



Why? - Answer
Proof
+ major functions numpy



Introduction to NumPy

Course : Data Science
Lecture On : NumPy
Instructor : Sumit Shukla

What we will cover in this session?

- 1 Introduction to NumPy library
- 2 Why we NumPy
- 3 How NumPy is more fast than Python Data Structures like list
- 4 Python Demonstration

Introduction to NumPy

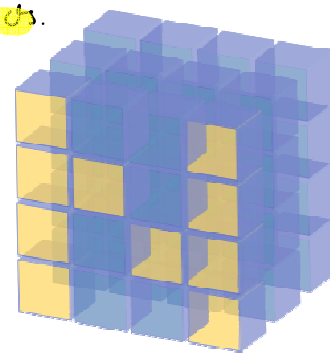
$\left. \begin{matrix} \text{List} \\ \text{tuple} \\ \text{Dicto} \end{matrix} \right\} \text{Python Base Data objects} = \left. \begin{matrix} \text{Slow} \\ + \\ \text{numbers (a lot)} \end{matrix} \right\}$ **upGrad**

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- A powerful N-dimensional array object
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

- * \rightarrow It is faster than python base data objects.
- $\checkmark \rightarrow$ faster execution
- \rightarrow useful mathematical functions.
- \rightarrow N-D Data data
- * \rightarrow It consumes less memory.



array
 \downarrow
NumPy

Why we need NumPy

Constraints

- Performance - they are faster than lists
- Need - Multidimensional array
- Functionality - NumPy have optimized functions such as linear algebra operations built in.

→ It can hold ~~data~~ data of one kind (homogeneous)
→ You need to define the length, data type ~~and~~
if you wanted to create an empty
numpy array.
→ Immutable

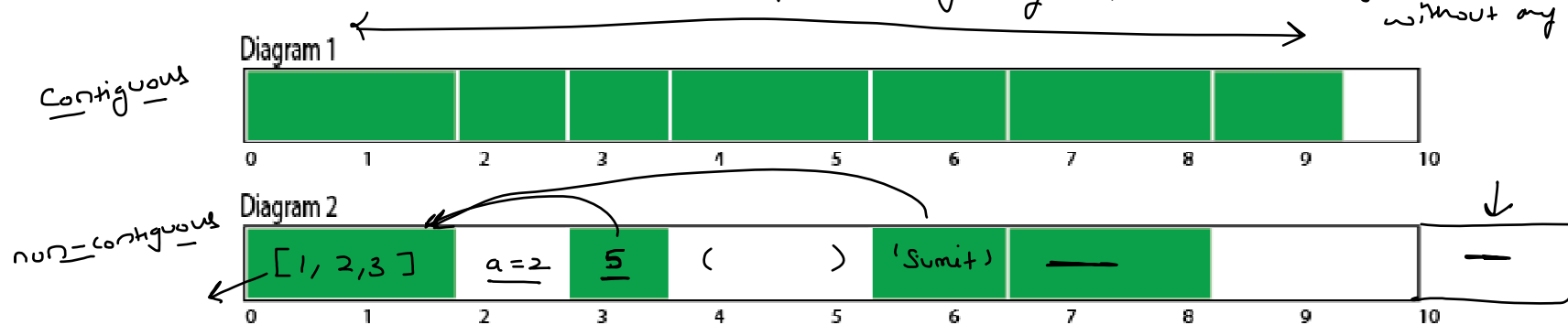
NumPy arrays are made to be created as homogeneous arrays, considering the mathematical operations that can be performed on them. It would not be possible with heterogeneous data sets.

How NumPy is faster than other Data Structures?

memory $\xrightarrow{\text{a = np.array()}}$

1. An array is a collection of homogeneous data-types which are stored in contiguous memory locations, on the other hand, a list in Python is collection of heterogeneous data types stored in non-contiguous memory locations.

Complete numpy array is saved in a memory address which is without any gaps



$\underline{l} = [1, 2, 3]$

$a = 2$

lists \rightarrow mutable

$\underline{l}.append(5)$

\rightarrow as we will add new elements to the list, the next available memory block will be assigned to the new element

How NumPy is faster than other Data Structures?

1. While declaring a NumPy array, it is necessary to describe the shape and the dtypes.
2. But this is not true in the case of the lists. While creating a list, you need not to specify how many elements you will append to your list nor do you have to define a data type. This is because the data in a lists is stored in a form of a pointers which describes the reference to the memory location. So, once you add a new data to Python lists, the lists will first create it's pointer and then it will save the data into the available memory location.
3. When you retrieve the first element in your list, python is taking two steps: First, retrieve the pointer. Second, go to the memory location of the pointer to finally get the object you want.

How NumPy is faster than other Data Structures?

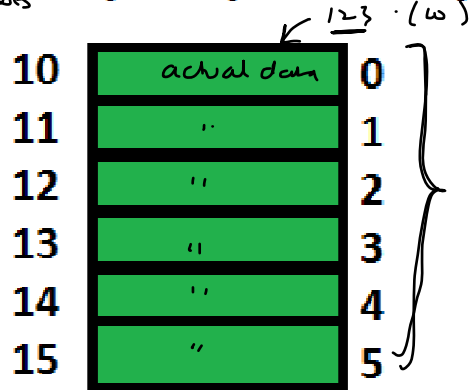
Base Python **upGrad**

- How many elements are there
- No new element is going to come
- the data type of all elements

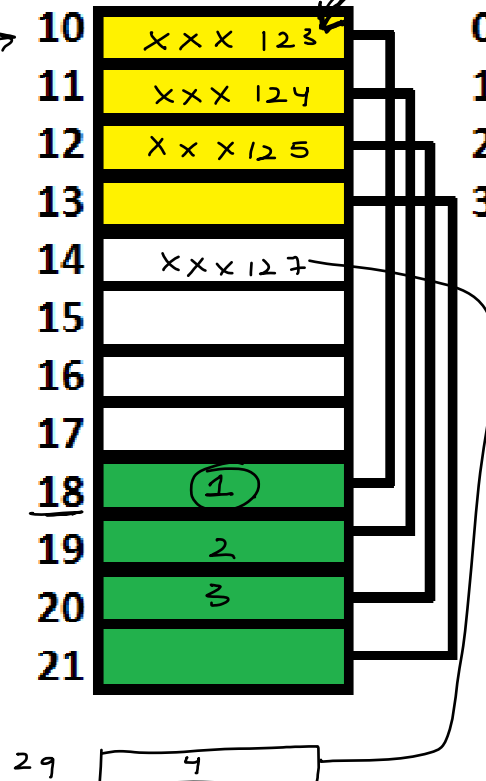
NumPy Array in memory

NumPy does not have any pointers ✓

It consumes less memory & They are faster.



Lists in memory



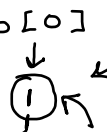
$l = [1, 2, 3]$

- Python will assign a pointer to each element of your list
- Python needs pointers to differentiate one element from another
- when we save a list into memory.
 - elements of list
 - pointers

when we try to fetch an element from list
 $l[0]$ → return
 Refer to the pointer → actual element

$b = np.array([1, 2, 3, 4])$

$b[0]$ → Python will assign a pointer on the fly



To recognise this element Python needs some ID.



→ b[0]

$b[3]$

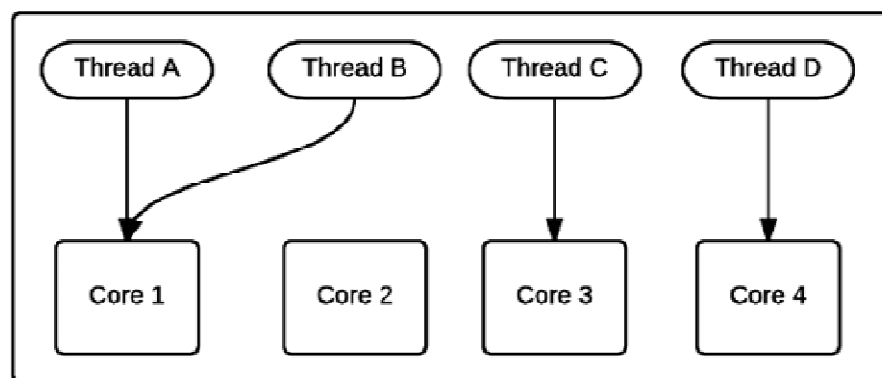
29

4

How NumPy is faster than other Data Structures?

Apart from a least time and space complexity, we have more advantages using NumPy arrays:

2. The NumPy package breakdowns a task into multiple fragments, and then processes all the fragments parallelly.
3. The NumPy package integrates C, C++ and Fortran codes in Python, these programming languages have very less execution time as compared to python.



$$\text{np-2d} [0:2, 0:2.]$$

slice

$\text{np-2d} = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

$\text{np-2d} [0:0]$

element

$\text{np-2d} [\text{---} , \text{---}]$

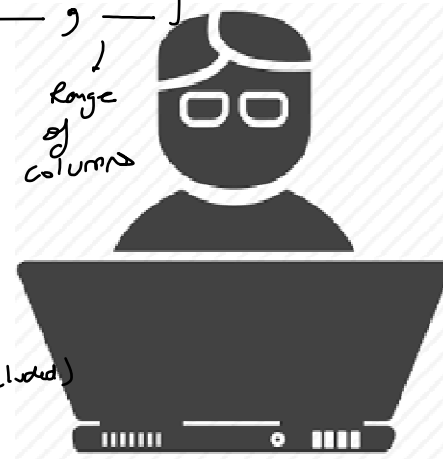
Range of
rows

Range
of
columns

a; b

start
index
(included)

end
index
(excluded)



Let's Code

$\text{np-2d} [0:1, 0:1]$

\downarrow
 $\text{np-2d} [0, 0]$

\downarrow
 1



Thank You!