NumPy stands for Numerical Python

Advantages of Numpy Arrays:

1. Allows several Mathematical Operations.
2. Faster Operations.

```
import numpy as np
```

List Vs Numpy- Time taken!

```
from time import process_time
```

Time taken by a list:

```
python_list = [ i for i in range(10000)]

start_time = process_time()

python_list = [i+5 for i in python_list]

end_time = process_time()
print(end_time - start_time)
```

```
    0.0008277310000002203
```

```
np_array = np.array([i for i in range(10000)])

start_time = process_time()

np_array += 5
```

```
 end_time = process_time()

 print(end_time - start_time)
```

```
    0.00010747599999992736
```

Numpy Arrays:

```
#list

list1 = [1, 2, 3, 4, 5]
print(list1)
type(list1)
```

```
     [1, 2, 3, 4, 5]
     list
```

```
np_array= np.array([1,2,3,4,5])
print(np_array)
type(np_array)
```

```
     [1 2 3 4 5]
     numpy.ndarray
```

```
#creating a 1 dim array

a = np.array([1,2,3,4,5])
print(a)
type(a)
```

```
     [1 2 3 4 5]
     numpy.ndarray
```

```
a.shape()
```

```
(5,)
```

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

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

```
b.shape
```

```
(2, 5)
```

```
c = np.array([(10,11, 12,14,16), (12,13,14,15,16), (10, 12,13,14,15)])
print(c)
```

```
[[10 11 12 14 16]
 [12 13 14 15 16]
 [10 12 13 14 15]]
```

```
c.shape
```

```
(3, 5)
```

```
d = np.array([(1,2,3,4), (5,6,7,8)],dtype=float)
print(d)
```

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

```
d.shape
```

```
(2, 4)
```

Initial Placeholders in Numpy Arrays:

```
#creating numpy array of zeroes

x = np.zeros((6,5))
print(x)
```

```
[[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.]]
```

```
#creating numpy array of ones!

y = np.ones((5,5))
print(y)
```

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

```
#creating an array of a given value:

z = np.full((4,5),3)
print(z)
```

```
[[3 3 3 3 3]
 [3 3 3 3 3]
 [3 3 3 3 3]
 [3 3 3 3 3]]
```

```
#creating the identity matrix.
w = np.eye(5)
```

```
print(w)
```

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

```
#creating a numpy array with random values

p = np.random.random((3,4))
print(p)
```

```
[[0.76640313 0.45830544 0.61054494 0.13230756]
 [0.52670166 0.04248605 0.46658022 0.70755062]
 [0.24483112 0.32464143 0.12838558 0.78120228]]
```

```
#creating a numpy array with random integers that too in a specific range

q = np.random.randint(1000,10000,(4,5))
print(q)
```

```
[[5586 6851 8709 1990 6505]
 [1901 6683 9885 9232 9081]
 [6095 9831 4713 6284 9330]
 [6376 8595 6086 2015 9758]]
```

```
#creating an array of evenly spaced values ----> specifying the number of values required

d = np.linspace(10,30,8)
print(d)
```

```
[10.         12.85714286 15.71428571 18.57142857 21.42857143 24.28571429
 27.14285714 30.        ]
```

```
#creating an array of evenly spaced values --> specifying the step

t = np.arange(10,20,3)
print(t)
```

```
[10 13 16 19]
```

```
#coverting a list to a numpy array

list2 = [2,3,4,5,6]

np_array = np.asarray(list2)
print(np_array)
type(np_array)
```

```
[2 3 4 5 6]
numpy.ndarray
```

Analysing a numpy array:

```
t = np.random.randint(20,60, (5,5))
print(t)
```

```
[[49 51 26 52 53]
 [44 55 39 30 55]
 [20 35 53 21 34]
 [53 49 42 29 50]
 [55 38 39 28 43]]
```

```
#checking the dimension of an array:

print(t.shape)
```

```
    (5, 5)
```

```
# checking number of dimension

print(t.ndim)
```

```
    2
```

```
# checking the number of elements in an array

print(t.size)
```

```
    25
```

```
#checking the data types of the value in the array.

print(t.dtype)
```

```
    int64
```

Mathematical operations in an numpy array:

```
list1 = [2,3,4,5,6,]

list2 = [3,4,5,6,7]

print(list1 + list2) #this will concatenate the elements of the two lists
```

```
    [2, 3, 4, 5, 6, 3, 4, 5, 6, 7]
```

```
a = np.random.randint(0,10,(4,4))
b = np.random.randint(10,20,(4,4))
print(a)
print(b)
```

```
[[4 6 6 7]
 [7 3 3 4]
 [0 0 9 3]
 [3 5 1 6]]
[[10 13 13 19]
 [12 17 19 10]
 [12 10 12 11]
 [15 11 19 15]]
```

```
print (a+b)
print (a-b)
print (a*b)
print (a/b)
```

```
[[14 19 19 26]
 [19 20 22 14]
 [12 10 21 14]
 [18 16 20 21]]
[[ -6  -7  -7 -12]
 [ -5 -14 -16  -6]
 [-12 -10  -3  -8]
 [-12  -6 -18  -9]]
[[ 40  78  78 133]
 [ 84  51  57  40]
 [  0   0 108  33]
 [ 45  55  19  90]]
[[0.4        0.46153846 0.46153846 0.36842105]
 [0.58333333 0.17647059 0.15789474 0.4       ]
 [0.         0.         0.75       0.27272727]
 [0.2        0.45454545 0.05263158 0.4       ]]
```

```
#another way of doing mathematical operations:

a = np.random.randint(0,10,(4,4))
```

```
b = np.random.randint(10,20,(4,4))
print(a)
print(b)
```

```
[[1 9 5 7]
 [0 2 8 9]
 [6 1 1 1]
 [3 6 8 6]]
[[15 18 19 17]
 [16 14 10 12]
 [16 15 10 15]
 [14 11 18 19]]
```

```
print(np.add(a,b))
print(np.subtract(a,b))
print(np.multiply(a,b))
print(np.divide(a,b))
```

```
[[16 27 24 24]
 [16 16 18 21]
 [22 16 11 16]
 [17 17 26 25]]
[[-14  -9 -14 -10]
 [-16 -12  -2  -3]
 [-10 -14  -9 -14]
 [-11  -5 -10 -13]]
[[ 15 162  95 119]
 [  0  28  80 108]
 [ 96  15  10  15]
 [ 42  66 144 114]]
[[0.06666667 0.5        0.26315789 0.41176471]
 [0.         0.14285714 0.8        0.75      ]
 [0.375      0.06666667 0.1        0.06666667]
 [0.21428571 0.54545455 0.44444444 0.31578947]]
```

Array Manipulation in Numpy arrays:

```
my_array = np.random.randint(0,10,(4,5))
print(my_array)
print(my_array.shape)
```

```
[[2 8 6 9 5]
 [3 6 8 1 3]
 [0 2 5 0 7]
 [4 6 4 8 1]]
(4, 5)
```

```
#calculating the transpose:

trans = np.transpose(my_array)
print(trans)
print(trans.shape)
```

```
[[2 3 0 4]
 [8 6 2 6]
 [6 8 5 4]
 [9 1 0 8]
 [5 3 7 1]]
(5, 4)
```

```
#another way of finding the transpose

trans2 = my_array.T
print(trans2)
print(trans2.shape)
```

```
[[2 3 0 4]
 [8 6 2 6]
 [6 8 5 4]
 [9 1 0 8]
 [5 3 7 1]]
(5, 4)
```

```
#reshaping an array:
```

```
a = np.random.randint(10,30, (4,3))
print(a)
print(a.shape)
```

```
[[21 13 12]
 [13 18 10]
 [14 21 27]
 [12 20 23]]
(4, 3)
```

```
b = a.reshape(2,6)
print(b)
print(b.shape)
```

```
[[21 13 12 13 18 10]
 [14 21 27 12 20 23]]
(2, 6)
```

✓ 0s completed at 16:36