numpy.ndarray.ctypes — NumPy v1.21 Manual

An object to simplify the interaction of the array with the ctypes module.

This attribute creates an object that makes it easier to use arrays when calling shared libraries with the ctypes module. The returned object has, among others, data, shape, and strides attributes (see Notes below) which themselves return ctypes objects that can be used as arguments to a shared library.

Parameters

None

Returns

cPython object

Possessing attributes data, shape, strides, etc.

Notes

Below are the public attributes of this object which were documented in “Guide to NumPy” (we have omitted undocumented public attributes, as well as documented private attributes):

_ctypes.data

A pointer to the memory area of the array as a Python integer. This memory area may contain data that is not aligned, or not in correct byte-order. The memory area may not even be writeable. The array flags and data-type of this array should be respected when passing this attribute to arbitrary C-code to avoid trouble that can include Python crashing. User Beware! The value of this attribute is exactly the same as `self._array_interface_['data'][0]`.

Note that unlike `data_as`, a reference will not be kept to the array: code like `ctypes.c_void_p((a + b).ctypes.data)` will result in a pointer to a deallocated array, and should be spelt `(a + b).ctypes.data_as(ctypes.c_void_p)`

_ctypes.shape

(c_intp*self.ndim): A ctypes array of length self.ndim where the basetype is the C-integer corresponding to `dtype('p')` on this platform. This base-type could be `ctypes.c_int`, `ctypes.c_long`, or `ctypes.c_longlong` depending on the platform. The c_intp type is defined accordingly in `numpy.ctypeslib`. The ctypes array contains the shape of the underlying array.

_ctypes.strides

(c_intp*self.ndim): A ctypes array of length self.ndim where the basetype is the same as for the shape attribute. This ctypes array contains the strides information from the underlying array. This strides information is important for showing how many bytes must be jumped to get to the next element in the array.

_ctypes.data_as(*obj*) [source]

Return the data pointer cast to a particular c-types object. For example, calling `self._as_parameter_` is equivalent to `self.data_as(ctypes.c_void_p)`. Perhaps you want to use the data as a pointer to a ctypes array of floating-point data: `self.data_as(ctypes.POINTER(ctypes.c_double))`.

The returned pointer will keep a reference to the array.

_ctypes.shape_as(*obj*) [source]

Return the shape tuple as an array of some other c-types type. For example: `self.shape_as(ctypes.c_short)`.

_ctypes.strides_as(*obj*) [source]

Return the strides tuple as an array of some other c-types type. For example: `self.strides_as(ctypes.c_longlong)`.

If the ctypes module is not available, then the ctypes attribute of array objects still returns something useful, but ctypes objects are not returned and errors may be raised instead. In particular, the object will still have the `as_parameter` attribute which will return an integer equal to the data attribute.

Examples

```
>>> import ctypes
>>> x = np.array([[0, 1], [2, 3]], dtype=np.int32)
>>> x
array([[0, 1],
       [2, 3]], dtype=int32)
>>> x.ctypes.data
31962608 # may vary
>>> x.ctypes.data_as(ctypes.POINTER(ctypes.c_uint32))
<__main__.LP_c_uint object at 0x7ff2fc1fc200> # may vary
>>> x.ctypes.data_as(ctypes.POINTER(ctypes.c_uint32)).contents
c_uint(0)
>>> x.ctypes.data_as(ctypes.POINTER(ctypes.c_uint64)).contents
c_ulong(4294967296)
>>> x.ctypes.shape
<numpy.core._internal.c_long_Array_2 object at 0x7ff2fc1fce60> # may vary
>>> x.ctypes.strides
<numpy.core._internal.c_long_Array_2 object at 0x7ff2fc1ff320> # may vary
```