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 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)

  Set an element by index.
- def __setitem__ (self, idx, value)

  Set an element by index.
- def fill (self, value)

  Fill all cells with a specified value This will set every index of the array to the same value.
- def slice (self, slice_indices)

  Slice an array to extract subsets.
- def filter (self, filter_func, secondary_indices, filter_type)

  Filter an array based on user-defined condition.
- def sum (self)

  Sum all indices of an array.

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'.

Example:

```
from zumpy import array
# create 3x2 array of 32-bit integers
myarray = array([3,2], 'int32')
```

Definition at line 100 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 106 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.

Example:
```
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 173 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.

Example:
```
print(myarray)
```

Definition at line 131 of file zumpy.py.

6.1.3.3 __setitem__()

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

Set an element by index.

This is a wrapper around the zumpy.array.set(self, idx, value) 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).

Example:
```
myarray[3] = 10     # 1D array
myarray[1,2] = 10   # 2D array
myarray[2,1,1] = 10 # 3D array
```

Definition at line 217 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.

Example:
```
print(myarray)
```

Definition at line 118 of file zumpy.py.

6.1.3.5 at()

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

Access an element by index.

Parameters

idx	A list (or integer for 1D) 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.

Example:
```
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 145 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'.

Example:

```
from zumpy import array
# create 3x2 array of 32-bit integers
myarray = array()
myarray.create([3,2], 'int32')
```

Definition at line 73 of file zumpy.py.

6.1.3.7 fill()

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

Fill all cells with a specified value This will set every index of the array to the same value.

Parameters

value	Value to set all indices to

Example:

```
from zumpy import array
# this will fill a 3x2 array with 10s
myarray = array([3,2], 'int32')
myarray.fill(10)
```

Definition at line 238 of file zumpy.py.

6.1.3.8 filter()

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

Filter an array based on user-defined condition.

Note

You will need to use ctypes in the filter function to convert values so the underlying C code knows what to do.

Currently this filter doesn't support different filters on different columns simultaneously, but that's planned soon.

Parameters

filter_func	A user-defined python function that takes one parameter and returns a boolean. You will need to use ctypes to convert this parameter into your array type. See example below.
secondary_indices	These are the indices to restrict the filter to and are analogous to columns. E.g if you pass [1] it will only check the filter against column 1. If you pass an empty list [], it will check all columns.
filter_type	A string specifying 'ANY' or 'ALL'. This only applies to arrays 2D or above and if you are applying the filter to multiple columns. If 'ANY' is used, then the filter must pass (be true) for AT LEAST one of the columns; then that row will be returned. If 'ALL' is used, then ALL columns must satisfy the filter in order for that row to be returned.

Example:

```
from zumpy import array
from ctypes import *
from random import randint, seed
# currently don't know a better way to make this more user-friendly
# so for now, you will have to use a bit of ctypes magic
def myfilter(x):
    x = cast(x, POINTER(c_int32)) # cast parameter to a pointer of our array type (int32)
    return x.contents.value > 20  # dereference and access the pointer value and check the condition
arr = array()
arr.create([5,2], 'int32')
# set seed for reproducibility
seed(5021)
# fill array with random values
for i in range(arr.shape[0]):
    for j in range(arr.shape[1]):
        arr[i,j] = randint(0,50)
print("Full Array:")
print(arr)
# this will check if EITHER column 0 or 1 match the condition
# we are passing an empty list in second parameter to check all columns.
print("Filtered ANY:")
filtered_any = arr.filter(myfilter, [], 'ANY')
print(filtered_any)
# this will check if BOTH column 0 and 1 match the condition
print("Filtered ALL:")
filtered_all = arr.filter(myfilter, [], 'ALL')
print(filtered_all)
```

Output:

```
Full Array:
37 39
32 21
49 44
0 35
12 18
Filtered ANY:
37 39
32 21
49 44
0 35
Filtered ALL:
37 39
32 21
49 44
```

Definition at line 427 of file zumpy.py.

6.1.3.9 set()

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

Set an element by index.

Parameters

idx	A list (or integer for 1D) specifying the index to set the value at. E.g [1, 2] will set a value at the second row, third column.
value	Value to set at the specified index. Will have to match the data type that the array is set at (e.g, int32, float).

Example:

```
myarray.set(3) = 10       # 1D array
myarray.set([1,3]) = 10   # 2D array
myarray.set([2,1,1]) = 10 # 3D array
```

Definition at line 192 of file zumpy.py.

6.1.3.10 slice()

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

Slice an array to extract subsets.

Parameters

slice_indices	A list of lists containing the indices to slice. First dimension corresponds to the array dimension and second dimension corresponds to the indices to pull from that dimension. See example below.

Example:

```
from zumpy import array
arr = array()
arr.create([3,2], 'int32')
arr.fill(10)
print("Full Array:")
print(arr)
# reading the parameter explicitly, this is saying take index 0-2 from array dimension 0
# and take index 0 from array dimension 1.
# In other words, take all rows from column 0.
sub = arr.slice([range(3), [0]])
print("Sliced Array:")
print(sub)
```

Output:

```
Full Array:
10 10
10 10
10 10
Sliced Array:
10
```

```
10
10
```

Example 2:

```
from zumpy import array
arr = array()
arr.create([3,2,3], 'int32')
arr.fill(10)
# set the right-most column to 20
for i in range(arr.shape[0]):
    for j in range(arr.shape[1]):
        arr[i,j,2] = 20
print("Full Array:")
print(arr)
# take all indices from dimension 0,
# all indices from dimension 1,
# and index 2 from dimension 2.
# In other words, this will extract the right-most column in a 3D array.
sub = arr.slice([range(3), range(2), [2]])
print("Sliced Array:")
print(sub)
```

Output:

```
Full Array:
10 10 20
10 10 20
10 10 20
10 10 20
10 10 20
10 10 20
Sliced Array:
20
20
20
20
20
20
```

Definition at line 332 of file zumpy.py.

6.1.3.11 sum()

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

Sum all indices of an array.

Returns

A float value representing the sum of all the elements

Example:

```
from zumpy import array
from random import randint, seed
arr = array()
arr.create([5,2], 'int32')
# set seed for reproducibility
seed(5021)
# fill array with random values
for i in range(arr.shape[0]):
    for j in range(arr.shape[1]):
        arr[i,j] = randint(0,50)
print("Full Array:")
print(arr)
print("Sum: ", arr.sum())
```

Output:

```
Full Array:
37 39
32 21
49 44
0 35
12 18
Sum:  287.0
```

Definition at line 499 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]
```

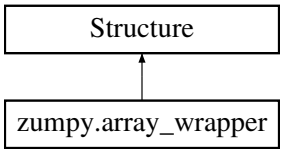
Definition at line 58 of file zumpy.py.

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/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

7.2 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
00073     def create(self, shape, dtype = 'int32'):
00074
00075         self.arr = array_wrapper()
00076         self.dtype = dtype
00077         self.shape = shape
00078
00079         arr_ptr = pointer(self.arr)
00080
00081         shape_size = len(self.shape)
00082         shape_arr = (c_size_t * len(self.shape))(*self.shape)
00083
00084         type_enum = self._get_type_enum(self.dtype)
00085
00086         _libZumpy.arr_init(arr_ptr, shape_arr, shape_size, type_enum)
00087
00088
00100     def __init__(self, shape = None, dtype = 'int32'):
00101         if shape != None:
00102             self.create(shape, dtype)
00103
00104
00106     def __del__(self):
00107         arr_ptr = pointer(self.arr)
00108         _libZumpy.arr_free(arr_ptr)
00109
00110
00118     def __str__(self):
00119         arr_ptr = pointer(self.arr)
00120         _libZumpy.arr_print(arr_ptr)
00121         return ""
00122
00123
00131     def __repr__(self):
00132         self.__str__()
00133
00134
00145     def at(self, idx):
00146         temp_idx = []
00147         if isinstance(idx, int):
00148             temp_idx.append(idx)
00149         else:
00150             temp_idx = idx
00151
00152         idx_arr = (c_size_t * len(temp_idx))(*temp_idx)
00153         # dereference different types
00154         if self.dtype == 'int32':
00155             return cast(cast(_libZumpy.arr_at(byref(self.arr), idx_arr), c_void_p),
    POINTER(c_int32)).contents.value
00156         elif self.dtype == 'float':
00157             return cast(cast(_libZumpy.arr_at(byref(self.arr), idx_arr), c_void_p),
    POINTER(c_float)).contents.value
00158
00159         return None
00160
00161
00173     def __getitem__(self, idx):
```

```
00174         temp_idx = []
00175         if isinstance(idx, int):
00176             temp_idx.append(idx)
00177         else:
00178             temp_idx = idx
00179         return self.at(temp_idx)
00180 
00181 
00192     def set(self, idx, value):
00193         temp_idx = []
00194         if isinstance(idx, int):
00195             temp_idx.append(idx)
00196         else:
00197             temp_idx = idx
00198 
00199         idx_arr = (c_size_t * len(idx))(*idx)
00200 
00201         if self.dtype == 'int32':
00202             _libZumpy.arr_set(byref(self.arr), idx_arr, byref(c_int32(value)))
00203         elif self.dtype == 'float':
00204             _libZumpy.arr_set(byref(self.arr), idx_arr, byref(c_float(value)))
00205 
00206 
00217     def __setitem__(self, idx, value):
00218         temp_idx = []
00219         if isinstance(idx, int):
00220             temp_idx.append(idx)
00221         else:
00222             temp_idx = idx
00223         self.set(temp_idx, value)
00224 
00225 
00238     def fill(self, value):
00239         val_ptr = None
00240         if self.dtype == 'int32':
00241             val_ptr = cast(byref(c_int32(value)), c_void_p)
00242         elif self.dtype == 'float':
00243             val_ptr = cast(byref(c_float(value)), c_void_p)
00244         _libZumpy.arr_fill(byref(self.arr), val_ptr)
00245 
00246 
00332     def slice(self, slice_indices):
00333         # slice indices should be a list of lists
00334         # convert slice_indices to size_t** (pointer to pointer of size_t)
00335         arr_inner = []
00336         for i in range(len(slice_indices)):
00337             arr_inner.append((c_size_t * len(slice_indices[i]))(*slice_indices[i]))
00338 
00339         arr_outer = []
00340         for i in range(len(arr_inner)):
00341             arr_outer.append(arr_inner[i])
00342 
00343         pp_slice_indices = (POINTER(c_size_t) * len(arr_outer))(*arr_outer)
00344 
00345         slice_idx_len = len(slice_indices)
00346         slice_dims = []
00347         for idx in slice_indices:
00348             slice_dims.append(len(idx))
00349 
00350         ref_arr = array_wrapper()
00351         p_slice_dims = (c_size_t * len(slice_dims))(*slice_dims)
00352         _libZumpy.arr_slice(byref(self.arr), pp_slice_indices, p_slice_dims, c_size_t(slice_idx_len),
      byref(ref_arr))
00353 
00354         ret_arr = array(slice_dims, self.dtype)
00355         ret_arr.arr = ref_arr
00356 
00357         return ret_arr
00358 
00359 
00427     def filter(self, filter_func, secondary_indices, filter_type):
00428         proto_filter_func = CFUNCTYPE(c_bool, c_void_p)
00429         p_filter_func = proto_filter_func(filter_func)
00430 
00431         p_secondary_indices = None
00432         if len(secondary_indices) != 0:
00433             p_secondary_indices = (c_size_t * len(secondary_indices))(*secondary_indices)
00434 
00435         ftype = None
00436         if (filter_type == 'ANY'):
00437             ftype = 0
00438         elif (filter_type == 'ALL'):
00439             ftype = 1
00440 
00441         dest_arr = array_wrapper()
00442 
00443         _libZumpy.arr_filter(byref(self.arr), p_filter_func, p_secondary_indices,
```

```
      c_size_t(len(secondary_indices)), c_uint(ftype), byref(dest_arr))
00444
00445         # grab contents from struct pointer and convert array
00446         # shape into Python list
00447         _shape_size = pointer(dest_arr).contents.shape_size
00448         _arr_shape = pointer(dest_arr).contents.arr_shape
00449         _total_size = pointer(dest_arr).contents.total_size
00450         _arr_shape_list = []
00451         for i in range(_shape_size):
00452             _arr_shape_list.append(_arr_shape[i])
00453         ret_arr = array(_arr_shape_list, self.dtype)
00454         ret_arr.arr = dest_arr
00455
00456         # return NULL if filter returned no results
00457         if _total_size == 0:
00458             return None
00459         return ret_arr
00460
00461
00499     def sum(self):
00500         return _libZumpy.arr_sum(byref(self.arr))
```

Index