Q1: NumPy stands for?

A: Num= numerical Py= Python

Q2:NumPy developed by?

A:Travis Oliphant

Q3:Where Numpy data is stored?

A:nd-Array= n-dimensional Array

Q3:Advantage?

A:

1. We (+,\*) Array with single line.
2. It’s Vectorized
3. Reading writng of array could be saved in it.
4. Igon value can be calculated.
5. Stores data in contigous Block of Memory

Q4: Vectorized operation?

A:Every Index Array Would Multiply with another at once

Q5:Contigous block?

A:Stores data in sequence

Q6:Link list is Contigous or Non-Contigous?

A:Data Stored randomly in an array.(Non-contigously)

Q7:Why %time used for?

A:It gives execution time of a line

Q8:numpy methods?

* np.zeros((4,4)):

Creates an (4X4) array where all elements are 0.

* np.ones((4,4))

Creates an (4X4) array where all elements are 1.

* np.empty((7,7))

Creates an (7X7) array where any elements can be placed.

* np.array(( 4,4))

Converts list into array.

* range():python method
* arange(par1,par2,par3):numpy method

1. Par1:inital range
2. Par2:final range
3. Par3:step size(number after which first index number starts.

* Np.shape()

Return length of shape

* Var.dtype()

Gives data type of variable.

* Var.ndim

Gives dimension of variable.

Q8:ARRAY Airthmetics?

A:

* A=np.random.randn(10):

Creates array randomly and assign it in A.

* A[w>0]

If w>0 passed in index then values are stored in w is compared with >0 and saved in A

* **Fancy indexing:**

**a=[1,3,5,7,9]**

**A=a[[4,2]]**

A beacomes [9,5]

* **np.sqrt(list/array):**

Takes square root of list/array and returns an array.

* **np.power(list/array,power\_variable):**

1. **List/array**: takes a list which need to be powered.
2. **Power\_variable**:the power which has to be raised.
3. Return type : array of power operation.

* **Np.maximum(array1,array2):**

1. **Array1:**takes first array to be compared.
2. **Array2:**takes second array to be compared.
3. **Return type:**return type would be the max array.

* **np.where(condition,option1,option2):**

1. **Condition:(**arr>0) specifies a condition of array

(array ki condition ko btana k agar number >0 aye to ye kaam kro array k index py)

1. **Option1:** runs if condition true
2. **Option2:**runs if condition false

* **Numerical and statistical methods:**

1. **Np.mean():**

Returns mean of an array/list in float.

1. **Np.cumsum():**

return cumulative sum of an array/list

Return Type would be array

**Commulative sum:**sum with all previous elements in an array.

1. **Np.cumprod():**

return type would be commulative product of array/list.

Return Type would be array

**Commulative product:**product with all previous elements in an array.

* **Boolean array:**

1. **np.sum():**

Counts number of true in a boolean array.

1. **Np.any():**

Test whether one or more values in an array is true.

1. **Np.all():**

Checks if all values in an array is true

* **Np.unique():**

Returns unique element in a list/array

* **Np.sort():**

Return sorted list in an array.

* **FILE INPUT/OUTPUT:**

1. **np.save(filename.npy,array):**

Used for saving a single array in a file having extension “.npy”

1. **Np.load(filename.npy):**

used for loading a file filename.npy and returns an array

1. **Np.savez(filename.npz,array1,array2):**

**X=[1,2,3] y=[4,6,1]**

**E.g:** np.savez(test.npz,a=x,b=y)

Save an arrays in the form of dictionary having a keys a and b and values as x and y.

If you want to retrive x then write as follows:

Z=np.load(“test.npz”)

Z[a] Output:[1,2,3]

Z[b] Output:[4,6,1]

* **Np.linalg.fn():**

**Linalg:**linear algorithm

**Fn(): fn includes the following things**

a=[[1,2],[,3,4]]

1. **Det:**

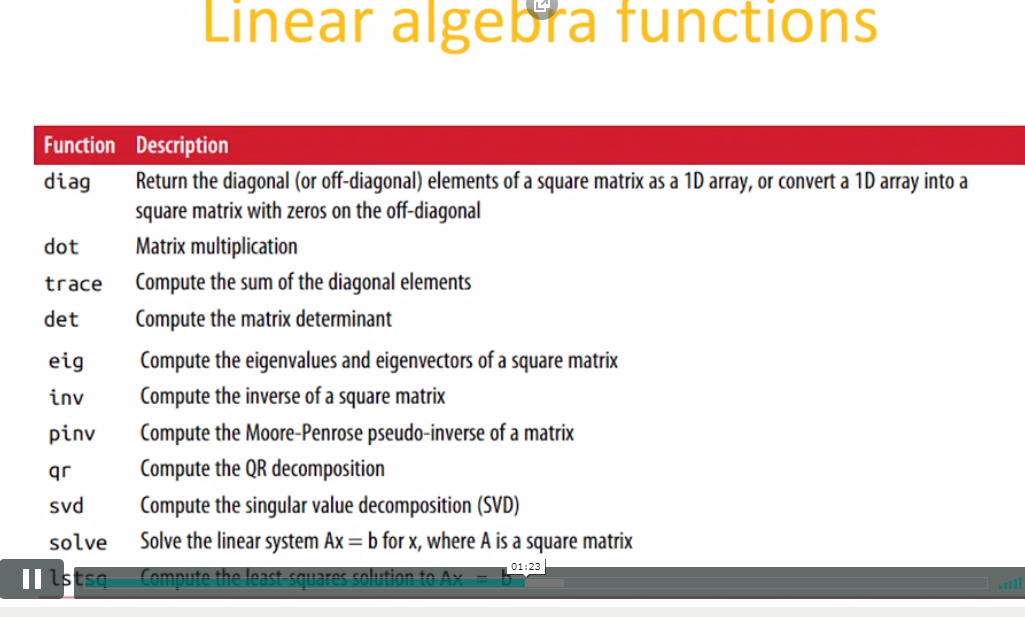
**E.G:** b=np.linalg.det(a)

It returns determinent of a

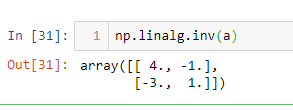
1. **Trace:**

**E.g:** b=np.trace(a)

Return sum of all daignols.



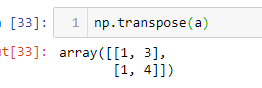
1. **Np.linalg.inv(b):**



Returns inverse of matrix.

1. **Np.Transpose():**

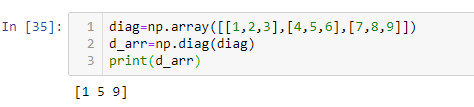
Generate transpose of matrix.



1. **Np.diag(a):**

**b=np.diag(a)**

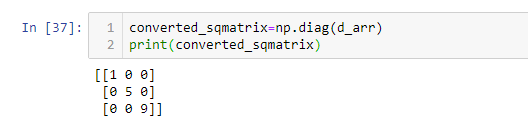
Create the array of daignols of matrix a.and assign it in b.



1. **Np.daig(b)**

**Diag\_sqmat=np.diag(b)**

Convert the 1D array into daignols of square matrix. Having upper and lower triangles with zero.

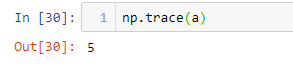


1. **Np.trace(a):**

Calculate the sum of daignols of matrix a.

**E.g:sum=np.trace(a)**

**Sum=5**



**NOTE:**conditions(>,<,<= etc) are not supported in list.