What is Numpy?

NumPy is the fundamental package for scientific computing in Python. NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.

Some important points about Numpy arrays:

- We can create a N-dimensional array in python using numpy.array().
- Array are by default Homogeneous, which means data inside an array must be of the same Datatype.
- Element wise operation is possible.
- Numpy array has the various function, methods, and variables, to ease our task of matrix computation.

Advantages of using Numpy Arrays Over Python Lists:

- consumes less memory.
- fast as compared to the python List.
- convenient to use.

In [1]:
```python
import numpy as np
```

In [70]:
```python
a= np.array([1,2,3],dtype='int16')
a
```
Out[70]:
```
array([1, 2, 3], dtype=int16)
```

In [74]:
```python
# Get type
print(type(a))
a.dtype
```
```
<class 'numpy.ndarray'>
```
Out[74]:
```
dtype('int16')
```

In [71]:
```python
# Get size
a.itemsize
```
Out[71]:
```
2
```

In [75]:
```python
# Get total size
a.nbytes
```
Out[75]:
```
6
```

In [5]:
```python
# Use a tuple to create a NumPy array:
```

In [6]:
```python
arr = np.array((1, 2, 3, 4, 5))
arr
```
Out[6]:
```
array([1, 2, 3, 4, 5])
```

Dimensions in Arrays

In [22]:
```python
# Create a 1-D array
arr = np.array([1, 2, 3, 4, 5])
print(arr.ndim) #Find the Number of dimensions
print(arr.shape) #Find the shape of array
```
```
1
(5,)
```

In [50]:
```python
# Create a 2-D array
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr.ndim) #Find the Number of dimensions
print(arr.shape) #Find the shape of array
arr
```
```
2
(2, 3)
```
Out[50]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

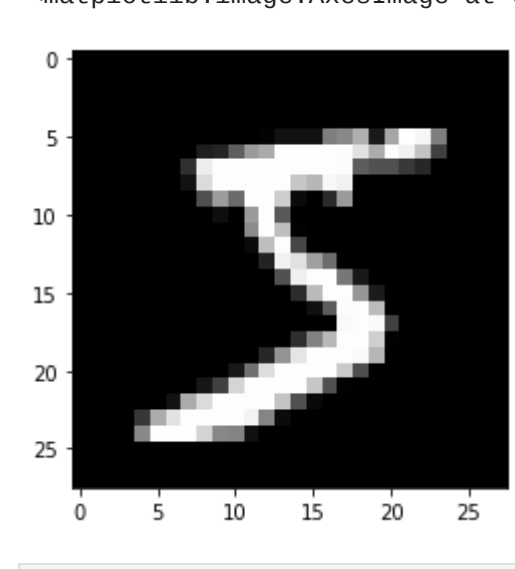
In [48]:
```python
arr = np.array([[[1, 2], [4, 5],[7, 8]], [[1, 2], [4, 5],[7, 8]],[[1, 2], [4, 5],[7, 8]],[[1, 2], [4, 5],[7, 8]]])
print(arr.ndim) #Find the Number of dimensions
print(arr.shape) #Find the shape of array
arr
```
```
3
(4, 3, 2)
```
Out[48]:
```
array([[[1, 2],
        [4, 5],
        [7, 8]],

       [[1, 2],
        [4, 5],
        [7, 8]],

       [[1, 2],
        [4, 5],
        [7, 8]],

       [[1, 2],
        [4, 5],
        [7, 8]]])
```

In [16]:
```python
#image in 2 D array
Image=np.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,   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,   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,   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,   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,   0,   0,   3,
         18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
        253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
        253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
        253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
        205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
         90,   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, 139, 253,
        190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
        253,  70,   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,  35,
        241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
         81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,  16,  93, 252, 253, 187,   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, 249, 253, 249,  64,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
        148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
        253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
        253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,
        195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244, 133,
         11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,   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,   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,   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,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0]])
```

In [4]:
```python
print(Image.ndim) #Find the Number of dimensions
print(Image.shape) #Find the shape of array
```
```
2
(28, 28)
```

In [5]:
```python
import matplotlib.pyplot as plt
```

In [6]:
```python
plt.imshow(Image,cmap='gray')
```
Out[6]:
```
<matplotlib.image.AxesImage at 0x24d2d42bf10>
```



In [52]:
```python
# Create an array with 5 dimensions and verify that it has 5 dimensions:
arr = np.array([1, 2, 3, 4], ndmin=5)

print(arr)
print('number of dimensions :', arr.ndim)
```
```
[[[[[1 2 3 4]]]]]
number of dimensions : 5
```

Accessing/Changing specfic elements, rows, colunms, etc

In [114…
```python
a=np.array([[1,2,3,4,5,6,7],[8,9,10,11,12,13,14]])
print(a)
```
```
[[ 1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14]]
```

In [78]:
```python
a.shape
```
Out[78]:
```
(2, 7)
```

In [79]:
```python
# Get a specific elements [r,c]
a[1,5]
```
Out[79]:
```
13
```

In [87]:
```python
# Get a specfic row
a[0,:]
```
Out[87]:
```
array([1, 2, 3, 4, 5, 6, 7])
```

In [88]:
```python
# Get a specfic row
a[:,0]
```
Out[88]:
```
array([1, 8])
```

In [94]:
```python
#slicing
a[0,1:6]
```
Out[94]:
```
array([2, 3, 4, 5, 6])
```

In [104…
```python
# changing Specfic elements
a[0,0]=28
print(a)
```
```
[[28  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14]]
```

In [115…
```python
# changing Specfic row
a[0,:]=5
print(a)
```
```
[[ 5  5  5  5  5  5  5]
 [ 8  9 10 11 12 13 14]]
```

In [116…
```python
# changing Specfic colunms
a[:,0]=1
print(a)
```
```
[[ 1  5  5  5  5  5  5]
 [ 1  9 10 11 12 13 14]]
```

In [118…
```python
# changing last colunms
a[:,-1]=6
print(a)
```
```
[[ 1  5  6  5  5  5  6]
 [ 1  9  6 11 12 13  6]]
```

3-d example

In [133…
```python
b=np.array([[[1,2,3],[3,4,9]],[[4,5,3],[6,7,3]]])
print(b)
```
```
[[[1 2 3]
  [3 4 9]]

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

In [130…
```python
b.shape
```
Out[130…
```
(2, 2, 3)
```

In [137…
```python
#Get the Accessing 9
b[0,1,2]
```
Out[137…
```
9
```

In [139…
```python
# changing Specfic elements ex: 7
b[1,1,1]=10
b
```
Out[139…
```
array([[[ 1,  2,  3],
        [ 3,  4,  9]],

       [[ 4,  5,  3],
        [ 6, 10,  3]]])
```

Joining Numpy Arrays (Using numpy.concatenate)

In [ ]:
```python
# axis = 0 refers to horizontal axis or rows
# axis = 1 refers to vertical axis or columns.
# axis=none
```

In [10]:
```python
a=np.array([1,2,3])
b=np.array([2,3,4])
```

In [22]:
```python
#axis=none arrays are flattened
none=np.concatenate((a,b),axis=None)
none
```
Out[22]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16])
```

In [13]:
```python
a=np.array([[1,2,3,4],[5,6,7,8]])
b=np.array([[9,10,11,12],[13,14,15,16]])
```

In [16]:
```python
# axis =0
row_con=np.concatenate((a,b),axis=0)
row_con
```
Out[16]:
```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12],
       [13, 14, 15, 16]])
```

In [18]:
```python
# axis=1
columm_con=np.concatenate((a,b),axis=1)
columm_con
```
Out[18]:
```
array([[ 1,  2,  3,  4,  9, 10, 11, 12],
       [ 5,  6,  7,  8, 13, 14, 15, 16]])
```

In [21]:
```python
# axis=0
np.concatenate((a,b),axis=None)
```
Out[21]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16])
```

Initializing Different Types of Arrays

In [146…
```python
# All zeros Matrix
zero_2d=np.zeros((2,3)) # 2 r * 3 c
zero_2d
```
Out[146…
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

In [147…
```python
zero_3d=np.zeros((2,3,4)) # 2 r * 3 c
zero_3d
```
Out[147…
```
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.]]])
```

In [161…
```python
# All ones matrix
ones_2d=np.ones((2,3),dtype='int32')
print(ones_2d)
```
```
[[1 1 1]
 [1 1 1]]
```

In [152…
```python
ones_3d=((2,3,3))
print(ones_3d)
```
```
(2, 3, 3)
```

In [162…
```python
# Any another Number
another_Numbers=np.full((2,3),77,dtype='float32')
print(another_Numbers)
```
```
[[77. 77. 77.]
 [77. 77. 77.]]
```

In [163…
```python
a=np.array([[1,2,3,4,5,6,7],[8,9,10,11,12,13,14]])
print(a)
```
```
[[ 1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14]]
```

In [165…
```python
#any another number (full like)
any_anothe = np.full_like(a,4)
any_anothe
```
Out[165…
```
array([[4, 4, 4, 4, 4, 4, 4],
       [4, 4, 4, 4, 4, 4, 4]])
```

In [169…
```python
# Random Decimal number
random=np.random.randn(4,2)
random
```
Out[169…
```
array([[ 0.93012707, -0.12500165],
       [ 0.52982761, -1.54954723],
       [-1.31225701, -0.83861898],
       [ 0.68235939, -0.8881385 ]])
```

In [172…
```python
# (full like) array of a
random_sample=np.random.random_sample(a.shape)
random_sample
```
Out[172…
```
array([[0.9976641 , 0.99090427, 0.48569696, 0.41370273, 0.56419191,
        0.00681625, 0.75958216],
       [0.70108839, 0.84717603, 0.21898563, 0.84767483, 0.4617885 ,
        0.86316318, 0.94781135]])
```

In [178…
```python
# Random integer values
rand_int=np.random.randint(3,8,size=(3,2)) # start=3 end=8
rand_int
```
Out[178…
```
array([[4, 4],
       [7, 7],
       [7, 7]])
```

In [180…
```python
# identity Matrix
identity=np.identity(3)
print(identity)
```
```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

In [220…
```python
arr_arrange=np.arange(1, 1000, 2, dtype=int)
# arr_arrange
```

In [184…
```python
#Be careful when copying arrays!!!
a=np.array([1,2,3])
b=a
print(a)
b[0]=98
print(a)
```
```
[1 2 3]
[98  2  3]
```

In [187…
```python
# copy array
b=a.copy()
print(b)
```
```
[98  2  3]
```

Mathematical Operation

In [188…
```python
a=np.array([1,2,3,4])
print(a)
```
```
[1 2 3 4]
```

In [189…
```python
#Adding with number
a+2
```
Out[189…
```
array([3, 4, 5, 6])
```

In [190…
```python
#subtraction with number
a-2
```
Out[190…
```
array([-1,  0,  1,  2])
```

In [191…
```python
# Multiplication with number
a*2
```
Out[191…
```
array([2, 4, 6, 8])
```

In [192…
```python
# float division
a/2
```
Out[192…
```
array([0.5, 1. , 1.5, 2. ])
```

In [193…
```python
# division
a//2
```
Out[193…
```
array([0, 1, 1, 2], dtype=int32)
```

In [194…
```python
# Adding two arrays
b=np.array([1,0,1,0])
a+b
```
Out[194…
```
array([2, 2, 4, 4])
```

In [195…
```python
# power
a**2
```
Out[195…
```
array([ 1,  4,  9, 16], dtype=int32)
```

In [196…
```python
# Taking sine of array
np.sin(a)
```
Out[196…
```
array([ 0.84147098,  0.90929743,  0.14112001, -0.7568025 ])
```

Reorganizing Arrays

In [200…
```python
before_array=np.array([[1,2,3,6],
                       [4,5,6,8]])
before_array.shape
```
Out[200…
```
(2, 4)
```

In [209…
```python
after_array=before_array.reshape((4,2))
after_array.shape
```
Out[209…
```
(4, 2)
```

In [207…
```python
print(after_array)
```
```
[[1 2]
 [3 6]
 [4 5]
 [6 8]]
```

In [17]:
```python
#Convert Image from 2-D to 1-D array
Image.shape
```
Out[17]:
```
(28, 28)
```

In [27]:
```python
Image_flatten=Image.reshape(-1)
Image_flatten.shape
```
Out[27]:
```
(784,)
```

In [28]:
```python
#Rescale the image from 0 to instead of 0 to 255
print(Image_flatten.dtype) # find the data type of array
Image_flatten.astype('float32')
Max=np.amax(Image_flatten)
print("The Max value in the array is: ",Max)
Image_flatten=Image_flatten/Max
Image_flatten.max()
```
```
int32
The Max value in the array is:  255
```
Out[28]:
```
1.0
```

Boolean Masking and Advanced indexing

In [40]:
```python
a=np.array([[1,2,3,4,5,6,7],[8,9,10,11,12,13,14]])
a>2
```
Out[40]:
```
array([[False, False,  True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True,  True,  True]])
```

In [60]:
```python
array=a[a>3]
array
```
Out[60]:
```
array([ 4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [ ]:

In [41]:
```python
#You can index with a list in numpy
A=np.array([1,2,3,4,5,6,7])
A[[1,4,5]]
```
Out[41]:
```
array([2, 5, 6])
```

Load and writting Data from files

In [29]:
```python
# Read a file
data=np.loadtxt('Data.txt' )
data
```
Out[29]:
```
array([ 1.,  3.,  5.,  7.,  9., 10., 11., 12., 13., 14., 15., 16., 17.,
       18., 19., 20.])
```

In [32]:
```python
print(type(data))
data.dtype
```
```
<class 'numpy.ndarray'>
```
Out[32]:
```
dtype('float64')
```

In [33]:
```python
# Write file
np.savetxt("Image.txt",Image_flatten)
```

In [36]:
```python
# Read the image
im=np.loadtxt('Image.txt')
```

In [44]:
```python
stra=np.array('18,67')
```

In [38]:

Out[38]:
```
array('sika', dtype='<U4')
```

In [45]:
```python
import numpy as np
```

In [46]:
```python
array=np.array([3,6,9,12])
array/3
```
Out[46]:
```
array([1., 2., 3., 4.])
```

In [25]:

In [ ]: