# NumPy Basics Cheat Sheet (2021), Python for Data Science

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

Check out the different sections below to learn the various array functions and tools NumPy offers.

***Sections:*** *1. [Creating Arrays](https://towardsdatascience.com/numpy-basics-cheat-sheet-2021-python-for-data-science-89c483773880" \l "7993)  
2. [Inspecting Your Array](https://towardsdatascience.com/numpy-basics-cheat-sheet-2021-python-for-data-science-89c483773880" \l "c7a8)  
3. [Array Mathematics](https://towardsdatascience.com/numpy-basics-cheat-sheet-2021-python-for-data-science-89c483773880" \l "0116)  
4. [Comparisons](https://towardsdatascience.com/numpy-basics-cheat-sheet-2021-python-for-data-science-89c483773880" \l "d2c8)  
5. [Aggregate Functions](https://towardsdatascience.com/numpy-basics-cheat-sheet-2021-python-for-data-science-89c483773880" \l "d86e)  
6. [Subsetting, Slicing, Indexing](https://towardsdatascience.com/numpy-basics-cheat-sheet-2021-python-for-data-science-89c483773880" \l "795b)  
7. [Adding/Removing Elements](https://towardsdatascience.com/numpy-basics-cheat-sheet-2021-python-for-data-science-89c483773880" \l "40d7)  
8. [Array Manipulation](https://towardsdatascience.com/numpy-basics-cheat-sheet-2021-python-for-data-science-89c483773880" \l "8416)  
9. [Copying Arrays](https://towardsdatascience.com/numpy-basics-cheat-sheet-2021-python-for-data-science-89c483773880" \l "a47c)  
10. [Sorting Arrays](https://towardsdatascience.com/numpy-basics-cheat-sheet-2021-python-for-data-science-89c483773880" \l "42b0)  
11.*[*Data Types*](https://towardsdatascience.com/numpy-basics-cheat-sheet-2021-python-for-data-science-89c483773880#9db0)

Creating Arrays

A NumPy array is a grid of values, all of the same type. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

In this section, you’ll learn how to create these various types of arrays.

* 1-dimensional array

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

* 2-dimensional array

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

* 3-dimensional array

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

* Array of zeros

>>> np.zeros((3,4))  
 array([[0., 0., 0., 0.],  
 [0., 0., 0., 0.],  
 [0., 0., 0., 0.]])

* Array of ones

>>> np.ones((3,4))  
 array([[1., 1., 1., 1.],  
 [1., 1., 1., 1.],  
 [1., 1., 1., 1.]])

* Array of evenly spaced values (step value)

>>> np.arange(10,25,5)  
 array([10, 15, 20])

* Array of evenly spaced values (number of samples)

>>> np.linspace(0,2,9)  
 array([0.0, 0.25, 0.5 , 0.75, 1.0, 1.25, 1.5 , 1.75, 2.0])

* 2x2 identity matrix

>>> np.eye(2)  
 array([[1., 0.],  
 [0., 1.]])

* Array with random values

>>> np.random.random((2,2))  
 array([[0.42326354, 0.56737208],  
 [0.01597192, 0.79065649]])

Inspecting Your Array

In this section, you’ll learn how to extract the specific characteristics of an array including length, data type, size, and dimensions.

**a** and **b** in the code below are used as examples of arrays throughout this section.

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

* Array dimensions

>>> b.shape  
 (2,3)

* Length of array

>>> len(a)  
 3

* Number of array dimensions

>>> b.ndim  
 2

* Number of array elements

>>> b.size  
 6

* Data type of array elements

>>> b.dtype  
 dtype('float64')

* Covert an array to a different type

>>> b.astype(int)  
 array([[1, 2, 3],  
 [4, 5, 6]])

Array Mathematics

In this section, you’ll learn how to perform various arithmetic operations using two different arrays.

**a** and **b** in the code below are used as examples of arrays throughout this section.

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

* Subtraction

>>> a - b  
 array([[-0.5, 0. , 0. ],  
 [-3. , -3. , -3. ]])

* Subtraction.v2

>>> np.subtract(a,b)  
 array([[-0.5, 0. , 0. ],  
 [-3. , -3. , -3. ]])

* Addition

>>> a + b  
 array([[2.5, 4. , 6. ],  
 [5. , 7. , 9. ]])

* Addition.v2

>>> np.add(a,b)  
 array([[2.5, 4. , 6. ],  
 [5. , 7. , 9. ]])

* Division

>>> a/b  
 array([[0.66, 1. , 1. ],  
 [0.25, 0.4, 0.5]])

* Division.v2

>>> np.divide(a,b)  
 array([[0.66, 1. , 1. ],  
 [0.25, 0.4, 0.5]])

* Multiplication

>>> a\*b  
 array([[ 1.5, 4. , 9. ],  
 [ 4. , 10. , 18. ]])

* Multiplication.v2

>>> np.multiply(a,b)  
 array([[ 1.5, 4. , 9. ],  
 [ 4. , 10. , 18. ]])

* Exponentiation

>>> np.exp(b)  
 array([[ 4.48168907, 7.3890561 , 20.08553692],  
 [ 54.59815003, 148.4131591 , 403.42879349]])

* Square root

>>> np.sqrt(b)  
 array([[1.22474487, 1.41421356, 1.73205081],  
 [2. , 2.23606798, 2.44948974]])

* Logarithm

>>> np.log(b)  
 array([[0.40546511, 0.69314718, 1.09861229],  
 [1.38629436, 1.60943791, 1.79175947]])

Comparisons

In this section, you’ll learn how to compare arrays using a specific element or another array.

**a** and **b** in the code below are used as examples of arrays throughout this section.

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

* Element-wise comparison

>>> a == b  
 array([[False, True, True],  
 [False, False, False]])

* Element-wise comparison.v2

>>> a < 2  
 array([True, False, False])

* Array-wise comparisons

>>> np.array\_equal(a,b)  
 False

Aggregate Functions

In this section, you’ll learn how to use various aggregate functions like sum, min, max, mean, and median on an array.

**a** and **b** in the code below are used as examples of arrays throughout this section.

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

* Array sum

>>> a.sum()  
 6

* Array minimum value

>>> a.min()  
 1

* Maximum value of an array row

>>> b.max(axis = 0)  
 array([4., 5., 6.])

* Cumulative sum of the elements

>>> b.cumsum(axis = 1)  
 array([[ 1.5, 3.5, 6.5],  
 [ 4. , 9. , 15. ]])

* Mean

>>> a.mean()  
 2

Subsetting, Slicing, Indexing

In this section, you’ll learn how to retrieve specific values from an array.

**a** and **b** in the code below are used as examples of arrays throughout this section.

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

* Select the element at the 2nd index

>>> a[2]  
 3

* Select the element at row 1 in column 2

>>> b[1,2]  
 6

* Select items at index 0 and 1

>>> a[0:2]  
 array([1, 2])

* Select items at row 0 and 1 in column 1

>>> b[0:2,1]  
 array([2., 5.])

* Select all items at row 0

>>> b[:1]  
 array([[1.5, 2. , 3. ]]

* Reverse array

>>> a[::-1]  
 array([3, 2, 1])

* Select elements that are less than 3

>>> a[a<3]  
 array([1, 2])

Adding/Removing Elements

In this section, you’ll learn how to add specific elements into an array and how to remove them.

**a** and **b** in the code below are used as examples of arrays throughout this section.

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

* Return a new array with shape (1,6)

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

* Append items to an array

>>> np.append(a,7)  
 array([1, 2, 3, 7])

* Insert items in an array

>>> np.insert(a,1,9)  
 array([1, 9, 2, 3])

* Delete items from an array

>>> np.delete(a,[1])  
 array([1, 3])

Array Manipulation

In this section, you’ll learn how to change the look of an array. You’ll learn how to flatten, transpose, reshape, and concatenate arrays.

**a, b,**and **d** in the code below are used as examples of arrays throughout this section.

>>> a = np.array([1, 2, 3])  
>>> b = np.array([[1.5, 2. , 3. ], [4. , 5. , 6. ]])  
>>> *d =* np.array([10, 15, 20])

* Flatten an array

>>> b.ravel()  
 array([1.5, 2. , 3. , 4. , 5. , 6. ])

* Reshape, but don’t change the data

>>> b.reshape(3,2)  
 array([[1.5, 2. ],  
 [3. , 4. ],  
 [5. , 6. ]])

* Concatenate arrays

>>> np.concatenate((a,d),axis = 0)  
 array([ 1, 2, 3, 10, 15, 20])

* Transpose array

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

Copying Arrays

In this section, you’ll learn how to create a copy of an array for future use.

**a**in the code below is used as an example of an array throughout this section.

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

* Create a copy of the array

>>> np.copy(a)  
 array([1, 2, 3])

* Create a deep copy of the array

>>> a.copy()  
 array([1, 2, 3])

Sorting Arrays

In this section, you’ll learn how to sort an array so it’s decreasing to increasing or vice versa.

**a**and**c**in the code below are used as examples of arrays throughout this section.

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

* Sort an array

>>> a.sort()  
 array([1, 2, 3])

* Sort the elements of an array’s axis

>>> c.sort(axis = 0)

Data Types

These are the different data types that can be in a NumPy array.

* 64-bit integer types: np.int64
* Double-precision floating point: np.float32
* Boolean type string TRUE and FALSE: np.bool
* Python object type values: np.object
* Fixed-length string type: np.string
* Fixed-length Unicode type: np.unicode\_

Python is the top dog when it comes to data science for now and in the foreseeable future. Knowledge of NumPy, one of its most powerful libraries is often a requirement for Data Scientists today.

Use this cheat sheet as a guide in the beginning and come back to it when needed, and you’ll be well on your way to mastering the NumPy library.