

[SciPy.org \(http://scipy.org/\)](http://scipy.org/) [Docs \(http://docs.scipy.org/\)](http://docs.scipy.org/)

[NumPy v1.9 Manual \(../index.html\)](#) [NumPy Reference \(../index.html\)](#)

[Array objects \(../arrays.html\)](#) [The N-dimensional array \(**ndarray**\) \(../arrays.ndarray.html\)](#)

[index \(../genindex.html\)](#) [next \(numpy.ndarray.item.html\)](#)

numpy.ndarray.ctypes

ndarray.ctypes¶

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: *c : Python object*

Possessing attributes data, shape, strides, etc.

See also:

[numpy.ctypeslib](#)

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):

- **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]`.
- **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 `c_int`, `c_long`, or `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.
- **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.

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

Be careful using the ctypes attribute - especially on temporary arrays or arrays constructed on the fly. For example, calling `(a+b).ctypes.data_as(ctypes.c_void_p)` returns a pointer to memory that is invalid because the array created as `(a+b)` is deallocated before the next Python statement. You can avoid this problem using either `c=a+b` or `ct=(a+b).ctypes`. In the latter case, `ct` will hold a reference to the array until `ct` is deleted or re-assigned.

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
array([[0, 1],
       [2, 3]])
>>> x.ctypes.data
30439712
>>> x.ctypes.data_as(ctypes.POINTER(ctypes.c_long))
<ctypes.LP_c_long object at 0x01F01300>
>>> x.ctypes.data_as(ctypes.POINTER(ctypes.c_long)).contents
c_long(0)
>>> x.ctypes.data_as(ctypes.POINTER(ctypes.c_longlong)).contents
c_longlong(4294967296L)
>>> x.ctypes.shape
<numpy.core._internal.c_long_Array_2 object at 0x01FFD580>
>>> x.ctypes.shape_as(ctypes.c_long)
<numpy.core._internal.c_long_Array_2 object at 0x01FCE620>
>>> x.ctypes.strides
<numpy.core._internal.c_long_Array_2 object at 0x01FCE620>
>>> x.ctypes.strides_as(ctypes.c_longlong)
<numpy.core._internal.c_longlong_Array_2 object at 0x01F01300>
```

Next topic

[numpy.ndarray.item \(numpy.ndarray.item.html\)](#)