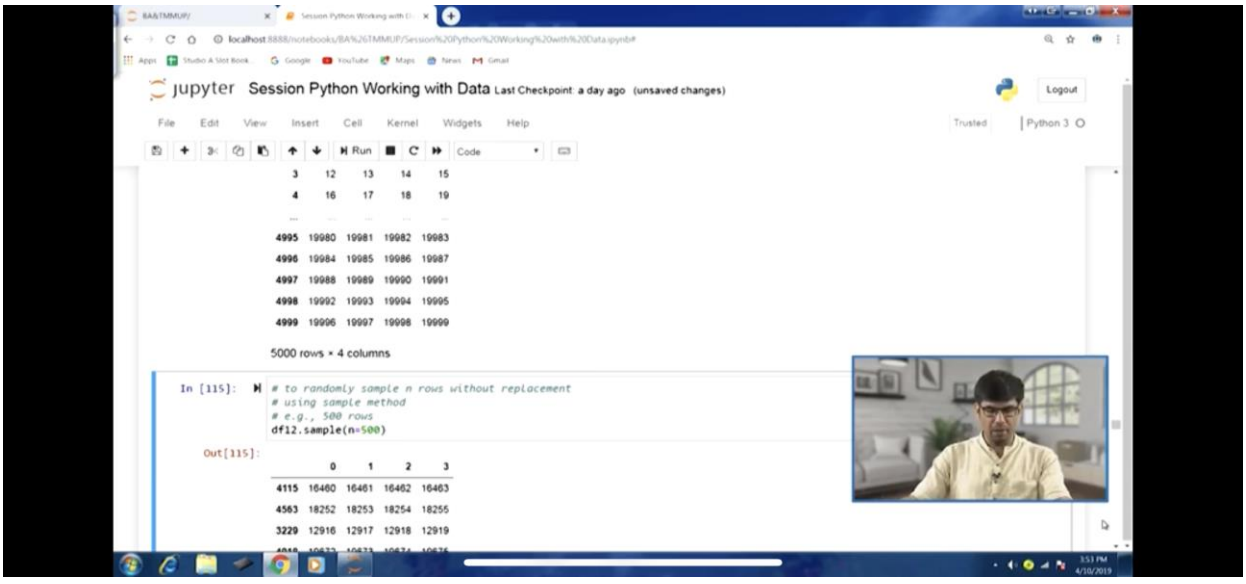
Now any transpose method you would see that if you do not pass the argument the default order would be you know executed or if you specify then that particular order is used for swapping the axis all the axis at once. Now sometimes you might want to just you know swap a pair of axis

and not go be the all the axis and you can still swap a pair of axis using transpose method but that would require you know indicating you know manipulating the transport method in a particular way.

However it can be run easily using another method that is swap axis method. So here essentially we would be switching any 2 axis or dimensions often nd array. For an example let us take this array 3d 1 and we will switch second and third axis second and third dimension. So this is array 3d 1 and shape of this array 3d1 is this 2, 2, 4. Now we can use swap axis method here and in the arguments you can see I am indicating the 2 axis that you know I would like to switch 1 and 2.

1 indicating you know second and dimension and 2 indicating third dimension. So essentially we would like to switch here. So if I run this you can see the output and you can compare the output number 3 2 1 and 3 2 3 and you can see how the switching has happened here. Now if you want to check this you know shape of you know this the output again we can use this shape attribute and you can see the original one was 2, 2, 4.

(Refer Slide Time: 18:57)



Now the you know after the switching it is 2 comma 4 comma 2 say can see the clear switching has happened. So looking at the number of elements in the you know after switching and how they change we can also understand you know which axis or dimensions have been switched or swept. Now let us focus on another aspect of array nd arrays that is element wise operations to

remember that when we started about discussion on NumPy we focused on the aspect that you know we can get rid of the those loop construct which are typically you know used to do certain kind of processing.

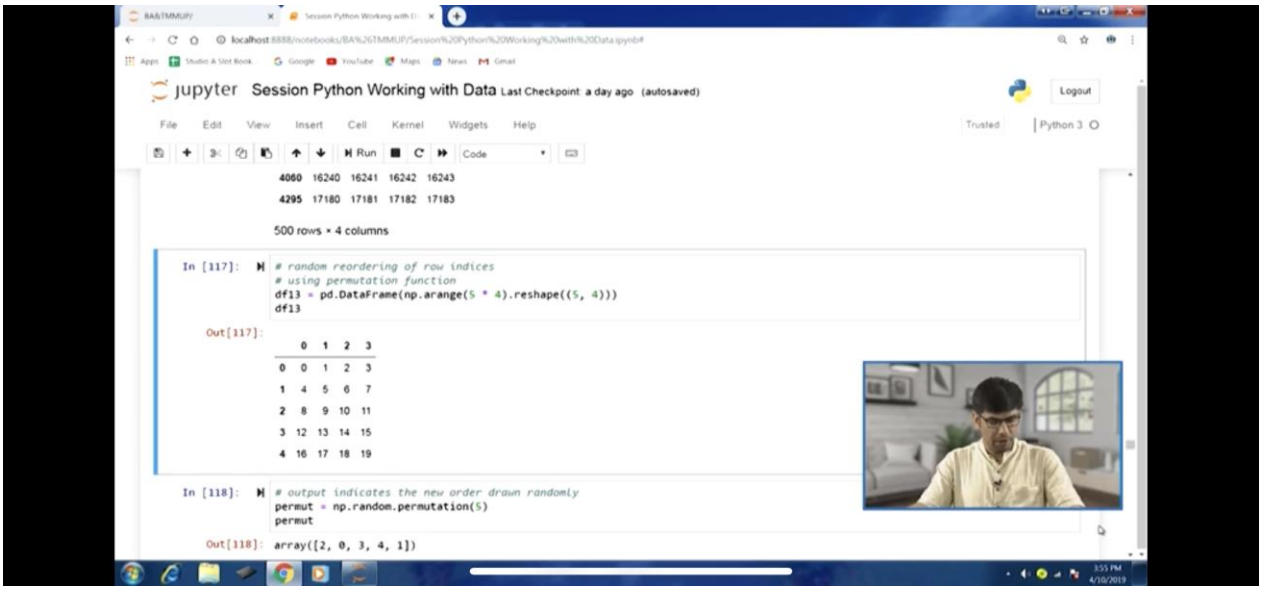
And element wise you know operations are facilitated you know in the using nd arrays. So we will focus on some of those aspects here. So let us take this array 9 and initialize this using a range function you can see the output 3 2 5 and these are the 10 values. Now if we want to take a square root of each number here. So this we can also achieve using the you know python basics in built-in capabilities.

And we can run a loop and perform this process for each of the element, however because we are using this data structure this arrays. And that can actually be used to perform this in one go and rather you know faster and better with better memory utilization. So for this we will have to use this np. sqrt function here and we will pass on this array 9. So np.sqrt array 9, and I run this. So we will get an array as an output and each of the element of the you know are a 9.

And that square root has been produced and generated in the form of an output array as you can see in the output number 3 2 6. So in this fashion just one line of code and we have achieved what we would have otherwise done using loops. Let us take another one more example exponential of each number. So lets you know if we have a sequence of values and we would like to take exponential for each number again arrays and this NumPy function could be really useful.

So np.exp to compute the exponential for this array 9 and each of the elements that are part of this array 9. So if I run this again and in the output you can see that exponential of each of the you know element has been taken and produced in the output. Now let us initialize another array np.random we are using np.random.random as you might have seen I have been using few functions without much discussion on them.

(Refer Slide Time: 21:32)



Because later on in the coming lectures we would be focusing more on those aspects of those kind of functions in more detail. So right now I am using this to initialize this array if I run this I will get array 10 like this and let me take another array 11 in a similar fashion using this function random function and now what we want to do is we want to find the maximum of 2 numbers you know.

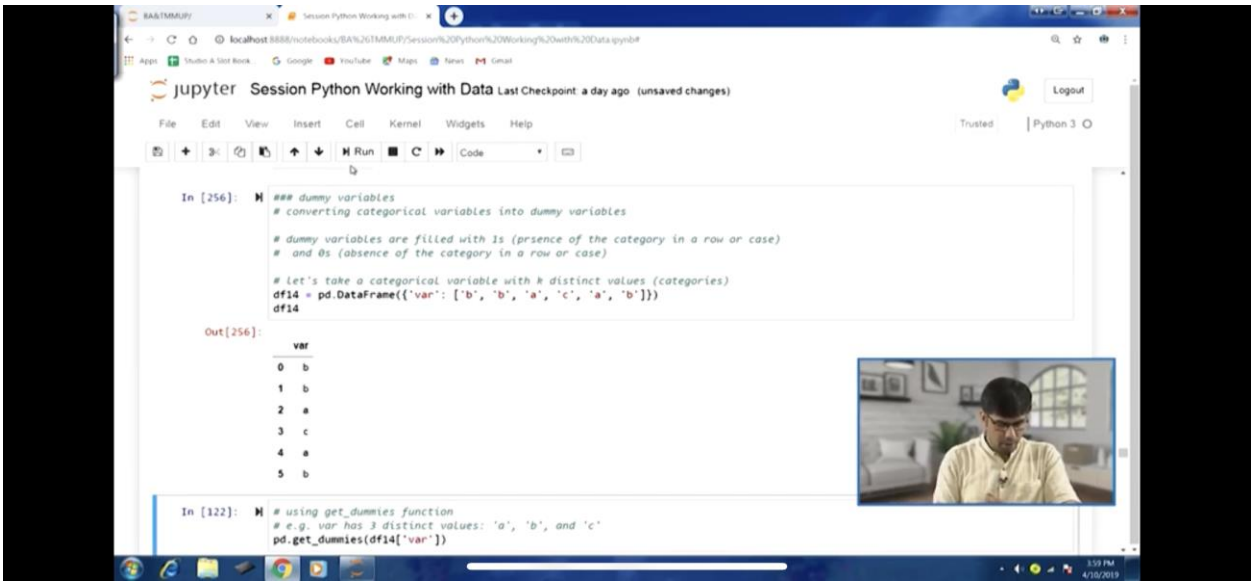
So you would like to do it. So essentially when we started this part of the discussion we said element wise operations. Now given these 2 arrays array 10 and array 11 we would like to compare you know first element in array 10 with the first element in the array 11. Similarly second element in the array 10 and second element in the array 11. So essentially an element-wise operations where we are comparing these 2 you know sequencers.

And each of the elements in these 2 sequences that can be done in one go. So we can have something like this np. so maximum of 2 numbers we can learn this np.maximum function. And we can pass on these 2 arrays 10 and array 11 and if I run this and you can see the output. So there you would see if you compare this with the output 328 and 329 and output 330 you can see in the output 330 whichever is the higher among 328 and 329 that has been produced **that is** that is part of output 330 as you can clearly see from here.

Now let us move to the next part. Now we might have an array which might be having you know floating-point numbers. So if we want to segregate the you know fractional part an integral part that can also be performed here. So that would also be quite similar to element wise operations that we typically do. So let us take this function np.modf and we pass on this argument array 10 and in the left hand side you can see we are segregating the fractional point will be stored in decimal object.

And the integral part will be stored in the whole object. So you can see decimal, whole and then equal to np.modf and array 10. So if I run this now you can see let us access the decimal part and you can see the decimal part we have got here and similarly the whole part and you can see in the output 3 3 3, the whole part has been segregated. So in this fashion this is another example of element wise operations using arrays where easily we have been able to do this you know if we do not use the NumPy.

(Refer Slide Time: 23:37)



And these you know and arrays then this would have to be performed using loops and slightly more lines of code .So you can see it is like you know more easier to work with data using this you know python modules python packages. Now will talk about an out argument to facilitate in place like output. So sometimes we would like to you know send certain scenarios we would like to have this kind of functionality.

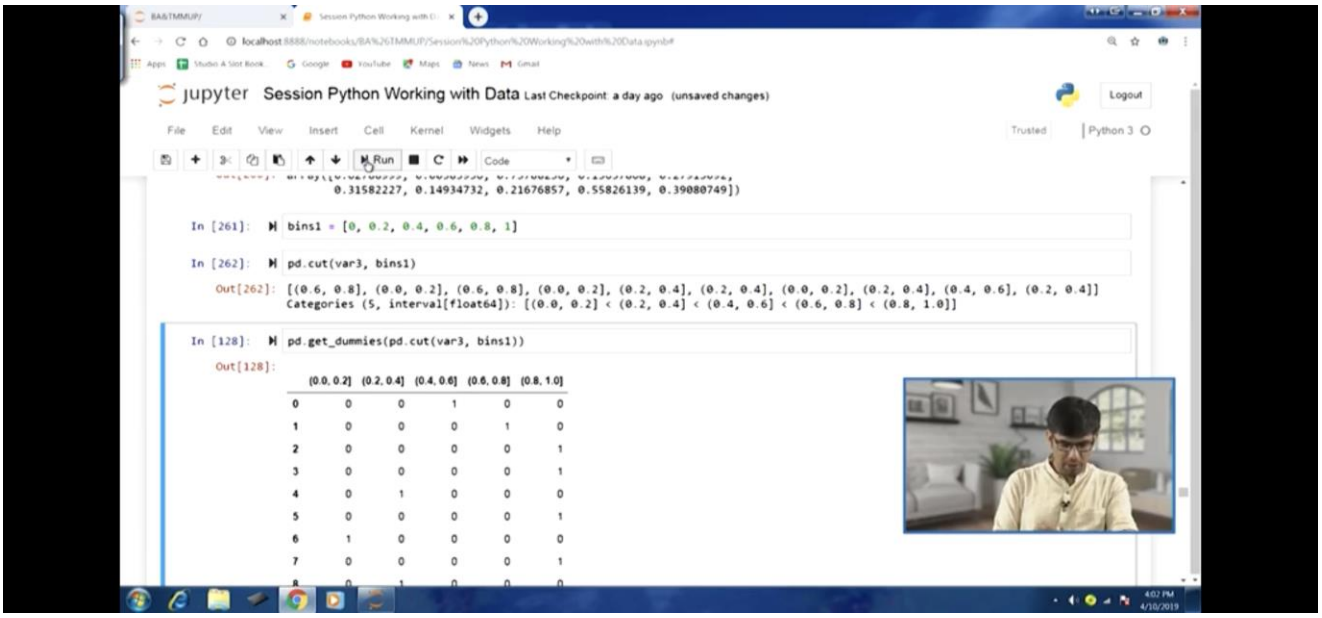
So before we start let us take this you know example np.sqrt array 9 and this would be the output. Now you know let me initialize another array array 12 with using the empty function and we are going to use the shape of array 9 because we would like to use later on this as and you know another arrays similar to array 9 to record the output. So let me initialize this find this and now you can see in the we are again using np.sqrt function.

So the line number 3 3 4 we directly got the output, now we could have recorded this output in another array .So that is one approach ah, now the second thing is that we can use the np.sqrt function itself where we have an out you know argument where we can indicate the array where we would like to you know you know store this output of np.sqrt. So in that sense it is quite similar to what we do when we use methods and we get in place output, now that means the original object is modified to produce the output.

So similar kind of thing with a different mechanism. So now we are going to call np.sqrt array 9, array 12 where array 12 which we have just defined that is going to be used to store the output of np. np.sqrt array 9. So if I run this I will get this output and you can check array 12 let me go back and you can you know compare the output number 3 3 7 with output number 3 3 4 and you can see it is the same output here.

In the same function we have you know again used another we have defined another object to record the output quite similar to in place thing. Now the next aspect is about the another another feature of arrays nd arrays which is vectorization,

(Refer Slide Time: 26:56)



(Video Ends: 29:19)

So now we will talk about the vectorized array operations. So for this we will talk about slightly more you know complex and mathematical operations. And how the nd arrays you know this data structure that we have in NumPy it can allow this element wise operation thing and vectorization and faster processing and faster education and better memory utilization for these kind of operations. So we will stop at this point and we will discuss these vectorized array operation in the next lecture, thank you.

Keywords: NumPy, Attributes, Vectorized operation, Text mining modelling, Matrix multiplication