* Numpy :- a Python Library used For working with arrays

Why use Numpy?

→ Python Lists are **very** important, but they are slow to process.

Numpy aims to provide an array object much Faster than the traditional python Lists.

* To start using Numpy.

→ import numpy as np → إذا لم تكن موجودة → pip install numpy

← كل العناصر في الـ numpy array يجب أن تكون من نفس نوع الـ data type

→ The array object in Numpy is called ndarray

n-dimensional array

```
arr = np.array([1,2,3,4,5])
print(arr) → [1 2 3 4 5]
print(arr.shape) → (5,)   (شبيه len() array يطبع عدد العناصر في الـ)
arr2 = np.arange(10) → array يدخل  يحفظ عدد من الأرقام بدايته من صفر
print(arr2) → [0 1 2 3 4 5 6 7 8 9]
arr3 = np.arange(2,6,2) → step ، النهاية هو العدد اللي قبله ، البداية
print(arr3) → [2 4]
```

.ones(3) → [1. 1. 1.]

```
arr4 = np.zeros(3)
print(arr4) → [0. 0. 0.]
```

```
arr5 = np.zeros((2,3))
print(arr5) → [[0. 0. 0.]
               [0. 0. 0.]]
```

shape ↖ ↗ value

```
arr6 = np.full((2,4), 3)
print(arr6) → [[3 3 3 3]
               [3 3 3 3]]
```

※ Convert python Lists to np

```
my_List = [1,2,3,4,5]
arr7 = np.array(my_List)
print(arr7) → [1 2 3 4 5]
```

※ Accessing a certain item

```
print(arr7[0]) → 1
```

* Slicing Numpy Arrays:→ يعتبر نفس الأكواد لـ Lists

```
import numpy as np
np1 = np.array([1,2,3,4,5,6,7,8])

print(np1[1:5]) → [2 3 4 5]
print(np1[3:]) → [4 5 ..... 8 9]
print(np1[-3:-1]) → [7 8]
print(np1[1:5:2]) → [2 4]

np2 = np.array([[1,2,3],[4,5,6]])
print(np2[1,2]) → 6
print(np2[0:1, 0:2] → [[1 2]]
print(np2[0:2, 0:2] → [[1 2]
                       [4 5]]
```

* Numpy Universal Functions:-

```
import numpy as np
np1=np.array([0,1,2,3,4])
```

* Square root of each element

```
print(np.sqrt(np1)) → [0.  1.  1.4142  1.73205  2.]
```

```
np2=np.array([-2,-1,0,1,2])
print(np.absolute(np1)) → [2 1 0 1 2]
```

* Exponential Function

```
print(np.exp(np1)) → [1  2.718...  7.38905...  20.08...
                                                54.598...]
```

```
print(np.max(np1)) → 4 , print(np.min(np1)) → 0
```

* sign positive or negative

```
print(np.sign(np2)) → [-1 -1  0  1  1]
```

* Trig sin cos Log

```
print(np.sin(np1)) → [0.  0.8414...  0.9092...
                      0.1411...  -0.7568...]
```

* Numpy Array Copy Vs. View :-

Difference :-

Copy → The copy owns the data and any changes made to the copy will not affect original array, and any changes made to the original array will not affect the copy

View → The view doesn't own the data and any changes made to the view will affect the original array, and any changes made to the original array will affect the view

Examples: Copy:-

```
import numpy as np

arr = np.array([1,2,3,4,5])
 x = arr.copy
arr[0] = 42
```

print(arr) → [42 2 3 4 5]

print(x) → [1 2 3 4 5]

View:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
arr[0] = 42
```

print(arr) → [42 2 3 4 5]
print(x) → [42 2 3 4 5]

```
x[0] = 31
```

print(arr) → [31 2 3 4 5]
print(x) → [31 2 3 4 5]

* Numpy Array Shape:- is the number of element in each dimension

** Shape of a 2-D array:

```
import numpy as np
arr = np.array([[1,2,3,4],[5,6,7,8]])
print(arr.shape) → (2,4) → the 1st-D has 2 elements
                         → the 2nd-D has 4 elements

arr2 = np.array([1,2,3,4], ndim=5)
print(arr2) → [[[[[1 2 3 4]]]]]
```

* Numpy Array Reshape:-

```
import numpy as np
arr = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
newarr = arr.reshape(4,3)
print(newarr) → [[ 1 2 3]
                 [4 5 6]
                 [7 8 9]
                 [10 11 12]]

newarr2 = arr.reshape(2,3,2)
print(newarr2) → [[[ 1 2]
                   [3 4]
                   [5 6]]

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

✱ UnKnown Dimension

```
arr2 = np.array([1,2,3,4,5,6,7,8])
newarr = arr2.reshape(2,2,-1)
print(newarr) → [[[1 2]
                  [3 4]]

                 [[5 6]
                  [7 8]]]
```

✱ Flattening the arrays

```
import numpy as np
arr = np.array([[1,2,3],[4,5,6]])
newarr = arr.reshape(-1)
```

print → [1 2 3 4 5 6]

* Iterating Through Numpy Arrays :- نفس الطريقة اللي List

```
import numpy as np          * 1-D
arr = np.array([1,2,3])
For x in arr:
    print(x) → 1
               2
               3
```

```
arr2 = np.array([[1,2,3],[4,5,6]])
For x in arr2:
    print(x) → [1 2 3]
               [4 5 6]
```

```
For x in arr2
    For y in x:
        print(y) → 1
                   2
                   3
                   4
                   5
                   6
```

```
For x in np.nditer(arr2):
    print(x) → 1
               2
               3
               4
               5
               6
```

* Numpy Sorting Array:-

```
import numpy as np
arr = np.array([3, 2, 0, 1])
print(np.sort(arr)) → [0 1 2 3]
```

※ When using strings → it sort the array alphabetically

```
arr2 = np.array(['banana', 'cherry', 'apple'])
print(np.sort(arr2)) → ['apple' 'banana' 'cherry']
```

※ BooLean

```
print(...) → [False True True]
```

→ Sort() Returns A Copy

* Searching Numpy Arrays:-

```
import numpy as np
arr = np.array([1,2,3,4,5,4,4])
x = np.where(arr==4)
print(x) → (array([3,5,6]),)
```

→ where returns a tuple

* The indexes for even values:

```
x = np.where(arr%2==0)
print(x) → (array([1,3,5,6]),)
```

* The searchsorted() method → search in the array as if it was sorted

```
arr2 = np.array([6,9,7,8])
x = np.searchsorted(arr2,7) → 1 → [6,7,8,9]
" " " " " (arr2,8) → 2
```

```
x = np.searchsorted(arr2,7, side='right')
print(x) → 7 →
```

* Multiple Values → Find the indexes where the values

* 2,4 and 6 should be inserted:

```
arr3 = np.array([1,3,5,7])
x = np.searchsorted(arr, [2,4,6])
print(x) → [1 2 3]
```

* Searching Numpy Arrays:-

```
import numpy as np
arr = np.array([1,2,3,4,5,4,4])
x = np.where(arr == 4)
print(x) → (array([3,5,6]),)
```

→ Where returns a tuple

※ The indexes For even values:

```
x = np.where(arr%2 == 0)
print(x) → (array([1,3,5,6]),)
```

※ The searchsorted() method → search in the array as if it was sorted

```
arr2 = np.array([6,9,7,8])
x = np.searchsorted(arr2, 7) → 1 → [6,7,8,9]
" " " " (arr2, 8) → 2
```

```
x = np.searchsorted(arr2, 7, side='right')
print(x) → 7 →
```

يبدأ بالعد من اليمين.

※ Multiple Values → Find the indexes where the values 2,4 and 6 should be inserted:

```
arr3 = np.array([1,3,5,7])
x = np.searchsorted(arr, [2,4,6])
print(x) → [1 2 3]
```

* Numpy Filter Array:- Getting some elements out of an existing array and creating a new array out of them

→ In Numpy, you filter an array using a boolean index list

→ A boolean index list is a list of booleans corresponding to indexes in the array.

Example:-

```
import numpy as np
arr = np.array([41, 42, 43, 44])
x = [True, False, True, False]
newarr = arr[x]
print(newarr) → [41 43] →
```

يطبع الـ True بس

※ print even values

```
Filtered = []
For obj in arr:
    if obj % 2 == 0:
        Filtered.append(True)
    else:
        Filtered.append(False)
print(filtered) → [False, True, False, True]
print(arr[Filtered]) → [42 44]
```

※ Shortcut

```
Filtered = arr % 2 == 0
```

※ Greater than 42

```
Filtered = arr > 42
print(arr[Filtered])
→ [43 44]
```

* Additional functions:-

```
v1 = np.array([1,2,3,4])
v2 = np.array([5,6,7,8])
print(np.vstack((v1, v2))) → [[1 2 3 4]
                              [5 6 7 8]]
print(np.hstack((v1, v2))) → [1 2 3 4 5 6 7 8]
```

* Loading data from a file:-

```
np.genfromtxt('data.txt', delimiter=',')
→ array([.....])
```

* checking if a variable in the array

```
→ 2 in v1 → True
  0 in v1 → False
```

* تحويل الـ array الى data type عادي

```
np1 = np.array([1,2,3])
np1.tolist() → List تحول الـ numpy array الى
List(np1) ——↗
```

```
→ print(np1[:, np.newaxis]) → [[1]
                               [2]
                               [3]]
```

```
→ np2 = np.array([6,2,5,-1,2,9,0,5])
print(np2.clip(0,5)) → [5 2 5 0 2 0 0 5]
print(np.unique(np2)) → [-1 0 2 5 6]
```

* Additional functions:-

```
v1 = np.array([1,2,3,4])
v2 = np.array([5,6,7,8])
print(np.vstack((v1, v2))) → [[1 2 3 4]
                              [5 6 7 8]]
print(np.hstack((v1, v2))) → [1 2 3 4 5 6 7 8]
```

* Loading data from a file:-

```
np.genfromtxt('data.txt', delimiter=',')
→ array([.....])
```

* checking if a variable in the array

```
→ 2 in v1 → True
  0 in v1 → False
```

* يختلف data type عن حفظ array للسماح لا

```
np1 = np.array([1,2,3])
np1.tolist() → List  تحول الـ numpy array إلى
list(np1) ——↗
```

```
→ print(np1[:, np.newaxis]) → [[1]
                               [2]
                               [3]]
```

```
→ np2 = np.array([6,2,5,-1,2,0,0,5])
print(np2.clip(0,5)) → [5 2 5 0 2 0 0 5]
print(np.unique(np2)) → [-1 0 2 5 6]
```

* Classes and Objects :- → oop

→ class ClassName :

x = 5 ※ Body

p1 = MyClass() ← object

print(p1.x) → 5

※ __init__ () Function ※ Constructor

→ class Person :

def __init__ (self, name, age): ← تستخدم علشان تـ access أي attribute موجود

self.name = name

self.age = age

(self.name, self.age → attributes)

p1 = Person("John", 36) ※ object

print (p1.name) → John

print(p1.age) → 36

※ def __del__ (self): ※ destructor

print (self.__class__.__name__, "destroyed") → Person destroyed

→ del p1

p1.func() → name 'p1' is not defined

* Inheritance :-

```
class Parent:          ## define parent class
    def myMethod(self):
        print('calling parent method')
class child(Parent):   ## define child class
    def myMethod(self):
        print('calling child method')
```