Lab 3.1

Introduction to Numpy module:

NumPy is a powerful Python library for numerical computing, providing support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions to operate on these arrays efficiently. It serves as the foundation for many scientific and mathematical Python libraries due to its speed and versatility. NumPy offers capabilities for array manipulation, mathematical operations, linear algebra, random number generation, and more, making it an essential tool for data manipulation and analysis in fields such as machine learning, data science, and engineering. With its efficient implementation in C, NumPy facilitates fast computations and enables seamless integration with other Python libraries and tools. Its simplicity and performance make it a cornerstone of the Python scientific computing ecosystem.

Installation

We can easily install this library using the terminal. Follow the steps given below to install Numpy: Step 1: Go to the Windows PowerShell. Step 2: We can install Numpy with the help of pip easily. Head over to the directory where you want to install the numpy and fire the below command: **pip install numpy OR conda install numpy** This is very simple to install. tall.

import numpy

Once numpy is installed, import it in your applications by adding the import keyword: **import numpy as pd** Now numpy is imported and ready to use.# se.

```
import numpy as np
```

Creating a list

```
mylist=[1,3,5,6]
```

Checking the type

```
type(mylist)
```

```
list
```

Converting the simple list to numpy array

```
np.array(mylist)
```

```
array([1, 3, 5, 6])
```

```
type(mylist)
```

```
list
```

What will happen in the above code ??

```
arr=np.array(mylist)
```

```
arr
```

```
array([1, 3, 5, 6])
```

Creating matrix

```
mylist=[[1,3,5],[4,5,6],[1,4,76],[1,3,5]]
```

```
mylist
```

```
[[1, 3, 5], [4, 5, 6], [1, 4, 76], [1, 3, 5]]
```

```
np.array(mylist)
```

```
array([[ 1,  3,  5],
       [ 4,  5,  6],
       [ 1,  4, 76],
       [ 1,  3,  5]])
```

```
mymatrix=np.array(mylist)
```

```
mymatrix.shape
```

```
(4, 3)
```

```
mynewmatrix=np.array(mylist)
mynewmatrix
```

```
array([[ 1,  3,  5],
       [ 4,  5,  6],
       [ 1,  4, 76],
       [ 1,  3,  5]])
```

Creating the numpy array by using range function

```
np.arange(0,15)
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

```
np.arange(0,10,2)
```

```
array([0, 2, 4, 6, 8])
```

Creating the numpy array of only Zeros by using range function

```
np.zeros(5)
```

```
array([0., 0., 0., 0., 0.])
```

```
type(0.)
```

```
float
```

Creating the matrix of only Zeros

```
np.zeros((4,10))
```

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

Creating the matrix of only sixes by incrementing of 5 in 1

```
np.ones((5,5)) +5
```

```
array([[6., 6., 6., 6., 6.],
       [6., 6., 6., 6., 6.],
       [6., 6., 6., 6., 6.],
       [6., 6., 6., 6., 6.],
       [6., 6., 6., 6., 6.]])
```

```
np.ones(4)*100
```

```
array([100., 100., 100., 100.])
```

```
np.ones(4)/100
```

```
array([0.01, 0.01, 0.01, 0.01])
```

Creating of array by using linespace function

The general syntax of numpy.linspace() is numpy.linspace(start, stop, num=50). It generates an array of evenly spaced numbers over a specified interval [start, stop], inclusive of both endpoints, with the number of elements determined by num.

```
np.linspace(0,30,3)
```

```
array([ 0., 15., 30.])
```

```
np.linspace(0,10,20)
```

```
array([ 0.        ,  0.52631579,  1.05263158,  1.57894737,
2.10526316,
        2.63157895,  3.15789474,  3.68421053,  4.21052632,
4.73684211,
        5.26315789,  5.78947368,  6.31578947,  6.84210526,
7.36842105,
        7.89473684,  8.42105263,  8.94736842,  9.47368421,
10.        ])
```

```
len(np.linspace(0,10,20))
```

```
20
```

Creating of matrix by eye function which has 1 on the diagnol

```
np.eye(5)
```

```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

numpy arrys

it will produce the random numbers

```
np.random.rand(1)
```

```
array([0.35754582])
```

```
np.random.rand(3,5)
```

```
array([[0.10514096, 0.81829136, 0.67571431, 0.4998239 , 0.56605004],
       [0.95347943, 0.56676474, 0.11846972, 0.66287605, 0.3576046 ],
       [0.29996218, 0.02434369, 0.32077685, 0.03250476, 0.81041917]])
```

```
np.random.randn(5)
```

```
array([-0.61021583, -1.12172603,  0.32826695, -0.06517884,
0.6730479 ])
```

```
np.random.randint(1,100,5)
```

```
array([49, 92, 10, 31, 62], dtype=int32)
```

```
np.random.seed(24)
```

```
np.random.rand(4)
```

```
array([0.9600173 , 0.69951205, 0.99986729, 0.2200673 ])
```

```
np.random.rand(4)
```

```
array([0.36105635, 0.73984099, 0.99645573, 0.31634698])
```

```
np.random.rand(4)
```

```
array([0.13654458, 0.38398001, 0.32051928, 0.36641475])
```

```
arr=np.arange(25)
```

```
arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
16,
       17, 18, 19, 20, 21, 22, 23, 24])
```

```
arr1=np.random.randint(0,50,16)
```

```
arr1
```

```
array([25, 35, 45, 31, 28,  0, 12, 31, 33,  1,  6, 33, 17,  6, 46,
1],
      dtype=int32)
```

```
arr1.shape
```

```
(16,)
```

```
arr1.reshape(4,4)
```

```
array([[25, 35, 45, 31],
       [28,  0, 12, 31],
       [33,  1,  6, 33],
       [17,  6, 46,  1]], dtype=int32)
```

Finding the maximum and minimum value

```
arr1.max()
```

```
np.int32(46)
```

```
arr1.min()
```

```
np.int32(0)
```

```
arr1
```

```
array([25, 35, 45, 31, 28,  0, 12, 31, 33,  1,  6, 33, 17,  6, 46,
1],
      dtype=int32)
```

Finding the index of maximum and minimum

```
arr1.argmax()

np.int64(14)

arr1.argmin()

np.int64(5)

arr1.dtype

dtype('int32')
```

numpy indeximg and selection

Run and observe the following code

```
import numpy as np

arr=np.arange(11)

arr

array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])

arr[8]

np.int64(8)

arr[2]

np.int64(2)

arr[1:5]

array([1, 2, 3, 4])

arr[:]

array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])

arr[5:]
```

```
array([ 5,  6,  7,  8,  9, 10])
arr[:5]
array([0, 1, 2, 3, 4])
arr[4:7]
array([4, 5, 6])
arr[0:6]
array([0, 1, 2, 3, 4, 5])
arr
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
arr+100
array([100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110])
arr / 2
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. ])
arr+100 / 2
array([50., 51., 52., 53., 54., 55., 56., 57., 58., 59., 60.])
new_arr=arr / 2
new_arr
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. ])
arr
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
arr**2
array([  0,   1,   4,   9,  16,  25,  36,  49,  64,  81, 100])



arr
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

Slicing

```
slice_of_arr=arr[0:8]
```

```
slice_of_arr
```

```
array([0, 1, 2, 3, 4, 5, 6, 7])
```

```
slice_of_arr[:]
```

```
array([0, 1, 2, 3, 4, 5, 6, 7])
```

```
slice_of_arr[:]=100
```

```
slice_of_arr
```

```
array([100, 100, 100, 100, 100, 100, 100, 100])
```

```
arr
```

```
array([100, 100, 100, 100, 100, 100, 100, 100,   8,   9,  10])
```

```
arr_copy = arr.copy()
```

```
arr_copy[:]=100
```

```
arr_copy
```

```
array([100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100])
```

```
arr
```

```
array([100, 100, 100, 100, 100, 100, 100, 100,   8,   9,  10])
```

INDEXING on 2D array

Run and observe the following code

```
arr_2d=np.array([[1,45,5],[2,45,6],[7,8,9]])
```

```
arr_2d
```

```
array([[ 1, 45,  5],
       [ 2, 45,  6],
       [ 7,  8,  9]])
```

```
arr_2d.shape
```

```
(3, 3)
```

```
arr_2d[1]
```

```
array([ 2, 45,  6])
```

```
arr_2d[1][0]
```

```
np.int64(2)
```

```
arr_2d[1][1]
```

```
np.int64(45)
```

```
arr_2d[1,1]
```

```
np.int64(45)
```

```
arr_2d
```

```
array([[ 1, 45,  5],
       [ 2, 45,  6],
       [ 7,  8,  9]])
```

```
arr_2d[:2]
```

```
array([[ 1, 45,  5],
       [ 2, 45,  6]])
```

```
arr_2d[:2,1:]
```

```
array([[45,  5],
       [45,  6]])
```

Comparision of whole array

```
arr=np.arange(11)
```

```
arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
arr < 4
```

```
array([ True,  True,  True,  True, False, False, False, False, False,
       False, False])
```

```
arr > 4
```

```
array([False, False, False, False, False,  True,  True,  True,  True,
        True,  True])
```

Condition seletion

```
bool_arr= arr>4
```

```
bool_arr
```

```
array([False, False, False, False, False,  True,  True,  True,  True,
        True,  True])
```

```
arr[bool_arr]  #condition seletion
```

```
array([ 5,  6,  7,  8,  9, 10])
```

```
arr[arr>4]
```

```
array([ 5,  6,  7,  8,  9, 10])
```

```
arr[arr ==2]
```

```
array([2])
```

```
arr[arr <=4]
```

```
array([0, 1, 2, 3, 4])
```

numpy operation

```
import numpy as np
```

```
arr=np.arange(10)
```

```
arr
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
arr+100
```

```
array([100, 101, 102, 103, 104, 105, 106, 107, 108, 109])
```

```
arr/2
```

```
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])
```

```
arr**2
```

```
array([ 0,  1,  4,  9, 16, 25, 36, 49, 64, 81])
```

```
(arr+2)/2
```

```
array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. , 5.5])
```

```
(arr+2)-2
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
arr+arr
```

```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
```

```
arr/0
```

```
C:\Users\PMLS\AppData\Local\Temp\ipykernel_97520\4291252909.py:1:
RuntimeWarning: divide by zero encountered in divide
```

```
  arr/0
C:\Users\PMLS\AppData\Local\Temp\ipykernel_97520\4291252909.py:1:
RuntimeWarning: invalid value encountered in divide
  arr/0

array([nan, inf, inf, inf, inf, inf, inf, inf, inf, inf])

1/arr

C:\Users\PMLS\AppData\Local\Temp\ipykernel_97520\255282349.py:1:
RuntimeWarning: divide by zero encountered in divide
  1/arr

array([       inf, 1.        , 0.5       , 0.33333333, 0.25      ,
       0.2       , 0.16666667, 0.14285714, 0.125     , 0.11111111])

arr/arr #bcoz zero are presnt in the array

C:\Users\PMLS\AppData\Local\Temp\ipykernel_97520\91700794.py:1:
RuntimeWarning: invalid value encountered in divide
  arr/arr #bcoz zero are presnt in the array

array([nan,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])
```

Some mathimathical operations

```
np.sqrt(arr)

array([0.        , 1.        , 1.41421356, 1.73205081, 2.        ,
       2.23606798, 2.44948974, 2.64575131, 2.82842712, 3.        ])

np.log(arr) #bcz first no is zro its makes it infinity

C:\Users\PMLS\AppData\Local\Temp\ipykernel_97520\3565305275.py:1:
RuntimeWarning: divide by zero encountered in log
  np.log(arr) #bcz first no is zro its makes it infinity

array([      -inf, 0.        , 0.69314718, 1.09861229, 1.38629436,
       1.60943791, 1.79175947, 1.94591015, 2.07944154, 2.19722458])

np.sin(arr)

array([ 0.        ,  0.84147098,  0.90929743,  0.14112001, -
0.7568025 ,
       -0.95892427, -0.2794155 ,  0.6569866 ,  0.98935825,
0.41211849])

arr.sum()

np.int64(45)

arr.max()
```

```
np.int64(9)
```

```
arr.mean()
```

```
np.float64(4.5)
```

```
arr_2d=np.array([[1,2,3],[2,2,3],[34,5,7]])
```

```
arr_2d
```

```
array([[ 1,  2,  3],
       [ 2,  2,  3],
       [34,  5,  7]])
```

```
arr_2d.sum()
```

```
np.int64(59)
```

```
arr_2d.shape
```

```
(3, 3)
```

```
arr_2d.sum()
```

```
np.int64(59)
```

```
# give me sum acroos the rows
arr_2d.sum(axis=0)
    #array  along the vertical axis when axis=0 that count colum
element
  #mean that produce the sum of the coluam
```

```
array([37,  9, 13])
```

```
arr_2d.sum(axis=1)
```

```
array([ 6,  7, 46])
```

numpy exericses

```
import numpy as np
```

create an array of 10 zeros

```
arr=np.zeros(10)
```

```
arr
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

create an array of 10 ones

```
arr=np.ones(10)

arr
```

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

create an array of 10 fives

```
arr=np.ones(10)*5

arr
```

```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

create an array of integer from 10 to 50

```
arr=np.arange(10,51)

arr
```

```
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])
```

create an array of all even integer from 10 to 50

```
arr=np.arange(10,51,2)

arr
```

```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40,
42,
       44, 46, 48, 50])
```

create 3x3 matrix with the values ranging of 0 to 8

```
arr=np.array([[0,1,2],[3,4,5],[6,7,8]])

arr
```

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])

matrix = np.arange(9).reshape((3, 3))
```

```
print(matrix)
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

```
matrix = np.arange(16)
matrix.reshape((4,4))
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15]])
```

create 3x3 identity matrix

```
identity_matrix = np.identity(3)
identity_matrix
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

```
print(identity_matrix)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

```
np.eye(3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

```
arr=np.random.rand(1)
```

```
arr
```

```
array([0.56243404])
```

```
random_matrix = np.random.randn(25)
```

```
random_matrix
```

```
array([-2.1628745 ,  0.1732177 ,  0.44980691, -0.15639909, -
0.5017551 ,
       -1.13070873, -2.29896186,  0.69752566, -1.17402782, -
0.73272521,
       -1.984332  ,  0.26950264, -0.77941255,  0.8706628 , -
0.32149986,
        1.9582427 ,  1.79617509,  1.352329  , -0.38124668,
```

```
0.36649793,
        0.99719722,  0.390556  , -0.16061569, -0.07517978, -
2.83762557])

arr=np.arange(1,101)

arr

array([  1,   2,   3,   4,   5,   6,   7,   8,   9,  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,  51,
52,
        53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
65,
        66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
78,
        79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
91,
        92,  93,  94,  95,  96,  97,  98,  99, 100])

arr=np.arange(1,101)/100
arr.reshape(10,10)

array([[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ],
       [0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ],
       [0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 ],
       [0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 ],
       [0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 ],
       [0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 ],
       [0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7 ],
       [0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8 ],
       [0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9 ],
       [0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.  ]])

arr

array([0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ,
0.11,
       0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21,
0.22,
       0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32,
0.33,
       0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43,
0.44,
       0.45, 0.46, 0.47, 0.48, 0.49, 0.5 , 0.51, 0.52, 0.53, 0.54,
0.55,
       0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64, 0.65,
```

```
0.66,
       0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75, 0.76,
0.77,
       0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87,
0.88,
       0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98,
0.99,
       1.  ])
```

create an arry of 20 linearly spaced point b/w 0 and 1

```
np.linspace(0,1,20)
```

```
array([0.        , 0.05263158, 0.10526316, 0.15789474, 0.21052632,
       0.26315789, 0.31578947, 0.36842105, 0.42105263, 0.47368421,
       0.52631579, 0.57894737, 0.63157895, 0.68421053, 0.73684211,
       0.78947368, 0.84210526, 0.89473684, 0.94736842, 1.        ])
```

numpy indexing and selection

```
mat=np.arange(1,26).reshape(5,5)
```

```
mat
```

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

```
mat[:3,1:2]
```

```
array([[ 2],
       [ 7],
       [12]])
```

```
mat[2:]
```

```
array([[11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

```
mat[2:,1:]
```

```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

```
mat[3,4]
```

```
np.int64(20)
```

```
mat[4:]
```

```
array([[21, 22, 23, 24, 25]])
```

```
mat[3:]
```

```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

```
mat.sum()
```

```
np.int64(325)
```

```
mat.std()
```

```
np.float64(7.211102550927978)
```

```
mat.sum(axis=0)
```

```
array([55, 60, 65, 70, 75])
```

```
np.random.seed(101)
```

```
np.random.rand(1)
```

```
array([0.51639863])
```