; NUMPY?

ITIDIMENSIONAL ARRAY LIBRARY ANALYSIS

- IN THOSE ARRAYS THERE IS HONOGENEOUS DATA TYPES, THIS ENABLES THE NUMPY ARRAY TO BE USED AND INCORPERATED WITH

ERATIONS:
EQUATTICAL (MULTIPLICATION, ---)
AL (COMPARISONS)
MANIPULATION
T, DFT, etc ---

CES BETWEEN NUMPY AND PYTHON SEQUENCES

UMPY ARR. HAVE FIXED SIZE AT CREATION,
LISTS GROW DYNAMICALLY (CHANGING THE SIZE OF AN ARR. WILL CREATE A NEW ONE)

NUMPY ARR. ALL ELEMENTS HAVE TO BE OF
E TYPE. → SAME SIZE IN MEMORY

UMPY PLOWS FOR ADVANCED MATHEMATICAL
VS ON HUGE DATA IN AN ARRAY (WITH NO FOR)

IFF:

```
]
range(len(a))
t(a[i]*b[i])
```

} LIST ITERATION

```
>> c = a*b
```

IF a & b ARE NumPY } WAY QUICKER

BASICS: IN NUMPY DIMENSIONS ARE CALLED AXIS (important)

NUMPY'S ARRAY CLASS IS CALLED (ndarray)

EXAMPLE + IMPORTANT Fn.s

```
>> import numpy as np
>> a = np.arange(15).reshape(3,5)
>> a
array([[ 0, 1, 2, 3, 4],
       [ 5, 6, 7, 8, 9],
       [10, 11, 12, 13, 14]])
>> a.shape
(3, 5)
>> a.ndim
2
>> a.dtype.name
'int64'
>> type(a)
<class 'numpy.ndarray'>
```

a.shape → dimensions of the array; for matrix in (n,m): rows, columns

(3,5): ① 3 rows ② 5 columns

a.ndim → number of axes of the array

① ndarray.ndim:
→ THE NUMBER OF AXES (DIMENSIONS OF THE ARRAY

② ndarray.shape
→ n rows, m columns (n, m)
⇒ len(ndarray.shape) = ndarray.ndim

③ ndarray.size
→ TOTAL NUMBER OF ELEMENTS IN THE ARRAY

④ ndarray.dtype
→ TYPE OF ELEMENTS IN THE ARRAY, ex:
numpy.int32, . . ---

⑤ ndarray.itemsize
SIZE IN BYTES OF ONE ELEMENT

⑥ ndarray.data

ARRAY CREATION: THERE ARE MULTIPLE WAYS

① FROM PYTHON LISTS OR TUPLES (WHEN THE ELEMENTS ARE KNOWN)

② OFTEN THE SIZE IS THE ONLY KNOWN

① np.zeros((3,4)) → 3x4 MATRIX OF ZEROS
② np.ones((3,4)) → 3x4 MATRIX OF ONES
③ np.arange(10,30,5) (from, to (excluding), step) → creates from → to with a step
[10, 15, 20, 25]
④ np.array(LIST OR TUPLE)
⑤ np.empty((3,4)) → 3x4 MATRIX WITH RANDOMS
⑥ np.linspace(2.0, 3.0, num=5) divisions
[2.0, 2.25, 2.5, 2.75, 3.]

⑦ np.zeros/empty/ones_like