|  |  |  |  |
| --- | --- | --- | --- |
| NUMPY  Introduction  NumPy is a package in Python used for Scientific Computing. NumPy package is used to perform different operations. The ndarray (NumPy Array) is a multidimensional array used to store values of same datatype. NumPy arrays are faster compared to Python lists.  Numeric, the ancestor of NumPy, was developed by Jim Hugunin. NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.  Features of Numpy  NumPy stands on CPython, a non optimizing bytecode interpreter. Multidimensional arrays. Functions & operators for these arrays Python alternatives to MATLAB. ndarray- n-dimensional arrays. Fourier transforms & shapes manipulation. Linear algebra & random number generation.  Instructions to follow before work with Numpy   * Install Numpy: pip install numpy * We have to import numpy package before we do any operations with it such as import numpy as np(np stands for numerical python).     Let’s see the various operations in Numpy as described below | | | |
| Methods/Functions | **Description** | **Example** | **Output** |
| Numpy Array Operations | | | |
| np.array([]) | Creating an Array | import numpy as np  a=np.array([1,2])  a | array([1, 2]) |
| a.dtype.name | It prints datatype of an array | a=np.array([[1,2,3]])  print(a.dtype.name) | int32 |
| np.shape() | It prints shape of an array | a=np.array([[1,2,3]])  print(np.shape(a)) | (1, 3) |
| np.ndim() | It prints dimensions of an array | a=np.array([[1,2,3]])  print(np.ndim(a),"dimensions of an array") | 2 dimensions of an array |
| np.size() | It prints size of an array(no,of elements present in it) | a=np.array([[1,2,3]])  print(np.size(a)) | 3 |
| a.itemsize | It prints itemsize of an array(bytes allocated to elements) | a=np.array([[1,2,3]])  print(a.itemsize,"no of elements present in array") | 4 no of elements present in array |
| np.arange().reshape() | Arange used to print an array, where as reshape is used to print that array in columns and rows(which are defined by user) | xy=np.arange(6).reshape(2,3)  xy | array([[0, 1, 2],  [3, 4, 5]]) |
| np.zeros() | It prints all zeros(0’s) | np.zeros((2,3)) | array([[0., 0., 0.],  [0., 0., 0.]]) |
| np.ones() | It prints all ones(1’s) | np.ones((2,2)) | array([[1., 1.],  [1., 1.]]) |
| np.eye() | It prints all ones in diagonal rest of zeros | np.eye(2) | array([[1., 0.],  [0., 1.]]) |
| np.identity() | It prints all ones in diagonal rest of zeros | np.identity(2) | array([[1., 0.],  [0., 1.]]) |
| np.full() | It fill the array with given value by user | np.full((2,3),2) | array([[2, 2, 2],  [2, 2, 2]]) |
| np.diag() | It prints diagonal elements which are defined by users | np.diag((1,2)) | array([[1, 0],  [0, 2]]) |
| np.tile() | It works like full function. | np.tile(1,(2,1)) | array([[1],  [1]]) |
| np.linspace() | It prints the values in the range between first 2 elements with equal distributions of 3rd element | np.linspace(1,2,3) | array([1. , 1.5, 2. ]) |
| a.copy() | It copies the values to another variable | a=np.arange(2)  b=a.copy()  b | array([0, 1]) |
| np.vstack() | It adds multiple arrays in vertical manner | import numpy as np  za=np.arange(2)  za=zb  zb  print(np.vstack((za,zb))) | [[0 1]  [0 1]] |
| np.hstack() | It adds multiple arrays in horizontal manner | import numpy as np  za=np.arange(2)  za=zb  zb  print(np.hstack((za,zb))) | [0 1 0 1] |
| np.empty() | It prints an empty array | np.empty((2,2)) | array([[1.64029794e-321, 1.06811015e-306],  [1.24611470e-306, 9.80720103e-312]]) |
| a.astype() | It prints array in user defined type by using astype function | a=np.ones((2,3))  print(a.astype(np.int32)) | [[1 1 1]  [1 1 1]] |
| np.zeros\_like() | It converts all array elements into zeros | a=np.ones((2,2))  b=np.zeros\_like(a)  b | array([[0., 0.],  [0., 0.]]) |
| np.ones\_like() | It converts all array elements into ones | a=np.zeros((2,2))  b=np.ones\_like(a)  b | array([[1., 1.],  [1., 1.]]) |
| np.ravel() | It prints the values in vector form only(1-Dimension) | np.ravel(np.arange(3)) | array([0, 1, 2]) |
| b.T or b.transpose() | It will exchange row to columns and columns to rows | b=np.arange(3).reshape(3,1)  b.T | array([[0, 1, 2]]) |
| np.concatenate() | It’s used to add two arrays | import numpy as np  a=np.ones((2,3))  b=np.ones((2,3))  np.concatenate(([a,b])) | array([[1., 1., 1.],  [1., 1., 1.],  [1., 1., 1.],  [1., 1., 1.]]) |
| np.empty\_like() | It prints array with empty values | a=np.zeros((2,2))  b=np.empty\_like(a)  b | array([[0., 0.],  [0., 0.]] |
|  |  |  |  |
| Numpy Random Functions | | | |
| np.random.random() | It prints random values between 0 & 1 | np.random.random(3) | array([0.64422714, 0.57728582, 0.25844165]) |
| np.random.rand() | It prints random values between given range of values | np.random.rand(1,2) | array([[0.81862519, 0.74427457]]) |
| np.random.randn() | It prints random values between given range of values which includes both +ve & -ve values | np.random.randn(2,2) | array([[ 0.34496698, 1.33766199],  [-1.46096012, -0.78954135]]) |
| np.random.randint() | It prints any integer value in given range | np.random.randint(3,5) | 2 |
| np.random.random\_sample() | It prints random sample values between 0 & 1 | np.random.random\_sample(3) | array([0.87745279, 0.34167318, 0.12124852]) |
| np.random.sample() | It prints ouput same as above function | np.random.sample(1) | array([0.68604656]) |
| np.random.ranf() | It prints ouput same as above function | np.random.ranf(3) | array([0.56290382, 0.22575919, 0.98310597]) |
| bb.max() | It prints maximum value | bb=np.arange(3,7)  bb.max() | 6 |
| bb.argmax() | It prints index of maximum value | bb=np.arange(5)  bb.argmax() | 4 |
| bb.min() | It prints minimum value | bb=np.arange(3,7)  bb.min() | 0 |
| bb.argmin() | It prints index of minimum value | bb=np.arange(3,7)  bb.argmin() | 0 |
|  |  |  |  |
| np.argsort() | It prints the index of an element in sorting order | a=np.random.random(3)  print(a)  a.argsort() | [0.7289258 0.30441564 0.77966584]  array([1, 0, 2], dtype=int64 |
| np.asarray() | It works same like numpy array | np.asarray([1,2,3]) | array([1, 2, 3]) |
| np.asfortranarray() | It works same like numpy array | np.asfortranarray([[3,4],[5,6]]) | array([[3, 4],  [5, 6]]) |
| np.atleast\_1d() | It gives ouput in 1-D | np.atleast\_1d([1,2]) | array([1, 2]) |
| np.atleast\_2d() | It gives ouput in 2-D | np.atleast\_2d([1,2]) | array([[1, 2]]) |
| np.atleast\_3d() | It gives ouput in 3-D | np.atleast\_3d([1,2]) | array([[[1],  [2]]]) |
| np.row\_stack() | It adds the array in row wise | e=np.array([1,2,3])  f=np.array([5,6,7])  np.row\_stack((e,f)) | array([[1, 2, 3],  [5, 6, 7]]) |
| np.column\_stack() | It adds the array in column wise | np.column\_stack((e,f)) | array([[1, 5],  [2, 6],  [3, 7]]) |
| np.vstack() | It adds the array in vertical way | np.vstack((e,f)) | array([[1, 2, 3],  [5, 6, 7]]) |
| np.hstack() | It adds the array in horizontal way | np.hstack((e,f)) | array([1, 2, 3, 5, 6, 7]) |
| np.dstack() | It prints the output in 3-D though we define arrays in 1-d/2-d,from 4-d it gives the same dimension as define by user | d=np.dstack((e,f))  print(d,d.ndim) | [[[1 5]  [2 6]  [3 5]  ]] 3 |
| np.convolve() | \_\_\_\_\_\_\_\_\_\_\_\_\_ | np.convolve([1,2,3],[0,1,0.5]) | array([0. , 1. , 2.5, 4. , 1.5]) |
| np.bincount() | \_\_\_\_\_\_\_\_\_\_\_\_\_ | np.bincount([1,1,2,1,2,3]) | array([0, 3, 2, 1], dtype=int64) |
| np.correlate() | It gives the summation of an entire arrays by multiplying elements of one another | np.correlate(e,f) | array([60]) |
| np.cumsum() | It gives output by adding of each element with previous element | np.correlate(e,f) | array([ 1, 3, 6, 10, 15, 21, 26, 33], dtype=int32) |
| np.cov() | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | np.cov(e,f) | array([[1.66666667, 0.83333333],  [0.83333333, 0.91666667]]) |
| np.expand\_dims() | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | np.expand\_dims(e,axis=0) | array([[1, 2, 3, 4]]) |
| np.extract() | It extracts the last list/element/variable | np.extract([1,2],[4,6]) | array([4, 6]) |
| np.frombuffer() | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | np.frombuffer(a) | array([2.12199579e-314, 6.36598737e-314, 1.06099790e-313]) |
| np.full\_like() | It fills the array with user defined element | a=np.arange(6).reshape(-3,2)  b=np.full\_like(a,3)  b | array([[3, 3],  [3, 3],  [3, 3]]) |
| np.histogram() | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | np.histogram([2]) | (array([0, 0, 0, 0, 0, 1, 0, 0, 0, 0], dtype=int64),  array([1.5, 1.6, 1.7, 1.8, 1.9, 2. , 2.1, 2.2, 2.3, 2.4, 2.5])) |
| np.ndenumerate() | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | np.ndenumerate([1]) | <numpy.ndenumerate at 0x1ebf99837f0> |
| np.ndindex() | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | np.ndindex(1,2) | <numpy.ndindex at 0x1ebf99d1358> |
| np.nditer() | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | np.nditer([1]) | <numpy.nditer at 0x1ebf9960990> |
| np.ptp() | It prints the before value of last element | np.ptp([1,50]) | 49 |
| np.repeat() | It repeats the given element | np.repeat('hi',2) | array(['hi', 'hi'], dtype='<U2') |
| np.roll() | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | np.roll(1,3) | array(1) |
| np.roots() | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | np.roots([1,2]) | array([-2.]) |
| np.round\_() | It gives roundoff value | np.round\_(2.8) | 3.0 |
|  |  |  |  |
| np.sinc() | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | np.sinc(5) | 3.898171832519376e-17 |
| np.sort() | It gives the ouput in sort way | np.sort([1,5,3,0,7,6]) | array([0, 1, 3, 5, 6, 7]) |
| np.stack() | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | np.stack([4,3,4]) | array([4, 3, 4]) |
| np.all() |  | c=np.array([1,2,3,4])  np.all(c) | True |
| np.any() |  | np.any(c) | True |
| np.conj() | It gives conjugate numbers | np.conj(c) | array([1, 2, 3, 4], dtype=int32) |
| np.conjugate() | It gives conjugate numbers | np.conjugate(c) | array([1, 2, 3, 4], dtype=int32) |
| np.cumprod() | It gives product of each element with previous element | np.cumprod(c) | array([ 1, 2, 6, 24], dtype=int32) |
| np.nonzero() | It gives indexes | np.nonzero([1,5,8,7,5,6]) | (array([0, 1, 2, 3, 4, 5], dtype=int64),) |
| np.percentile() |  | np.percentile(c,2) | 1.06 |
| np.prod() | It gives the product of all elements | np.prod([1,2,3]) | 6 |
|  | | | |
| Statistical Operations | | | |
| np.var() | It gives variance of a given value | yz=np.arange((3))  np.var(yz) | 0.6666666666666666 |
| np.std() | It gives standarddeviation of a given value | np.std(yz) | 0.0 |
| np.corrcoef() | It gives correlationcoefficient of a given value | np.corrcoef(yz) | 1.0 |
| np.mean() | It gives mean of a given value | np.mean(yz) | 1.0 |
| np.median() | It gives mean of a given value | np.median(yz) | 1.0 |
|  |  |  |  |
|  |  |  |  |
| Arithmatic Operations | | | |
| np.square() | It prints square of the given value | np.square(2) | 4 |
| np.sqrt() | It prints square root value of the given element | np.sqrt(4) | 2.0 |
| np.random.logistic() | It gives log value | np.random.logistic(i) | array([[2.4014999 , 3.53932717],  [3.22251036, 3.56391501]]) |
| np.random.beta() | It gives beta value | np.random.beta(i,j) | array([[0.79993404, 0.83603517],  [0.49896728, 0.42563197]]) |
| np.add() | It performs addition | i=np.full((2,2),3)  j=np.full((2,2),2)  np.add(i,j) | array([[5, 5],  [5, 5]]) |
| np.subtract() | It performs subtraction | np.subtract(i,j) | array([[1, 1],  [1, 1]]) |
| np.multiply() | It does multiplication | np.multiply(i,j) | array([[6, 6],  [6, 6]]) |
| np.divmod() | It does modulodivision | np.divmod(i,j) | (array([[1, 1],  [1, 1]], dtype=int32), array([[1, 1],  [1, 1]], dtype=int32)) |
| np.divide() | It does division | np.divide(i,j) | array([[1.5, 1.5],  [1.5, 1.5]]) |
| np.exp() | It performs exponential | np.exp(i) | array([[20.08553692, 20.08553692],  [20.08553692, 20.08553692]]) |
|  |  |  |  |
| Trignometric Operations | | | |
| np.sin() | It prints sin value of element | np.sin(i) | array([[0.14112001, 0.14112001],  [0.14112001, 0.14112001]]) |
| np.cos() | It prints cos value of given element | np.cos(i) | array([[-0.9899925, -0.9899925],  [-0.9899925, -0.9899925]]) |
| np.tan() | It prints tan value of given element | np.tan(i) | array([[-0.14254654, -0.14254654],  [-0.14254654, -0.14254654]]) |
| 1/np.sin() | It gives cosec value of an array | a=np.array([1,2])  1/np.sin(a) | array([1.18839511, 1.09975017]) |