~ NumPy :

Creating a NumPy Array:

Single dimensional array

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

In [5]:
```python
a=np.array([10,20,30,40])
a
```

Out[5]:
```
array([10, 20, 30, 40])
```

Multi dimensional array

In [7]:
```python
b=n.array([[10,20,30,40],[50,60,70,80]])
b
```

Out[7]:
```
array([[10, 20, 30, 40],
       [50, 60, 70, 80]])
```

In [8]:
```python
type(a)
type(b)
```

Out[8]:
```
numpy.ndarray
```

Initializing NumPy with zeros->

In [11]:
```python
c=np.zeros((1,2))
c
```

Out[11]:
```
array([[0., 0.]])
```

In [13]:
```python
d=np.zeros((4,4))
d
```

Out[13]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

Initializing NumPy array with same number ->

In [15]:
```python
e=np.full((2,2),(17))
e
```

Out[15]:
```
array([[17, 17],
       [17, 17]])
```

In [16]:
```python
_=np.full((3,4),2)
_
```

Out[16]:
```
array([[2, 2, 2, 2],
       [2, 2, 2, 2],
       [2, 2, 2, 2]])
```

Initializing NumPy within a range->

In [17]:
```python
_=np.arange(10,20)
_
```

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

In [19]:
```python
_=np.arange(10,20,2)
_
```

Out[19]:
```
array([10, 12, 14, 16, 18])
```

Initializing Numpy with Random numbers- >

In [25]:
```python
_=np.random.randint(1,100,5) #start-end-no. want
_
```

Out[25]:
```
array([43, 70, 11, 43, 26])
```

Checking the shape of arrays- >

In [26]:
```python
b
```

Out[26]:
```
array([[10, 20, 30, 40],
       [50, 60, 70, 80]])
```

In [27]:
```python
b.shape
```

Out[27]:
```
(2, 4)
```

In [29]:
```python
b.shape=(4,2)
b
```

Out[29]:
```
array([[10, 20],
       [30, 40],
       [50, 60],
       [70, 80]])
```

Joining NumPy Arrays-->

--vstack()
--hstack()
--column_stack()

In [3]:
```python
a=np.array([10,20,30,40])
b=np.array([40,50,60,70])
np.vstack((a,b))
np.sum([a,b],axis=1)
```

Out[3]:
```
array([100, 220])
```

In [35]:
```python
np.hstack((a,b))
```

Out[35]:
```
array([10, 20, 30, 40, 40, 50, 60, 70])
```

In [36]:
```python
np.column_stack((a,b))
```

Out[36]:
```
array([[10, 40],
       [20, 50],
       [30, 60],
       [40, 70]])
```

Intersection & Difference-->

--intersect1d
--setdiff1d

In [7]:
```python
a
```

Out[7]:
```
array([10, 20, 30, 40])
```

In [8]:
```python
b
```

Out[8]:
```
array([40, 50, 60, 70])
```

In [9]:
```python
np.intersect1d(a,b)
```

Out[9]:
```
array([40])
```

In [11]:
```python
np.setdiff1d(a,b)
```

Out[11]:
```
array([10, 20, 30])
```

In [12]:
```python
np.setdiff1d(b,a)
```

Out[12]:
```
array([50, 60, 70])
```

NumPy Maths-->

In [23]:
```python
a=np.array([10,20])
b=np.array([30,40])
print(a)
print(b)
```

```
[10 20]
[30 40]
```

In [15]:
```python
np.sum([a,b])
```

Out[15]:
```
100
```

In [16]:
```python
np.sum([a,b],axis=0)
```

Out[16]:
```
array([40, 60])
```

In [17]:
```python
np.sum([a,b],axis=1)
```

Out[17]:
```
array([30, 70])
```

In [24]:
```python
a=a+1
b=b+1
print(a)
print(b)
```

```
[11 21]
[31 41]
```

In [25]:
```python
a=a-1
b=b-1
print(a)
print(b)
```

```
[10 20]
[30 40]
```

In [26]:
```python
a=a*2
b=b*2
print(a)
print(b)
```

```
[20 40]
[60 80]
```

In [27]:
```python
a=a/5
b=b/5
print(a)
print(b)
```

```
[4. 8.]
[12. 16.]
```

Math Functions-->

In [31]:
```python
a=np.array([10,20,30,40])
a
```

Out[31]:
```
array([10, 20, 30, 40])
```

In [32]:
```python
np.mean(a)
```

Out[32]:
```
25.0
```

In [33]:
```python
np.median(a)
```

Out[33]:
```
25.0
```

In [34]:
```python
np.std(a)
```

Out[34]:
```
11.180339887498949
```

Save & Load

In [35]:
```python
n1=np.array([10,20,30,0,232])
```

In [36]:
```python
np.save('final_numpy_array',n1)
```

In [38]:
```python
n2=np.load('final_numpy_array.npy')
```

In [39]:
```python
n2
```

Out[39]:
```
array([ 10,  20,  30,   0, 232])
```

~ Pandas :

In [7]:
```python
import pandas as pd
```

In [44]:
```python
a=pd.Series([10,20,30,40])
a
```

Out[44]:
```
0    10
1    20
2    30
3    40
dtype: int64
```

In [45]:
```python
a=pd.Series([10,20,30,40],index=['a','b','c','d'])
a
```

Out[45]:
```
a    10
b    20
c    30
d    40
dtype: int64
```

In [46]:
```python
b=pd.Series({'a':10,'b':20,'c':30})
b
```

Out[46]:
```
a    10
b    20
c    30
dtype: int64
```

In [48]:
```python
b=pd.Series({'a':10,'b':20,'c':30},index=['d','c','b','a'])
b
```

Out[48]:
```
d     NaN
c    30.0
b    20.0
a    10.0
dtype: float64
```

In [49]:
```python
a=pd.Series([1,2,3,4,5,6,7])
a[4]
```

Out[49]:
```
5
```

In [50]:
```python
a[:3]
```

Out[50]:
```
0    1
1    2
2    3
dtype: int64
```

In [51]:
```python
a[-3:]
```

Out[51]:
```
4    5
5    6
6    7
dtype: int64
```

In [53]:
```python
a+2
```

Out[53]:
```
0    3
1    4
2    5
3    6
4    7
5    8
6    9
dtype: int64
```

In [56]:
```python
a=pd.Series([1,2,3,4,5])
b=pd.Series([6,7,8,9,7])
```

In [57]:
```python
a+b
```

Out[57]:
```
0     7
1     9
2    11
3    13
4    12
dtype: int64
```

In [58]:
```python
a*2
```

Out[58]:
```
0     2
1     4
2     6
3     8
4    10
dtype: int64
```

2-Dimensional labelled data structure

~Data Frame

In [61]:
```python
df1=pd.DataFrame({'Name':['bhargav','bhavya','sravani'],'Marks':[100,99,98]})
df1
```

Out[61]:

	Name	Marks
0	bhargav	100
1	bhavya	99
2	sravani	98

In-Built Functions >

-head()
-tail()
-shape()
-describe()

In [12]:
```python
## importing from iris file
iris=pd.read_csv('iris.csv')
```

In [13]:
```python
iris.head()
```

Out[13]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [15]:
```python
iris.tail()
```

Out[15]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

In [17]:
```python
iris.shape
```

Out[17]:
```
(150, 5)
```

In [18]:
```python
iris.describe()
```

Out[18]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [19]:
```python
iris.iloc[0:3,0:2]
```

Out[19]:

	Sepal.Length	Sepal.Width
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2

In [23]:
```python
iris.loc[5:11,('Petal.Length','Species')]
```

Out[23]:

	Petal.Length	Species
5	1.7	setosa
6	1.4	setosa
7	1.5	setosa
8	1.4	setosa
9	1.5	setosa
10	1.5	setosa
11	1.6	setosa

In [25]:
```python
iris.drop('Sepal.Length',axis=1)
```

Out[25]:

	Sepal.Width	Petal.Length	Petal.Width	Species
0	3.5	1.4	0.2	setosa
1	3.0	1.4	0.2	setosa
2	3.2	1.3	0.2	setosa
3	3.1	1.5	0.2	setosa
4	3.6	1.4	0.2	setosa
...	...	...	...	...
145	3.0	5.2	2.3	virginica
146	2.5	5.0	1.9	virginica
147	3.0	5.2	2.0	virginica
148	3.4	5.4	2.3	virginica
149	3.0	5.1	1.8	virginica

150 rows × 4 columns

In [26]:
```python
iris.drop([1,2,3,4,5],axis=0)
```

Out[26]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
6	4.6	3.4	1.4	0.3	setosa
7	5.0	3.4	1.5	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
9	4.9	3.1	1.5	0.1	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

145 rows × 5 columns

Functions

~ iiris.mean()
~ iris.median()
~ iris.min()
~ iris.max()

In [27]:
```python
iris.mean()
```

Out[27]:
```
Sepal.Length    5.843333
Sepal.Width     3.057333
Petal.Length    3.758000
Petal.Width     1.199333
dtype: float64
```

In [28]:
```python
iris.median()
```

Out[28]:
```
Sepal.Length    5.80
Sepal.Width     3.00
Petal.Length    4.35
Petal.Width     1.30
dtype: float64
```

In [29]:
```python
iris.min()
```

Out[29]:
```
Sepal.Length       4.3
Sepal.Width        2.0
Petal.Length       1.0
Petal.Width        0.1
Species         setosa
dtype: object
```

In [30]:
```python
iris.max()
```

Out[30]:
```
Sepal.Length          7.9
Sepal.Width           4.4
Petal.Length          6.9
Petal.Width           2.5
Species         virginica
dtype: object
```

In [31]:
```python
def double_make(s):
    return s*0.5
```

In [33]:
```python
iris[['Sepal.Width','Petal.Width']].apply(double_make)
```

Out[33]:

	Sepal.Width	Petal.Width
0	1.75	0.10
1	1.50	0.10
2	1.60	0.10
3	1.55	0.10
4	1.80	0.10
...	...	...
145	1.50	1.15
146	1.25	0.95
147	1.50	1.00
148	1.70	1.15
149	1.50	0.90

150 rows × 2 columns

In [34]:
```python
iris['Species'].value_counts()
```

Out[34]:
```
versicolor    50
setosa        50
virginica     50
Name: Species, dtype: int64
```

In [35]:
```python
iris.sort_values(by='Sepal.Length')
```

Out[35]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
13	4.3	3.0	1.1	0.1	setosa
42	4.4	3.2	1.3	0.2	setosa
38	4.4	3.0	1.3	0.2	setosa
8	4.4	2.9	1.4	0.2	setosa
41	4.5	2.3	1.3	0.3	setosa
...	...	...	...	...	...
122	7.7	2.8	6.7	2.0	virginica
118	7.7	2.6	6.9	2.3	virginica
117	7.7	3.8	6.7	2.2	virginica
135	7.7	3.0	6.1	2.3	virginica
131	7.9	3.8	6.4	2.0	virginica

150 rows × 5 columns

In [ ]:
```python

```