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.

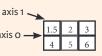
Use the following import convention: >>> import numpy as np

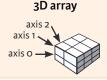


NumPy Arrays



2D array





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)]],
                 dtvpe = float)
```

Initial Placeholders

>>> np.zeros((3,4)) >>> np.ones((2,3,4),dtype=np.int16) >>> d = np.arange(10,25,5)	С
>>> np.linspace(0,2,9)	St C St
>>> e = np.full((2,2),7) >>> f = np.eye(2) >>> np.random.random((2,2))	C
>>> np.empty((3,2))	Č

Create an array of zeros Create an array of ones Create an array of evenly paced values (step value) reate an array of evenly paced values (number of samples) reate a constant array Create a 2X2 identity matrix Create an array with random values Create an empty array

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")
>>> np.genfromtxt("my file.csv", delimiter=',')
>>> np.savetxt("myarray.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		Subtraction
array([[-0.5, 0. , 0.],		
[-3. , -3. , -3.]])		
>>> np.subtract(a,b)		Subtraction
>>> b + a		Addition
array([[2.5, 4. , 6.],		
[5., 7., 9.]])		
>>> np.add(b,a)		Addition
>>> a / b		Division
],	
]])	
>>> np.divide(a,b)		Division
>>> a * b		Multiplication
array([[1.5, 4., 9.],		
[4. , 10. , 18.]])		
>>> np.multiply(a,b)		Multiplication
>>> np.exp(b)		Exponentiation
>>> np.sqrt(b)		Square root
>>> np.sin(a)		Print sines of an array
>>> np.cos(b)		Element-wise cosine
>>> np.log(a)		Element-wise natural logarithm
>>> e.dot(f)		Dot product
array([[7., 7.],		'

Comparison

<pre>>>> a == b array([[False, True, True],</pre>	Element-wise comparison
[False, False, False]], dtype=bool)	-1
>>> a < 2 array([True, False, False], dtype=bool)	Element-wise comparison
>>> np.array_equal(a, b)	Array-wise comparison

Aggregate Functions

[7., 7.]])

>>> 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

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

Subsetting, Slicing, Indexing

Subsetting

>>> a[2]

6.0 Slicing

>>> b[1,2]

>>> a[0:2]

>>> b[:1]

array([1, 2])

array([2., 5.])

array([[1.5, 2., 3.]]) >>> c[1,...]

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

>>> b[0:2,1]

>>> a[: :-1]

>>> a[a<2]

array([1])

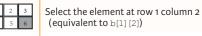
Fancy Indexing

array([3, 2, 1]) **Boolean Indexing**

Also see Lists

1 2 3 Select the element at the 2nd index 1.5 2 3

1 2 3



Select items at index 0 and 1

Select items at rows 0 and 1 in column 1

Select all items at row o (equivalent to b[0:1, :]) Same as [1,:,:]

Reversed array a

Select elements from a less than 2

Select elements (1,0), (0,1), (1,2) and (0,0)

Select a subset of the matrix's rows and columns

Array Manipulation

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

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

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

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

```
>>> i.T
Changing Array Shape
```

```
>>> b.ravel()
>>> g.reshape(3,-2)
```

Adding/Removing Elements

>>> np.concatenate((a,d),axis=0)

```
>>> h.resize((2,6))
>>> np.append(h,g)
>>> np.insert(a, 1, 5)
>>> np.delete(a,[1])
```

Combining Arrays

```
array([ 1, 2, 3, 10, 15, 20])
>>> np.vstack((a,b))
 >>> 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

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

