|  |  |
| --- | --- |
| >>np.linalg.eig(np.array([[1, -1], [1, 1]]))  Return two matrices – Eigen value, Eigen Vector  >>np.linalg.svd(a = np.random.randn(9, 6))  Return 3 vectors | Eigen decomposition.  Singular Value Decoposition. |

Input and Output

|  |  |
| --- | --- |
| **Saving and Loading numpy arrays**  >> np.save('outfile', x)  >> y = np.load('outfile.npy')  >>np.savetxt('textfile.out',x) | Saving as .npy file  Loading .npy file  Saving in a textfile. |

|  |  |
| --- | --- |
| >> d = c.view()  [11, 12, 13]  >> f = e.copy()  [21, 22, 23] | Creating a shallow copy.  Creating a deep copy. |

Linear Algebra Module

Copies and Views

|  |  |
| --- | --- |
| **Sorting**  >> np.sort(a)  array([4,5], [1,3])  >> np.argsort(a)  array([1,0],[1,0)  **Searching**  >> argmax(a)  0  >> np.argwhere(a>=2)  Array([0,0],[0,1],[1,1])  **Counting**  >>np.count\_nonzero(a)  array([0,0],[0,1],[1,0],[1,1]) | >> By default by rows sorting.  >> Sorting, returning index sorted.  >> Returning the index of max value.  Returning index of where condition matches.  Returning index of non-zero values. |

Sorting, Searching and Counting

|  |  |
| --- | --- |
| >> a>2  array([True, False, True])  >> a ==b  array([True, False, True])  >> np.bitwise\_and(13, 17)  1  >>np.invert(array(True, False))  array([False, True]) | Condition checking on arrays elementwise.  Elementwise comparision.  Performing bitwise and on bits of number.  Inverting or negating values. |

Comparision and Boolean Operators

Statistical Functions

Trignometric Functions

|  |  |
| --- | --- |
| >>np.matmul(A.reshape((3,2)),B)  array([[ 9, 12, 15], [19, 26, 33],  [29, 40, 51]]) | Matmult-iplication. |

|  |  |
| --- | --- |
| >> np.sin(np.pi /2.)  1.0  >> np.deg2rad(180)  3.1415 | Sine trigonometric operation.  Convert degree to radians |

|  |  |
| --- | --- |
| >> np.amin(a, axis =0)  array([0,1])  >> np.perctentile(a, q=50)  2.5  >> np.median(a)  2.5  >> np.var(a)  2.16666 | Finding minimum along any axis.  Calculating qth percentile.  Median of data  Variance of data/arrays. |

|  |  |
| --- | --- |
| >> c = a+b  array([[ 2, 4, 6], [ 8, 10, 12]])  >> np.add(a,b)  array([[ 2, 4, 6], [ 8, 10, 12]])  >> c = a-b  array([[0, 0, 0], [0, 0, 0]])  >> np.subtract(a,b)  array([[0, 0, 0], [0, 0, 0]])  >> a/b  array([[1,1,1], [1,1,1]])  >> np.divide(a,b)  array([[1,1,1], [1,1,1]])  >> a\*b  array([[ 1, 4, 9], [16, 25, 36]])  >> np.multiply(a,b)  array([[ 1, 4, 9], [16, 25, 36]])  >> np.dot(a.reshape(3,2),b)  array([[ 9, 12, 15], [19, 26, 33],  [29, 40, 51]]) | Addition of arrays  Addition of arrays using numpy function.  Subtraction of arrays  Sbtraction of arrays using numpy function.  Division of arrays  Division of arrays using numpy function.  Elementwise multiplication.  Elementiwise multiplication using numpy function.  Dot product of arrays. |

Arithmetic Operators

Matrix and Vector Product

Array Operators and Functions

|  |  |
| --- | --- |
| **Changing Shape**  >> a.reshape(2,3)  array([[0,1,2],[3,4,5]])  >> a.ravel()  array([0,1,2,3,4,5])  >>a.flatten()  array([0,1,2,3,4,5])  **Transpose Operation**  >> np.transpose(a)  array([0, 2],[1, 3])  >> a.T  array([1,3],[2,4])  **Combining Arrays**  >> np.vstack((a,b))  array([1,2],[3,4])  >> np.hstack((a,b))  array([1,2,3,4])  **Broadcasting Arrays**  >>np.ones((3,2))/np.ones((1,2))  array([[1,1],[1,1],[1,1]]) | Gives new shape to array.  Contiguous flattened array.  Copy of the array collapsed into 1-D  Reverses the dimensions.  Same as transpose  Stacking arrays vertically.  Stacking arrays  horizontally.  Broadcast smaller array if not of same size automatically. |

Array Manipulation

|  |  |
| --- | --- |
| **Indexing**  >> a[1]  2  >> a[1,2]  7  **Slicing**  >> a[0:3]  array((1,2,3))  >> a[0:2,2:]  array([[2,3],[3,4]  ])  **Boolean Indexing**  >> x[x>10]  array([11,12,13,14,15]) | Single element indexing  Multidimensional Indexing  Slicing on 1-D array.  Slicing with selected columns.  Using the Boolean i.e, True and False by applying condition. |

|  |  |
| --- | --- |
| >> for x in np.nditer(a):  print(x) | Iterating over each value. |

Iterating over Arrays

Slicing and Indexing

Array Attributes

a is a some created numpy array.

|  |  |
| --- | --- |
| >> a.ndim  >> a.shape  >> a .size  >> a.dtype  >> a.data  >> | Return no. of axes  Return shape  Return no. of elements  Data type of elements  Return memory location. |

Creating arrays

* Numpy’s array class is called ndarray.
* The rank 1 array is like – [2,3,4], it is similar to a 1-D array

|  |  |
| --- | --- |
| >> np.array([1.,2.,3.])  >> np.array([1.,2.,3.]) | Creating 1-D array  Creating 2-D array |

* Conversion from other Python structures (e.g. lists, tuples).
* Intrinsic NumPy array creation objects (e.g. arange, ones, zeros, etc).
* Use of special library functions (e.g. random).

|  |  |
| --- | --- |
| >> np.array([2,3,1,0])  >> np.zeros((3, 3))  >> np.arrange(10)  >> np.eye(2)  >> np.random.rand((3,3)) | Creating array with list  Creating array with zeros(intrinsic approach).  Creating array with arrange.  It will generate numbers btw 0 to 9 with equal steps.  2\*2 Identity matrix  Random matrix |

NumPy Arrays (ndarray)

Installing and importing numPy to use

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object and tools for working with these arrays.

>> pip install numpy

>> import numpy as np

Introduction to NumPy