|  |  |  |  |
| --- | --- | --- | --- |
| CREATING NUMPY ARRAYS | | | |
| 1 | CREATING NUMPY ARRAYS FROM LISTS/DICTS/TUPLES | | |
| 1.1 | understand parameters of np.array() function | full command for using np.array() function |  |
| 1.2 | creating a 1 dim array from list/tuple/dict defining type | create a 1 dimensional(1 axis) numpy array of type byte(-128-127) from a list [1,2,3,4,5] | lst = [1,2,3,4,5]  arr\_1dim = np.array(lst, dtype = uint8)  print(arr\_1dim) |
| 1.3 | creating a 2 dim array from list/tuple/dict without defining type | create a 2 dimensional(2 axis) numpy array without defining type from a multidimensional list [[1,2,3],[4,5,6],[7,8,9]] | lst = [[1,2,3],[4,5,6],[7,8,9]]  arr\_2dim = np.array(lst)  print(arr\_2dim) |
| 2 | CREATING SEQUENTIAL NUMPY ARRAYS FROM ARANGE FUNCTION | | |
| 2.1 | understand parameters of np.arange() function | full command for using np.arange() function | numpy.arange([start, ]stop, [step, ], dtype=None) -> numpy.ndarray |
| 2.2 | creating a 1 dim array from a range given start,stop and step values | create array for numbers between 1 and 12 step 2 | arr = np.arange(start=1,stop=12,step=2)  print(arr) |
| 2.3 | creating a 1 dim array from a range given start,stop values | create array for numbers between 1 and 12 | arr = np.arange(start=1,stop=12)  print(arr) |
| 2.4 | creating a 1 dim array from a range given one argument | creating a 1 dim array from a range given single argument 10 | arr = np.arange(10) # Stop is 10, start is 0, and step is 1!  print(arr) |
| 2.5 | creating a 1 dim empty array from a range given start and stop | creating a 1 dim empty array from a range given start and stop | arr = np.arange(2, 2) # start/stop can be any other value other than 2  print(arr) |
| 2.6 | DType operation/Dominance | Whats the dtype of the following arrays:   1. np.arange(1, 5.1) 2. np.arange(0, 9, 1.5) 3. np.arange(5, dtype=float) | Float type |
|  | Inconsistency using float steps | Why do we avoid using a float step value? | When step is not an integer, the results might be inconsistent due to the limitations of floating-point arithmetic. |
| 3 | CREATING SEQUENTIAL NUMPY ARRAYS FROM LINSPACE | | |
| 3.1 | understand parameters of np.array() function | full command for using np.linspace() function |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  | numpy.fromiter(iterable, dtype, count=-1)  var = "Geekforgeeks"  arr = np.fromiter(var, dtype = 'U2')  print("fromiter() array :", arr)  numpy.linspace(start, stop, num=50, endpoint=True, retstep=False, dtype=None, axis=0)  np.linspace(3.5, 10, 3,  dtype = np.int32)  numpy.empty(shape, dtype=float, order=’C’)  np.empty([4, 3],  dtype = np.int32,  order = 'f')  numpy.ones(shape, dtype=None, order=’C’)  np.ones([4, 3],  dtype = np.int32,  order = 'f')  numpy.zeros(shape, dtype=None)  np.zeros([4, 3],  dtype = np.int32,  order = 'f') |  | Load NumPy Arrays From Text  loadtxt()  genfromtxt()  save array to file  savetxt()  save()  savez()  savez\_compressed() |
|  |  |  | # Create an array of ones  np.ones((3,4))  # Create an array of zeros  np.zeros((2,3,4),dtype=np.int16)  # Create an array with random values  np.random.random((2,2))  # Create an empty array  np.empty((3,2))  # Create a full array  np.full((2,2),7)  # Create an array of evenly-spaced values  np.arange(10,25,5)  # Create an array of evenly-spaced values  np.linspace(0,2,9)  # creating an array by sampling 10 numbers  # randomly from a mean-1, std-dev-5 normal  # distribution  np.random.normal(1, 5, 10)  arr = np.fromiter(iterable, float) |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Area |  |  |
| 1 | Dtypes operations | | |
|  | integer fixed-sized dtypes | NumPy offers you several integer fixed-sized dtypes that differ in memory and limits:  np.int8: 8-bit signed integer (from -128 to 127)  np.uint8: 8-bit unsigned integer (from 0 to 255)  np.int16: 16-bit signed integer (from -32768 to 32767)  np.uint16: 16-bit unsigned integer (from 0 to 65535)  np.int32: 32-bit signed integer (from -2\*\*31 to 2\*\*31-1)  np.uint32: 32-bit unsigned integer (from 0 to 2\*\*32-1)  np.int64: 64-bit signed integer (from -2\*\*63 to 2\*\*63-1)  np.uint64: 64-bit unsigned integer (from 0 to 2\*\*64-1) |  |
|  | Define dtype as integer | The argument dtype=np.int32 (or dtype='int32') |  |
| 2 | Attributes | | |
|  | query dtype attributes | x = arr.dtype  data  shape  strides  itemsize  # Print the number of `my\_array`'s dimensions  print(my\_array.ndim)  # Print the number of `my\_array`'s elements  print(my\_array.size)  # Print information about `my\_array`'s memory layout  print(my\_array.flags)  # Print the length of one array element in bytes  print(my\_array.itemsize)  # Print the total consumed bytes by `my\_array`'s elements  print(my\_array.nbytes) |  |
| 3 | Operations | | |
|  | np.flip |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| 4 | Misc support functions | | |
|  | Using timeit | import timeit  n = 1  timeit.timeit(f'x = [i\*\*2 for i in range({n})]')  timeit.timeit(f'x = np.arange({n})\*\*2', setup='import numpy as np') |  |

From shape or value

|  |  |
| --- | --- |
| [empty](https://numpy.org/doc/stable/reference/generated/numpy.empty.html#numpy.empty)(shape[, dtype, order, like]) | Return a new array of given shape and type, without initializing entries. |
| [empty\_like](https://numpy.org/doc/stable/reference/generated/numpy.empty_like.html#numpy.empty_like)(prototype[, dtype, order, subok, ...]) | Return a new array with the same shape and type as a given array. |
| [eye](https://numpy.org/doc/stable/reference/generated/numpy.eye.html#numpy.eye)(N[, M, k, dtype, order, like]) | Return a 2-D array with ones on the diagonal and zeros elsewhere. |
| [identity](https://numpy.org/doc/stable/reference/generated/numpy.identity.html#numpy.identity)(n[, dtype, like]) | Return the identity array. |
| [ones](https://numpy.org/doc/stable/reference/generated/numpy.ones.html#numpy.ones)(shape[, dtype, order, like]) | Return a new array of given shape and type, filled with ones. |
| [ones\_like](https://numpy.org/doc/stable/reference/generated/numpy.ones_like.html#numpy.ones_like)(a[, dtype, order, subok, shape]) | Return an array of ones with the same shape and type as a given array. |
| [zeros](https://numpy.org/doc/stable/reference/generated/numpy.zeros.html#numpy.zeros)(shape[, dtype, order, like]) | Return a new array of given shape and type, filled with zeros. |
| [zeros\_like](https://numpy.org/doc/stable/reference/generated/numpy.zeros_like.html#numpy.zeros_like)(a[, dtype, order, subok, shape]) | Return an array of zeros with the same shape and type as a given array. |
| [full](https://numpy.org/doc/stable/reference/generated/numpy.full.html#numpy.full)(shape, fill\_value[, dtype, order, like]) | Return a new array of given shape and type, filled with *fill\_value*. |
| [full\_like](https://numpy.org/doc/stable/reference/generated/numpy.full_like.html#numpy.full_like)(a, fill\_value[, dtype, order, ...]) | Return a full array with the same shape and type as a given array. |

From existing data

|  |  |
| --- | --- |
| [array](https://numpy.org/doc/stable/reference/generated/numpy.array.html#numpy.array)(object[, dtype, copy, order, subok, ...]) | Create an array. |
| [asarray](https://numpy.org/doc/stable/reference/generated/numpy.asarray.html#numpy.asarray)(a[, dtype, order, like]) | Convert the input to an array. |
| [asanyarray](https://numpy.org/doc/stable/reference/generated/numpy.asanyarray.html#numpy.asanyarray)(a[, dtype, order, like]) | Convert the input to an ndarray, but pass ndarray subclasses through. |
| [ascontiguousarray](https://numpy.org/doc/stable/reference/generated/numpy.ascontiguousarray.html#numpy.ascontiguousarray)(a[, dtype, like]) | Return a contiguous array (ndim >= 1) in memory (C order). |
| [asmatrix](https://numpy.org/doc/stable/reference/generated/numpy.asmatrix.html#numpy.asmatrix)(data[, dtype]) | Interpret the input as a matrix. |
| [copy](https://numpy.org/doc/stable/reference/generated/numpy.copy.html#numpy.copy)(a[, order, subok]) | Return an array copy of the given object. |
| [frombuffer](https://numpy.org/doc/stable/reference/generated/numpy.frombuffer.html#numpy.frombuffer)(buffer[, dtype, count, offset, like]) | Interpret a buffer as a 1-dimensional array. |
| [fromfile](https://numpy.org/doc/stable/reference/generated/numpy.fromfile.html#numpy.fromfile)(file[, dtype, count, sep, offset, like]) | Construct an array from data in a text or binary file. |
| [fromfunction](https://numpy.org/doc/stable/reference/generated/numpy.fromfunction.html#numpy.fromfunction)(function, shape, \*[, dtype, like]) | Construct an array by executing a function over each coordinate. |
| [fromiter](https://numpy.org/doc/stable/reference/generated/numpy.fromiter.html#numpy.fromiter)(iter, dtype[, count, like]) | Create a new 1-dimensional array from an iterable object. |
| [fromstring](https://numpy.org/doc/stable/reference/generated/numpy.fromstring.html#numpy.fromstring)(string[, dtype, count, like]) | A new 1-D array initialized from text data in a string. |
| [loadtxt](https://numpy.org/doc/stable/reference/generated/numpy.loadtxt.html#numpy.loadtxt)(fname[, dtype, comments, delimiter, ...]) | Load data from a text file. |

Creating record arrays (**numpy.rec**)

**Note**

**numpy.rec** is the preferred alias for **numpy.core.records**.

|  |  |
| --- | --- |
| [core.records.array](https://numpy.org/doc/stable/reference/generated/numpy.core.records.array.html#numpy.core.records.array)(obj[, dtype, shape, ...]) | Construct a record array from a wide-variety of objects. |
| [core.records.fromarrays](https://numpy.org/doc/stable/reference/generated/numpy.core.records.fromarrays.html#numpy.core.records.fromarrays)(arrayList[, dtype, ...]) | Create a record array from a (flat) list of arrays |
| [core.records.fromrecords](https://numpy.org/doc/stable/reference/generated/numpy.core.records.fromrecords.html#numpy.core.records.fromrecords)(recList[, dtype, ...]) | Create a recarray from a list of records in text form. |
| [core.records.fromstring](https://numpy.org/doc/stable/reference/generated/numpy.core.records.fromstring.html#numpy.core.records.fromstring)(datastring[, dtype, ...]) | Create a record array from binary data |
| [core.records.fromfile](https://numpy.org/doc/stable/reference/generated/numpy.core.records.fromfile.html#numpy.core.records.fromfile)(fd[, dtype, shape, ...]) | Create an array from binary file data |

Creating character arrays (**[numpy.char](https://numpy.org/doc/stable/reference/routines.char.html" \l "module-numpy.char" \o "numpy.char)**)

**Note**

[**numpy.char**](https://numpy.org/doc/stable/reference/routines.char.html#module-numpy.char) is the preferred alias for **numpy.core.defchararray**.

|  |  |
| --- | --- |
| [core.defchararray.array](https://numpy.org/doc/stable/reference/generated/numpy.core.defchararray.array.html#numpy.core.defchararray.array)(obj[, itemsize, ...]) | Create a [chararray](https://numpy.org/doc/stable/reference/generated/numpy.chararray.html" \l "numpy.chararray" \o "numpy.chararray). |
| [core.defchararray.asarray](https://numpy.org/doc/stable/reference/generated/numpy.core.defchararray.asarray.html#numpy.core.defchararray.asarray)(obj[, itemsize, ...]) | Convert the input to a **[chararray](https://numpy.org/doc/stable/reference/generated/numpy.chararray.html" \l "numpy.chararray" \o "numpy.chararray)**, copying the data only if necessary. |

Numerical ranges

|  |  |
| --- | --- |
| [arange](https://numpy.org/doc/stable/reference/generated/numpy.arange.html#numpy.arange)([start,] stop[, step,][, dtype, like]) | Return evenly spaced values within a given interval. |
| [linspace](https://numpy.org/doc/stable/reference/generated/numpy.linspace.html#numpy.linspace)(start, stop[, num, endpoint, ...]) | Return evenly spaced numbers over a specified interval. |
| [logspace](https://numpy.org/doc/stable/reference/generated/numpy.logspace.html#numpy.logspace)(start, stop[, num, endpoint, base, ...]) | Return numbers spaced evenly on a log scale. |
| [geomspace](https://numpy.org/doc/stable/reference/generated/numpy.geomspace.html#numpy.geomspace)(start, stop[, num, endpoint, ...]) | Return numbers spaced evenly on a log scale (a geometric progression). |
| [meshgrid](https://numpy.org/doc/stable/reference/generated/numpy.meshgrid.html#numpy.meshgrid)(\*xi[, copy, sparse, indexing]) | Return coordinate matrices from coordinate vectors. |
| [mgrid](https://numpy.org/doc/stable/reference/generated/numpy.mgrid.html#numpy.mgrid) | *nd\_grid* instance which returns a dense multi-dimensional "meshgrid". |
| [ogrid](https://numpy.org/doc/stable/reference/generated/numpy.ogrid.html#numpy.ogrid) | *nd\_grid* instance which returns an open multi-dimensional "meshgrid". |

Building matrices

|  |  |
| --- | --- |
| [diag](https://numpy.org/doc/stable/reference/generated/numpy.diag.html#numpy.diag)(v[, k]) | Extract a diagonal or construct a diagonal array. |
| [diagflat](https://numpy.org/doc/stable/reference/generated/numpy.diagflat.html#numpy.diagflat)(v[, k]) | Create a two-dimensional array with the flattened input as a diagonal. |
| [tri](https://numpy.org/doc/stable/reference/generated/numpy.tri.html#numpy.tri)(N[, M, k, dtype, like]) | An array with ones at and below the given diagonal and zeros elsewhere. |
| [tril](https://numpy.org/doc/stable/reference/generated/numpy.tril.html#numpy.tril)(m[, k]) | Lower triangle of an array. |
| [triu](https://numpy.org/doc/stable/reference/generated/numpy.triu.html#numpy.triu)(m[, k]) | Upper triangle of an array. |
| [vander](https://numpy.org/doc/stable/reference/generated/numpy.vander.html#numpy.vander)(x[, N, increasing]) | Generate a Vandermonde matrix. |

The Matrix class

|  |  |
| --- | --- |
| [mat](https://numpy.org/doc/stable/reference/generated/numpy.mat.html#numpy.mat)(data[, dtype]) | Interpret the input as a matrix. |
| [bmat](https://numpy.org/doc/stable/reference/generated/numpy.bmat.html#numpy.bmat)(obj[, ldict, gdict]) | Build a matrix object from a string, nested sequence, or array. |

*Table 4-1. Array creation functions*

|  |  |
| --- | --- |
| Function | Description |
| array | Convert input data (list, tuple, array, or other sequence type) to an ndarray either by inferring a dtype or explicitly specifying a dtype. Copies the input data by default. |
| asarray | Convert input to ndarray, but do not copy if the input is already an ndarray |
| arange | Like the built-in range but returns an ndarray instead of a list. |
| ones, ones\_like | Produce an array of all 1’s with the given shape and dtype. ones\_like takes another array and produces a ones array of the same shape and dtype. |
| zeros, zeros\_like | Like ones and ones\_like but producing arrays of 0’s instead |
| empty, empty\_like | Create new arrays by allocating new memory, but do not populate with any values like ones and zeros |
| eye, identity | Create a square N x N identity matrix (1’s on the diagonal and 0’s elsewhere) |