

Introduction to NumPy

Exercises:

1. Import numpy as np and see the version

```
import numpy as np
np.__version__
'1.26.2'
```

2. How to create a 1D array?

```
x = np.arange(10)

x

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

3. How to create a boolean array?

4. How to extract items that satisfy a given condition from 1D array?

```
 \begin{array}{l} x = \text{np.arange(30)} \\ x[x\%2==1] \\ \text{array([ 1,  3,  5,  7,  9,  11,  13,  15,  17,  19,  21,  23,  25,  27,  29])} \end{array}
```

5. How to replace items that satisfy a condition with another value in numpy array?

```
 \begin{array}{l} x = \text{np.arange(30)} \\ x[x\%2==1] = -1 \\ x \\ \\ \text{array([ 0, -1, \ 2, -1, \ 4, -1, \ 6, -1, \ 8, -1, \ 10, -1, \ 12, -1, \ 14, -1, \ 16, \\ \\ & -1, \ 18, \ -1, \ 20, \ -1, \ 22, \ -1, \ 24, \ -1, \ 26, \ -1, \ 28, \ -1]) \\ \end{array}
```

6. How to get the common items between two python numpy arrays?

```
a = np.array([1,2,3,2,3,4,3,4,5,6,7])
b = np.array([7,2,10,2,7,4,9,4,9,8])
np.intersectld(a,b)
array([2, 4, 7])
```

7. How to get all the items between two python numpy arrays?

```
np.union1d(a,b)
array([ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

8. How to extract all numbers between a given range from a numpy array?

```
a = np.array([7,2,10,2,12,7,4,14,9,4,9,8]) a[(a>6) \& (a<=10)] array([7, 10, 7, 9, 9, 8])
```

9. How to reverse the rows of a 2D array?

```
arr = np.arange(12).reshape(4,3)
print("Original Array: ")
print(arr)
```

```
print("\nModified Array: ")
print(arr[::-1,])

Original Array:
[[ 0  1  2]
  [ 3  4  5]
  [ 6  7  8]
  [ 9  10  11]]

Modified Array:
[[ 9  10  11]
  [ 6  7  8]
  [ 3  4  5]
  [ 0  1  2]]
```

10. How to swap two columns in a 2d numpy array?¶

```
arr = np.arange(20).reshape(4,5)

print('Original array')
print(arr)
print("\nModified array")
print(arr[:, [4,1,0,3,2]])

Original array
[[ 0  1  2  3  4]
  [ 5  6  7  8  9]
  [10  11  12  13  14]
  [15  16  17  18  19]]

Modified array
[[ 4  1  0  3  2]
  [ 9  6  5  8  7]
  [14  11  10  13  12]
  [19  16  15  18  17]]
```

Python For Data Science Cheat Sheet

NumPv Basics

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



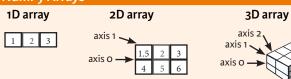
NumPy

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



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>
>>>	<pre>np.savetxt("myarray.txt", a, delimiter=" ")</pre>

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)	Subtraction
>>> b + a array([[2.5, 4., 6.],	Addition
[5. , 7. , 9.]]) >>> np.add(b,a)	Addition
>>> a / b array([[0.66666667, 1. , 1.], [0.25 , 0.4 , 0.5]]	
>>> np.divide(a,b)	Division
>>> a * b array([[1.5, 4., 9.], [4., 10., 18.]])	Multiplication
>>> 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 logarithr
>>> e.dot(f) array([[7., 7.],	Dot product
[7., 7.]])	

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

Subsetting

>>> a[2]

>>> b[1,2]

>>> a[0:2]

>>> b[:1]

array([1, 2])

array([2., 5.])

>>> b[0:2,1]

>>> c[1,...]

>>> a[: :-1]

>>> a[a<2]

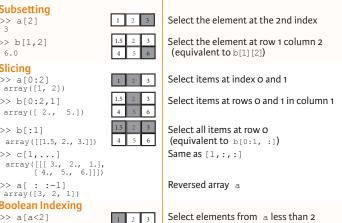
array([1])

Fancy Indexing

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

6.0 Slicina

```
Also see Lists
```



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

```
Select a subset of the matrix's rows
and columns
```

Array Manipulation

array([1, 2, 3, 10, 15, 20])

[7., 7., 0., 1.]])

[array([1]),array([2]),array([3])]

>>> np.hstack((e,f))
array([[7., 7., 1., 0.],

>>> np.column stack((a,d))

>>> np.vstack((a,b)) array([[1. , 2. , 3.], [1.5, 2. , 3.], [4. , 5. , 6.]])

>>> np.r [e,f]

array([[1, 10], 2, 15], [3, 20]])

Splitting Arrays

>>> np.hsplit(a,3)

>>> np.vsplit(c,2)

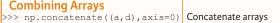
>>> np.c [a,d]

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

array([4. , 2. , 6. , 1.5]) >>> b[[1, 0, 1, 0]][:,[0,1,2,0]]

array dimensions array dimensions
ne array but don't change data

Adding/ Kemoving Elements	
>>> h.resize((2,6))	Return a new array with shape (2,6)
>>> np.append(h,g)	Append items to an array
>>> np.insert(a, 1, 5)	Insert items in an array
>>> np.delete(a,[1])	Delete items from an array



DataCamp Learn Python for Data Science Interactively

Stack arrays verti	cally (row-wise)
--------------------	------------------

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

Create stacked column-wise arrays

Create stacked	column-w	ise arrays
----------------	----------	------------

Split the array horizontally at the 3rd	
index	

Split the array vertically at the 2nd in	dex