tinynumpy Documentation

Release 0.0.1dev

Almar Klein, Wade Brainerd

December 08, 2014

Contents

A lightweight, pure Python, numpy compliant ndarray class.

The documenation in this module is rather compact. For details on each function, see the corresponding documentation at: http://docs.scipy.org/doc/numpy/reference/index.html Be aware that the behavior of tinynumpy may deviate in some ways from numpy, or that certain features may not be supported.

tinynumpy.**arange**([*start*], *stop*[, *step*], *dtype=None*)
Return evenly spaced values within a given interval.

Values are generated within the half-open interval `[start, stop)` (in other words, the interval including *start* but excluding *stop*). For integer arguments the function is equivalent to the Python built-in range function, but returns an ndarray rather than a list.

When using a non-integer step, such as 0.1, the results will often not be consistent. It is better to use `linspace` for these cases.

tinynumpy.**array**(*obj*, *dtype=None*, *copy=True*, *order=None*)
Create a new array. If obj is an ndarray, and copy=False, a view of that array is returned. For details see: http://docs.scipy.org/doc/numpy/reference/generated/numpy.array.html

tinynumpy.**empty**(*shape*, *dtype=None*, *order=None*)
Return a new array of given shape and type, without initializing entries

tinynumpy.**empty_like**(*a*, *dtype=None*, *order=None*)
Return a new array with the same shape and type as a given array.

tinynumpy.**linspace**(*start*, *stop*, *num=50*, *endpoint=True*, *retstep=False*, *dtype=None*)
Return evenly spaced numbers over a specified interval. Returns num evenly spaced samples, calculated over the interval [start, stop]. The endpoint of the interval can optionally be excluded.

tinynumpy.**ones**(*shape*, *dtype=None*, *order=None*)
Return a new array of given shape and type, filled with ones

tinynumpy.**ones_like**(*a*, *dtype=None*, *order=None*)
Return an array of ones with the same shape and type as a given array.

tinynumpy.**squeeze_strides**(*s*)
Pop strides for singular dimensions.

tinynumpy.**zeros**(*shape*, *dtype=None*, *order=None*)
Return a new array of given shape and type, filled with zeros

tinynumpy.**zeros_like**(*a*, *dtype=None*, *order=None*)
Return an array of zeros with the same shape and type as a given array.

---

class tinynumpy.**ndarray**(*shape*, *dtype='float64'*, *buffer=None*, *offset=0*, *strides=None*, *order=None*)
Array class similar to numpy's ndarray, implemented in pure Python. This class can be distinguished from a real numpy array in that the repr always shows the dtype as a string, and for larger arrays (more than 100 elements) it shows a short one-line repr.

An array object represents a multidimensional, homogeneous array of fixed-size items. An associated data-type property describes the format of each element in the array.

Arrays should be constructed using *array*, *zeros* or *empty* (refer to the See Also section below). The parameters given here refer to a low-level method (*ndarray(...)*) for instantiating an array.

Parameters **shape** : tuple of ints

Shape of created array.

dtype : data-type, optional

Any object that can be interpreted as a numpy data type.

buffer : object containing data, optional

Used to fill the array with data. If another ndarray is given, the underlying data is used. Can also be a ctypes.Array or any object that exposes the buffer interface.

offset : int, optional

Offset of array data in buffer.

strides : tuple of ints, optional

Strides of data in memory.

order : {'C', 'F'}, optional NOT SUPPORTED

Row-major or column-major order.

See also:

array Construct an array.

zeros Create an array, each element of which is zero.

empty Create an array, but leave its allocated memory unchanged (i.e., it contains "garbage").

Notes

There are two modes of creating an array:

1. If *buffer* is None, then only *shape*, *dtype*, and *order* are used.
2. If *buffer* is an object exposing the buffer interface, then all keywords are interpreted.

Attributes

Methods

T

all(*axis=None*)

any(*axis=None*)

argmax(*axis=None*)

argmin(*axis=None*)

astype(*dtype*)

base

clip(*a_min*, *a_max*, *out=None*)

copy()

cumprod(*axis=None*, *out=None*)

cumsum(*axis=None*, *out=None*)

data

dtype

fill(*value*)

flags

flat

flatten()

itemsize

max(*axis=None*)

mean(*axis=None*)

min(*axis=None*)

nbytes

ndim

prod(*axis=None*)

ravel()

repeat(*repeats*, *axis=None*)

reshape(*newshape*)

shape

size

strides

sum(*axis=None*)

transpose()

view(*dtype=None*, *type=None*)

t

t