Zumpy

Generated by Doxygen 1.9.3

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

Class Documentation

3.1 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.
- def to_array (self, list_arr, dtype='int32')

  Convert a Python list (of lists) to an array.

Static Public Attributes

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

3.1.1 Detailed Description

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

3.1.2 Constructor & Destructor Documentation

3.1.2.1 __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')
```

3.1.2.2 __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.

3.1.3 Member Function Documentation

3.1.3.1 __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
```

3.1.3.2 __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)
```

3.1.3.3 __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
```

3.1.3.4 __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)
```

3.1.3.5 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
```

3.1.3.6 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')
```

3.1.3.7 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)
```

3.1.3.8 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
```

3.1.3.9 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
```

3.1.3.10 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
```

3.1.3.11 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
```

3.1.3.12 to_array()

```
def zumpy.array.to_array (
                self,
                list_arr,
                dtype = 'int32' )
```

Convert a Python list (of lists) to an array.

Parameters

list_arr	The Python list (of lists) to convert into an array. This assumes the length of each list within the same dimension is the same (i.e, no jagged arrays)
dtype	The data type of the array. One of ('int32', 'float'). By default is 'int32'.

Example:

```
from zumpy import array
# convert a 3x4 python list into a zumpy array
x = [
        [1,2,3,4],
        [5,6,7,8],
        [9,10,11,12]
]
arr = array()
arr.to_array(x) # by default the dtype is 'int32' so no need to specify here
print("Shape: ", arr.shape)
print("Array:")
print(arr)
```

Output:

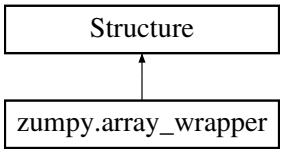
```
Shape:  [3, 4]
Array:
1 2 3 4
5 6 7 8
9 10 11 12
```

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

- src/python/zumpy.py

3.2 zumpy.array_wrapper Class Reference

Inheritance diagram for zumpy.array_wrapper:



Static Public Attributes

- **argtypes**
- **restype**

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

- src/python/zumpy.py

Index