TPU → Tensorflow Processing unit
Python and opensource ecosystem.

Tensorflow → google
Pytorch → facebook.
cloud and AI Supp.

Introduction to Numpy:—

→ Numpy stands for Numerical Python, powerful Python library that is widely used for scientific computing, data analysis and numerical computing task.

→ In numpy, an array is the fundamental object.

→ Array are used to store homogenous data elements in a contiguous block of memory.

Install numpy:—

pip install numpy

→ We create ndarray. (n-dimensional array).

Array vs List:—

→ All element of the array are same data-type.
→ List can have element of different data type.

→ Elements are stored in contiguous memory location.
→ Not stored contiguously in memory.

→ Arrays are static and cannot be changed once they are created.	→ List can be re-sized and modified easily.
→ Support element wise operation.	→ List do not support element wise operation.
→ Numpy arrays takes up less space in memory.	→ More space in memory.

How to create numpy array:-

1-D

Eg:
```
import numpy as np
list1 = [10, 20, 30, 40, 50]
array1 = np.array(list1)  # np.array(list1, dtype='float')
print(array1)
type(array1) → numpy.ndarray
```

2-D

Eg:
```
import numpy as np
list1 = [[10,20,30], [40,50,60], [70,80,90]]
array1 = np.array(list1)
print(array1)
```

Eg:
```
import numpy as np
array1 = np.arange(1,8)
print(array1)
```

O/P:- [1, 2, 3, 4, 5, 6, 7]

Eg:
```
import numpy as np
array1 = np.arange(11,17).reshape((2,3))
print(array1)
```
(2,3) → 2 Row, 3 Col.

O/P :- $\begin{bmatrix} 11 & 12 & 13 \\ 14 & 15 & 16 \end{bmatrix}$

Eg:
```
import numpy as np
array1 = np.zeros(4)
print(array1)
```
np.zeros((4,2)) → 2-D Array.

O/P :- [0, 0, 0, 0]

Eg:
```
import numpy as np
array1 = np.ones(4)
print(array1)
```
O/P :- [1, 1, 1, 1]

Attributes of numpy Array :-

→ Attributes of numpy array
1) ndim
2) shape
3) size
4) dtype
5) itemsize.

Eg:
array1.ndim → show the dimension of array.
array1.shape → Display the no. of Rows and columns.

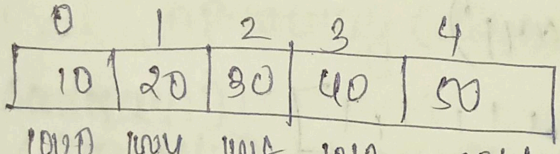
array1.size → Display the total no. of element in the array.

array1.dtype → Data-type of the array. dtype 'int32', dtype float64

array1.itemsize → size of each element of the array.

Indexing in Numpy array:-

```
import numpy as np
array1 = np.array([10,20,30,40,50])
Print(array1[0]) → 10.
Print(array1[-1]) → 50.
```




```
Print(array1[-7]) → Error.
```

Eg: array1 = np.array([[10,20,30],[40,50,60],[70,80,90]])

```
Print(array1[1,2])
Print(array1[0,:])
Print(array1[:,0])
```

	0	1	2
0	10	20	30
1	40	50	60
2	70	80	90

O/P :- 60
10 20 30
20 50 80

Slicing in Numpy array:-

→ slicing is a way to extract a subset of data from a numpy array.

Eg:

```
import numpy as np
array1 = np.array([10,20,30,40,50,60,70])
Print(array1[1:3])      → 20 30
Print(array1[1:6:2])    → 20 40 60
Print(array1[-1:-3:-1]) → 70, 60
Print(array1[::2])      → 10, 30 50 70
Print(array1[::-1])     → 70 60 50 40 30 20 10
```

-7	-6	-5	-4	-3	-2	-1
10	20	30	40	50	60	70
0	1	2	3	4	5	6

Eg:

```
array1 = np.array([[15,16,17],[25,26,27],[35,36,37],
                   [45,46,47]])
Print(array1[1,])       25 26 27
Print(array1[:,1])      16 26 36 46
Print(array1[1:3,1:3])  26 27 36 37
Print(array1[1:3,])     25 26 27 35 36 37
Print(array1[,1:3])     16 17 26 27 36 37 46 47
Print(array1[1:3,1])    26 36
Print(array1[1:3,:1])   25 35
Print(array1[1:3,1:])   26 27 36 37
```

	0	1	2
0	15	16	17
1	25	26	27
2	35	36	37
3	45	46	47

Arithmetic operation on Numpy array :—

1) Addition (+)
2) subtraction (-)
3) Multiplication (*)
4) matrix multiplication (@)
5) Division (/)
6) Floor division (//)
6) Exponentiation (**)
7) Modulo (%)
8) Transpose,

Eg:

```
import numpy as np
x = np.array([[1,2],
              [3,4]])
y = np.array([[11,12],
              [13,14]])
z = x+y
Print(z).
z = x-y
Print(z)
z = x*y
Print(z).
```

```
z = x@y
Print(z)
z = x/y
Print(z)
z = x//y
Print(z).
```

```
z = x**y
Print(z)
z = x%y
Print(z)
```

print(x.transpose())

Sorting 2-D Array:-

import numpy as np

x = np.array([[12, 11, 15],
[21, 25, 20],
[18, 27, 16]])

y = np.sort(x) → row-wise sort

print(y).

Eg:

y = np.argsort(x, axis=0)

print(y) → column-wise sort

axis=1 → row-wise sort

argsort → print the index position.

O/P :- $\begin{bmatrix} 0 & 0 & 0 \\ 2 & 1 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ ⇐ $\begin{bmatrix} 12 & 11 & 15 \\ 18 & 25 & 16 \\ 21 & 27 & 20 \end{bmatrix}$

x.sort()

Sorting in numpy array:-

1) np.sort() → It will return a sorted copy of an array.

2) np.argsort() → will return the indices that would sort an array

iii) ndarray.sort() → arrayname and sort in-place.

Eg: import numpy as np
x = np.array([7,2,3,9,6])
y = np.sort(x)
print(y). → [2,3,6,7,9]
y = np.argsort(x)
print(y) → [1, 2, 4, 0, 3]

x.sort()
print(x) → [2,3,6,7,9]
y = np.sort(x)[::-1]
[9,7,6,3,2]

Statistical operations:—

1) max()
2) min()
3) sum()
4) mean()
5) median()
6) prod
7) var.
8) std()

Eg: import numpy as np
x = np.array([4,2,3,9,7])
y = np.max(x)
print(y) → 9
y = np.min(x)
print(y) → 2

```
y = np.sum(x)
Print(y) → 25
y = np.mean(x)
Print(y) → 5.0
y = np.prod(x)
Print(y) → 1512
y = np.var(x)
Print(y) → 6.8
```

```
y = np.median(x)
Print(y) → 4
```

Eg: 2 3 4 7 9
Mid

Eg: $\frac{(5-4)^2+(5-2)^2+(5-3)^2+(5-9)^2+(5-7)^2}{5}$

5 → Mean.

```
y = np.std(x)
Print(y) → 2.607
```

$\boxed{\text{Std} = \sqrt{\text{var}}}$

Pandas

→ Pandas store tabular data using a dataframe.
→ A dataframe is a 2-D labelled data structure like a table in data base.
→ Every dataframe contains rows and columns and therefore has both a row and column index.
→ Each column can have a different types of values.

Eg:

```
import pandas as pd
std_data = [(1, 'varun', 30, 'male', 'Chandigarh'),
            (2, 'ravi', 31, 'male', 'delhi'),
            (3, 'Preeti', 29, 'female', 'Jaipur')]

df = pd.DataFrame(std_data, columns =
        ['stud_id', 'name', 'age',
         'gender', 'location'])

df
```

O/P:-

	stud_id	name	age	gender	location
0	1	varun	30	male	Chandigarh
1	2	Ravi	31	male	delhi
2	3	Preeti	29	female	Jaipur

Eg:
```
import pandas as pd
df = pd.read_csv("student.csv")
df
```

→ df.head() :— Display TOP 5 Rows.
→ df.head(2) :— " " 2 Rows
→ df.tail() → Display Last 5 Rows.
→ df.shape() → No. of Rows and No. of columns.
→ df.columns() → Display all column names.
→ df.age → Display age value.
df['age', 'address'] → Display age, address.
→ df.size → Total No. of values in the dataframe.
→ df.dtypes → Display data-type of each columns.

Eg:
```
stud_id    int64
name       object
age        int64
```

→ df.values → Display all the values in list format.
→ df.index → Display index value.

Eg:
```
RangeIndex (start=0, stop=2, step=1)
```

Operation on Pandas :—

Select a single Row by index value :—

```
df.loc[0]
df.iloc[0]
```

Select multiple rows by index label:—

```
df.loc[[0,2,4]]
df.iloc[0,2]
```

Filtering Rows:—

```
df[df['age']>29]
```

Adding a new column:—

```
df['Phone_no'] = [10,20,30,40,50]
df.insert(3, 'Phone_no', [10,20,30,40,50])
print(df)
```

Deleting a column:—

```
df = df.drop(columns = ['Phone_no'])
```

Rename:—

```
df = df.rename(columns={'old name':'new name'})
df = df.rename(columns={'age':'student_age'})
```

Deleting a column:—

```
del df['Phone_no']
df
```

Deleting a Row:—

```
df = df.drop(4)
```

Adding a new Row:—

```
df.loc[4] = [5, 'Pinki', 28, 'female', 'bengluru']
df
```

updating the values:—

```
df.loc[2, 'student_age'] = 71
df
```

updating multiple values:—

```
df.loc [[0,2], 'address'] = ['anyaman', 'nilobari']
df
```