Python For Data Science Cheat Sheet

NumPy Basics

Learn Python for Data Science Interactively at www.DataCamp.com



NumPy

The **NumPy** library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

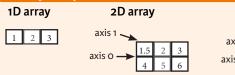
Use the following import convention: >>> import numpy as np



3D array

axis 2

NumPy Arrays



Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([(1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]],
                 dtype = float)
```

Initial Placeholders

>>> np.zeros((3,4))	Create an array of zeros
>>> np.ones((2,3,4),dtype=np.int16)	
>>> d = np.arange(10,25,5)	Create an array of evenly
	spaced values (step value)
>>> np.linspace(0,2,9)	Create an array of evenly
	spaced values (number of samples)
>>> e = np.full((2,2),7)	Create a constant array
>>> f = np.eye(2)	Create a 2X2 identity matrix
>>> np.random.random((2,2))	Create an array with random values
>>> np.empty((3,2))	Create an empty array

1/0

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my array.npy')
```

Saving & Loading Text Files

>>>	np.loadtxt("myfile.txt")
>>>	<pre>np.genfromtxt("my file.csv", delimiter=',')</pre>
>>>	np.savetxt("mvarrav.txt", a, delimiter=" ")

Data Types

>>> np.int64	Signed 64-bit integer types
>>> np.float32	Standard double-precision floating point
>>> np.complex	Complex numbers represented by 128 floats
>>> np.bool	Boolean type storing TRUE and FALSE values
>>> np.object	Python object type
>>> np.string_	Fixed-length string type
>>> np.unicode_	Fixed-length unicode type

Inspecting Your Array

>>> a.shape	Array dimensions
>>> len(a)	Length of array
>>> b.ndim	Number of array dimensions
>>> e.size	Number of array elements
>>> b.dtype	Data type of array elements
>>> b.dtype.name	Name of data type
>>> b.astype(int)	Convert an array to a different type

Asking For Help

>>> np.info(np.ndarray.dtype)

Array Mathematics

Arithmetic Operations

>>> g = a - b array([[-0.5, 0. , 0.],	Subtraction
[-3., -3., -3.]]) >>> np.subtract(a,b) >>> b + a array([[2.5, 4., 6.],	Subtraction Addition
[5. , 7. , 9.]]) >>> np.add(b,a) >>> a / b array([[0.66666667, 1. , 1.],	Addition Division
>>> np.divide(a,b) >>> a * b array([[1.5, 4. , 9.],	Division Multiplication
<pre>[4., 10., 18.]]) >>> np.multiply(a,b) >>> np.exp(b) >>> np.sqrt(b) >>> pp.sin(a)</pre>	Multiplication Exponentiation Square root Print sines of an array
>>> np.sin(a) >>> np.cos(b) >>> np.log(a) >>> e.dot(f) array([[7., 7.],	Element-wise cosine Element-wise natural logarithm Dot product

Comparison

>>> a == b array([[False, True, True],	Element-wise comparison
<pre>[False, False, False]], dtype=bool) >>> a < 2 array([True, False, False], dtype=bool)</pre>	Element-wise comparison
	Array-wise comparison

Aggregate Functions

>>> a.sum()	Array-wise sum
>>> a.min()	Array-wise minimum value
>>> b.max(axis=0)	Maximum value of an array row
>>> b.cumsum(axis=1)	Cumulative sum of the elements
>>> a.mean()	Mean
>>> b.median()	Median
>>> a.corrcoef()	Correlation coefficient
>>> np.std(b)	Standard deviation

Copying Arrays

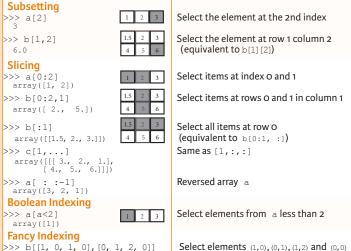
>>> h = a.view()	Create a view of the array with the same data
>>> np.copy(a)	Create a copy of the array
>>> h = a.copy()	Create a deep copy of the array

Sorting Arrays

>>> a.sort()	Sort an array
>>> c.sort(axis=0)	Sort the elements of an array's axis

Subsetting, Slicing, Indexing

Also see Lists



Array Manipulation

array([4. , 2. , 6. , 1.5])

>>> b[[1, 0, 1, 0]][:,[0,1,2,0]]

Transposing Array
>>> i = np.transpose(b)
>>> i.T

Changing Array Shape >>> b.ravel() >>> g.reshape(3,-2)

Adding/Removing Elemen
>>> h.resize((2,6))
>>> np.append(h,g)
>>> np.insert(a, 1, 5)

>>> np.delete(a,[1]) Combining Arrays

```
>>> np.concatenate((a,d),axis=0)
  array([ 1, 2, 3, 10, 15, 20])
>>> np.vstack((a,b))
 array([[ 1., 2., 3.], [ 1.5, 2., 3.], [ 4., 5., 6.]])
>>> np.r [e,f]
>>> np.hstack((e,f))
 array([[ 7., 7., 1., 0.],
         [ 7., 7., 0., 1.]])
>>> np.column stack((a,d))
 array([[ 1, 10],
          2, 15],
         [ 3, 20]])
>>> np.c [a,d]
```

Splitting Arrays

```
>>> np.hsplit(a,3)
[array([1]),array([2]),array([3])]
>>> np.vsplit(c,2)
```

Permute array dimensions Permute array dimensions

Select a subset of the matrix's rows

and columns

Flatten the array Reshape, but don't change data

Return a new array with shape (2,6) Append items to an array Insert items in an array Delete items from an array

Concatenate arrays

Stack arrays vertically (row-wise)

Stack arrays vertically (row-wise) Stack arrays horizontally (column-wise)

Create stacked column-wise arrays

Create stacked column-wise arrays

Split the array horizontally at the 3rd

Split the array vertically at the 2nd index