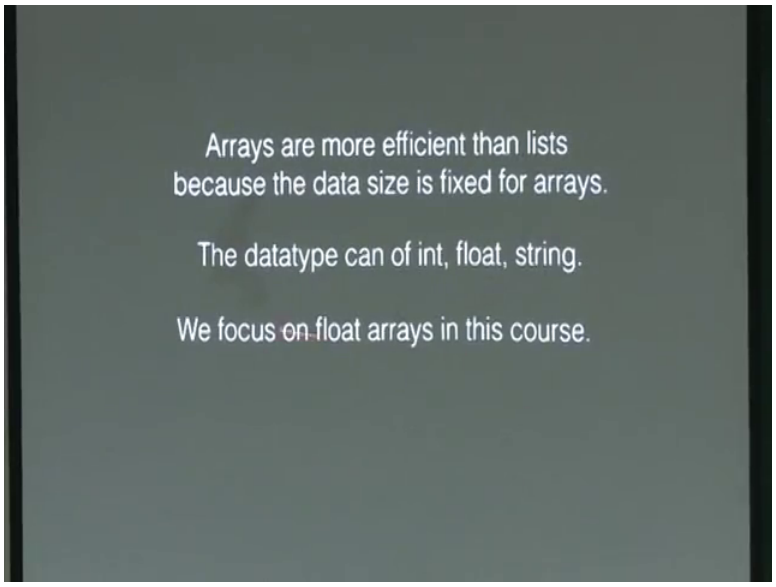
Computational Science and Engineering using Python

Prof. Mahendra K. Verma
Department of Physics
Indian Institute of Technology, Kanpur

Lecture - 03
Python: Numpy Arrays

So, let us look at Numpy Arrays in bit more detail.

(Refer Slide Time: 00:22)



Arrays are more efficient than lists because the data size is fixed for arrays.

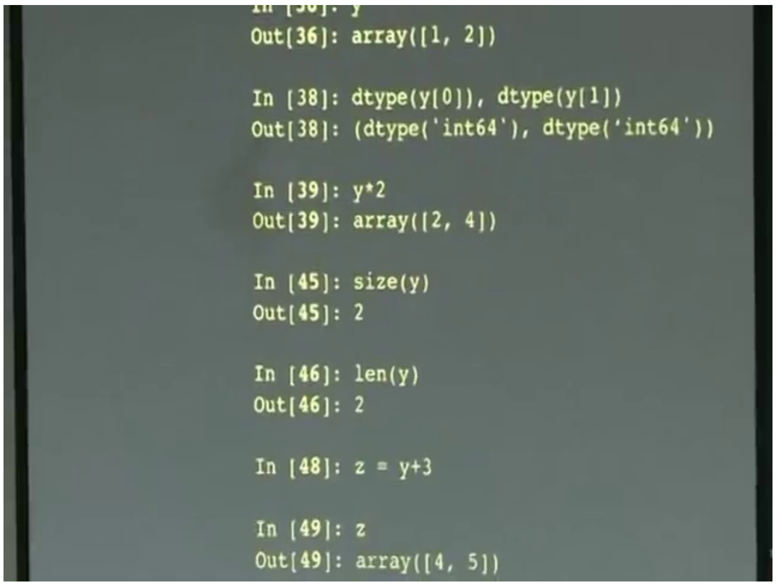
The datatype can of int, float, string.

We focus ~~on~~ float arrays in this course.

So, this is efficient implementation of list. So, we should not use list for our course because we want to do things fast. Computer science datas are not of the same type and there you sacrifice speed for some processing.

So, here we want to make it efficient and data type, data size are all fixed. So, a number of elements in the array is fixed; it cannot be changed. Data type can be int float or string, but it is one type. There are exceptions, but in this course I will not discuss it. So, either it is of integer type or float type or string type. We also do not want to deal with strength. So, basically float an int and that is what we will do. Also, it is a homogeneous data type and that is what makes it fast and data size also fixed. So, I will give an example after this.

(Refer Slide Time: 01:20)

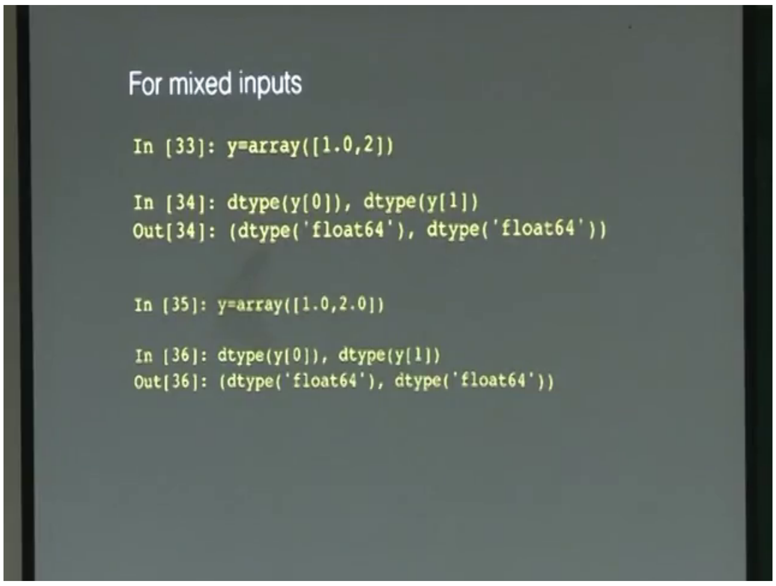


```
In [36]: y
Out[36]: array([1, 2])

In [38]: dtype(y[0]), dtype(y[1])
Out[38]: (dtype('int64'), dtype('int64'))

In [39]: y*2
Out[39]: array([2, 4])

In [45]: size(y)
Out[45]: 2

In [46]: len(y)
Out[46]: 2

In [48]: z = y+3

In [49]: z
Out[49]: array([4, 5])
```

So, this is how we define an array. So, this array is a function array with. So, a small bracket means function. A function argument will go inside. So, array is a function whose argument is a list, but these are all integer type, 1, 2. Is that clear? So, array always will come with small brackets and not with large bracket. Now, if I just say y is a variable which contains a arrays. So, it will just print this. Now, I can look at the type of the elements. So, here my elements are y 0 and y 1. Y 0 is the first element and y 1 is the second element and to get the type, I would say d type, data type d for data and these are integer 64. So these, our 64 bit, ok.

So, this is what is stored here. Now, y star 2 will multiply each element by 2. I can get the size. This array is number of element is 2. Length also gives you same answer. I can get a new array z as y plus 3. So, 3 will be added to each element. So, most operations like addition subtraction division and so on will be acting on each element of the array. So, the array answer is 4, 5 because just add 3 to each element now.

(Refer Slide Time: 02:57)

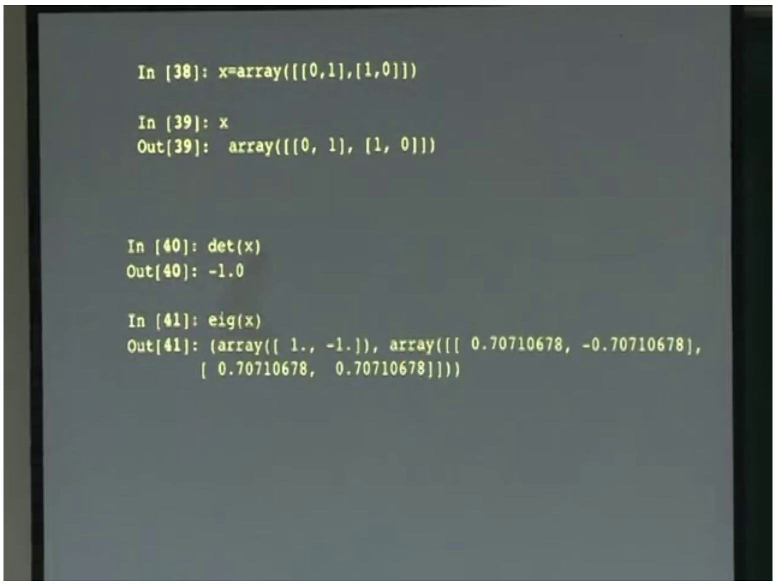


For mixed inputs

```
In [33]: y=array([1.0,2])

In [34]: dtype(y[0]), dtype(y[1])
Out[34]: (dtype('float64'), dtype('float64'))


In [35]: y=array([1.0,2.0])

In [36]: dtype(y[0]), dtype(y[1])
Out[36]: (dtype('float64'), dtype('float64'))
```

What if you put mixed input? So, you may be saying some integer, some float will focus only on that and not on the string. So, if I have mixed input, so my input for array y is 1.0, 2. So, this is float means real float is real floating point. So, it is real. 2 is the integer. Now, if I say d type, so it gives you both float. So, what has it done? It has converted that one rather 2.02 to 2.0. It will convert it automatically. So, it makes all the data as same type, that what makes it efficient. List will not do as list will keep one as float and other one as int which I showed in the last class. So, that is the difference between list in array. What if I put both of them as 1.0 2.0? This is straightforward. It should be both; float 64. Its 64 bit is a double float. 64 is same as double of c. So, let us do some more.

(Refer Slide Time: 03:56)



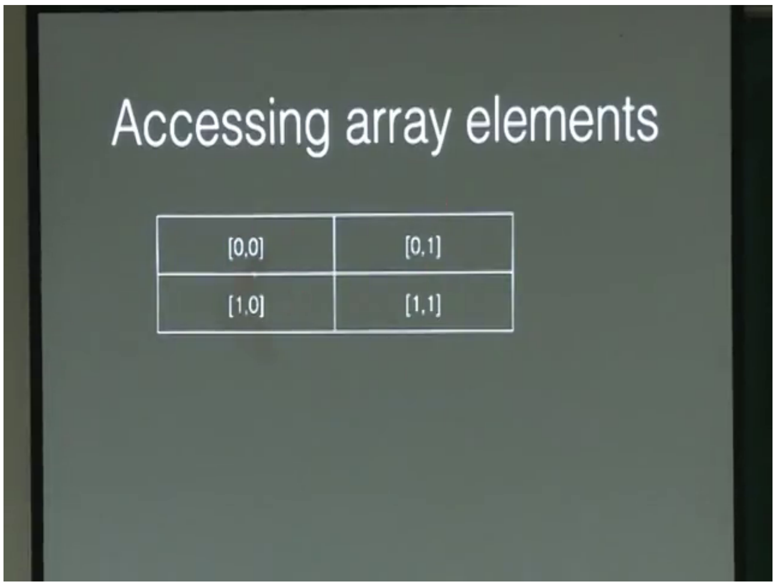
```
In [38]: x=array([[0,1],[1,0]])

In [39]: x
Out[39]:  array([[0, 1], [1, 0]])


In [40]: det(x)
Out[40]: -1.0

In [41]: eig(x)
Out[41]: (array([ 1., -1.]), array([[ 0.70710678, -0.70710678],
       [ 0.70710678,  0.70710678]]))
```

With this a 2 d array means there is a square bracket. Within that there are two elements with square brackets. So, it is 0 1 1 0. It is a matrix. If I say x, then I will get the same thing.

Now, I can do the determinant of x. So, I get answer is minus 1 eigen value and eigen vectors can be obtained by the function ie g ei g. It will give you eigen values are here and eigen vectors are two of them. So, it is a 2d array. You see the first is the first eigen vector and the second element is the second eigen vector. Now, this array is of type c. It has two elements. This is the first element which is a vector with 2, 1 and this is an array with 2, 2. It has four entries and it is one entry. It is an array.

This is inhomogeneous. We do not want to start working on this, but Python will output this. So, Python is very kind of you know is a loose language, where if data can be of different types, it will manipulate it ease of flexibility, but of course it makes the slope a little bit look at how to access array elements; so how to access array elements?

(Refer Slide Time: 05:10)



Accessing array elements

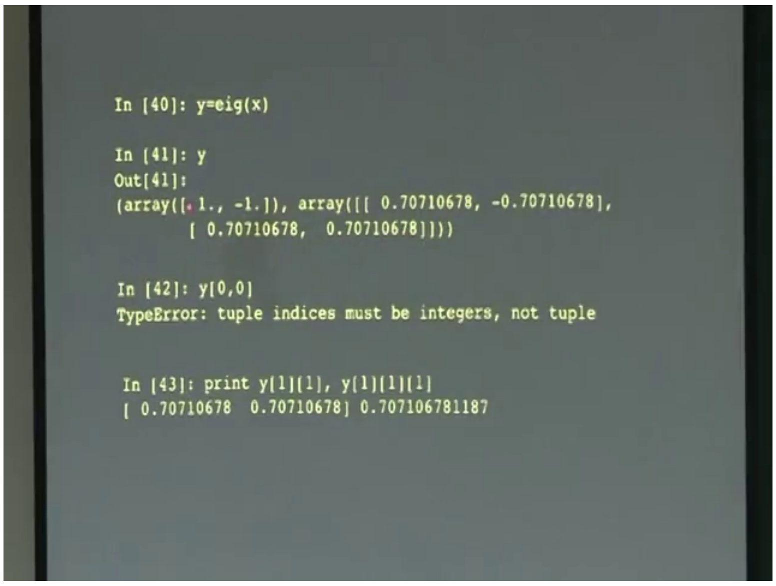
[0,0]	[0,1]
[1,0]	[1,1]

So, in 1 d straight forward one ds like list; so y 0 y 1 y 2 like that, but in 2 d you have to access like this. So, again start with 0. So, 0 is a valid address. So, 0, 0 is the first element of this 2 d array 0, 1. So, this is a row and this is called column. So, the second index is the column index and the first index is the row index. Is that clear? So, these are rows 0 and row 1.

So, that is the first index row 0 row 1 and this is column 0 column 1. So, column 0 is here right and column 1 is here. So, it can be bigger arrays. It is 2 d or we can also do 3 d. 3 d will have three indices i j k. 3 d is not a matrix. No, 3 d is an array like in this room I want to represent position of a point, then it is going to be x y z and you can represent by 3 d array or it can also be 4 d, 5 d. Whatever dimension you want, programming language is created and Python does it too.

So, we can access the 0, 1 which will be this element. You remember there was 0 1 1 0. So, it is correct. This is the answer one print. Now, we can also access this element by putting two; bracket 0 bracket 1. These give the same answer. So, this says you bit of writing, but you can use this as well.

(Refer Slide Time: 06:42)

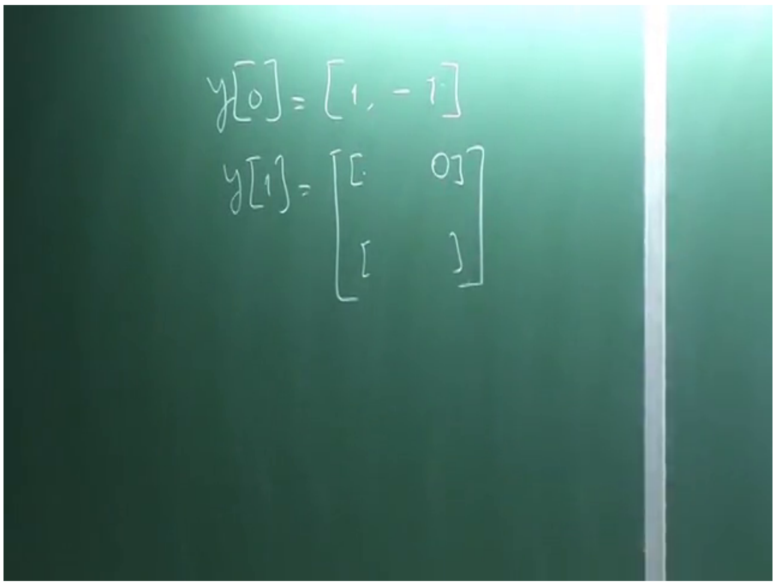


```
In [40]: y=eig(x)

In [41]: y
Out[41]:
(array([ 1., -1.]), array([[ 0.70710678, -0.70710678],
       [ 0.70710678,  0.70710678]]))

In [42]: y[0,0]
TypeError: tuple indices must be integers, not tuple

In [43]: print y[1][1], y[1][1][1]
[ 0.70710678  0.70710678] 0.707106781187
```

Let us look at this eigen function again. So, I can assign this answer to a array. Why? It is because it gives an array i e i g returns an array. So, if I say why, so it is going to give me the array. I want to access some element; let say I want to access this guy 1.0. How do I access it? This is an array y y 0 or 1 y 0, 0.

Will it worth 0, 0? It turns out it does not work because it is an array. So, this would not work and it gives error like this. It will come in your screen. So, what you have to do is, you have say 0 bracket. So, if you say 0, put a bracket here, put a bracket here. It will work like this. So, y 1 1 will give you in fact these are another example. Y 1 1 will be what? So, this is 0, this is 1, but 1 1 will give you this. This one this is y 0 and this whole thing is y 1. So, it is a structure if you know this c. So, let us write this.

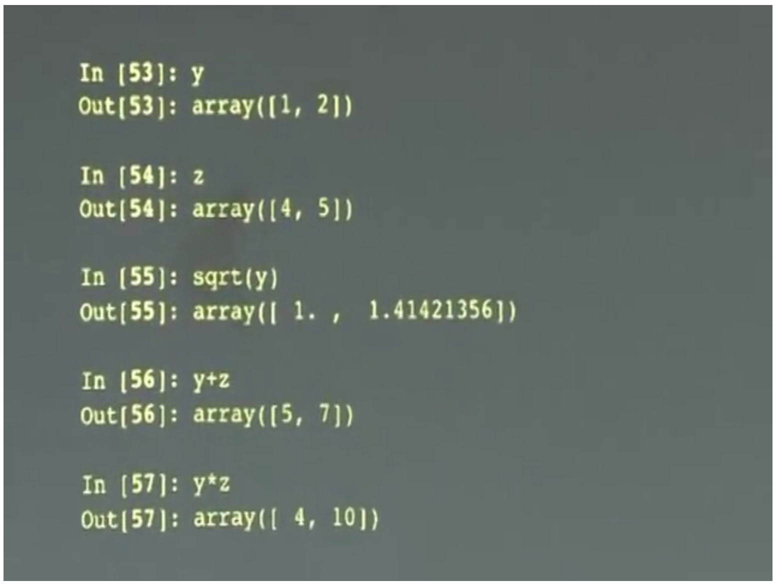
(Refer Slide Time: 07:47)



So, for this example y 0 is a structure which is that 1 minus 1. Is it correct? 1 minus 1 and y 1 is a 2 d array. Now, I want to get y 1 1. So, if I say y 1 1, what will I get? So, this consists of two of these. So, I will get here and if you want to go within that, so I want to get here. Then, I have to put this y 1 1 1 in three levels. So, you go in a building, then you go inside the one corridor, then go inside a room and this part of structure, actually this is not an array really. In strict sense, it is a structure which has sub arrays inside, but do not worry too much about it. I just wanted to illustrate that a dressing has to be carefully done.

For you, you just focus on matrices and arrays which are 3 by 3, 5 cube like that, but if you want to access data from these kind of objects, then you need to know how to do it.

(Refer Slide Time: 09:05)

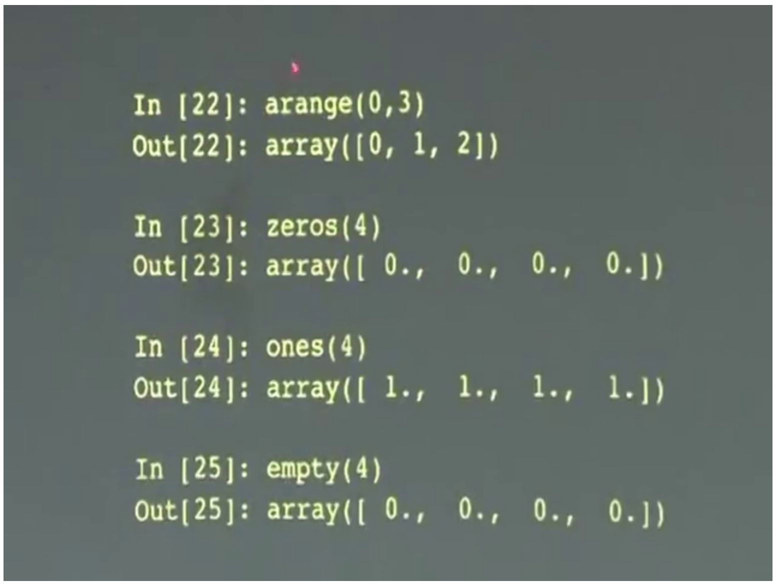


```
In [53]: y
Out[53]: array([1, 2])

In [54]: z
Out[54]: array([4, 5])

In [55]: sqrt(y)
Out[55]: array([ 1. ,  1.41421356])

In [56]: y+z
Out[56]: array([5, 7])

In [57]: y*z
Out[57]: array([ 4, 10])
```

So, we can also do operations of mathematical operations; so it PyLab, Python lab, pi for Python. Python library has square root as a function. So, if I do square root of this, I will get this. It is an array square root will act on each element. Similarly, I can add two arrays y plus z. So, we will add element by element, but what if you do y star z? What will I get? It is not a matrix multiplication or array. It is element by element multiplication.

If I take matrix a and matrix b, I say e star b, it is just going to do element by element multiplication. It is not going to do matrix multiplication. Matrix multiplication you have to write your own, where rather it is there, but it is not a star b. So, one has to be careful. This kind of operation, this is a dot product. If you like it is not a complex number, but it is a dot product. Well, actually it is not a dot product. I apologize. It is not adding things, but you just multiply. It is just multiplication of element by element. That is what you should say it is not a dot product.

(Refer Slide Time: 10:05)

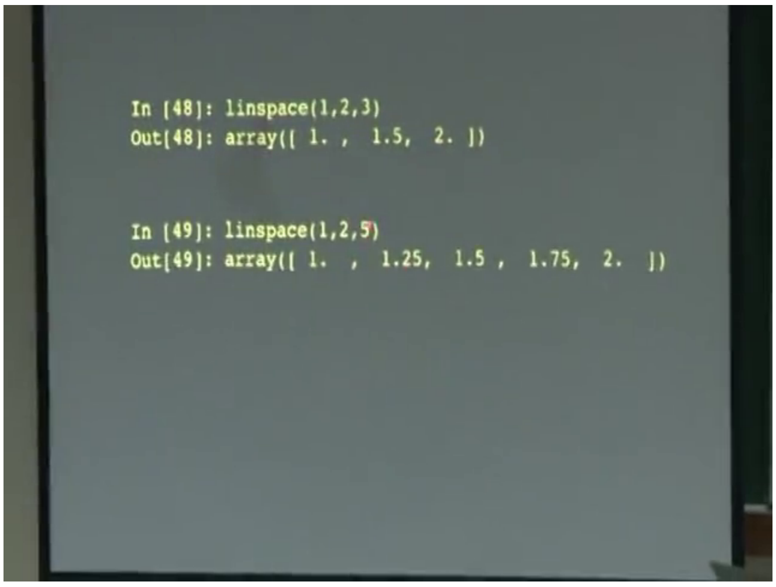


```
In [22]: arange(0,3)
Out[22]: array([0, 1, 2])

In [23]: zeros(4)
Out[23]: array([ 0.,  0.,  0.,  0.])

In [24]: ones(4)
Out[24]: array([ 1.,  1.,  1.,  1.])

In [25]: empty(4)
Out[25]: array([ 0.,  0.,  0.,  0.])
```

These are very useful things, especially when I want to create an array. I do not know the entries. So, I am not going to write like this. So, suppose I know that it is 10 array of 10 elements. So, you say 0 10, right. You just say 0, 10, all entries will be 0 and 0, 10. This is a useful function or you can say 1. Well, I start with 0, 10, 0, 4. So, it will create an array and on that I will operate. We can create an empty array or with arrays with ones, this is a range function which I described earlier. So, 0, 3 will give you equi spaced numbers 1 0 1 2 and not 3. It has no name. If you like, you can just say x equal to. If you say x equal to, then x will get that and it gives the output. It is the same thing.

So, you are saying except we do have act on arrays, it gives an array and not list. So, that gives a list and these gives an array. Sometimes you want arrays, so it give you arrays. There is one more useful function called rand. So, rand will give random numbers between 0 and 1. So, these are random numbers with 4 entries, 4 elements. If I do not put anything, here will give one number, but I found connect generate 100 random numbers. You can just do it.

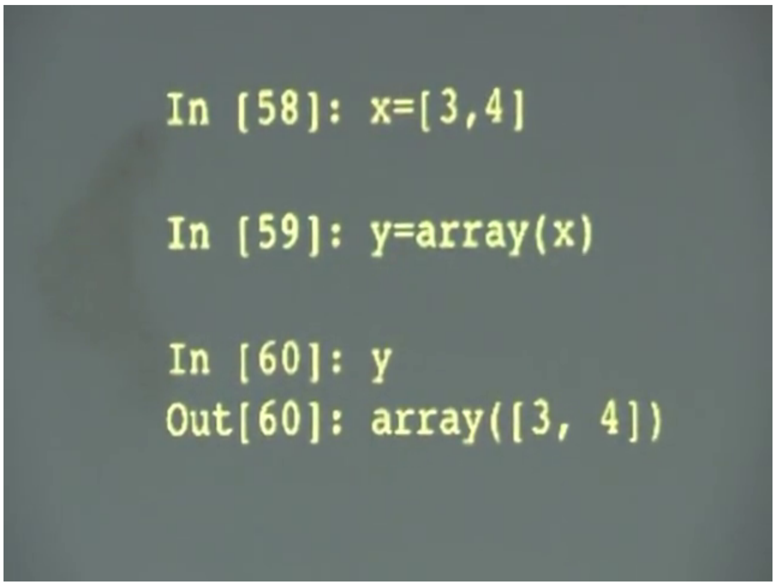
(Refer Slide Time: 11:45)



```
In [48]: linspace(1,2,3)
Out[48]: array([ 1. ,  1.5,  2. ])


In [49]: linspace(1,2,5)
Out[49]: array([ 1.  ,  1.25,  1.5 ,  1.75,  2.  ])
```

Linspace was a bit confusing in the last class. So, linspace if I want linear space points between 1 and 2 including 1 and 2, but three points, then the middle point must be 0.5 1.5. So, this is number of elements in that array 3 and starting and the end n values. So, in one line it just gives in one line. Linspace gives that function. We can also with five elements; it is going to be this.

(Refer Slide Time: 12:12)



```
In [58]: x=[3,4]

In [59]: y=array(x)

In [60]: y
Out[60]: array([3, 4])
```

We can convert a list to an array. So, how do you convert a list to an array? So, this is list, right x equal to 3, 4. It is a list. List will not allow you to do square root. Square root is not possible on a list. List could have string and so on. So, you say well I do not know how to do square root of, I mean it is impossible to do it square root of a list, but I can convert that to numpy array by saying this array x. So, y will be numpy array on which you can do a square root and all the math operations. That is about it.

So, this for numpy arrays which is going to be a key part of our course, all the things which you will be using, you have to be conversant with how to use numpy.