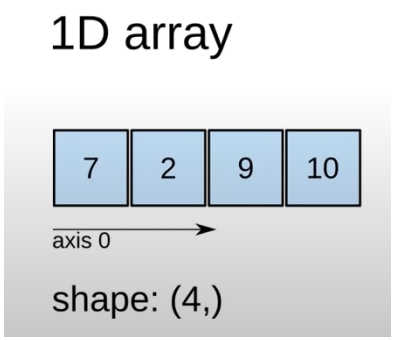
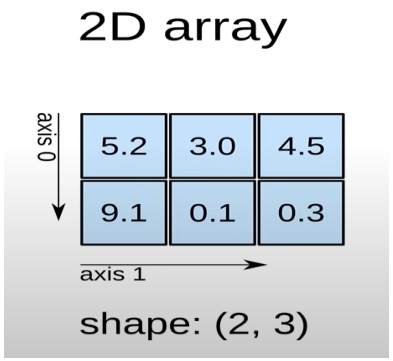
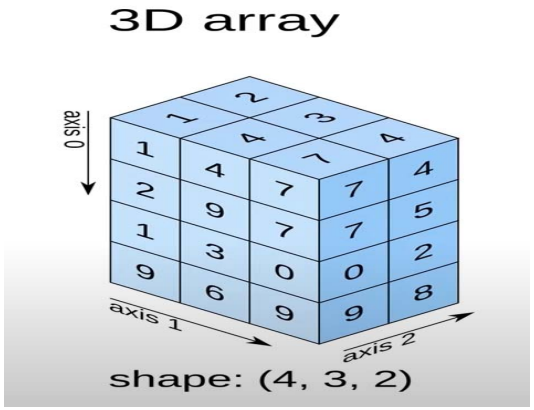
Numpy Libraries

- It is for large, **multi-dimensional array** and **matrix processing**.
- It is very useful for fundamental scientific computations in Machine Learning.





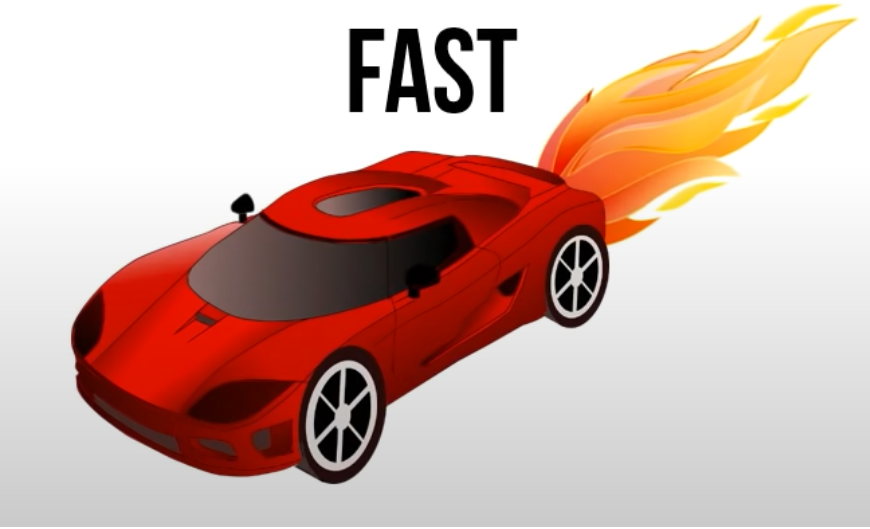


Dhirubhai Ambani University
Formerly known as DA-IICT

How are Lists different from Numpy

Lists

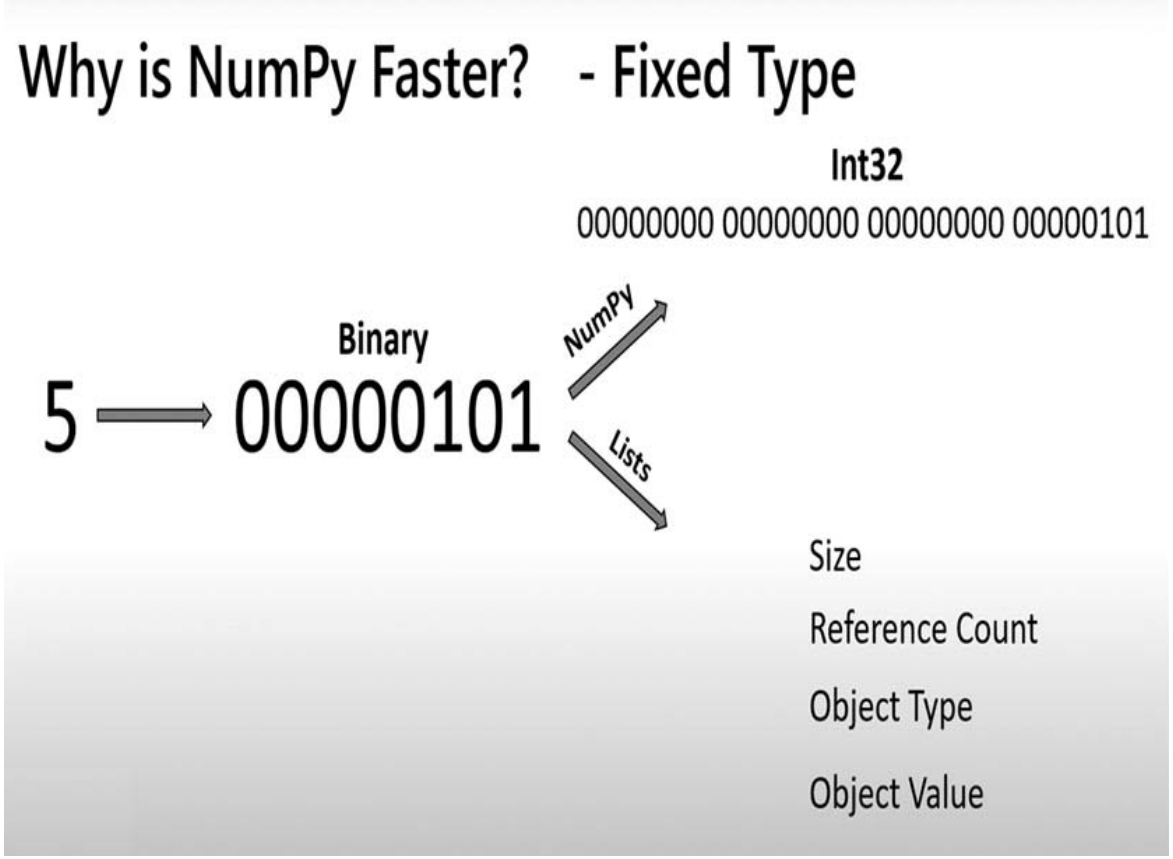
NumPy



Speed

Dhirubhai Ambani University
Formerly known as DA-IICT

How are Lists different from Numpy



Numpy:

- NumPy uses a fixed type, e.g., int32.
- That means the number is stored in exactly 32 bits (4 bytes).
- So 5 becomes:

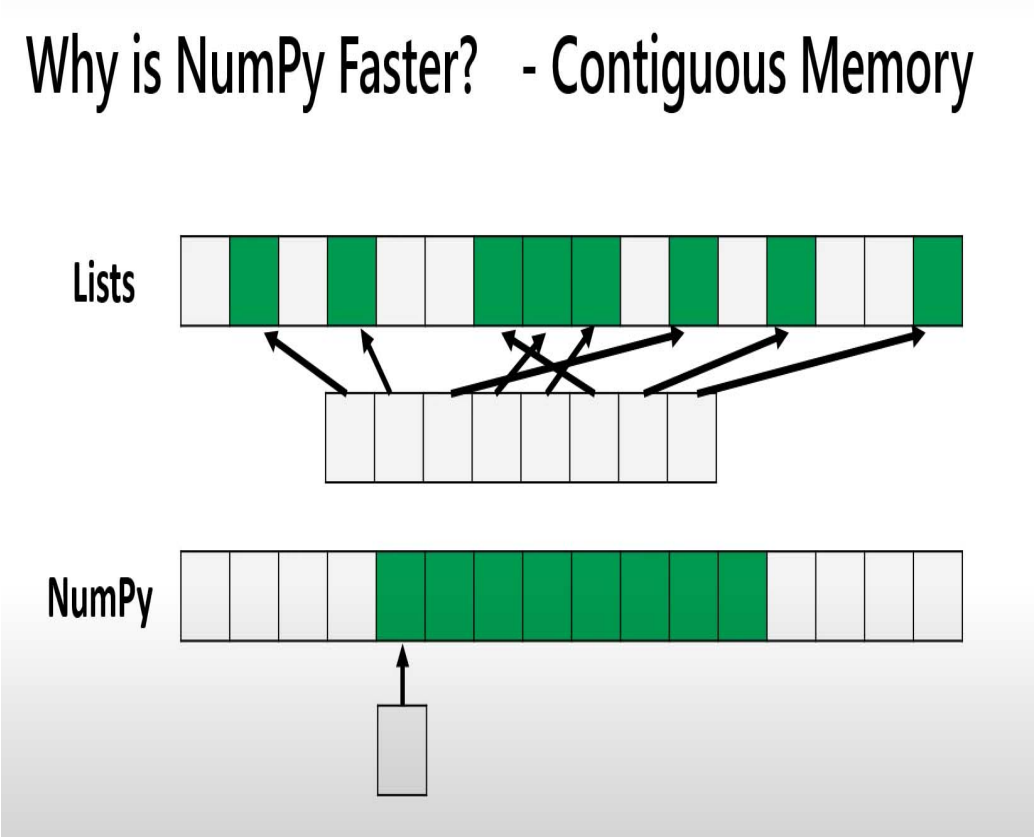
  00000000 00000000 00000000 00000101
- Stored directly in memory as raw binary.

List:

- Python lists don't store raw numbers, they store references to Python objects.
- Each integer in Python is a full object with: Size, Reference count, Object type, Object value.
- So instead of just 00000101, Python needs to keep extra metadata for every element.

Dhirubhai Ambani University
Formerly known as DA-IICT

How are Lists different from Numpy



List:

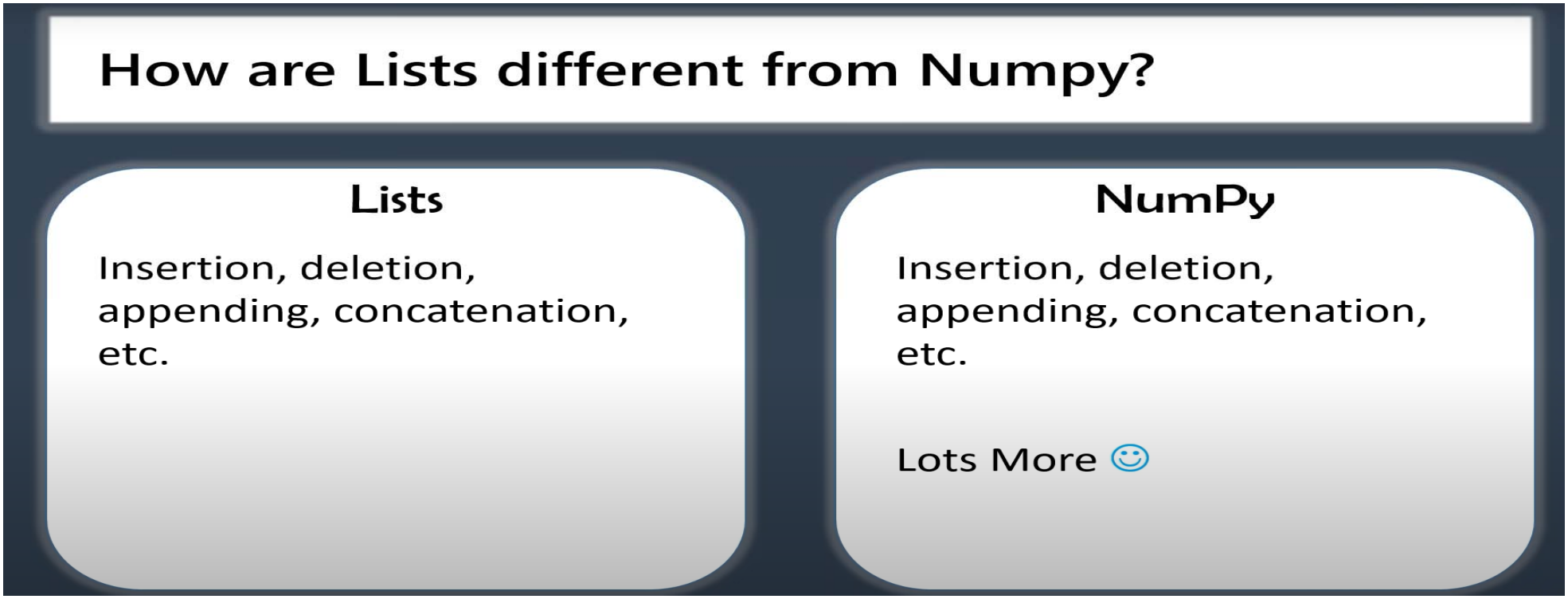
- A Python list is essentially an array of pointers.
- Each list element doesn't store the raw data directly; instead, it stores a reference (pointer) to an object somewhere else in memory.
- This means:
  - Data is not contiguous (scattered across memory).
  - Accessing each element requires following a pointer.

Numpy:

- A NumPy array stores all values as raw binary data in contiguous memory.
- In this diagram, you can see that only one pointer is needed.
- This means:
  - Data is contiguous (tightly packed).
  - The CPU can load and process data much faster.

Dhirubhai Ambani University
Formerly known as DA-IICT

How are Lists different from Numpy



How are Lists different from Numpy

How are Lists different from Numpy?

Lists

a = [1,3,5]
b = [1,2,3]

*a*b = ERROR*

NumPy

a = np.array([1,3,5])
b = np.array([1,2,3])

*a*b = np.array([1,6,15])*

Dhirubhai Ambani University
Formerly known as DA-IICT

How are Lists different from Numpy

Applications of NumPy?

Mathematics (MATLAB Replacement)

Plotting (Matplotlib)

Backend (Pandas, Connect 4, Digital Photography)

Machine Learning

Numpy

Install, import, and upgrade

```
# pip install
!pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\administrator\anaconda3\lib\site-packages (2.3.3)
```

```
import numpy  #Import
import numpy as np #standard in python community
```

```
numpy.__version__
```

```
'2.3.3'
```

```
pip install --upgrade numpy
```

```
Requirement already satisfied: numpy in c:\users\administrator\anaconda3\lib\site-packages (2.3.3)Note: you may need to restar
```

Numpy

Initializing Array

One-dimensional Array

```
a=np.array([1,2,3])
a
```

```
array([1, 2, 3])
```

Two-dimensional Array

```
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
a
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

Three-dimensional Array

```
# 3-D example
a=np.array([[[0,1],[2,3]],[[4,5],[6,7]],[[8,9],[10,11]]])
a
```

```
array([[[ 0,  1],
        [ 2,  3]],

       [[ 4,  5],
        [ 6,  7]],

       [[ 8,  9],
        [10, 11]]])
```

Dhirubhai Ambani University
Formerly known as DA-IICT

Why NumPy is Better for Mathematical Computations

```
a = [[1, 3, 5],
     [4, 5, 6]]

b = [[2, 4, 6],
     [7, 8, 9]]

c = a * b   # ❌ This does NOT do element-wise multiplication
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[8], line 7
      1 a = [[1, 3, 5],
      2      [4, 5, 6]]
      4 b = [[2, 4, 6],
      5      [7, 8, 9]]
----> 7 c = a * b

TypeError: can't multiply sequence by non-int of type 'list'
```

```
import numpy as np

a = np.array([[1, 3, 5],
              [4, 5, 6]])

b = np.array([[2, 4, 6],
              [7, 8, 9]])

c = a * b
print(c)
```

```
[[ 2 12 30]
 [28 40 54]]
```

Dhirubhai Ambani University
Formerly known as DA-IICT

Numpy

```
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
a
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

Dimension

```
#Get the dimension (1D array/2D array.....)
a.ndim
```

```
2
```

Shape

```
#Get the shape
a.shape
```

```
(3, 3)
```

Datatype

type of elements stored in a NumPy array

```
#Get type
a.dtype
```

```
dtype('int64')
```

Itemsize

(number of bytes used to store each element in the array)

```
a.itemsize # information of byte
```

```
8
```

```
a= np.array([1,2,2.3])
a
```

```
array([1. , 2. , 2.3])
```

```
a.itemsize # information of byte
```

```
8
```

```
a.dtype
```

```
dtype('float64')
```

```
a=np.array([1,"Ankit",2.3])
a.dtype # Unicode
```

```
dtype('<U32')
```

```
a.itemsize
#In Unicode,every character is stored in exactly 4 bytes
```

```
128
```

 # string type -not exact length- always bigger length

Dhirubhai Ambani University
Formerly known as DA-IICT

Numpy

Assessing the elements. rows, columns, etc

```
a=np.array([[1,2,3,11,21,23,25,36],[4,5,6,7,8,9,10,0]])
a
```

```
array([[ 1,  2,  3, 11, 21, 23, 25, 36],
       [ 4,  5,  6,  7,  8,  9, 10,  0]])
```

Get a specific element

```
#Get a specific element [r,c]
print(a[0,1]) # a[r,c]
```

```
2
```

Get a specific row

```
#Get a specific row
print(a[0])
```

```
[1 2 3]
```

Get a specific Column

```
print(a[:,1:2]) # Get a specific Column
```

```
[[2]
 [5]]
```

- **print(a[0,0:2])**

```
print(a[0,0:2])
```

```
[1 2]
```

- **print(a[:,1:6:2])** # start:stop:step

```
print(a[:,1:6:2])
```

```
[[ 2 11 23]
 [ 5  7  9]]
```

- **print(a[:,6:1:-1])**

```
print(a[:,6:1:-1])
```

```
[[25 23 21 11  3]
 [10  9  8  7  6]]
```

Numpy

3-dimensional array

```
# 3-D example
a=np.array([[[0,1],[2,3]],[[4,5],[6,7]],[[8,9],[10,11]]])
a
```

```
array([[[ 0,  1],
        [ 2,  3]],

       [[ 4,  5],
        [ 6,  7]],

       [[ 8,  9],
        [10, 11]]])
```

- **print(a[1,0,1]) # get a specific element**

```
#Get a specific element
print(a[1,0,1])
```

```
5
```

- **print(a[0])**

```
print(a[0])
```

```
[[0 1]
 [2 3]]
```

- **print(a[0,1,:])**

```
print(a[0,1,:])
```

```
[2 3]
```

- **print(a[:,1,:])**

```
print(a[:,1,:])
```

```
[[ 2  3]
 [ 6  7]
 [10 11]]
```

Dhirubhai Ambani University
Formerly known as DA-IICT

Numpy

Null Matrix

```
# null Matrix
a=np.zeros(5)
print(a)
```

```
[0. 0. 0. 0. 0.]
```

```
np.zeros((2,5))
```

```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

```
np.zeros(((3,2,5)))
```

```
array([[[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]],

       [[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]],

       [[0., 0., 0., 0., 0.],
        [0., 0., 0., 0., 0.]]])
```

Full Matrix

```
#Any other number
a=np.full((2,2),1)
a
```

```
array([[1, 1],
       [1, 1]])
```

```
np.full_like(a,3)
```

```
array([[3, 3],
       [3, 3]])
```

```
np.full(a.shape,5)
```

```
array([[5, 5],
       [5, 5]])
```

```
np.ones((2,2))
```

```
array([[1., 1.],
       [1., 1.]])
```

Identity Matrix

```
#identity matrix
np.identity(2)
```

```
array([[1., 0.],
       [0., 1.]])
```

```
#identity matrix
matrix = np.eye(2)
print(matrix)
```

```
[[1. 0.]
 [0. 1.]]
```

Dhirubhai Ambani University
Formerly known as DA-IICT

Numpy

Create an array with random numbers (between 0 and 1)

```
#create a array with random number (between 0 and 1)
np.random.rand(4)
#np.random.rand(4,2)
#np.random.rand(4,2,3)
```

```
array([0.80068123, 0.12780885, 0.97459564, 0.5483972 ])
```

```
#create a array with random number (between 0 and 1)
#np.random.rand(4)
np.random.rand(4,2)
#np.random.rand(4,2,3)
```

```
array([[0.75973859, 0.95567886],
       [0.03436674, 0.89176753],
       [0.24576945, 0.99682463],
       [0.29832416, 0.57195902]])
```

```
#create a array with random number (between 0 and 1)
#np.random.rand(4)
#np.random.rand(4,2)
np.random.rand(4,2,3)
```

```
array([[[0.42567349, 0.5836056 , 0.74776345],
        [0.99171107, 0.00296539, 0.98263174]],

       [[0.38910825, 0.44418267, 0.8932749 ],
        [0.04631678, 0.29927347, 0.36632794]],

       [[0.9390877 , 0.75264851, 0.50588025],
        [0.68145279, 0.02544418, 0.060096  ]],

       [[0.23270351, 0.47428333, 0.98592488],
        [0.4426028 , 0.8688685 , 0.86829031]]])
```

Dhirubhai Ambani University
Formerly known as DA-IICT

Numpy

Create a random number in a given range (eg, between 0 and 50)

```
# create random number between 0 t0 50
np.random.rand(4) *50
```

```
array([37.05448718, 46.92889767, 27.54244767, 37.11042244])
```

```
#Random Integer value
np.random.randint(10,size=(3,3)) #between 0 to 9
```

```
array([[4, 1, 2],
       [5, 6, 5],
       [2, 4, 9]], dtype=int32)
```

```
np.random.randint(5,10,size=(3,3)) #between 5 to 9
```

```
array([[9, 6, 8],
       [7, 7, 5],
       [5, 5, 5]], dtype=int32)
```

Numpy

Concatenate: Joining the arrays (existing axis)

```
arr_1=np.array([[1,2,3],[4,5,6]])
arr_2=np.array([[7,8,9],[10,11,12]])
```

```
arr_1
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
arr_2
```

```
array([[ 7,  8,  9],
       [10, 11, 12]])
```

Joint vertically rows

```
arr= np.concatenate([arr_1,arr_2]) # axis =0
```

```
arr
```

```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 11, 12]])
```

Joint the horizontal columns

```
arr= np.concatenate((arr_1,arr_2),axis=1) #axis=1
arr
```

```
array([[ 1,  2,  3,  7,  8,  9],
       [ 4,  5,  6, 10, 11, 12]])
```

Dhirubhai Ambani University
Formerly known as DA-IICT

Numpy

Stack: **Joining the arrays along a new axis (Concatenate, merge along an existing axis)**

```
arr_1=np.array([[1,2,3],[4,5,6]])
arr_2=np.array([[7,8,9],[10,11,12]])
```

```
arr_1
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
arr_2
```

```
array([[ 7,  8,  9],
       [10, 11, 12]])
```

```
arr=np.stack((arr_1,arr_2),axis=0)
print(arr)
```

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

 [[ 7  8  9]
  [10 11 12]]]
```

Explore: hstack, vstack, dstack

Dhirubhai Ambani University
Formerly known as DA-IICT

Splitting

np.split()	:	Split array into equal parts (requires size divisible)
np.array_split()	:	Split array into nearly equal parts (works for unequal sizes)
np.hsplit()	:	Split horizontally (columns for 2D arrays)
np.vsplit()	:	Split vertically (rows for 2D arrays)
np.dsplit()	:	Split along the third axis (depth for 3D arrays)

Numpy

np.split()

```
arr=np.array([1,2,3,4,5,6,7,8])
```

```
new_arr=np.split(arr,2)
print(new_arr)
```

```
[array([1, 2, 3, 4]), array([5, 6, 7, 8])]
```

np.array_split()

```
arr=np.array([1,2,3,4,5,6,7,8,9])
new_arr=np.array_split(arr,2)
print(new_arr)
```

```
[array([1, 2, 3, 4, 5]), array([6, 7, 8, 9])]
```

np.hsplit()

```
arr=np.array([[1,2,3,4],[5,6,7,8]])
new_arr=np.hsplit(arr,2)
print(new_arr)
```

```
[array([[1, 2],
       [5, 6]]), array([[3, 4],
       [7, 8]])]
```

np.vsplit()

```
arr=np.array([[1,2,3,4],[5,6,7,8]])
new_arr=np.vsplit(arr,2)
print(new_arr)
```

```
[array([[1, 2, 3, 4]]), array([[5, 6, 7, 8]])]
```

np.dsplit()

```
arr=np.array([[[0,1],[2,3]],[[4,5],[6,7]],[[8,9],[10,11]]])
print(new_arr)
new_arr=np.dsplit(arr,2)
print(new_arr)
```

```
[[[ 0  1]
  [ 2  3]]

 [[ 4  5]
  [ 6  7]]

 [[ 8  9]
  [10 11]]]
[array([[[ 0],
         [ 2]],

        [[ 4],
         [ 6]],

        [[ 8],
         [10]]]), array([[[ 1],
         [ 3]],

        [[ 5],
         [ 7]],

        [[ 9],
         [11]]])]
```

Searching in Numpy Arrays

```
arr=np.array([[1,2,3,4],[5,6,7,8]])
x=np.where(arr== 4)
print(x)
```

```
(array([0]), array([3]))
```

```
arr=np.array([[1,2,3,4],[5,4,7,8]])
x=np.where(arr==4)
print(x)
```

```
(array([0, 1]), array([3, 1]))
```

```
arr=np.array([[1,2,3,4],[5,6,7,8]])
x=np.where((arr==4) | (arr==7))
print(x)
```

```
(array([0, 1]), array([3, 2]))
```

Numpy

Sorting

```
arr=[3,6,4,1,7,9,8,3]
print(np.sort(arr))   #assending order
```

```
[1 3 3 4 6 7 8 9]
```

```
arr_sort=np.sort(arr)
print(arr_sort[::-1]) # reverse order
```

```
[9 8 7 6 4 3 3 1]
```

```
print(np.flip(arr)) # flip
```

```
[3 8 9 7 1 4 6 3]
```

```
arr=np.array(["cat", "bananan", "dog", "apple"])
print(np.sort(arr))
```

```
['apple' 'bananan' 'cat' 'dog']
```

```
arr=np.array(["cat", "bananan", "Dog", "apple"])
print(np.sort(arr))              # Case senitive
```

```
['Dog' 'apple' 'bananan' 'cat']
```

Dhirubhai Ambani University
Formerly known as DA-IICT

Numpy

Arithmetic Operation with Numpy

```
arr_1=np.array([1,2,3,4,5,6,7])
arr_2= np.array([8,9,10,11,12,13,14])
```

```
arr_1
```

```
array([1, 2, 3, 4, 5, 6, 7])
```

```
arr_2
```

```
array([ 8,  9, 10, 11, 12, 13, 14])
```

```
new_arr=np.add(arr_1,arr_2)
print(new_arr)
```

```
[ 9 11 13 15 17 19 21]
```

```
new_arr=np.subtract(arr_2,arr_1)
print(new_arr)
```

```
[7 7 7 7 7 7 7]
```

```
new_arr=np.multiply(arr_2,arr_1)
print(new_arr)
```

```
[ 8 18 30 44 60 78 98]
```

```
new_arr=np.divide(arr_2,arr_1)
print(new_arr)
```

```
[8.         4.5        3.33333333 2.75       2.4        2.16666667
 2.        ]
```

```
new_arr=np.power(arr_2,arr_1)
print(new_arr)
```

```
[        8        81      1000     14641    248832   4826809 105413504]
```

```
new_arr=np.power(arr_2,2)
print(new_arr)
```

```
[ 64  81 100 121 144 169 196]
```

Numpy

Arithmetic Operation with Numpy

```
new_arr=np.mod(arr_2,arr_1)
print(new_arr)
```

```
[0 1 1 3 2 1 0]
```

```
new_arr=np.remainder(arr_2,arr_1)
print(new_arr)
```

```
[0 1 1 3 2 1 0]
```

```
arr=np.array([-1,4,-5,7,-9])
print(np.absolute(arr))
```

```
[1 4 5 7 9]
```

```
new_arr=np.divide(arr_2,arr_1)            #Rounding
new_arr_1=np.around(new_arr,2)
print(new_arr,new_arr_1)
```

```
[8.         4.5        3.33333333 2.75       2.4        2.16666667
 2.        ] [8.   4.5  3.33 2.75 2.4  2.17 2.  ]
```

```
arr_1=np.array([1,2,3,4,5,6,7])
arr_2= np.array([8,9,10,11,12,13,14])

print(np.sum([arr_1]))
```

```
28
```

```
print(np.sum([arr_1,arr_2]))    #Sum
```

```
105
```

```
print(np.prod([arr_1]))    # product
```

```
5040
```

and many more …