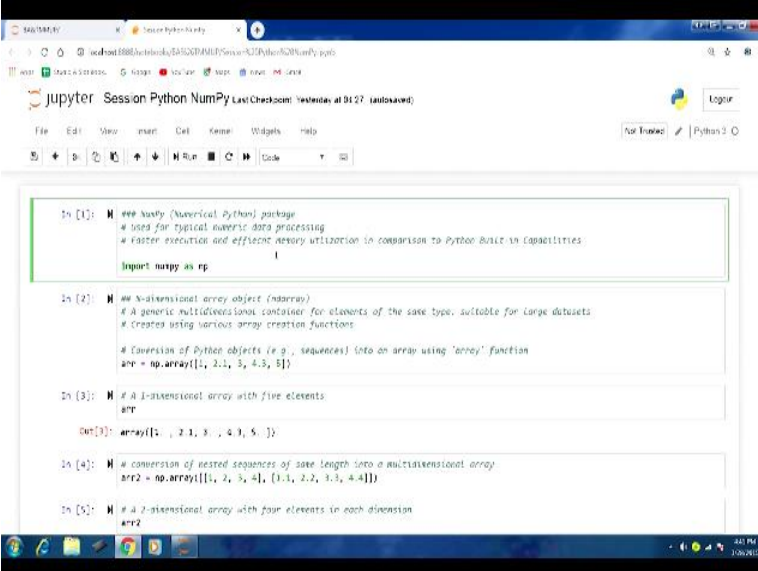


Business Analytics And Text Mining Modeling Using python
Prof. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology Roorkee

Lecture-20
Numerical Python -Part II

Welcome to the course business analytics and text mining modeling using python. So in previous few lectures we have started our discussion on python NumPy. And so let us quickly zero through what we discussed in the previous lecture and then we will pick up from where we stopped in the last one.

(Refer Slide Time: 00:45)



```
In [1]: # NumPy (Numerical Python) package
# Used for typical numeric data processing
# Faster execution and efficient memory utilization in comparison to Python built-in capabilities

import numpy as np

In [2]: # N-dimensional array object (ndarray)
# A generic multidimensional container for elements of the same type, suitable for large datasets
# Created using various array creation functions

# Conversion of Python objects (e.g., sequences) into an array using 'array' function
arr = np.array([1, 2, 3, 4, 5])

In [3]: # A 1-dimensional array with five elements
arr

Out[3]: array([ 1,  2,  3,  4,  5])

In [4]: # A conversion of nested sequences of same length into a multidimensional array
arr2 = np.array([[1, 2, 3, 4], [1, 1, 2, 2, 3, 3, 4, 4]])

In [5]: # A 2-dimensional array with four elements in each dimension
arr2
```

So, we talked about the numerical python package, the NumPy package and how it could be really useful for numeric data processing. And quite efficient in comparison to in terms of execution and in terms of memory utilization in comparison to python built in capabilities. So let us quickly zero through some of the code that we have discussed before.

(Video Starts: 01:04)

So first thing is coding this library module and we talked about the ndarray that is ndimensional array object and you know, we talked about you know, 1 dimensional array let us take for this is just an example array with 5 elements then how we can create multi dimensional arrays. So,

you can see that we can use a list of lists, and we can create array2, this is a 2 dimensional array here. And you know, in this fashion, we can access you know, elements similarly.

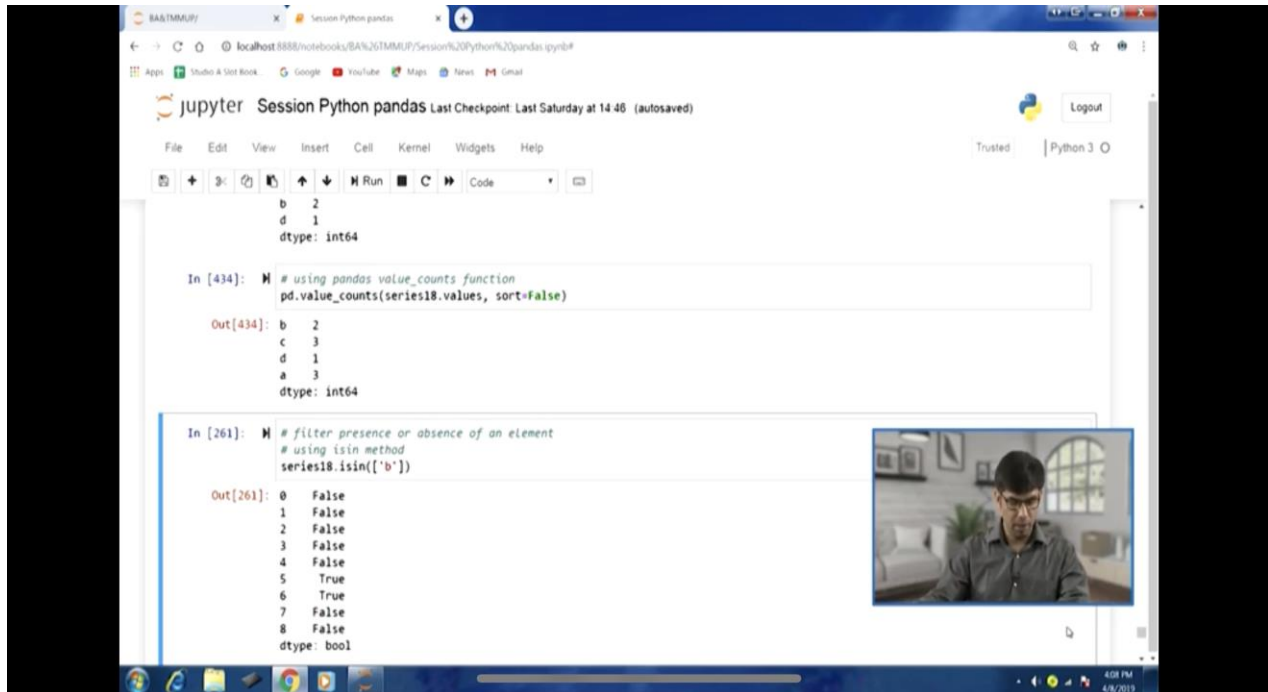
Another example here, here we are indicating the data type as well for this array using the dtype argument. So, all these aspects we have discussed before, let us move further. Now, we talked about the array properties in terms of number of dimensions and the shape of array, which also includes the you know, number of dimension and the elements that are there in those dimensions.

So, let us quickly zero through this number of dimensions for array is 1, number of dimension array 2 into and then the shape for array is 5 which is just 1 dimensional array, so just 1 dimensional array with 5 elements and for array2 here this 2, 4. So, this is indicating that you know, 2 dimensions, because there are 2 elements for this array. And first one having 2 elements, second one having 4 elements.

So, this is about array2. Now, we also talked about data type and you know, how we can find out that the datatype that is there for a particular array. So, in this fashion, we can find out for array2, array3 right. So, then we can use other array creation function this part also we have discussed zeros and how we can create arrays here. Now 3 dimensional array using zeros you can see here then this one you can see this is 4 dimensional array here.

So, using zeros function, we would be able to create this similarly, once function is there, which is very similar to zeros function. So, here also you can see 3 dimensional array one more 3 dimensional array with the datatype also specified. And then we talked about a range function which is also very similar to range function in this part we discussed. So, after that supported datatype, so some most of the data types that we have discussed.

(Refer Slide Time: 03:58)



So, far you know, floating point you know complex is also there integer, boolean. So, all these are supported. So, you can see that now smaller into bigger and converting one datatype to another that part is also you know, how we can use as type function to cast to change the type. So this part also we have discussed and to float and all those things flow to end and the data loss also that zeroes with it.

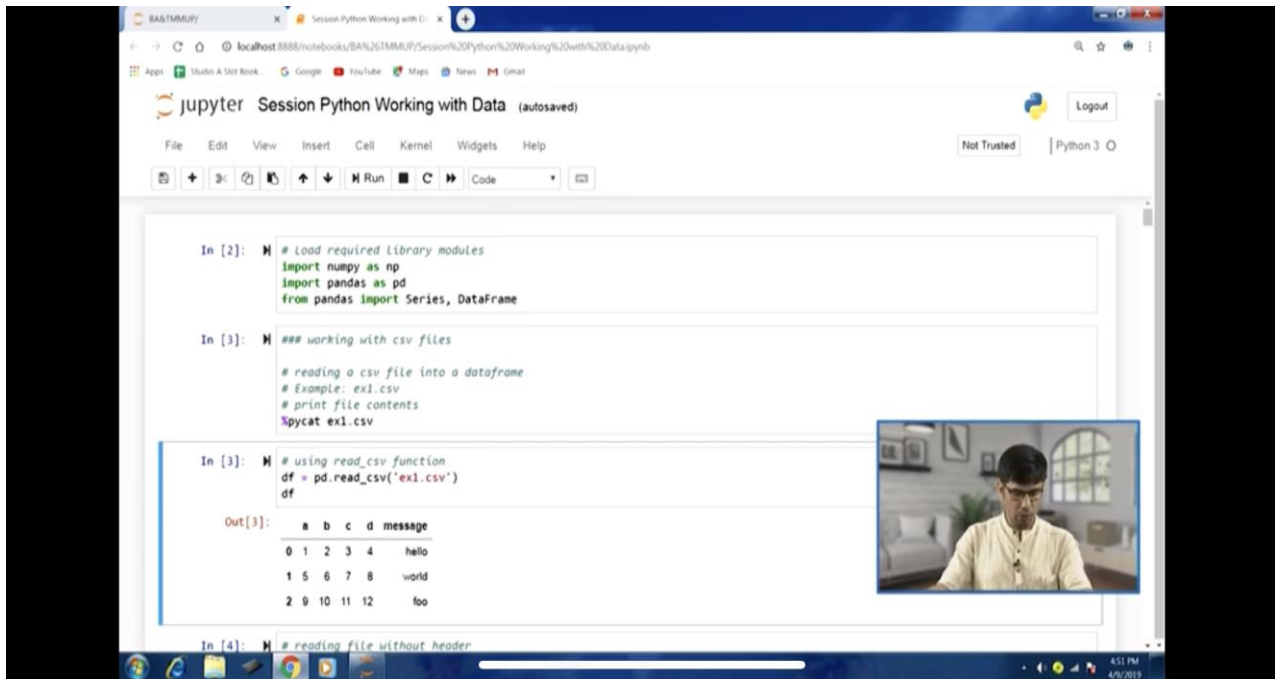
So, those aspects we have discussed talked about a string representing numbers and how we can change the in datatype and all that string to float as well. Similarly, we talked about this, the number of bits that are used to store an element and how that can create a problem in terms of when we change one type to another type using as type function. So, those aspects also be discussed in the previous lecture.

So, let us move forward. Now, we can also use in other arrays objects dtype so, that part also be discussed here. So, you can see here that we can use you can see in this example in this 43 that for array4 dtype is being used to you know change the cause the data of array3. So all this we have discussed. Now we stopped at this point in the previous lecture. So, let us pick up from here.

So, now there are some short hand type quotes also available in python environment, especially for this you know, numerical pattern, which can also be used to specify these types data types. So, let us take an example let us create a whole byte long string using empty function. So, this is empty functions is other function another array creation function. So, here we are trying to create this array8 and np.empty.

And then you know you can see the second argument we are specifying the U4 the shorthand you try that is this is for a string. This is for unique code of 4 bytes. So, if I run this and if I look at it, so the empty function would actually create an array with 9 elements and second U4 shorthand we had used to specify the data type there and you can see array in the output number 45 you can see that each of these numbers you can see.

(Refer Slide Time: 06:50)



These are randomly given and the datatype you can see U in 32. So, that is comes out to be 4 bytes. So, the same is indicated in U4. So U4 is essentially representing unit 32. So this is what we meant by shorthand type codes. So these shorthand types codes can sometimes be really

useful in that sense. Now another aspect that we would like to mention is that if you want to perform batch operations on data, then typically we would require loops right.

So however element wise operations can be performed using arrays. So what we can do, where we require loops to perform batch operations, the element wise kind of operations that can be very easily done and efficiently done using arrays, of course of the same size. So this is something that we refer to as vectorization, so numerical python, particular package.

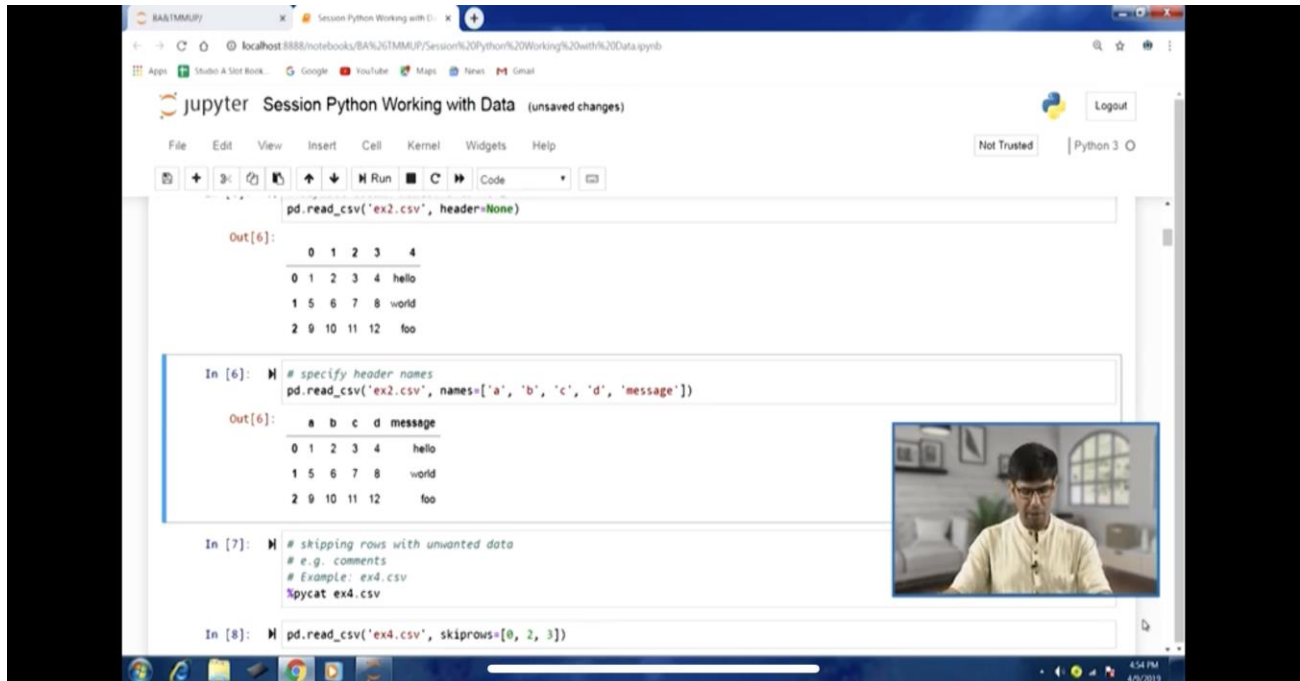
So these library models, they really facilitate this vectorization. So let us take few examples here. So we will talk about few arithmetic operations, and we will see how these arrays this particular data structure really facilitates this element wise operations. So, let us take array2. So, these are the values that are part of array2. Now we multiply this array2 with an array with itself $\text{array2} * \text{array2}$ then in one zero we would be able to get the output which is also an array here.

And now if we were to perform the same you know operation you know without using arrays, then we would require loops. So, probably that would not be that much efficient. Similarly, another example for subtraction $\text{array2} - \text{array2}$ and you can see that quickly it is performed similarly power operation here $\text{array2}^{0.5}$. So, essentially, these are the square root values here.

Comparison, if you want to perform again element wise comparison will happen and the output of a 50 you can see for each element whether you know those values were equal or not, we have zeroth a you know, boolean value return there. So, because we are comparing the same array, then therefore all the, values are true. So, you can see these kind of operation. So can easily perform and this is what we refer as vectorization.

And let us take division. Similarly, if we want to perform, you know, these conditional you know statements, zero conditional statement, you can see, $\text{array2} > 1 / \text{array2}$. So we will get you know boolean array here as an output. So, if I run this, you can see output number 52.

(Refer Slide Time: 09:08)



Now, let us talk about the another aspect of arrays which is accessing array elements. So, this is also will, you know, referred as indexing, so, how we index you know, this particular data and structure arrays. So therefore, the same indexing mechanism is to be used to accessing those array elements. So this is quite similar to what we have learned in python list. So let us take a few examples. So let us take array2 that we had created.

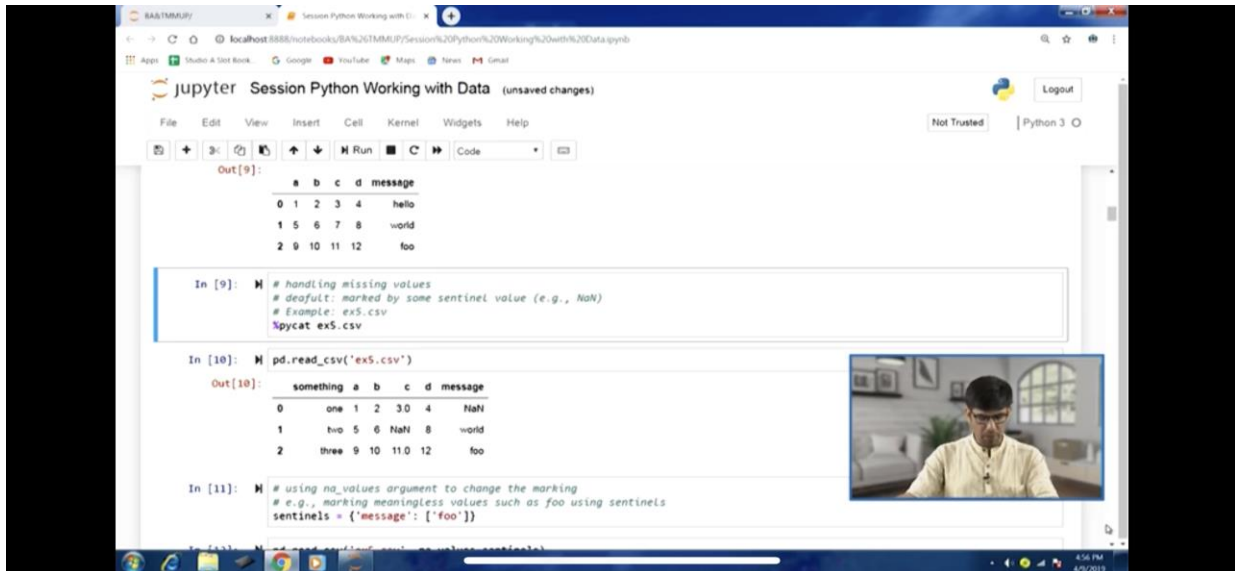
Now, we want to access the you know, array2 to 1 within the brackets if you just specify 1. And let us run this, let us look at the output, output number 54. If you see, the output is also an array, and you can see, if you compare the output number 53 and 54, you can see the you know first you know we had a list of list kind of array there and the first list has been written there.

So, if we specify array to 1, it is actually the you know first dimension that is being returned over here. So, this is how so, first dimension elements happened to be you know this array element with 1 array element there. So, that has been written here, and you can see the 1 is

because all these are zero index, we can see 1 is actually zero is the first list element and the 1 is second list element.

So, you can see that in the output number 54 we have got the second element there. So, that is how we can get this. Now, let us talk about the comma separated list of indices. So, this works like a coordinate system and you know, for a 2d array we can access an element using a coordinate kind of system. So, first index like a coordinate along the first dimension and second is like a coordinate longest second dimension.

(Refer Slide Time:13:08)



So, we use the same array here array2 and we specify the you know our list of indices 1,1. So, one will indicate the second dimension and you know, one will indicate the, you know second element in the first dimension and comma, comma 1 the second one value will indicate the second element in the second dimension. So, what will happen is if we run this you can see we got the value 2.2.

If you look at the output number of you know 53. So, second dimension is the you know, this list 1 comprising of 1.1 2.2 3.3 4.4 and the second element in this list has been returned. In this fashion just like a coordinate system and we would get indices to indicate you know a particular

index along those dimensions and we will be able to access the elements. Now and let us take another example array2 zero1.

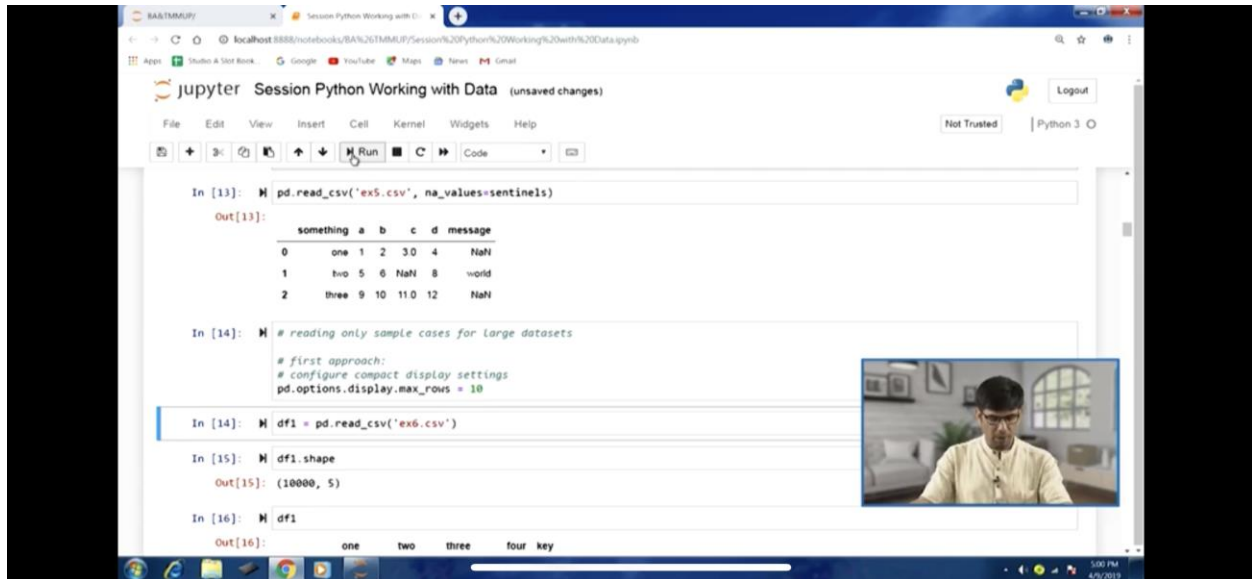
So, you know, so, first dimension first element second dimension second element, so, that comes out to be 2. So, you can see another example, first dimension first element zero means first element and it is coming first. So, first dimension, then next element is for the second dimension zero means the first element. So, this is very first element of the array that we had generated 2D array that we had.

So, you can see the value is 1. So, similarly, you know array3 if we take array3 here, so, if we take the array example and we just access the third know element here, this is we have just 1 dimension This is a 1D de array array arr is 1D array as you can see in the output 58. so, this is 1d array with 5 elements and when we specify within the brackets, just 1 value 3, this is for the first dimension which is the only dimension.

And the third value over there, third index over there that means 4th value 4th element in the array. So, that comes out to be 4.3. So, in this fashion you can understand how on these values are being accessed. Now, let us talk about the another aspect about arrays, which is slicing. So slicing something that we have discussed in list and other data structures as well.

So let us talk about slicing you know on array. So, is this slightly different from python built in list. So, modification when we perform modification in the slides array, they will reflect in the source array. So these slices are going to be views and not copies. So let us take an example, let us take arr this array, and this is the 1d array here. Now, you can see remember you can recall the syntax for to perform the slicing.

(Refer Slide Time: 18:54)



```
In [13]: pd.read_csv('ex5.csv', na_values=sentinals)
Out[13]:
something a b c d message
0 one 1 2 3 0 4 NaN
1 two 5 6 NaN 8 world
2 three 9 10 11 0 12 NaN

In [14]: # reading only sample cases for large datasets
# first approach:
# configure compact display settings
pd.options.display.max_rows = 10

In [14]: df1 = pd.read_csv('ex6.csv')
In [15]: df1.shape
Out[15]: (10000, 5)
In [16]: df1
Out[16]:
one two three four key
```

So arrays and within brackets we are indicating 2:4. So starting from the index number 2 to index number 4 that means element number 3 to element number 5. So, here the last element the stopping point is not typically included as you would remember, as we had learned in during list. So, you can see in the output 3 and 4, 4.3, these 2 elements have been returned and 5 has not been returned because there is a corresponding to the stop you know point.

Similarly, let us take another slicing you know, 2:4 here. So, you can see here another slicing. And let us take another one in the code line number 62 we had made an assignment. So all the values from you know 2 when we perform this slicing all the values starting around next 2 to 4, 4 not included that means 2 and 3, they will be assigned this value 3.5. So, if we look at the output number 63 the values will be 3.5 3.5.

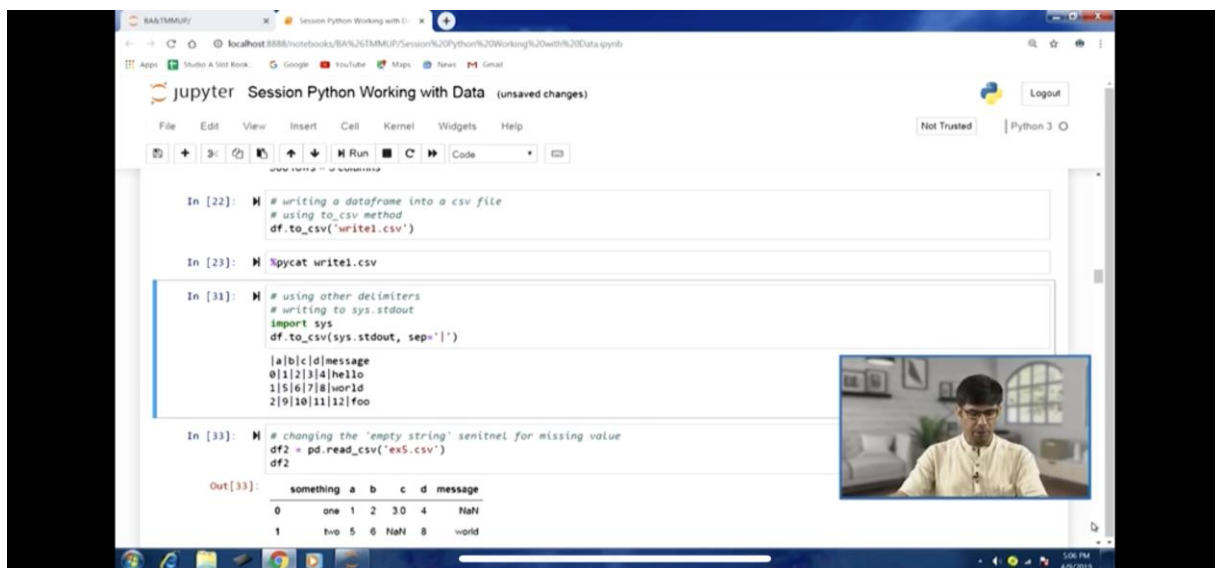
Now, if you look at the original array here in the output 64 you can see these 2 values are third element and 4th element they have been they have changed to 3.5. So the assignment has taken place. So in 1 go we can slice an array and in one go we can assign all the values that are there part of that slice, you know array. So in this fashion, this assignment can happen. Now another way to assign you can see here array 1, we are going to assign 5.5.

So let us take a look at the output. So you can see second element has been changed to 5.5. Now there is another thing where we do not mention any start point or a stop point, just the colon is there, that means the bare slice so all the elements would be included in this. So if I run this bare slice here, you can see array is actually equivalent of the original array arr in this case. Now if I perform a assignment operation using bare slice, let us look at the line number 68 array and just column there and I assign the value 0.5 here.

And if I look at the output here, then all the elements of this 1d array they have been changed 0.5. Now let us move forward. Now, next thing is you know multi dimensional arrays. So, in this case, we need to first understand how we can supplies multinational arrays in python first. So, elements of a higher dimensional array or you know, lower dimensional arrays and not values. So, this we need to understand. So, let us take an example here.

So, let us take a 3 cross 3 array. So, essentially, it means it is a 2d array. So array t2d let us create this np.array and we can see this is a list of lists and that we are using to create this array and you can see first list within list is 1 2 3 and 4 5 6 7 8 9. So, if I run this and you will see this is the array that we have just created. So, lower dimension within this 2d array would again be you know w-arrays.

(Refer Slide Time: 22:34)



So, in this case, this is 2 dimensional. So, lower dimension would be 1 dimensional arrays. So, let us look at the shape of this 2 dimensional array here. So, few points that we can understand here is that number of dimensions in an array is typically number of elements that we have in the you know, the output tuple of the shape object. So, if I run this `array2d.shape`, and this we have seen before `3,3`.

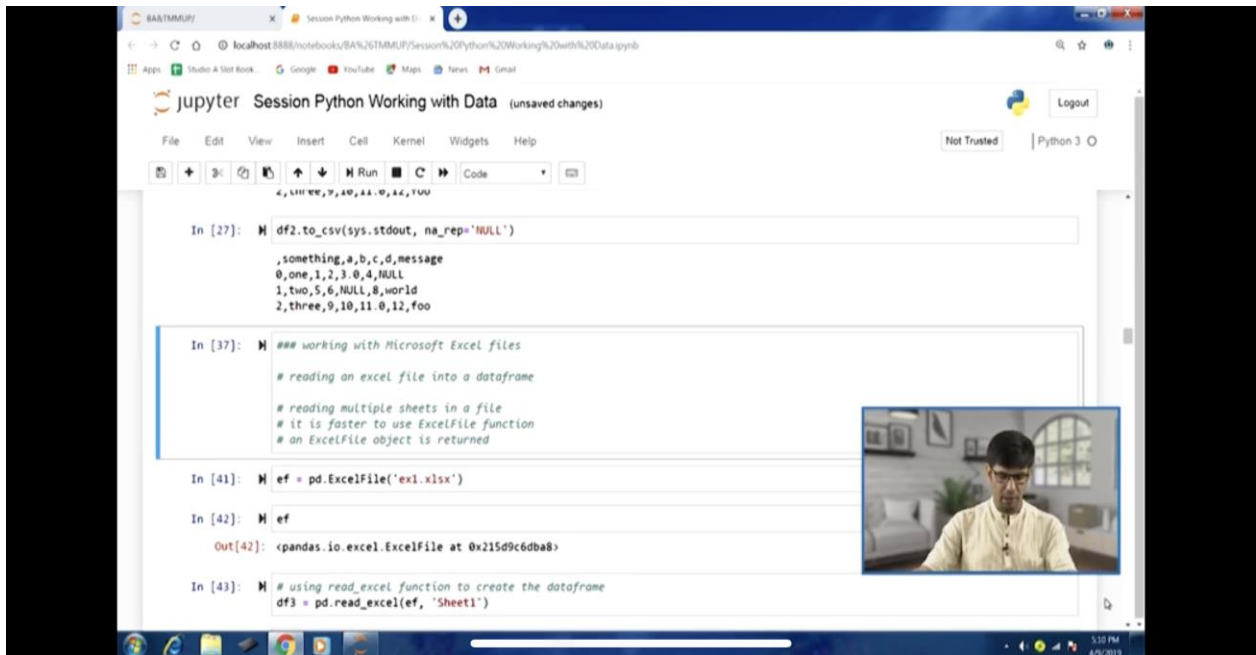
So, number of elements here are 2, so that is the dimension of this array. So, this is first thing now, first value in this tuple indicates the number of elements in the first dimension that is 3 and the second value in this tuple that we have got from the shape object, this indicates the number of elements in the second dimension . Also we need to understand that these elements of a dimension could be arrays or values.

It is not necessarily that they are going to be just the scalar values they could be arrays itself. So, just like the output number 71 that is the original arrays 2d, but if you look at the first element here in the first dimension, it is an 1d array itself. So, this array 2d this is 3 cross 3 first dimension having 3 elements which are actually happens to be 1d arrays, then the second dimensions has 3 elements, which actually happens to be scalar values 1 2 3.

These are scalar values, but these you know 1 2 3 as an array 4 5 6 7 8 9, these are 3 arrays that we have. So therefore, 3 cross 3. Now, if you want to access the lower dimension, so, this is how like we did before also array 2d and if you just specify 1 value zero. So, we would be just accessing the you know sub arrays there. So, in this case, if i run this and the output number 73 we got a 1d array 1 2 3.

Similarly array 2d 1 we will get 4 5 6 which is 1d array. Similarly array 2d 2 this is the third 1d array that we have in this. Now we can have a look at the number of dimension and shape object as well. So, here you can see dimension of this array 2d brackets 1 this is 1 and you know similarly, if you look at the shape object, you can see just 1 you know element there in the shape output indicating the number of elements you know in that just 1 dimension there.

(Refer Slide Time: 25:28)



Now, if you compare this with the array 2d itself, then you would see that and you know return the dimensions as 2. Now let us take bigger array in the sense multi dimensional array, 3 dimensional array, so, let us take 2 cross 2 cross 3 array here. So array 3d and you can see here this is a 3 dimensional array here. So, this is a list of list. So, you can see 3 you know nested brackets there, that is another way to find out the kind of you know, which the kind of array that we are creating.

So if there just to you know 1 bracket no nesting and 1d, if there are 2 brackets then 2d, if there are 3 brackets here at the start, you can see the 79 , then 3 brackets are there that is 3d array that we are going to create here. So, that is another way to understand this. So, if I run this will have this 3 dimensional array array 3d and you can see in the output. So, if we look at this, then 1 2 3 and 4 5 6 list comprising of these 2 lists is the first element of the in the first dimension.

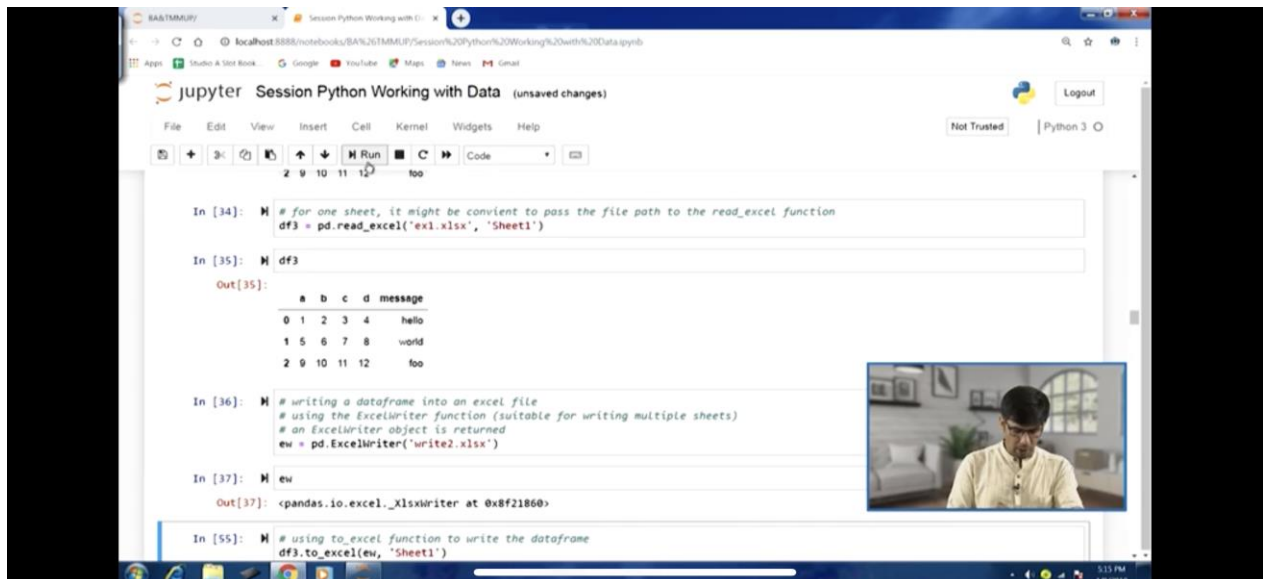
And a list comprising of these 2 lists 7 8 9 and 10 11 12 is you know is the second element in the first dimension. Now, in the second dimension, we have 2 elements you know, that is a list comprising of lists. So, these are 1 2 3 and 4 5 6. These are 2 elements in the second dimension and we go deeper than then there are you know most nested list there, we have 3 scalar values.

So, that is in the third dimension and third dimension we have 3 elements which are scalar values in the second dimension we have you know 2 elements, which are list. And in the first dimension we again have 2 elements which are list of lists. So, this is how we need to understand this. Now again, if I try to access the shape object for this particular array `array3d.shape`, you can see in the output I have got 2, 2, 3.

So that is indicating that we have total number of elements 3 that means this is a 3 dimensional array, which we clearly understand and in the first dimension we have 2 element, 2 elements in the second dimension we have 2 elements, the third dimension we have 3 elements. In the third dimension we have 3 elements which are actually scalar values like 1, 2, 3. And in the second dimension we have you know, 2d array, where we have 1 2 3 and 4 5 6.

These are 2 you know 1d array. And in the first dimension again we have 2 elements, which are list of you know, which are 2d arrays themselves. So, first element is 1 2 3 4 5 6, second element is 7 8 9 10 11 12. So, clearly the understanding part has to be very clear here. So that we are able to work with all these you know multi dimensional array. So another way to access a multi dimensional array.

(Refer Slide Time: 06:50)



So, here you would see array 3ds, so earlier we used to specify, you know list of indices. Now, instead we can specify like this in a recursive fashion, array 3d first bracket 0, then second bracket 1. So, in this fashion also we can specify that means the first dimension zeroth element that means first element, second dimension, you know, index is 1, that means second element.

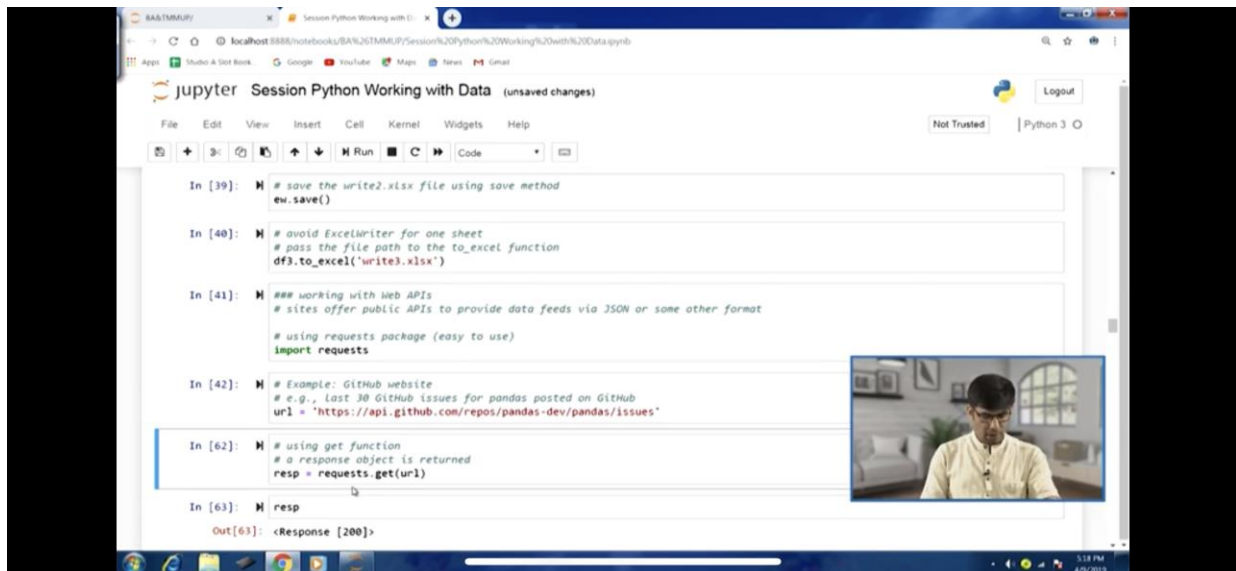
So, that comes out to be 4 5 6, if I run this, you can see the output number 82 and you can compare this with the output number 80. So, you can find out how this is working again dimensions 3. So, let us move forward. Now, dimensions of array 3d that we have just you know, just talked about this is 3 dimension array. So, these are 2 . So, these dimensions are two 2 cross 3 arrays.

So, this is another way to understand this, that 3 dimension array in this case array 3d is having two 2 dimensions which happens to be 2 cross 3 arrays. So, if I access array 3d in brackets zero if I run this, this is the first 2 dimensional array, which is a 2 cross 3 array. Similarly array 3d 1, this is the second 2 dimensional array, and this is the second element there in the first dimension of array 3d.

So different ways to understand these multi dimensional arrays. So I am trying to give you and different perspectives on how to understand these multi dimensional arrays. Now , if we want to create a copy, so, we can use the copy method, so, you can see at a 3d and in brackets 0. So, this 3d array 2 cross 3array, we can use dot copy method, and you can record array 3d_0. So, this is one way to copy these arrays.

Now, let us talk about the assignment of values and arrays here. So, let us first start with the value assignment. So, if we perform an operation like this array 3d and brackets 0 that means the first element in the first dimension, that means the 2d array first 2d array and in the right hand side we have 55. So, all the elements in the first element of you know first dimension they are going to be assigned this value 55.

(Refer Slide Time: 28:26)



So if I run this, and if I try to have a look at the array 3d, and you can see that you know, first dimension first element, which is a list of list 55 55 55 first one and 55 55 55 second list together list of tests. So, this is how this value assignment can make changes in the original array, because as you know, so, we are working with the view, so, any changes will be reflected in the original array.

Now, you can compare this with the array that we had recorded array2 _0 you can see the values have changed. Now, let us talk about the you know this assignment using array. So, here you can see that in the left hand side we have array 3d 0, 1. So, essentially what we are talking about is 0 indicating that you know first element in the first dimension 1 indicating that the second element in the second dimension.

Once we identified that list, we are in the right hand side you can see we have 1d array. So that is zeroing to be replaced with this 1d array having values you know 66 66 66. So, if I execute this, and if I look at the output, you can see the output here and then reflected back to you know, if I look at 0 0, so, we had already created 55 55 55. So, you can see here and we can have a look at here array 3d 0 we can this another way of assigning this.

So if I run again, you can see you know, we can restore this in this fashion 1 2 3 4 5 6. So, earlier the array 3d 0 1 that assignment that we did, it would have changed the you know a

second list there. Second list that we had an output at 88 55 55 55. So, first would be possible remain as 55 55 55 after performing operation of our 90 but the second one would be changed to 66 66 66 right.

So in this fashion these assignment of arrays either using arrays or values can actually take place. Now dimensions of these arrays 3d 0 and array 3d 1 which are 2 dimensional arrays, or 2 1 dimensional arrays, each having 3 elements. So if I try to access array 3d 1,0, the output would be this array 7 8 9. So this happens to be 1 dimension array. So this is you know, part of the dimension of array 3d 0.

Let us take another example array 3d 1, 1. So, if I take this, you can see we again 1d array here. Now next thing is about slicing multi dimensional arrays. So we will discuss this aspect in the next lecture.

(Video Ends: 29:27)

Let us stop here, thank you.

Keywords: Array, Multi- Dimensional array, Slicing, Exception, classification.