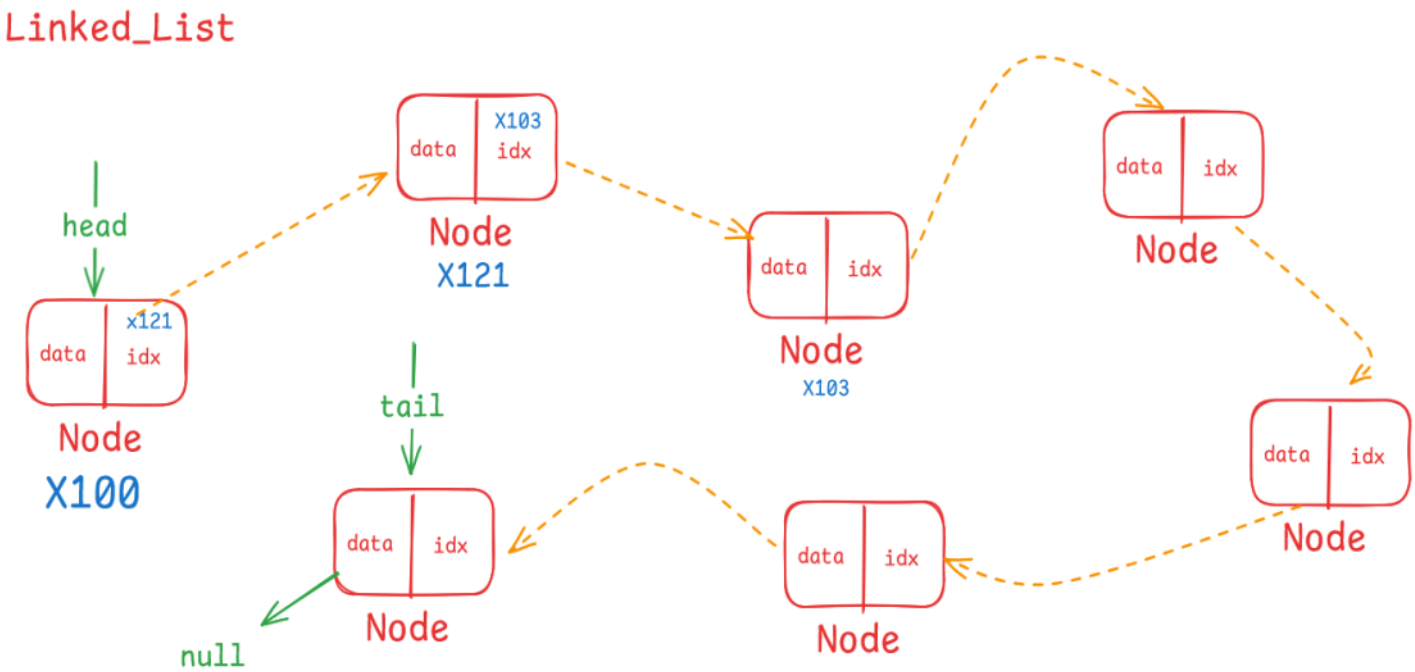
NumPy-I

Session Objectives:

- Understand what NumPy is and why it's important
- Understand what an array is and how it differs from lists/tuples
- Create NumPy arrays using various methods
- Explore the attributes of NumPy arrays
- Apply indexing and slicing on arrays

Numpy - np.array

Contiguous in Nature



What is Numpy?

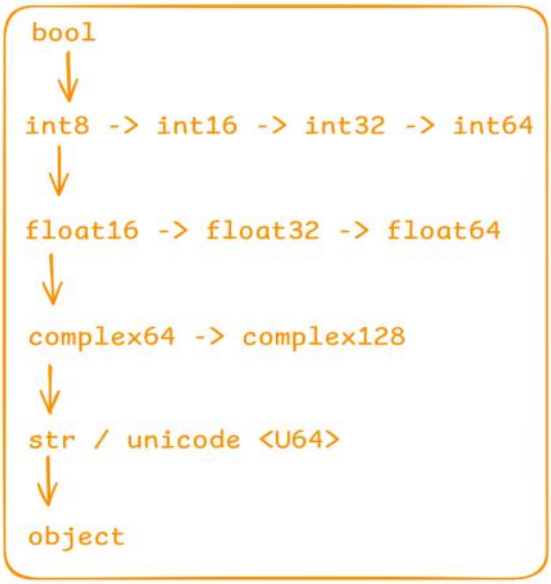
Numpy (Numerical Python) is a core library for scientific and numerical computing in Python.

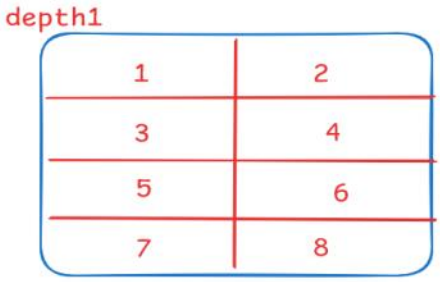
Why use Numpy?

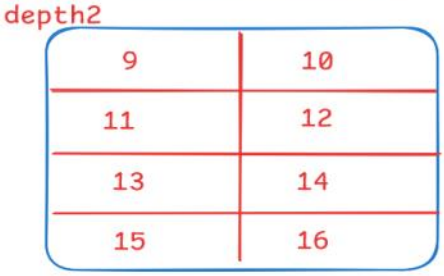
- Fast : Works with Large, multi Dimensional Arrays stored in continuguous memory blocks
- Effecient : Operations are optimized via 'C' under the hood.
- Foundational : Powered with many Scientific Libraries (eg. Pandas, Scipy , Tensorflow)

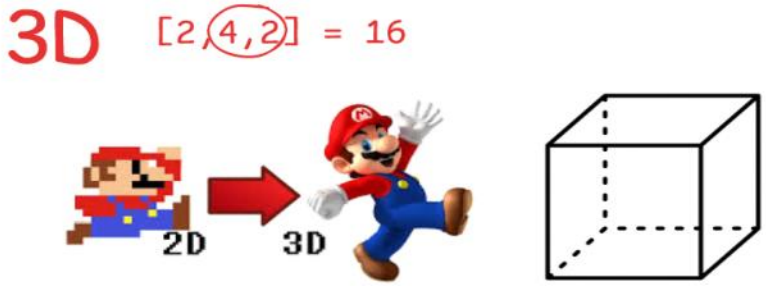
Numpy Also:

- offers the statistical tools like - mean, median, std, variance , etc.
- Intergrates well with Visualization Libraries Like - Matplotlib.



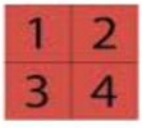






Vector

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



Matrix

```
np.array([[1, 2], [3, 4]])
```

3D Matrix

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

```
# Install
conda install numpy
OR
pip install numpy
```

```
import numpy as np
dir(np)
```

```
['ALLOW_THREADS',
 'BUFSIZE',
 'CLIP',
 'DataSource',
 'ERR_CALL',
 'ERR_DEFAULT',
 'ERR_IGNORE',
 'ERR_LOG',
 'ERR_PRINT',
 'ERR_RAISE',
 'ERR_WARN',
 'FLOATING_POINT_SUPPORT',
 'FPE_DIVIDEBYZERO',
 'FPE_INVALID',
 'FPE_OVERFLOW',
 'FPE_UNDERFLOW',
```

```
num = np.array([11,22,33,44,55,66,77,88,99])
print(num)
```

```
[11 22 33 44 55 66 77 88 99]
```

```
# Indexation starts with 0
num[5] # 66
```

```
66
```

```
num[-1] #99
```

```
99
```

```
num[-3] # 77
```

```
77
```

```
arr_from_list = np.array([11,22,33,44,55,66,77,88,99])
arr_from_tuple = np.array((11,22,33,44,55,66,77,88,99))
print(arr_from_list)
print(arr_from_tuple)
```

```
[11 22 33 44 55 66 77 88 99]
[11 22 33 44 55 66 77 88 99]
```

```
# Common Mistake : 9 arguments can't be fetched w.r.t 1 parameter -> we always have to pass iterable
arr_from_tuple = np.array(11,22,33,44,55,66,77,88,99)
# TypeError: array() takes from 1 to 2 positional arguments but 9 were given
```

```
arr_from_tuple = np.array(11)
arr_from_tuple
```

```
array(11)
```

```
print(type(arr_from_list))
print(type(arr_from_tuple))
```

```
<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

```
arr_from_list.ndim
```

```
1
```

```
# 1D Array
arr_1d = np.array([11,121.9,True,'k','Coding',11+5j,99]) # mixed dtypes
print(arr_1d)
```

```
['11' '121.9' 'True' 'k' 'Coding' '(11+5j)' '99']
```

```
type(arr_1d)
```

```
numpy.ndarray
```

```
print(type(arr_1d))
```

```
<class 'numpy.ndarray'>
```

```
arr_1d.dtype
```

```
dtype('<U64')
```

```
arr_1d.ndim
```

```
1
```

```
# 2D Array [2 Dimensional] [rows, cols] (n*m) (5,3)
arr_2d = np.array([
   [11,22,33] ,
   ['a','b','c'] ,
   [True,False,True] ,
   ['learn','python','programming'] ,
   [9.99,11.11,77.77]
]) # Mixed dtypes [maintaining 15 elements] [rows * cols] = 5*3 = 15
print(arr_2d)
```

```
[['11' '22' '33']
 ['a' 'b' 'c']
 ['True' 'False' 'True']
 ['learn' 'python' 'programming']
 ['9.99' '11.11' '77.77']]
```

```
print(type(arr_2d))
```

```
<class 'numpy.ndarray'>
```

```
arr_2d.dtype
```

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

```
arr_2d.ndim # 2
```

```
2
```

```
# 1D Array -> U32/U64 -> to the max length of a string
arr_1d = np.array([11,121.9,True,'k','Coding',11+5j,99,
                   'Awek$#@##@lkdsahbk2972ry3oFWuighaslk7238rghjv,cadbvdsck.hvalgbvsablviabha,dsbv12']) # mixed
print(arr_1d)
```

```
['11' '121.9' 'True' 'k' 'Coding' '(11+5j)' '99'
 'Awek$#@##@lkdsahbk2972ry3oFWuighaslk7238rghjv,cadbvdsck.hvalgbvsablviabha,dsbv12']
```

```
arr_1d.dtype
```

```
dtype('<U80')
```

```
arr_1d = np.array(['Python Programming',
                   'Today, We are learning Numpy Array']) # mixed dtypes
print(arr_1d)
```

```
['Python Programming' 'Today, We are learning Numpy Array']
```

```
arr_1d.dtype
```

```
dtype('<U34')
```

```
arr_1d = np.array(['Python Programming',99,77]) # mixed dtypes
print(arr_1d)
```

```
['Python Programming' '99' '77']
```

```
arr_1d.dtype
```

```
dtype('<U18')
```

```
# 3D Array [3 Dimensional] [depth ,rows, cols] (d*n*m)  = Total Number of Elements
# 3 depth , 3 rows , 3 columns
arr_3d = np.array([
    [[1,2,3],
    [2,4,6],
    [1,3,5]],

    [['a','b','c'],
    ['d','e','f'],
    ['g','h','i']],

    [[True,True,True],
    [True,False,True],
    [False,False,False]]
])
print(arr_3d)
```

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

 [['a' 'b' 'c']
  ['d' 'e' 'f']
  ['g' 'h' 'i']]

 [['True' 'True' 'True']
  ['True' 'False' 'True']
  ['False' 'False' 'False']]]
```

```
type(arr_3d)
```

```
numpy.ndarray
```

```
arr_3d.dtype
```

```
dtype('<U11')
```

```
arr_3d.ndim
```

```
3
```

```
# 27 = 3*3*3
arr_1d = np.array([1,2,3,4,5,6,7,8,9,9,8,7,6,5,4,3,2,1,1,2,3,4,5,6,7,8,9])
print(arr_1d)
```

```
[1 2 3 4 5 6 7 8 9 9 8 7 6 5 4 3 2 1 1 2 3 4 5 6 7 8 9]
```

```
# 3D reshape (depth,rows,cols) - (3,3,3)
arr_1d.reshape(3,3,3)
```

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

       [[9, 8, 7],
        [6, 5, 4],
        [3, 2, 1]],

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

```
# Specifying Data Types [Upcasting]
nested_list = np.array([
    [11,22],
    [33,44],
    [55,66],
    [77,88],
    [99,99.99]
])
print(nested_list)
```

```
[[11.   22.  ]
 [33.   44.  ]
 [55.   66.  ]
 [77.   88.  ]
 [99.   99.99]]
```

```
 [99.   99.99]]
```

```
nested_list = np.array([
    [11,22],
    [33,44],
    [55,66],
    [77,88],
    [99,121]
])
float_arr = np.array(nested_list , dtype = float)
print(float_arr)
```

```
[[ 11.  22.]
 [ 33.  44.]
 [ 55.  66.]
 [ 77.  88.]
 [ 99. 121.]]
```

```
# upcast ['str'] VS dtype ['float']
nested_list = np.array([
    [11,22],
    [33,44],
    [55,66],
    [77,88],
    ['99.99','121']
])
float_arr = np.array(nested_list , dtype = float)
print(float_arr)
```

```
[[ 11.    22.  ]
 [ 33.    44.  ]
 [ 55.    66.  ]
 [ 77.    88.  ]
 [ 99.99 121.  ]]
```

```
# upcast ['str'] VS dtype ['float']
# ValueError: could not convert string to float: 'k'
nested_list = np.array([
    [11,22],
    [33,44],
    [55,66],
    [77,88],
    [99.99,'k']
])
float_arr = np.array(nested_list , dtype = float)
print(float_arr)
```

```
# upcast ['str'] VS dtype ['float']
```

```python
# upcast ['str'] VS dtype ['float']
nested_list = np.array([
    [11,22],
    [33,44],
    [55,66],
    [77,88],
    [False,True]
])
float_arr = np.array(nested_list , dtype = float)
print(float_arr)
```

```
[[11. 22.]
 [33. 44.]
 [55. 66.]
 [77. 88.]
 [ 0.  1.]]
```

```python
float_arr.dtype
```

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

```python
# ndarray [N-Dimensional Array] -> [.shape]
arr_1d.shape
```

```
(27,)
```

```python
arr_2d.shape
```

```
(5, 3)
```

```python
arr_3d.shape # 3d [depth,rows,cols]
```

```
(3, 3, 3)
```

```python
# .size Attribute -> Total Numbers of Elements
arr_1d.size
```

```
27
```

```python
arr_2d.size
```

```
15
```

```python
arr_3d.size
```

```
27
```

```python
# Reshape 1D -> 4D [2,2,2,2] = 2*2*2*2 = 16 elements
arr_1d = np.array([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16])
print(arr_1d)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]
```

```
print(arr_1d)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16]
```

```
arr_1d.reshape(2,4,2) # 3D
```

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

       [[ 9, 10],
        [11, 12],
        [13, 14],
        [15, 16]]])
```

```
arr_1d.reshape(2,2,4) # 3D
```

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

       [[ 9, 10, 11, 12],
        [13, 14, 15, 16]]])
```

```
arr_1d.reshape(2,2,2,2) # 4D
```

```
array([[[[ 1,  2],
         [ 3,  4]],

        [[ 5,  6],
         [ 7,  8]]],


       [[[ 9, 10],
         [11, 12]],

        [[13, 14],
         [15, 16]]]])
```

```
arr_1d.reshape(2,3,2) # 3D -> 12 elements
ValueError: cannot reshape array of size 16 into shape (2,3,2)
```