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

In [2]:
```python
#Creating numpy array
#Single dimensional array
n1 = np.array([1,2,3,4,5,6])
n1
```

Out[2]:
```
array([1, 2, 3, 4, 5, 6])
```

In [4]:
```python
#Checking its type
type(n1)
```

Out[4]:
```
numpy.ndarray
```

In [9]:
```python
#Multi-dimensinal array
n2 = np.array([[1,2,3,4,5,6],[10,20,30,40,50,60]])
n2
```

Out[9]:
```
array([[ 1,  2,  3,  4,  5,  6],
       [10, 20, 30, 40, 50, 60]])
```

In [10]:
```python
#Checking its type
type(n2)
```

Out[10]:
```
numpy.ndarray
```

In [11]:
```python
#Initiallizing numpy array
#Initiallizing numpy array with zeros
n3 = np.zeros((2,3))
n3
```

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

In [12]:
```python
#Initiallizing numpy array with zeros
n4 = np.zeros((6,6))
n4
```

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.],
       [0., 0., 0., 0., 0., 0.]])
```

In [13]:
```python
#Initiallizing numpy array with same number
n5 = np.full((3,3),5)
n5
```

Out[13]:
```
array([[5, 5, 5],
       [5, 5, 5],
       [5, 5, 5]])
```

In [14]:
```python
#Initiallizing numpy array with same number
n6 = np.full((3,6),9)
n6
```

Out[14]:
```
array([[9, 9, 9, 9, 9, 9],
       [9, 9, 9, 9, 9, 9],
       [9, 9, 9, 9, 9, 9]])
```

In [15]:
```python
#Initiallizing numpy array with a range
n7 = np.arange(10,51)
n7
```

Out[15]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
       27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
       44, 45, 46, 47, 48, 49, 50])
```

In [16]:
```python
#Initiallizing numpy array with a range
n8 = np.arange(1,105,10)
n8
```

Out[16]:
```
array([  1,  11,  21,  31,  41,  51,  61,  71,  81,  91, 101])
```

In [17]:
```python
#Initiallizing numpy array with a random numbers
n9 = np.random.randint(1,1000,10)
n9
```

Out[17]:
```
array([ 59, 159, 915, 246, 589,  59, 204, 626, 537, 127])
```

In [18]:
```python
#Numpy-shape
#Checking the shape of numpy arrays
n10 = np.array([[1,2,3],[4,5,6]])
```

In [19]:
```python
n10
```

Out[19]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [20]:
```python
n10.shape
```

Out[20]:
```
(2, 3)
```

In [21]:
```python
#Changing the shape of a numpy array
n10.shape = (3,2)
```

In [22]:
```python
n10
```

Out[22]:
```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

In [24]:
```python
#Joining numpy array
n1 = np.array([10,20,30])
n2 = np.array([100,200,300])
```

In [25]:
```python
#vstack
np.vstack((n1,n2))
```

Out[25]:
```
array([[ 10,  20,  30],
       [100, 200, 300]])
```

In [26]:
```python
#hstack
np.hstack((n1,n2))
```

Out[26]:
```
array([ 10,  20,  30, 100, 200, 300])
```

In [27]:
```python
#column_stack
np.column_stack((n1,n2))
```

Out[27]:
```
array([[ 10, 100],
       [ 20, 200],
       [ 30, 300]])
```

In [28]:
```python
#Numpy interaction and difference
n1 = np.array([10,20,30,40,50,60])
n2 = np.array([50,60,70,80,90])
```

In [29]:
```python
#intersect1d
np.intersect1d(n1,n2)
```

Out[29]:
```
array([50, 60])
```

In [32]:
```python
#setdiff1d(n1,n2)
np.setdiff1d(n1,n2)
```

Out[32]:
```
array([10, 20, 30, 40])
```

In [33]:
```python
#setdiff1d(n2,n1)
np.setdiff1d(n2,n1)
```

Out[33]:
```
array([70, 80, 90])
```

In [34]:
```python
#Numpy Array Mathematics
#Addition of numpy array
n1 = np.array([10,20])
n2 = np.array([30,40])
```

In [36]:
```python
#np.sum([n1,n2])
np.sum([n1,n2])
```

Out[36]:
```
100
```

In [37]:
```python
#np.sum([n1,n2],axis=0)
np.sum([n1,n2],axis=0)
```

Out[37]:
```
array([40, 60])
```

In [38]:
```python
#np.sum([n1,n2],axis=1)
np.sum([n1,n2],axis=1)
```

Out[38]:
```
array([30, 70])
```

In [39]:
```python
#Basic addition
n1 = np.array([10,20,30])
n1 = n1 + 1
n1
```

Out[39]:
```
array([11, 21, 31])
```

In [40]:
```python
#Basic subtraction
n1 = np.array([10,20,30])
n1 = n1 - 1
n1
```

Out[40]:
```
array([ 9, 19, 29])
```

In [41]:
```python
#Basic multiplication
n1 = np.array([10,20,30])
n1 = n1*2
n1
```

Out[41]:
```
array([20, 40, 60])
```

In [42]:
```python
#Basic division
n1 = np.array([10,20,30])
n1 = n1/2
n1
```

Out[42]:
```
array([ 5., 10., 15.])
```

In [45]:
```python
#Basic maths function
#Mean
n1 = np.array([10,20,30,40,50,60])
np.mean(n1)
```

Out[45]:
```
35.0
```

In [46]:
```python
#Median
n1 = np.array([11,44,5,96,67,85])
np.median(n1)
```

Out[46]:
```
55.5
```

In [47]:
```python
#Standard deviation
n1 = np.array([1,5,3,100,4,48])
np.std(n1)
```

Out[47]:
```
36.59424666377065
```

In [50]:
```python
#Basic maths function
n1 = np.array([10,48,23,9,45,61])
```

In [51]:
```python
#Mean
np.mean(n1)
```

Out[51]:
```
32.666666666666664
```

In [52]:
```python
#Standard deviation
np.std(n1)
```

Out[52]:
```
19.821424996424675
```

In [53]:
```python
#Median
np.median(n1)
```

Out[53]:
```
34.0
```

In [54]:
```python
#Saving and loading a numpy array
n1 = np.array([10,20,87,0,12,65,232,9])
```

In [58]:
```python
#Saving numpy array
np.save('final_numpy_array',n1)
```

In [59]:
```python
#Loading numpy array
n2 = np.load('final_numpy_array.npy')
n2
```

Out[59]:
```
array([ 10,  20,  87,   0,  12,  65, 232,   9])
```

In [60]:
```python
#Pandas
#Pandas series object
#Creating a series
import pandas as pd
s1 = pd.Series([1,2,3,4,5])
s1
```

Out[60]:
```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

In [61]:
```python
#Checking its type
type(s1)
```

Out[61]:
```
pandas.core.series.Series
```

In [62]:
```python
#Creating a series
l1 = [1,2,3,4,5]
s1 = pd.Series(l1)
s1
```

Out[62]:
```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

In [63]:
```python
#Checking its type
type(s1)
```

Out[63]:
```
pandas.core.series.Series
```

In [64]:
```python
#Creating a series
s1 = pd.Series([10,30,43,12,9302])
s1
```

Out[64]:
```
0      10
1      30
2      43
3      12
4    9302
dtype: int64
```

In [65]:
```python
#Checking its type
type(s1)
```

Out[65]:
```
pandas.core.series.Series
```

In [66]:
```python
#Changing the index
s1 = pd.Series([1,2,3,4,5])
s1
```

Out[66]:
```
0    1
1    2
2    3
3    4
4    5
dtype: int64
```

In [67]:
```python
s1 = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])
s1
```

Out[67]:
```
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

In [68]:
```python
#Series object from dictionary
s1 = pd.Series({'a':10,'b':20,'c':30})
s1
```

Out[68]:
```
a    10
b    20
c    30
dtype: int64
```

In [70]:
```python
#Series object from dictionary
s1 = pd.Series({'a':100,'b':200,'c':300})
s1
```

Out[70]:
```
a    100
b    200
c    300
dtype: int64
```

In [74]:
```python
#Changing the index position
s1 = pd.Series({'a':100,'b':200,'c':300},index = ['d','c','a','b'])
s1
```

Out[74]:
```
d      NaN
c    300.0
a    100.0
b    200.0
dtype: float64
```

In [75]:
```python
#Extracting indiviual elements
#Extracting a single element
s1 = pd.Series([1,2,3,4,5,6,7,8,9])
s1[3]
```

Out[75]:
```
4
```

In [77]:
```python
#Extracting a sequence of element
s1 = pd.Series([1,2,3,4,5,6,7,8,9])
s1[:4]
```

Out[77]:
```
0    1
1    2
2    3
3    4
dtype: int64
```

In [80]:
```python
#Extracting elements from back
s1 = pd.Series([1,2,3,4,5,6,7,8,9])
s1[-3:]
```

Out[80]:
```
6    7
7    8
8    9
dtype: int64
```

In [81]:
```python
#Basic maths operation on series
#Adding scaler values to a series
s1 = pd.Series([1,2,3,4,5,6,7,8,9])
s1 +5
```

Out[81]:
```
0     6
1     7
2     8
3     9
4    10
5    11
6    12
7    13
8    14
dtype: int64
```

In [82]:
```python
#Adding two series objects
s1 = pd.Series([1,2,3,4,5,6,7,8,9])
s2 = pd.Series([10,20,30,40,50,60,70,80,90])
s1+s2
```

Out[82]:
```
0    11
1    22
2    33
3    44
4    55
5    66
6    77
7    88
8    99
dtype: int64
```

In [83]:
```python
#Subtracting scaler value from a series
s1 = pd.Series([1,2,3,4,5,6,7,8,9])
s1 -100
```

Out[83]:
```
0   -99
1   -98
2   -97
3   -96
4   -95
5   -94
6   -93
7   -92
8   -91
dtype: int64
```

In [84]:
```python
#Multiplying with a scaler value
s1 = pd.Series([1,2,3,4,5,6,7,8,9])
s1 *2
```

Out[84]:
```
0     2
1     4
2     6
3     8
4    10
5    12
6    14
7    16
8    18
dtype: int64
```

In [85]:
```python
#Maths operations on two series object
s1 = pd.Series([1,2,3,4,5,6,7,8,9])
s2 = pd.Series([10,20,30,40,50,60,70,80,90])
s1,s2
```

Out[85]:
```
(0    1
 1    2
 2    3
 3    4
 4    5
 5    6
 6    7
 7    8
 8    9
 dtype: int64,
 0    10
 1    20
 2    30
 3    40
 4    50
 5    60
 6    70
 7    80
 8    90
 dtype: int64)
```

In [86]:
```python
#Addition of two series objects
s1+s2
```

Out[86]:
```
0    11
1    22
2    33
3    44
4    55
5    66
6    77
7    88
8    99
dtype: int64
```

In [87]:
```python
#Subtraction of two series objects
s1-s2
```

Out[87]:
```
0    -9
1   -18
2   -27
3   -36
4   -45
5   -54
6   -63
7   -72
8   -81
dtype: int64
```

In [88]:
```python
#Creating a dataframe
df1 = pd.DataFrame({'Name':['Sam','Julia','Anne','Matt'],'Marks':[99,32,100,78]})
df1
```

Out[88]:

	Name	Marks
0	Sam	99
1	Julia	32
2	Anne	100
3	Matt	78

In [89]:
```python
#Dataframe in-build functions
college_1 = pd.read_csv('college_1.csv')
```

In [91]:
```python
#head()
college_1.head()
```

Out[91]:

	Name	python	mysql	Previous Geekions	CodeKata Score	Department	Rising
0	A.Dharani	82.0	20.0	24500	24500	Computer Science and Engineering	0
1	V.JEEVITHA	82.0	20.0	21740	21740	Computer Science and Engineering	0
2	HEMAVATHI.R	100.0	100.0	19680	19680	Computer Science and Engineering	0
3	Mugunthan S	100.0	47.0	10610	10610	Computer Science and Engineering	0
4	Sathammai.S	100.0	8.0	8980	8980	Computer Science and Engineering	0

In [94]:
```python
#tail()
college_1.tail()
```

Out[94]:

	Name	python	mysql	Previous Geekions	CodeKata Score	Department	Rising
79	KarthikeyanS	45.0	0.0	0	0	Electronics and Electrical Engineering	0
80	BARATH.P	29.0	0.0	0	0	Electronics and Electrical Engineering	0
81	N.Ajith kumar	82.0	0.0	0	0	Electronics and Electrical Engineering	0
82	mohamed nabi	0.0	0.0	0	0	Electronics and Electrical Engineering	0
83	yaser ahamed.A	0.0	27.0	0	0	Electronics and Electrical Engineering	0

In [96]:
```python
#shape
college_1.shape
```

Out[96]:
```
(84, 7)
```

In [97]:
```python
#describe()
college_1.describe()
```

Out[97]:

	python	mysql	Previous Geekions	CodeKata Score	Rising
count	84.000000	84.00000	84.000000	84.000000	84.000000
mean	73.392857	27.60119	2904.523810	2906.071429	1.547619
std	35.589280	36.30562	4372.425826	4374.184197	14.184163
min	0.000000	0.00000	0.000000	0.000000	0.000000
25%	52.250000	0.00000	292.500000	292.500000	0.000000
50%	90.250000	11.62500	1580.000000	1580.000000	0.000000
75%	100.000000	41.00000	3727.500000	3727.500000	0.000000
max	100.000000	100.00000	24500.000000	24500.000000	130.000000

In [98]:
```python
#Extracting indiviual rows and columns .iloc[]
college_1.iloc[0:3,0:2]
```

Out[98]:

	Name	python
0	A.Dharani	82.0
1	V.JEEVITHA	82.0
2	HEMAVATHI.R	100.0

In [99]:
```python
#Extracting indiviual rows and columns .loc[]
college_1.loc[5:11,('Name','python')]
```

Out[99]:

	Name	python
5	NIVEESHWAR S	100.0
6	KALAIARASAN K	85.0
7	MOHAMED ZUBAIR AHMED	82.0
8	J.SUGANTHI	27.0
9	thamizhpaana	29.0
10	Iyappan Samiraj	50.0
11	Ponniyamma.R	100.0

In [100…
```python
#Droping a particular column from a given dataframe .drop( ,axis=1)
#DataFrame.drop('column_name',axis=1)
```

In [101…
```python
#Droping a particular rows from a given dataframe .drop( ,axis=0)
#DataFrame.drop([rows_values],axis=0)
```

In [102…
```python
#Pandas functions
#mean()
college_1.mean()
```

```
C:\Users\swara\AppData\Local\Temp/ipykernel_27228/3366505761.py:3: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numer
ic_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  college_1.mean()
```

Out[102…
```
python                 73.392857
mysql                  27.601190
Previous Geekions    2904.523810
CodeKata Score       2906.071429
Rising                  1.547619
dtype: float64
```

In [103…
```python
#median()
college_1.median()
```

```
C:\Users\swara\AppData\Local\Temp/ipykernel_27228/3826758542.py:2: FutureWarning: Dropping of nuisance columns in DataFrame reductions (with 'numer
ic_only=None') is deprecated; in a future version this will raise TypeError.  Select only valid columns before calling the reduction.
  college_1.median()
```

Out[103…
```
python                 90.250
mysql                  11.625
Previous Geekions    1580.000
CodeKata Score       1580.000
Rising                  0.000
dtype: float64
```

In [104…
```python
#Minimum
college_1.min()
```

Out[104…
```
Name                                        A.Dharani
python                                            0.0
mysql                                             0.0
Previous Geekions                                   0
CodeKata Score                                      0
Department           Computer Science and Engineering
Rising                                              0
dtype: object
```

In [105…
```python
#Maximum
college_1.max()
```

Out[105…
```
Name                                   yaser ahamed.A
python                                          100.0
mysql                                           100.0
Previous Geekions                               24500
CodeKata Score                                  24500
Department           Electronics and Electrical Engineering
Rising                                            130
dtype: object
```

In [108…
```python
#apply()
#To reduce the values in the column by half
#def half(s):
     #return s*0.5
#DataFrame[['column_names']].apply(half-name of a function defined above)
```

In [110…
```python
#value_counts()
college_1['Department'].value_counts()
```

Out[110…
```
Computer Science and Engineering             41
Electronics and Communication Engineering    31
Electronics and Electrical Engineering       12
Name: Department, dtype: int64
```

In [111…
```python
#sort_values(by= )
college_1.sort_values(by='python')
```

Out[111…

	Name	python	mysql	Previous Geekions	CodeKata Score	Department	Rising
83	yaser ahamed.A	0.0	27.00	0	0	Electronics and Electrical Engineering	0
82	mohamed nabi	0.0	0.00	0	0	Electronics and Electrical Engineering	0
49	bavithra	0.0	100.00	1020	1020	Electronics and Communication Engineering	0
50	Deivani.S	0.0	86.25	980	980	Electronics and Communication Engineering	0
51	Nivetha.S	0.0	50.00	950	950	Electronics and Communication Engineering	0
...	...	...	...	...	...	...	...
19	S.srinivasan	100.0	12.00	4090	4090	Computer Science and Engineering	0
56	pravina	100.0	100.00	720	720	Computer Science and Engineering	0
18	Abirami Anbazhagan	100.0	0.00	4320	4320	Electronics and Communication Engineering	0
61	Aravind	100.0	0.00	380	380	Computer Science and Engineering	0
41	vijayakumar	100.0	0.00	1580	1580	Computer Science and Engineering	0

84 rows × 7 columns

In [ ]: