```
python -m venv numpy_env
source numpy_env/bin/activate
pip install numpy
```

```
import numpy as np
a = np.array([1,2,3])
print(a)
print(type(a))
```

dtype -> numpy ndarray

Array & Matrix

```
my_matrix = [[1,2,3],[4,5,6],[7,8,9]]
print(my_matrix)
print(type(my_matrix))
b = np.array(my_matrix)
print(b)
print(type(b))
m = np.matrix(my_matrix)
print(m)
print(type(m))
```

To print number of dimension `.ndim`

Zeros
`print(np.zeros(3))

Multi-dimensional Array: Provide a tuple of integers (rows, columns).
`print(np.zeros((3,3)))`

Ones

`print(np.ones(3,dtype='int'))`

Arange

`print(np.arange(3,34,2))`
start,stop,step

`print(np.linspace(1,50))` linear spacing from start to stop with 50

Identitiy Matrix

```
identity_matrix = np.eye(3)
print(identity_matrix)
print(identity_matrix.shape)
```

`

Translation AI = IA = A

Operation in Number

```
number_list = [1,2,3,4]
print(number_list *2)

numpy_list = np.arange(1,5)
print(numpy_list *2)
```

Reshape

```
arr = np.arange(25)

print(arr.reshape(5,5))
print(arr)
```

Random Number Generation

```
print(np.random.rand()*10)

#Returns from Gaussian Distribution
print(np.random.randn())

print(np.random.randint(1,100,10))
```

Shallow Copy

```
new_list = [1,2,3,4]
new_copied_list = new_list
new_copied_list +=[1]
print(new_copied_list,new_list)
copied_list =new_list.copy()
print(id(new_list))
print(id(new_copied_list))
print(id(copied_list))
```

```
import numpy as np

arr = np.array([1, 2, 3, 4])
arr_copied = arr

arr_copied += 1

print(id(arr_copied))
print(id(arr))
```

`.copy()` is a shallow copy

Slicing & Broadcasting

```
import numpy as np
ranarr = np.random.randint(0,50,10)
print(ranarr)
print(ranarr.max())
print(ranarr.argmax())
print(ranarr[0:3])
```

```
#Broadcasting
ranarr[0:3] = 0
print(ranarr)
nan = ranarr/ranarr
print(nan)
print(np.isnan(nan))
```

2d multiplication

```
import numpy as np
x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])
v = np.array([9,10])
w = np.array([11, 12])
print(x+v)
print(v.dot(w))
print(v@w)
print(y*x)
print(np.multiply(v,w))
print(np.multiply(x,y))
print(np.multiply(x,y))
print(np.dot(x,y))
```

Pandas

```
labels = ['Anish','Sanam','Puja']
my_list = [1,4,5]
arr = np.array([1,4,5])
d = {'Anish':1,'Sanam':4,'Puja':5}

print(pd.Series(data=my_list))
print(pd.Series(data=arr))
print("Familarity With Python")
ser =pd.Series(data=my_list,index=labels)
print(ser)
print(ser['Anish'])
```

Addition of series

```
import pandas as pd

jan_sales = pd.Series(
    [1200, 950, 700, 1100],
    index=['USA', 'Germany', 'Italy', 'Japan']
)

feb_sales = pd.Series(
```

```
    [1350, 1000, 800, 1250],
    index=['USA', 'Germany', 'France', 'Japan']
)

total_sales = jan_sales + feb_sales

print(total_sales)
```

pd.to_datetime

```
import pandas as pd

ser = pd.Series(['01 Jan 2026', '02-02-2026', '20260303', '2026/04/04',
'2026-05-05', '2026-06-06T12:20'])

print(ser)

print(pd.to_datetime(ser))
```

implication

```
import pandas as pd
import numpy as np
import numpy as np
import pandas as pd

df = pd.DataFrame(
    np.random.randint(0, 50, size=(5, 4)),
    index=['Player', 'Enemy', 'Tree', 'House', 'Car'],
    columns=['X_Position', 'Y_Position', 'Scale', 'Rotation_Degrees']
)

df['new'] = df['X_Position'] + df['Y_Position']
df.drop('new',axis=1,inplace=True)
print(df.loc['Player'])
print(df.loc[['Player', 'Enemy'], ['Scale', 'Rotation_Degrees']])
#Give me rows where Scale > 4, and show only Scale and X_Position
print(df.loc[df['Scale'] > 4, ['Scale', 'X_Position']])
```

#Drop

```
df['new'] = df['X_Position'] + df['Y_Position']
```

```
print(df.drop('new',axis=1,inplace=True))
print(df)
```

`print(df.loc[['Player', 'Enemy'], ['Scale']])`

Rows:
0 = Player
1 = Enemy
2 = Tree
3 = House
4 = Car

Columns:
0 = X_Position
1 = Y_Position
2 = Scale
3 = Rotation_Degrees

`.iloc` when we know row and col position

```
print(df.iloc[0:3, 1:4])
```

SALARIES.CSV

```
sal = pd.read_csv('Salaries.csv')
print(sal.info())
print(sal['BasePay'])
print(sal.loc[0,'BasePay'])
print(sal.iloc[0,1])
print(sal.iloc[:,1])
```

```
print(sal['BasePay'].isnull().sum())
sal['BasePay']=sal['BasePay'].fillna(sal['BasePay'].mean())
print(sal['BasePay'].isnull().sum())

print(sal['OvertimePay'].max())
print(sal['OvertimePay'].min())

print(sal[sal['EmployeeName']=='ANISH MANANDHAR']['JobTitle'])
print(sal.loc[sal['EmployeeName'] == 'ANISH MANANDHAR', 'JobTitle'])
print(sal['JobTitle'].value_counts().tail())
```

```
print(len(sal[sal['JobTitle'] == 'Transit Operator']))
```