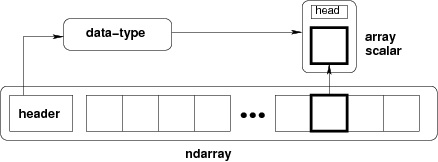
The most important object defined in NumPy is an N-dimensional array type called ndarray. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.

Every item in an ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called dtype).

Any item extracted from ndarray object (by slicing) is represented by a Python object of one of array scalar types. The following diagram shows a relationship between ndarray, data type object (dtype) and array scalar type −



An instance of ndarray class can be constructed by different array creation routines described later in the tutorial. The basic ndarray is created using an array function in NumPy as follows −

numpy.array

# minimum dimensions

import numpy as np

a = np.array([1, 2, 3,4,5], ndmin = 2)

print a

The output is as follows −

[[1, 2, 3, 4, 5]]

# dtype parameter

import numpy as np

a = np.array([1, 2, 3], dtype = complex)

print a

The output is as follows −

[ 1.+0.j, 2.+0.j, 3.+0.j]

In this chapter, we will discuss the various array attributes of NumPy.

## ndarray.shape

This array attribute returns a tuple consisting of array dimensions. It can also be used to resize the array.

### Example 1

[Live Demo](http://tpcg.io/SPvAYQ)

import numpy as np

a = np.array([[1,2,3],[4,5,6]])

print a.shape

The output is as follows −

(2, 3)

### Example 2

[Live Demo](http://tpcg.io/SVO3RX)

# this resizes the ndarray

import numpy as np

a = np.array([[1,2,3],[4,5,6]])

a.shape = (3,2)

print a

The output is as follows −

[[1, 2]

[3, 4]

[5, 6]]

### Example 3

NumPy also provides a reshape function to resize an array.

[Live Demo](http://tpcg.io/FKvABq)

import numpy as np

a = np.array([[1,2,3],[4,5,6]])

b = a.reshape(3,2)

print b

The output is as follows −

[[1, 2]

[3, 4]

[5, 6]]

## ndarray.ndim

This array attribute returns the number of array dimensions.

### Example 1

[Live Demo](http://tpcg.io/za4WJQ)

# an array of evenly spaced numbers

import numpy as np

a = np.arange(24)

print a

The output is as follows −

[0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]

### Example 2

[Live Demo](http://tpcg.io/sFgnoR)

# this is one dimensional array

import numpy as np

a = np.arange(24)

a.ndim

# now reshape it

b = a.reshape(2,4,3)

print b

# b is having three dimensions

The output is as follows −

[[[ 0, 1, 2]

[ 3, 4, 5]

[ 6, 7, 8]

[ 9, 10, 11]]

[[12, 13, 14]

[15, 16, 17]

[18, 19, 20]

[21, 22, 23]]]

## numpy.itemsize

This array attribute returns the length of each element of array in bytes.

### Example 1

[Live Demo](http://tpcg.io/pmlHmj)

# dtype of array is int8 (1 byte)

import numpy as np

x = np.array([1,2,3,4,5], dtype = np.int8)

print x.itemsize

The output is as follows −

1

### Example 2

[Live Demo](http://tpcg.io/DyQL6L)

# dtype of array is now float32 (4 bytes)

import numpy as np

x = np.array([1,2,3,4,5], dtype = np.float32)

print x.itemsize

The output is as follows −

4

## numpy.empty

It creates an uninitialized array of specified shape and dtype. It uses the following constructor −

numpy.empty(shape, dtype = float, order = 'C')

The constructor takes the following parameters.

|  |  |
| --- | --- |
| **Sr.No.** | **Parameter & Description** |
| 1 | Shape  Shape of an empty array in int or tuple of int |
| 2 | Dtype  Desired output data type. Optional |
| 3 | Order  'C' for C-style row-major array, 'F' for FORTRAN style column-major array |

### Example

The following code shows an example of an empty array.

[Live Demo](http://tpcg.io/0oGlTq)

import numpy as np

x = np.empty([3,2], dtype = int)

print x

The output is as follows −

[[22649312 1701344351]

[1818321759 1885959276]

[16779776 156368896]]

Note − The elements in an array show random values as they are not initialized.

## numpy.zeros

Returns a new array of specified size, filled with zeros.

numpy.zeros(shape, dtype = float, order = 'C')

# array of five zeros. Default dtype is float

import numpy as np

x = np.zeros(5)

print x

Similarly numpy.ones

numpy.asarray(a, dtype = None, order = None)

|  |  |
| --- | --- |
| **Sr.No.** | **Parameter & Description** |
| 1 | a  Input data in any form such as list, list of tuples, tuples, tuple of tuples or tuple of lists |
| 2 | dtype  By default, the data type of input data is applied to the resultant ndarray |
| 3 | order  C (row major) or F (column major). C is default |

# convert list to ndarray

import numpy as np

x = [1,2,3]

a = np.asarray(x)

print a

# dtype is set

import numpy as np

x = [1,2,3]

a = np.asarray(x, dtype = float)

print a

# ndarray from tuple

import numpy as np

x = (1,2,3)

a = np.asarray(x)

print a

Its output would be −

[1 2 3]

### Example 4

[Live Demo](http://tpcg.io/Y1paYp)

# ndarray from list of tuples

import numpy as np

x = [(1,2,3),(4,5)]

a = np.asarray(x)

print a

Here, the output would be as follows −

[(1, 2, 3) (4, 5)]

## numpy.arange

This function returns an ndarray object containing evenly spaced values within a given range. The format of the function is as follows −

## numpy.linspace

This function is similar to arange() function. In this function, instead of step size, the number of evenly spaced values between the interval is specified. The usage of this function is as follows −

numpy.linspace(start, stop, num, endpoint, retstep, dtype)

### Example 1

[Live Demo](http://tpcg.io/LweALo)

import numpy as np

x = np.linspace(10,20,5)

print x

Its output would be −

[10. 12.5 15. 17.5 20.]

### Example 2

[Live Demo](http://tpcg.io/S0koVZ)

# endpoint set to false

import numpy as np

x = np.linspace(10,20, 5, endpoint = False)

print x

The output would be −

[10. 12. 14. 16. 18.]

### Example 3

[Live Demo](http://tpcg.io/WqFdFX)

# find retstep value

import numpy as np

x = np.linspace(1,2,5, retstep = True)

print x

# retstep here is 0.25

Now, the output would be −

(array([ 1. , 1.25, 1.5 , 1.75, 2. ]), 0.25)

**SyntaxError:** positional argument follows keyword argument

Sort\_values

Groupby .mean()

.sort in pandas

usually, a Python dictionary throws a KeyError if you try to get an item with a key that is not currently in the dictionary. The defaultdict in contrast will simply create any items that you try to access (provided of course they do not exist yet).

**meaning of r in regex??**