Zumpy

Generated by Doxygen 1.9.3

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 example Namespace Reference

Variables

- arr = array([5], 'int32')
- arr2 = array([3,3], 'int32')

5.1.1 Variable Documentation

5.1.1.1 arr

```
example.arr = array([5], 'int32')
```

Definition at line 3 of file example.py.

5.1.1.2 arr2

```
example.arr2 = array([3,3], 'int32')
```

Definition at line 9 of file example.py.

5.2 zumpy Namespace Reference

Classes

- class array

  Array Module A simple array class that handles arbitrary dimensions for integer and float types.
- class array_wrapper

Chapter 6

Class Documentation

6.1 zumpy.array Class Reference

Array Module A simple array class that handles arbitrary dimensions for integer and float types.

Public Member Functions

- def create (self, shape, dtype='int32')

  Create/Initialize an empty array with specified size/dimension and data type.
- def __init__ (self, shape=None, dtype='int32')

  Constructor for array class.
- def __del__ (self)

  Destructor to deallocate memory from the array.
- def __str__ (self)

  Override print() call to print the contents of an array.
- def __repr__ (self)

  Override print() call to print the contents of an array.
- def at (self, idx)

  Access an element by index.
- def __getitem__ (self, idx)

  Access an element by index.
- def set (self, idx, value)
- def __setitem__ (self, idx, value)
- def fill (self, value)
- def slice (self, slice_indices)
- def filter (self, filter_func, secondary_indices, filter_type)
- def sum (self)

Static Public Attributes

- arr = None
- dtype = None
- shape = None

6.1.1 Detailed Description

Array Module A simple array class that handles arbitrary dimensions for integer and float types.

Definition at line 49 of file zumpy.py.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 __init__()

```
def zumpy.array.__init__ (
              self,
              shape = None,
              dtype = 'int32' )
```

Constructor for array class.

Calls create(self, shape, dtype) method.

Parameters

shape	A list specifying the shape/dimension, e.g [3, 2] for a 3x2 array.
dtype	A string specifying the data type of the array. One of ('int32', 'float'). By default, it's 'int32'.

Definition at line 81 of file zumpy.py.

6.1.2.2 __del__()

```
def zumpy.array.__del__ (
              self )
```

Destructor to deallocate memory from the array.

This probably won't ever need to be manually called by the user. This should handle the memory management behind the scenes interacting with the C code to avoid memory leaks.

Definition at line 87 of file zumpy.py.

6.1.3 Member Function Documentation

6.1.3.1 __getitem__()

```
def zumpy.array.__getitem__ (
              self,
              idx )
```

Access an element by index.

This is a wrapper around the zumpy.array.at(self, idx) method to use convenient square bracket syntax.

Parameters

idx	A list specifying the index. E.g [1, 2] will access the element at the second row and third column (zero-indexed).

Returns

Returns the value at the specified index.

```
myarray[3]     # access the fourth element in a 1D array
myarray[1,2]   # access the (1,2)th element in a 2D array
myarray[2,1,1] # so on and so forth...I think you get the idea
```

Definition at line 136 of file zumpy.py.

6.1.3.2 __repr__()

```
def zumpy.array.__repr__ (
              self )
```

Override print() call to print the contents of an array.

Calls custom print() function implemented in C to output contents in the console.

Definition at line 100 of file zumpy.py.

6.1.3.3 __setitem__()

```
def zumpy.array.__setitem__ (
              self,
              idx,
              value )
```

Definition at line 152 of file zumpy.py.

6.1.3.4 __str__()

```
def zumpy.array.__str__ (
              self )
```

Override print() call to print the contents of an array.

Calls custom print() function implemented in C to output contents in the console.

Definition at line 93 of file zumpy.py.

6.1.3.5 at()

```
def zumpy.array.at (
              self,
              idx )
```

Access an element by index.

Parameters

idx	A list specifying the index. E.g [1, 2] will access the element at the second row and third column (zero-indexed).

Returns

Returns the value at the specified index.

```
myarray.at(2) # access third element in 1D array
# note that higher dimensions require list syntax as below:
myarray.at([1,4]) # access (1,4)th element in 2D array
```

Definition at line 111 of file zumpy.py.

6.1.3.6 create()

```
def zumpy.array.create (
              self,
              shape,
              dtype = 'int32' )
```

Create/Initialize an empty array with specified size/dimension and data type.

Parameters

shape	A list specifying the shape/dimension, e.g [3, 2] for a 3x2 array.
dtype	A string specifying the data type of the array. One of ('int32', 'float'). By default, it's 'int32'.

Definition at line 63 of file zumpy.py.

6.1.3.7 fill()

```
def zumpy.array.fill (
              self,
              value )
```

Definition at line 160 of file zumpy.py.

6.1.3.8 filter()

```
def zumpy.array.filter (
              self,
              filter_func,
              secondary_indices,
              filter_type )
```

Definition at line 195 of file zumpy.py.

6.1.3.9 set()

```
def zumpy.array.set (
              self,
              idx,
              value )
```

Definition at line 144 of file zumpy.py.

6.1.3.10 slice()

```
def zumpy.array.slice (
              self,
              slice_indices )
```

Definition at line 168 of file zumpy.py.

6.1.3.11 sum()

```
def zumpy.array.sum (
              self )
```

Definition at line 230 of file zumpy.py.

6.1.4 Member Data Documentation

6.1.4.1 arr

```
zumpy.array.arr = None  [static]
```

Definition at line 56 of file zumpy.py.

6.1.4.2 dtype

```
zumpy.array.dtype = None  [static]
```

Definition at line 57 of file zumpy.py.

6.1.4.3 shape

```
zumpy.array.shape = None  [static]
```

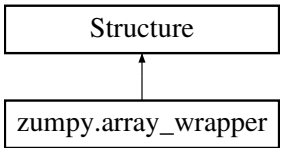
Definition at line 58 of file zumpy.py.

The documentation for this class was generated from the following file:

- src/python/zumpy.py

6.2 zumpy.array_wrapper Class Reference

Inheritance diagram for zumpy.array_wrapper:



Static Public Attributes

- argtypes
- restype

6.2.1 Detailed Description

Definition at line 9 of file zumpy.py.

6.2.2 Member Data Documentation

6.2.2.1 argtypes

```
zumpy.array_wrapper.argtypes  [static]
```

Definition at line 20 of file zumpy.py.

6.2.2.2 restype

```
zumpy.array_wrapper.restype  [static]
```

Definition at line 21 of file zumpy.py.

The documentation for this class was generated from the following file:

- src/python/zumpy.py

Chapter 7

File Documentation

7.1 src/python/example.py File Reference

Namespaces

- namespace example

Variables

- example.arr = array([5], 'int32')
- example.arr2 = array([3,3], 'int32')

7.2 example.py

Go to the documentation of this file.

```
00001 from zumpy import array
00002
00003 arr = array([5], 'int32')
00004 arr.fill(10)
00005 arr[2] = 22
00006
00007 print(arr.at(2))
00008
00009 arr2 = array([3,3], 'int32')
00010 arr2.fill(20)
00011
00012 print(arr2.at([0,2]))
```

7.3 src/python/zumpy.py File Reference

Classes

- class zumpy.array_wrapper
- class zumpy.array

  Array Module A simple array class that handles arbitrary dimensions for integer and float types.

Namespaces

- namespace zumpy

7.4 zumpy.py

Go to the documentation of this file.

```
00001 # python binding for libZumpy.so
00002 from ctypes import *
00003 import faulthandler
00004
00005 # load library
00006 _libZumpy = CDLL('./ext/libZumpy.so')
00007
00008 # wrapper class
00009 class array_wrapper(Structure):
00010     _fields_ = [
00011         ("data", c_void_p),
00012         ("arr_shape", POINTER(c_size_t)),
00013         ("shape_size", c_size_t),
00014         ("type_size", c_size_t),
00015         ("total_size", c_size_t),
00016         ("type", c_uint)
00017     ]
00018
00019 # function prototypes
00020 _libZumpy.arr_init.argtypes = [POINTER(array_wrapper), POINTER(c_size_t), c_size_t, c_size_t]
00021 _libZumpy.arr_init.restype = None
00022
00023 _libZumpy.arr_free.argtypes = [POINTER(array_wrapper)]
00024 _libZumpy.arr_free.restype = None
00025
00026 _libZumpy.arr_at.argtypes = [POINTER(array_wrapper), POINTER(c_size_t)]
00027 _libZumpy.arr_at.restype = c_void_p
00028
00029 _libZumpy.arr_set.argtypes = [POINTER(array_wrapper), POINTER(c_size_t), c_void_p]
00030 _libZumpy.arr_set.restype = None
00031
00032 _libZumpy.arr_fill.argtypes = [POINTER(array_wrapper), c_void_p]
00033 _libZumpy.arr_fill.restype = None
00034
00035 _libZumpy.arr_sum.argtypes = [POINTER(array_wrapper)]
00036 _libZumpy.arr_sum.restype = c_float
00037
00038 _libZumpy.arr_slice.argtypes = [POINTER(array_wrapper), POINTER(POINTER(c_size_t)), POINTER(c_size_t),
       c_size_t, POINTER(array_wrapper)]
00039 _libZumpy.arr_slice.restype = None
00040
00041 _libZumpy.arr_print.argtypes = [POINTER(array_wrapper)]
00042 _libZumpy.arr_slice.restype = None
00043
00044 _libZumpy.arr_filter.argtypes = [POINTER(array_wrapper), CFUNCTYPE(c_bool, c_void_p),
       POINTER(c_size_t), c_size_t, c_uint, POINTER(array_wrapper)]
00045 _libZumpy.arr_filter.restype = None
00046
00047
00049 class array():
00050     def _get_type_enum(self, dtype):
00051         if dtype == 'int32':
00052             return 0
00053         elif dtype == 'float':
00054             return 1
00055
00056     arr = None
00057     dtype = None
00058     shape = None
00059
00060
00063     def create(self, shape, dtype = 'int32'):
00064
00065         self.arr = array_wrapper()
00066         self.dtype = dtype
00067         self.shape = shape
00068
00069         arr_ptr = pointer(self.arr)
00070
00071         shape_size = len(self.shape)
00072         shape_arr = (c_size_t * len(self.shape))(*self.shape)
00073
00074         type_enum = self._get_type_enum(self.dtype)
```

```
00075
00076         _libZumpy.arr_init(arr_ptr, shape_arr, shape_size, type_enum)
00077
00078
00081     def __init__(self, shape = None, dtype = 'int32'):
00082         if shape != None:
00083             self.create(shape, dtype)
00084
00085
00087     def __del__(self):
00088         arr_ptr = pointer(self.arr)
00089         _libZumpy.arr_free(arr_ptr)
00090
00091
00093     def __str__(self):
00094         arr_ptr = pointer(self.arr)
00095         _libZumpy.arr_print(arr_ptr)
00096         return ""
00097
00098
00100     def __repr__(self):
00101         self.__str__()
00102
00103
00111     def at(self, idx):
00112         temp_idx = []
00113         if isinstance(idx, int):
00114             temp_idx.append(idx)
00115         else:
00116             temp_idx = idx
00117
00118         idx_arr = (c_size_t * len(temp_idx))(*temp_idx)
00119         # dereference different types
00120         if self.dtype == 'int32':
00121             return cast(cast(_libZumpy.arr_at(byref(self.arr), idx_arr), c_void_p),
    POINTER(c_int32)).contents.value
00122         elif self.dtype == 'float':
00123             return cast(cast(_libZumpy.arr_at(byref(self.arr), idx_arr), c_void_p),
    POINTER(c_float)).contents.value
00124
00125         return None
00126
00127
00136     def __getitem__(self, idx):
00137         temp_idx = []
00138         if isinstance(idx, int):
00139             temp_idx.append(idx)
00140         else:
00141             temp_idx = idx
00142         return self.at(temp_idx)
00143
00144     def set(self, idx, value):
00145         idx_arr = (c_size_t * len(idx))(*idx)
00146
00147         if self.dtype == 'int32':
00148             _libZumpy.arr_set(byref(self.arr), idx_arr, byref(c_int32(value)))
00149         elif self.dtype == 'float':
00150             _libZumpy.arr_set(byref(self.arr), idx_arr, byref(c_float(value)))
00151
00152     def __setitem__(self, idx, value):
00153         temp_idx = []
00154         if isinstance(idx, int):
00155             temp_idx.append(idx)
00156         else:
00157             temp_idx = idx
00158         self.set(temp_idx, value)
00159
00160     def fill(self, value):
00161         val_ptr = None
00162         if self.dtype == 'int32':
00163             val_ptr = cast(byref(c_int32(value)), c_void_p)
00164         elif self.dtype == 'float':
00165             val_ptr = cast(byref(c_float(value)), c_void_p)
00166         _libZumpy.arr_fill(byref(self.arr), val_ptr)
00167
00168     def slice(self, slice_indices):
00169         # slice indices should be a list of lists
00170         # convert slice_indices to size_t** (pointer to pointer of size_t)
00171         arr_inner = []
00172         for i in range(len(slice_indices)):
00173             arr_inner.append((c_size_t * len(slice_indices[i]))(*slice_indices[i]))
00174
00175         arr_outer = []
00176         for i in range(len(arr_inner)):
00177             arr_outer.append(arr_inner[i])
00178
00179         pp_slice_indices = (POINTER(c_size_t) * len(arr_outer))(*arr_outer)
```

```
00180
00181         slice_idx_len = len(slice_indices)
00182         slice_dims = []
00183         for idx in slice_indices:
00184             slice_dims.append(len(idx))
00185
00186         ref_arr = array_wrapper()
00187         p_slice_dims = (c_size_t * len(slice_dims))(*slice_dims)
00188         _libZumpy.arr_slice(byref(self.arr), pp_slice_indices, p_slice_dims, c_size_t(slice_idx_len),
      byref(ref_arr))
00189
00190         ret_arr = array(slice_dims, self.dtype)
00191         ret_arr.arr = ref_arr
00192
00193         return ret_arr
00194
00195     def filter(self, filter_func, secondary_indices, filter_type):
00196         proto_filter_func = CFUNCTYPE(c_bool, c_void_p)
00197         p_filter_func = proto_filter_func(filter_func)
00198
00199         p_secondary_indices = None
00200         if len(secondary_indices) != 0:
00201             p_secondary_indices = (c_size_t * len(secondary_indices))(*secondary_indices)
00202
00203         ftype = None
00204         if (filter_type == 'ANY'):
00205             ftype = 0
00206         elif (filter_type == 'ALL'):
00207             ftype = 1
00208
00209         dest_arr = array_wrapper()
00210
00211         _libZumpy.arr_filter(byref(self.arr), p_filter_func, p_secondary_indices,
      c_size_t(len(secondary_indices)), c_uint(ftype), byref(dest_arr))
00212
00213         # grab contents from struct pointer and convert array
00214         # shape into Python list
00215         _shape_size = pointer(dest_arr).contents.shape_size
00216         _arr_shape = pointer(dest_arr).contents.arr_shape
00217         _total_size = pointer(dest_arr).contents.total_size
00218         _arr_shape_list = []
00219         for i in range(_shape_size):
00220             _arr_shape_list.append(_arr_shape[i])
00221         ret_arr = array(_arr_shape_list, self.dtype)
00222         ret_arr.arr = dest_arr
00223
00224         # return NULL if filter returned no results
00225         if _total_size == 0:
00226             return None
00227         return ret_arr
00228
00229
00230     def sum(self):
00231         return _libZumpy.arr_sum(byref(self.arr))
```

Index