



# IIT Madras

## ONLINE DEGREE

**Programming in Python**  
**Professor Sudarshan Iyengar**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Ropar**  
**Introduction to NumPy library**

(Refer Slide Time: 0:16)

## Introduction to NumPy library

NumPy: Numerical Python  
ndarray: N-dimensional array

Introduction to NumPy library



Hello python students. In this lecture we will briefly introduce one more external library called NumPy. NumPy stands for numerical Python and this library is very popular for scientific computing in Python. The core of this NumPy package is ndarray object. ndarray stands for n dimensional arrays. Many of you might think Python list and this NumPy arrays as same, but that is a misconception.

There are multiple differences between list and arrays. Therefore, in order to use NumPy library effectively we should first understand these differences. Then only we will be able to decide when to use list and when to use arrays.

(Refer Slide Time: 1:14)

Parameters	Python list	NumPy array
Installation and importing	Not required	Required
Type of elements	Heterogenous	Homogenous
Dimension of elements	No restriction	Has to be same
Memory allocation	Non-contiguous	Contiguous
Size	Requires more space	Requires less space
Performance	Slower	Faster
Element wise operations	Not possible	Possible
Functionality	Can not handle arithmetic operations	Can handle arithmetic operations

Introduction to NumPy library



So, these are the differences between Python list and NumPy array. Most of these differences are self-explanatory whereas few requires some additional explanations because of technical complexity involved in it. So, let us go through these differences one by one. First installation and importing. As we know Python list is the core data structure in Python language itself. Hence, we do not require any installation or import statement.

Whereas, NumPy is an external library, hence we have to install this library and then only we can import it and use it just like what we did with Pandas. Second point, type of elements. Python list can store elements of any type in a single list which means we can have integer, character, float, another list, a dictionary, tuple, set, string or any kind of data element inside a single list.

Whereas, for NumPy arrays, there is an restriction; every element in the NumPy array must be of same type, which means if we create a NumPy array we have to use it either only for integers or only for strings and so on. But we cannot mix and match all these together in an single array.

Third point; dimension of elements. There is no restriction on the dimension of elements stored inside list. For example, we are using nested list. The inner list can be of any size. There is no restriction on that. Whereas, with respect to NumPy array, every inner list should be of same size.

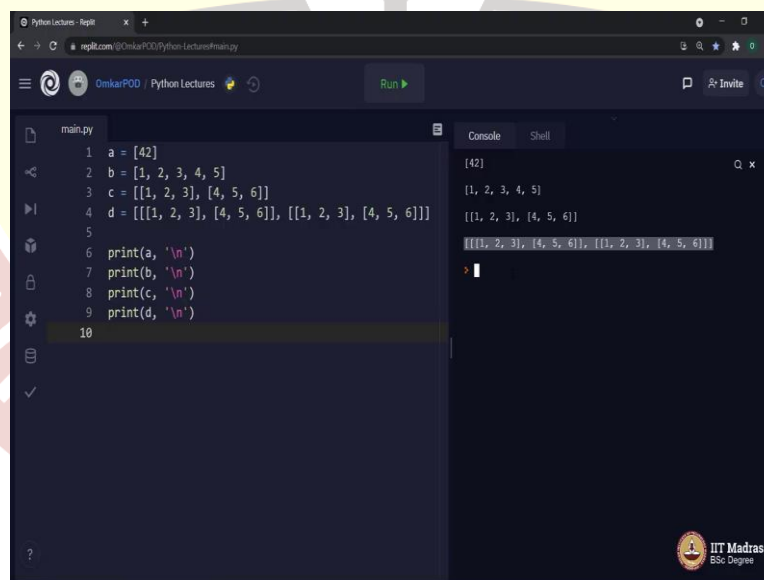
Next point. Memory allocation. Whenever we create list all the elements in the list are stored in memory in non-contiguous manner. Whereas, with respect to NumPy array, every single

element stored in that array is always in contiguous manner in the main memory. Next point; size. As would have seen it earlier, Python list requires more memory as in it requires more bytes of memory to store some n number of elements. On the other hand, in order to store those same n number of elements, NumPy arrays takes less space.

Next parameter is performance and as it says NumPy arrays are faster than Python list. Moving to next parameter; element wise operations. In Python lists, we cannot do element wise operations in a single instruction, instead we have to iterate over the list and access individual element in order to do such operations. Whereas for NumPy it is very much possible to perform element wise operations in single statement.

And then the last difference is related to functionality. Python list cannot handle arithmetic operations. Whereas NumPy arrays can handle arithmetic operations and not just that NumPy array also provides a wide range of inbuilt functions which are very much useful for these arithmetic operations. And because of that NumPy array is very popular with respect to scientific computing in Python. So, now let us move to our Python editor and look at the core concept of NumPy array which is ndarrays.

(Refer Slide Time: 5:41)



```
main.py
1 a = [42]
2 b = [1, 2, 3, 4, 5]
3 c = [[1, 2, 3], [4, 5, 6]]
4 d = [[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]]
5
6 print(a, '\n')
7 print(b, '\n')
8 print(c, '\n')
9 print(d, '\n')
10
```

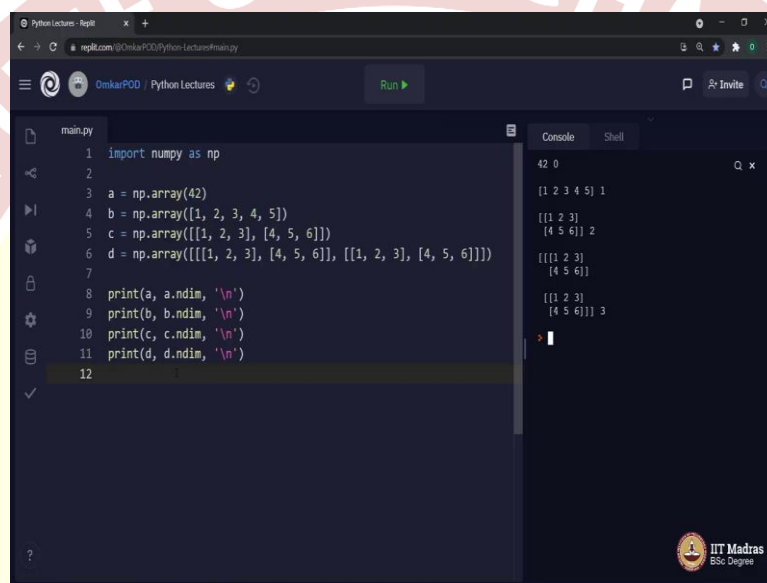
Console

```
[42]
[1, 2, 3, 4, 5]
[[1, 2, 3], [4, 5, 6]]
[[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]]
```

Now, look at this particular code, here we have declared 4 variables, all are of type list, first with single element, second with 5 elements, third is a nested list where we have two inner list stored inside one outer list. So, it is like a matrix stored in a form of list of lists. Whereas, the last variable d is like a three-dimensional matrix where we have 3 levels of nesting with respect to list.

So, let us execute and we got the output as expected and there is nothing new about it. But the point to remember is even though we said that this is a nested list and we are trying to store matrix inside this particular list, still Python list consider this list as a one-dimensional list. Same thing happens even with respect to this list where we are trying to store values in three dimensions. Python is considering the entire thing as a one-dimensional entity with some nesting. Now, let us try to implement same example using NumPy arrays and then we will be able to understand the difference between regular list and ndarrays provided by NumPy.

(Refer Slide Time: 7:22)



The screenshot shows a Python REPL window with the following code in the editor:

```
1 import numpy as np
2
3 a = np.array(42)
4 b = np.array([1, 2, 3, 4, 5])
5 c = np.array([[1, 2, 3], [4, 5, 6]])
6 d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
7
8 print(a, a.ndim, '\n')
9 print(b, b.ndim, '\n')
10 print(c, c.ndim, '\n')
11 print(d, d.ndim, '\n')
12
```

The console output on the right shows the execution results:

```
42 0
[[1 2 3 4 5] 1
[[1 2 3]
 [4 5 6]] 2
[[[1 2 3]
  [4 5 6]]
 [[1 2 3]
  [4 5 6]]] 3
```

Now, look at this particular code, import NumPy as np, then in order to create a NumPy array we will use this particular function np dot array, then we will pass this simple number as a parameter. This will create an array of 0 dimensions. In second case we are creating an array of one dimension. In third case we are trying to creating an array with two dimensions. Whereas, this will be a three-dimensional array.

Let us print the values stored in these four variables along with the dimensions of each particular variable. Let us execute. As you can observe in the output 42 has 0 dimensions because it is just a single number whereas, this is a one-dimensional array, whereas for third variable it says two dimensional because it is a matrix and as you can see it actually considers this as a matrix instead of list of list or in this case array of arrays.

This is the first row of matrix and this will be the second row of matrix and that is the biggest difference between NumPy and list. Fourth variable d is even more interesting because this is a 3d entity. So, we have this as a first matrix and this as a second matrix in that third dimension.

I hope you must have understood the major difference between NumPy and a regular Python list. The idea behind this particular lecture was to make you all aware that there exists some entity called as NumPy which is an external library and it is very much useful with respect to arithmetic computation in Python. I am sure you must have explored this particular library in depth in statistics 2 course and if not, you will do it in upcoming terms.

Beyond that this particular library will be used in Machine Learning and deep learning courses extensively. But with respect to this particular Python course, this library is irrelevant. Hence, we will stop with this brief introduction and allow you to explore it as you go on. Thank you for watching this lecture. Happy learning.

