notes created by swapan chetri

Libraries in python

python library is a collection of functions and methods that allows us to perform many actions without writing our code.

pandas -> data manipulation

numpy -> numerical computing

matplotlib -> visualization

Numpy LIBRARY

->Numpy stands for numerical python and it is the core library for numeric and scientific computing.

->consists of multi dimensional array objects and a collection of routines for processing those arrays.

importing numpy

In [2]:

```
#if numpy is not installed try this in the cell(!pip install numpy)
import numpy as np
```

single dimensional array

In [15]:

```
#initialization
n1=np.array([10,20,30,40])
n1
```

Out[15]:

```
array([10, 20, 30, 40])
```

multi dimensional array

In [7]:

```
n2=np.array([[10,20,30,40],[50,60,70,80]])
n2
```

Out[7]:

```
array([[10, 20, 30, 40],
       [50, 60, 70, 80]])
```

In [8]:

```
type(n2)
```

Out[8]:

```
numpy.ndarray
```

initialization of numpy arrays with zeros

In [12]:

```
n1=np.zeros((5,6))
#here 5 means (rows) and 6 means (columns)
n1
```

Out[12]:

```
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.]])
```

initialization of numpy arrays with same number

In [18]:

```
n1=np.full((2,3),10)
# here 2 is number of (rows) and 3 is no. of (columns) and 10 is the number
n1
```

Out[18]:

```
array([[10, 10, 10],
       [10, 10, 10]])
```

initialization of numpy arrays within a range

In [20]:

```
n1=np.arange(10,20)
n1
```

Out[20]:

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

In [22]:

```
n2=np.arange(10,50,5)
n2
```

Out[22]:

```
array([10, 15, 20, 25, 30, 35, 40, 45])
```

initialization of numpy array with random numbers

In [29]:

```
n1=np.random.randint(1,100,5)
# here 1 to 100 is the range and 5 means we need 5 numbers
n1
```

Out[29]:

```
array([81, 25,  1, 58, 26])
```

numpy shape

In [33]:

```
# checking the shape of numpy arrays
n1=np.array([[1,2,3],[4,5,6],[7,8,9],[11,12,13]])
n1.shape
```

Out[33]:

```
(4, 3)
```

In [35]:

```
# change the shape of numpy arrays
n1.shape=(3,4)
n1
```

Out[35]:

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 11, 12, 13]])
```

In [37]:

```
# change the shape of numpy arrays with function
n1.reshape(4,3)
```

Out[37]:

```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [11, 12, 13]])
```

joining numpy arrays

In [38]:

```
#initialization
n1=np.array([10,20,30])
n2=np.array([40,50,60])
```

In [40]:

```
# vstack()
np.vstack((n1,n2))
```

Out[40]:

```
array([[10, 20, 30],
       [40, 50, 60]])
```

In [42]:

```
# hstack()
np.hstack((n1,n2))
```

Out[42]:

```
array([10, 20, 30, 40, 50, 60])
```

In [43]:

```
# column_stack()
np.column_stack((n1,n2))
```

Out[43]:

```
array([[10, 40],
       [20, 50],
       [30, 60]])
```

numpy intersention and Difference

In [44]:

```
#initialization
n1=np.array([10,20,30,40,50,60])
n2=np.array([50,60,70,80,90])
```

In [46]:

```
# intersection(return common from both the arrays)
np.intersect1d(n1,n2)
```

Out[46]:

```
array([50, 60])
```

In [51]:

```
#set difference
# return uncommon from n1
np.setdiff1d(n1,n2)
```

Out[51]:

```
array([10, 20, 30, 40])
```

In [50]:

```
# return uncommon from n2
np.setdiff1d(n2,n1)
```

Out[50]:

```
array([70, 80, 90])
```

numpy array mathematics

In [52]:

```
#initialization
n1=np.array([10,20,30])
n2=np.array([40,50,60])
```

In [62]:

```
# addition in numpy arrays
# axis=0 means summation column wise
np.sum([n1,n2],axis=0)
```

Out[62]:

```
array([50, 70, 90])
```

In [61]:

```
# axis=1 means summation row wise
np.sum([n1,n2],axis=1)
```

Out[61]:

```
array([ 60, 150])
```

In [65]:

```
# basic addition
n3=n1+n2
n3
```

Out[65]:

```
array([50, 70, 90])
```

In [68]:

```
# multiplication
n1=n1*2
n1
```

Out[68]:

```
array([ 80, 160, 240])
```

numpy math functions

In [70]:

```
#initialization
n1=np.array([10,20,30,40,50,60])
```

In [71]:

```
# mean
np.mean(n1)
```

Out[71]:

```
35.0
```

In [72]:

```
# median
np.median(n1)
```

Out[72]:

```
35.0
```

sort in numpy

In [80]:

```
#initialization
n1=np.array([10,20,50,60,30,40])
```

In [81]:

```
np.argsort(n1)
# it retuens the index of the smallest to larget number
```

Out[81]:

```
array([0, 1, 4, 5, 2, 3], dtype=int64)
```

In [82]:

```
np.argmin(n1)
# returns the index of the minimum number
```

Out[82]:

```
0
```

In [83]:

```
np.argmax(n1)
# returns the index of the maximum number
```

Out[83]:

```
3
```

numpy save and load

In [84]:

```
#initialization
n1=np.array([10,20,50,60,30,40])
```

In [85]:

```
# save
np.save("my_numpy",n1)
```

In [86]:

```
# load numpy array
n2=np.load("my_numpy.npy")
```

In [87]:

```
n2
```

Out[87]:

```
array([10, 20, 50, 60, 30, 40])
```