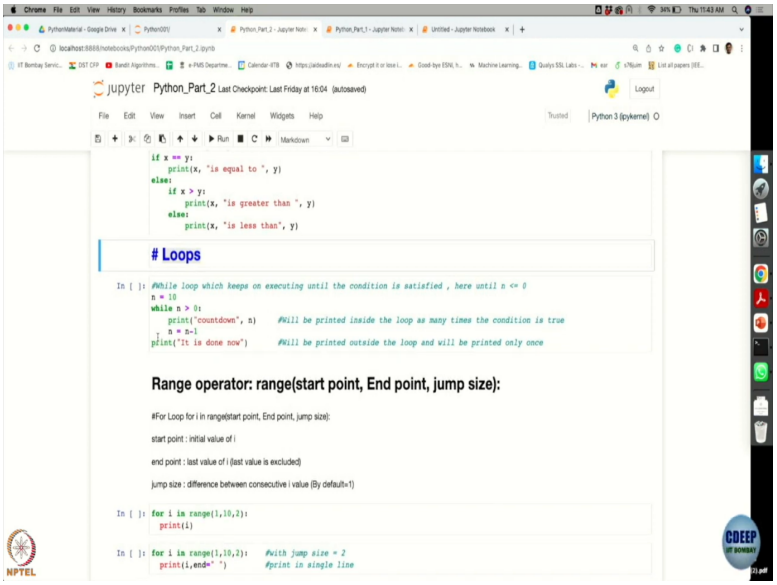
Engineering Statistics
Professor Mahesh Hanawal
Industrial Engineering and Operations Research
Indian Institute of Technology, Bombay
Lecture 30
Python-loops and Numpy Library

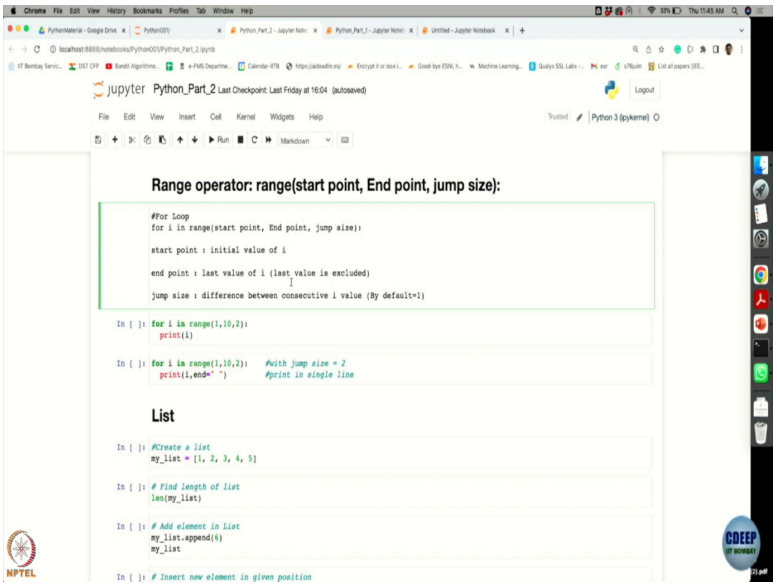
(Refer Slide Time: 00:23)



Okay. Now, after this if-else another important thing for us is loopings or loops. Python also provides very easy and intuitive ways of writing in the loops. Suppose, let us say I want to have 10 numbers and I want to print them one after another. So, they say like I have to do this printing activity 10 times repeatedly. So, for this we can say n is assigned a number 10 and when n and we can write 10 printing statements for by decrementing the value of n.

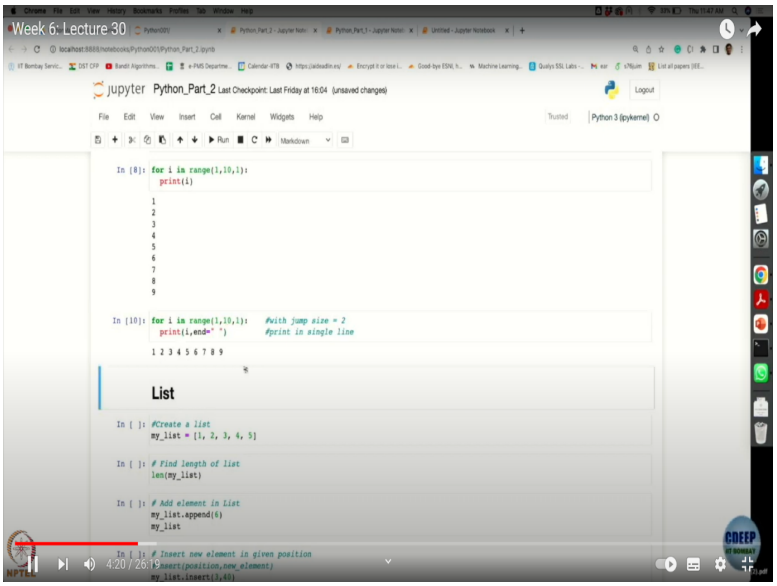
So, this we can simply do by while n is positive, that is achieved by the statement n is greater than 0 and you end it with a colon here and then print this number and then you reduce it by 1 and this while continues still n is greater than 0 and when this is done and when n becomes a 0 here, the while statement is not correct, the loop will come out of this and then it is print it is done now and this is how the output will look.

(Refer Slide Time: 01:39)



Now range operators are often like when you are doing this looping, we need to decide how many times you wanted to want to loop it and all for that range function comes a very handy its format is if I want to loop let us say a certain number of times, what I have to do is I have to tell what is my starting point, what is my end point and how much is a jump size. So, starting point here, we will say what is the initial value of i, i is going to take a different possible values here which I want to take in my range that I am going to specify and here endpoint is simply the last value that I want i to take. And usually, this last value is excluded when I am assigning these values to i and jump size is basically the difference between the consecutive values of i.

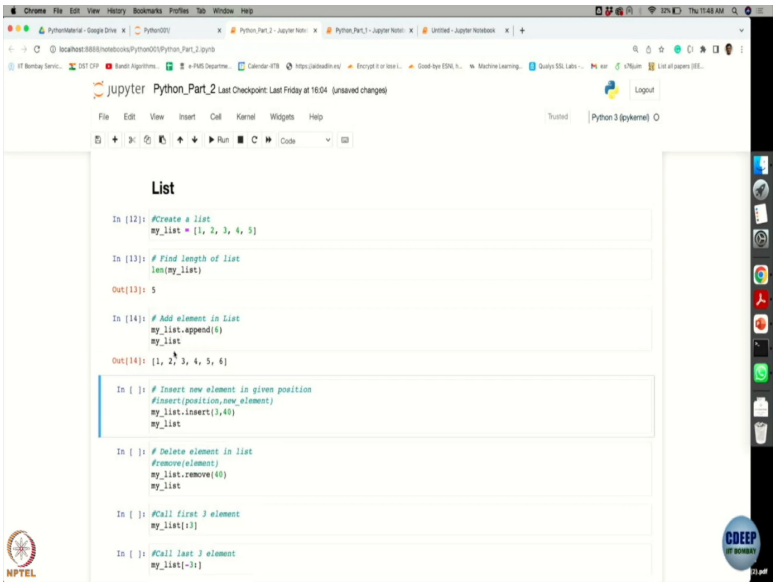
(Refer Slide Time: 02:52)



So, for example, if I want to print the numbers from 1 to 10 with a jump of 2, all I have to do is for i in range 1 to 10 to 2 print i and when I execute this you will see that I will see the value 1 3 5 7 9 and here anyway 10 is not printed because after 9 if I have to do a value of jump of 2 it is going to be 11 which is not in my range.

Suppose let us make it 1. Now, you see that the starting value is 1 and it is incremented the value of 1 but the last value here is only 9. This is mainly because as I said the last value is skipped. Okay, i is not assigned the last value 10 here and now, you see that every time a print command is executed a new line is automatically inserted. So, because of that you are seeing it as a column. In case you do not want to do this python provides this option to add this end equals to whitespace if you do this, you will get the things written in a horizontal way in a row and if you want to just see what we did in the previous cell, so now this column is written as this row here.

(Refer Slide Time: 04:24)



Now creation of the list. A list can be created simply by putting them between 2 square brackets and separating them by commas. Here are how created five items, numbered 1, 2, 3, 4, 5. And so, this length function gives how many elements are there? First this is exit this give me error here because I did not execute this cell and that is why I did not know what is my list. So, now I have executed now I am coming back and execute is now it shows what is 5. Now there are many functions which makes manipulation of the list very handy in Python.

One thing is append function. If you want to append a new element to this list, all you need to simply do is my list dot append and add the number you want to append. And if you do this

and print my list, you see that a new element is getting appended at the end of my list. Similarly, we can insert new values at any place not at the end, maybe at any place here I am trying to insert at the third location number 40. And if I do this, you see that 40 is coming here. So, notice that indexing always starts from 0. So here, when I say third means actually in the list, it is like a four position because indexing starting from 0. So, if this is a 0th position, 1, 2 and 3 and 3 is where this is where the 40 has come.

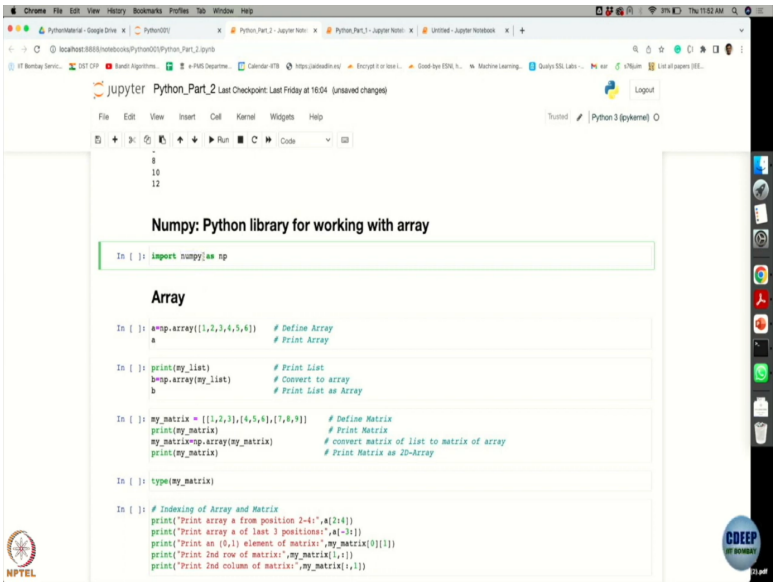
And removing is also as easy as adding it any place. If you want to remove a particular element, wherever it is in your list, all you need to say that number to be removed by using this function remove and it gets removed. And also, it is easy to get some part of your list by saying if you are only interested in the first three elements, all you need to do is just give a colon and 3 here and you will get these three elements. And similarly, if you are interested in only the last three elements, so you need to do minus 3 and give a colon.

Okay, so this comes very handy this minus things like so that you do not need to reverse any of your list, we Python will automatically take a ref and reads from the end. And reversing, if you want to reverse the list, all you need to do is colon colon minus 1 is 1 to do and you are listing get automatically reverse you see that it has been reversed here.

And to add or to append to list, all you need to do is write a plus symbol between them and if you do that, you see that two or like maybe concatenation happening here like when you use plus on list, the concatenation happens and the two lists get merged and if you want to do some element wise operation in your list, then loop operations comes pretty handy.

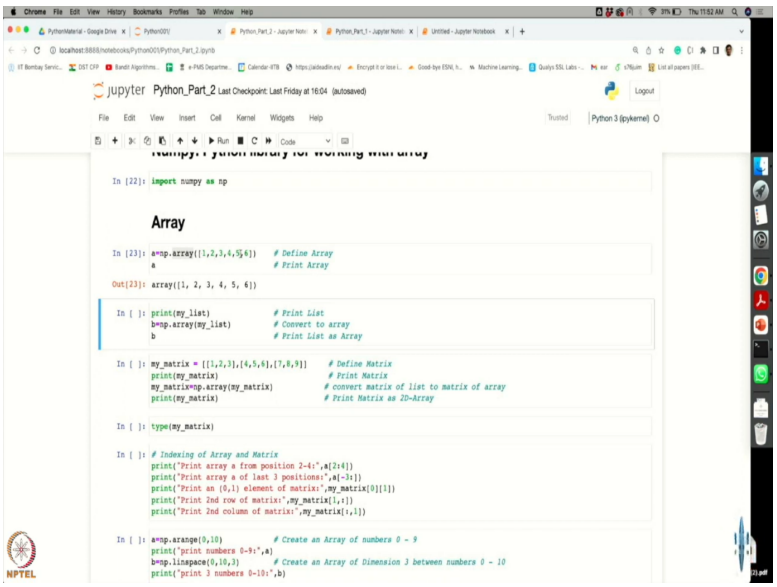
Like now here, let us say you want to multiply every element in your list by two. So, then you can run through each element in the list and multiply that by a factor of 2 and print it. So, if you do that, you see that there were 6 elements and each of them are getting by multiplied by 2 and this first 6 elements are here, multiplied to and listed here.

(Refer Slide Time: 08:21)



This Numpy is one of the very useful libraries in Python which help us to deal with the arrays. So obviously, when you have to do a lot of data, data may be put you have to put it in matrix format to manipulate them and this Numpy library comes as very handy. And if you want to use this library in your coding, first you need to import it and the option to import it is simply import Numpy and all the time you do not want to use Numpy maybe you want to use a shorthand and so you can even give the name like the shorthand name here is np.

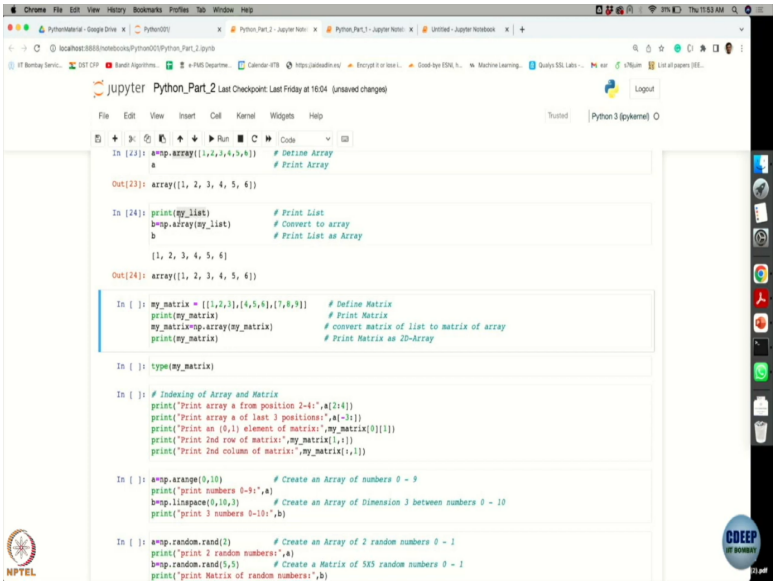
(Refer Slide Time: 09:10)



Now using this function may be we can now create arrays. So, now see here when I executed this 21, cell number 21 here it throw me an error because I have not executed this previous cell here. So, it do not know what is np yet. So let me first execute this cell. So, my cell is

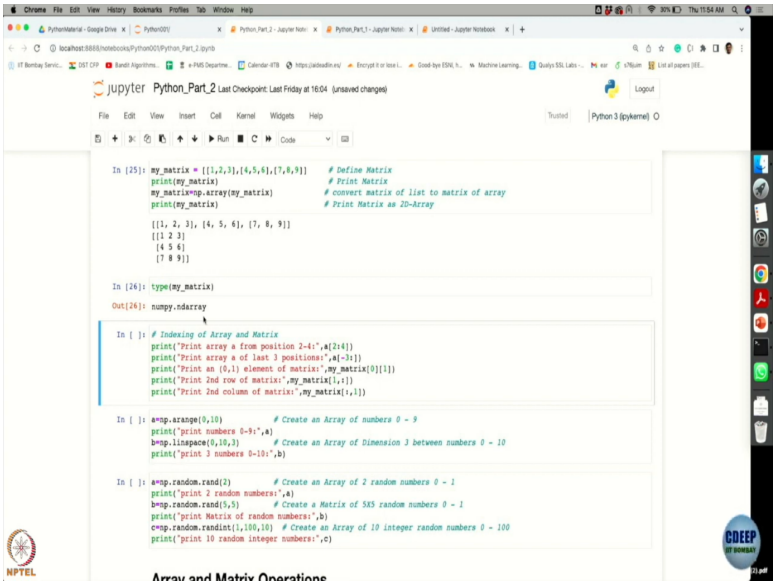
executed. Now my Python knows what is np. Now I can use that to create an array and now see like I have created an array of elements 1 up to 6 and when I print it, it shows me as a array.

(Refer Slide Time: 09:54)



Okay, so let us see what was the difference between the list and the arrays here. So, earlier I had created a list, if I print that list, you will get these numbers 1, 2, 3, 4, 5. And now when I created this array it and so here, okay, what I have done is basically I have converted my list into array and now it shows me as an array.

(Refer Slide Time: 10:19)

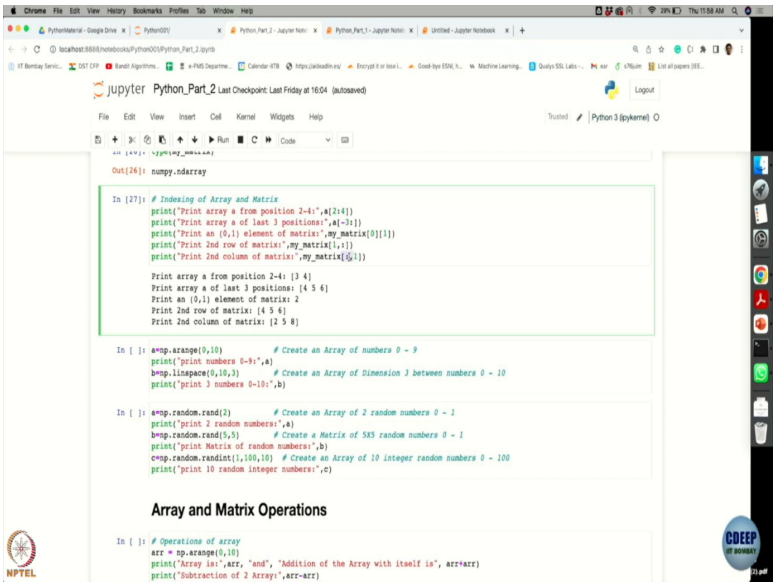


And now we can also create matrices like the way we created list here, I have created one matrices which is a 3 cross 3 matrix, the here I have put numbers 1, 2, 3 in the first row, 4, 5,

6 in the second row and 7, 8, 9 in the third. So, the rows gets separated by these commas and elements within a row are written within the square bracket with each element separated by commas.

And now, whatever this matrix I have, I can convert it to an array, again by using np array. So, let us see this. So, when I have this array matrix created here, this is what I got, but when I want to convert it to an array, you see that I will get this, what is the difference here, when I convert it to array, all the commas get removed, and they have, rows are now appearing one below the other. Okay, and now if I want to see what is the type of this my matrix here, it will show it as a numpy array. Okay.

(Refer Slide Time: 11:36)



So, when we are doing to do with this matrices and list, we have to be very careful in indexing them and know what convention Python follows to index them and as I said, indexing always begins with 0. So, if you want to access a particular element in your matrix or array, you should be properly indexing them or like properly use the right index to get those elements. For example, if you have a, what is a for us so far, I have an A here, which is defined as an array here, and it is a numpy array and I want to get the elements in position 2 to 4.

So, I have if I want to that I can give this as a 2 to 4. So, notice that 2 here means actually the third position, okay and I want till the number 4. But by convention, this last index is dropped, let us see what I what I get. So, if I execute that, a elements between 2 to 4, it will

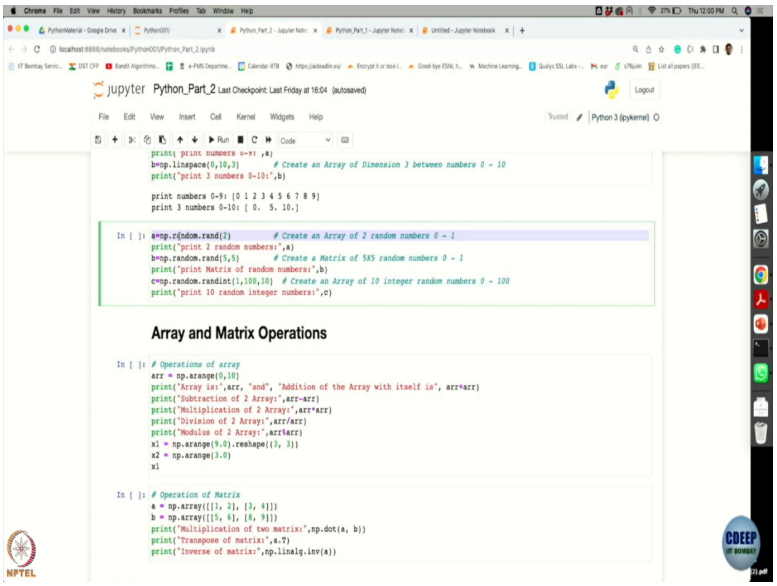
show me the value 3 and 4. So, 3 is fine because the position 2 actually the index 2 actually corresponds to the position 3, where 3 is there.

And after that, we have element 4 and this 4 actually, index 4 actually corresponds to position 5 but that is not displayed here. Okay, so it is only taking when there is basically taking difference between these two indexes and that difference is that many elements it is showing here.

Now, in this array, if you want to see the last three position, we have to simply do minus 3, this is the same thing we did also for the list. And the same thing works for the matrix also. Sorry, same thing works for this Numpy arrays also. And if you want to look into a particular element, suppose I want to look into the zeroeth row and the first I want to let us say 01 element of the matrix, then this is the same as saying I want I am interested in the zeroth row and the first column.

And that one for us is this is let us say zeroth row and first column this one. So, this is yeah, this one, so I should get a value of 2 here and if I want to print only the second row, I have to pass on the index 1 and write a colon here and I will get only the second row. And if you similarly I want to get the second column, I have to put the colon here and pass on the index for the column as simply 1.

(Refer Slide Time: 15:01)



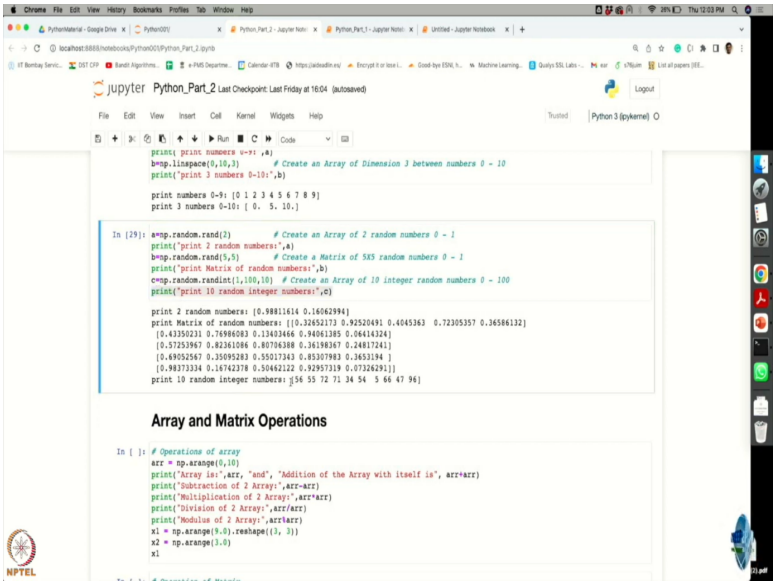
Like range function we used earlier Numpy has similar function which is arange array arrange maybe array range, that is what that a has come. And here if you want to print the elements like okay if you want to get the store the elements between 0 to 10 in A you can use

this command A np dot arange 0 comma 10. Let us execute this and see what happens now, if you do this see you are getting all the values from 0 1 2 up to 9. So, notice that this is because this is now an array, you do not see a comma between them comma between the numbers here and again the last number is not displayed.

Now, if you want to see only certain parts of these elements, let us say I want to see that only three dimension only take three elements that are equally spaced between 0 to 10 you can use the command np dot linspace and here this is your range and you have to give this number 3 here represent the spacing between them and you are going to get this value 0, 5 and 10.

But here you see that the last number 10 is also included because we wanted spacing to be and we wanted three numbers which also are equally separated, okay and here you see that this dot here, what this dot mean here this is basically is saying that this number or this number got saved as floats, okay. So, because of that, you see that instead of even though this 1 2 3 4 5 are all can be simply integers, but because of this spacing, you did now they are stored as float numbers and you are going to see a dot here.

(Refer Slide Time: 17:42)



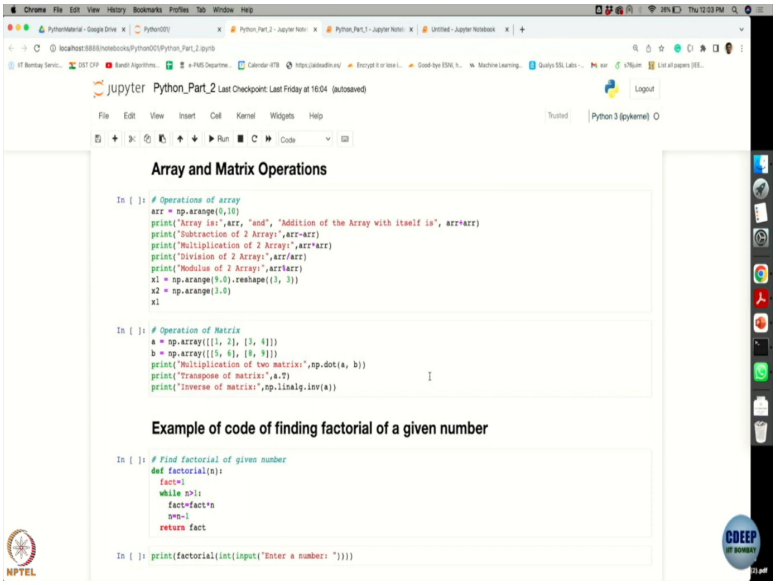
Now, instead of you putting all the times the numbers into lists or arrays, you can use sometimes you may have to deal with some randomly generated numbers, you want some random numbers and for that this random functions comes handy that is there in the numpy. So, if you want to generate let us say 2 random numbers which are between the interval 0 1 then you can use the RAND function and give how many numbers have some, numbers of that random numbers you want and that will generate. Let us execute this.

So, here when I execute this command and print its value, you will see that I get 2 numbers which are between 0 1 and these are randomly generated. How it generated random numbers? We have discussed in the course maybe like it uses inversion sorry like maybe like, we discussed the various possible methods of generating this random numbers uniformly in the interval 0 1, we said linear registers and I think other methods of the shift based on the shifts we discussed. So, maybe it will implement one all those methods and generate this.

And if you want a matrix of the random numbers, suppose you want a matrix of size 5 cross 5 filled with random numbers, we can use this again RAND function with argument 5, comma 5, it creates a 5 cross 5 random matrix. So here, it has displayed the random matrix it had generated. Now, instead of all the times generate, instead of getting the random numbers in the interval 0 1, you may be interested in getting random numbers which are integers, maybe in some range.

Suppose you want to generate random numbers in the interval 0 sorry, 1 to 100 and you want 10 such numbers, you can use this rand int function and whatever the value we get, if you print you will see that I have gotten some 10 numbers here which are between 1 to 100 and these are like a randomly generated.

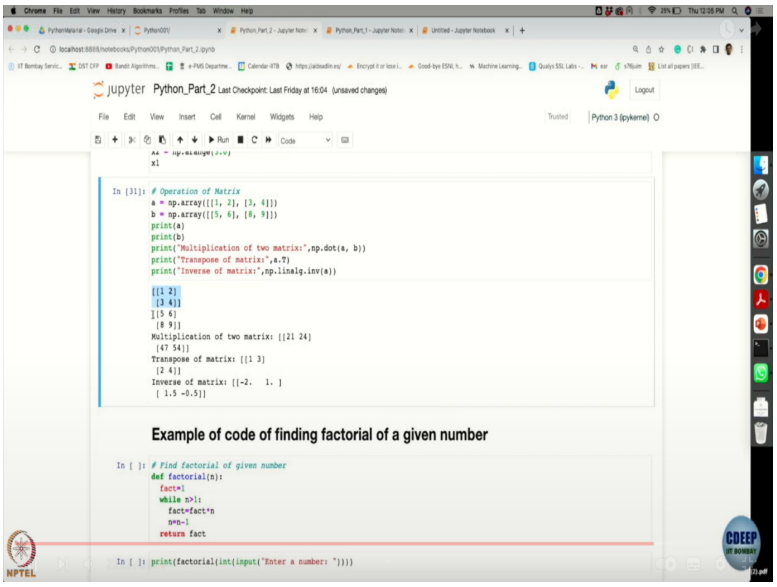
(Refer Slide Time: 20:02)



So, this random function is going to be very useful for us if you want to generate later samples according to a given inverse given function. If you recall, we said that if you want to generate a sample according to a function f, let us assume that f is invertible, then you can

apply this f inverse on this uniformly generated random variables and whatever the transform numbers you get, they are distributed as per your required distribution function.

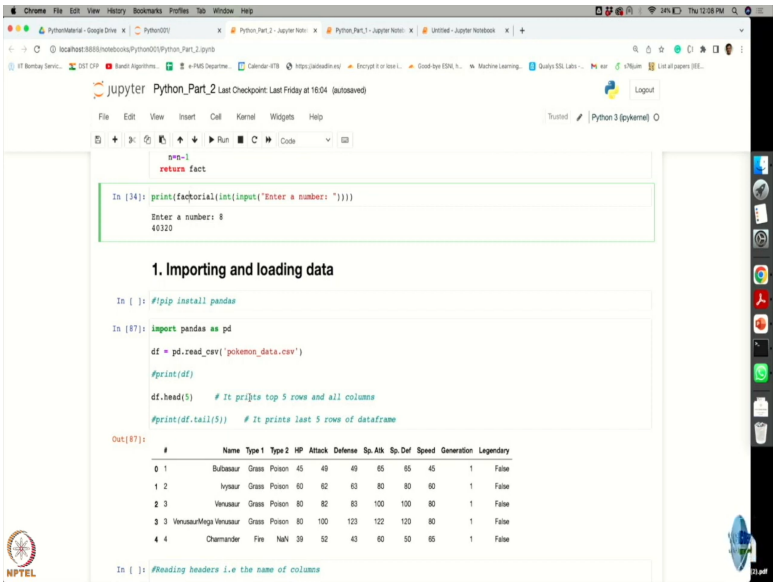
(Refer Slide Time: 20:43)



Operation on the matrix is also very straightforward in Python. So, some of the operations we do in matrix or we want to multiply them or invert them or do the transpose them, they can be easily accomplished. Suppose we have these two matrix A and B and now you want to take a dot product between them, all you need to do is np dot and pass on these two matrices A comma B, A and B, it will give you the multiplication. And if you want to transpose, all you need to do is A dot T. Actually, this is straightforward you do not need even NumPy. For that to do the transpose, you can simply put A dot T.

And if you also want to do inversion of that, so we are going to use the inverse function available in the linear algebra part of the numpy library and you are going to use this function. So, let us execute this. So, you can just verify that if you multiply this, you are going to get this matrix and it is clear that the transpose of this matrix maybe it should I, maybe I will just print a and b for better visualization. Yeah, so this is one matrix for us and this is another matrix and it is clear that if I have to transpose, I will get this and if I have the inverse, I will get this.

(Refer Slide Time: 22:41)



Now Python also helps us to provide some functions which we can call by passing certain arguments it is not that we have to use all the time the inbuilt functions, we can write our own function. And as x first exercise, here, we are writing our first function. So, function is going to do a factorial of a given number, find a factorial of a given number. So, to define a function, I have to follow the syntax define factorial n. And now, n is the argument it is going to take. Here the actual computation of the factorial is working.

So, I will start with a variable which I am initialized to 1 and as long as this number n is greater than 1, this variable is going to get multiplied with itself. And this n keep on decrementing and obviously, when n riches value 1, this comes out and you see that n is multiplied by with all the values less than n up to 1 and that is exactly the factorial of n. And when this is completed, it will return me the value of fact, that is the factorial of the number n that I have passed.

And we can make it so let us execute let us understand what we are doing here like first here, I am giving as input. And now whatever the input I am going to give I am going to convert it to integer and then going to pass it to my factorial function and that is going to compute the factorial of that and give me. So, see now when I executed this cell 32 it says set an error because I have not executed this cell previous cell where I have defined my factorial function.

Let me execute this first and then execute this. Now see it is asking me let us say I want to compute the factorial of 8 and now it has computed its value to be this much. Okay. So, here I could have written the same step by dividing into three parts first I asked okay enter a number

then stored it in some number and then convert it in the second line, I could have convert it into integer and in the third line I would have passed that number to the factorial function instead of that I could do everything in one line and achieve the same thing.

So that is another thing very handy here when we want to play with all I mean when we want to compress or we want to write a very short codes, this kind of writing these concatenated commands are very handy.