# Theory: Intro to NumPy



30 minutes 0 / 5 problems solved



**NumPy** (short for *Numerical Python*) is a Python library fundamental for scientific computing. It supports a variety of high-level mathematical functions and is broadly used in data science, machine learning, and big data applications. With its help, you will be able to perform linear algebra calculations easily, as well as statistical, logical, and other operations, making use of numerous built-in functions.

Most parts of NumPy that require high execution speed are written in C, which makes the operations much faster than the corresponding ones in Python itself. Owing to its efficiency and convenience, the library has gained vast popularity among data scientists who work with large datasets and need to perform speed computations.

## §1. Installation

Firstly, to start working with NumPy, we need to install it, which can be easily done with pip:

```
pip install numpy
```

You can read more about the installation on the official page of the scientific python distribution.

Then, import the library before starting your work:

```
import numpy as np
```

## §2. NumPy arrays

The core NumPy object is an n-dimensional **array**, also known as **ndarray**. The simplest way to create a NumPy array is to convert a Python list:

```
first = np.array([1, 2, 3, 4, 5])
print(first)                    # [1 2 3 4 5]
print(type(first))              # <class 'numpy.ndarray'>
```

In the example above, `first` is a one-dimensional array that is treated as a **vector**. As you can see, when printed, it is rendered without commas, as opposed to Python lists.

Similarly, we can create a two-dimensional Numpy array from the corresponding Python list. Two- and more dimensional arrays are treated as **matrices**.

```
second = np.array([[1, 1, 1],
                   [2, 2, 2]])

print(second)    # [[1 1 1]
                 #  [2 2 2]]
```

## §3. NumPy arrays vs Python lists

As you can see, NumPy arrays resemble a Python built-in list data type. However, there are a few crucial differences:

- Unlike Python lists, NumPy arrays can only contain **elements of the same type**, usually numbers, due to the specifics of application fields.
- NumPy arrays are much more **memory-efficient** and much **faster** than Python lists when working with large datasets.
- **Arithmetic operations** differ when executed on Python lists or NumPy arrays.

Current topic:

Intro to NumPy

Topic depends on:

✕ Introduction to matrices

✓ Type casting 12★

✓ List 13★ Stage 1

✓ Declaring a function 10★ Stage 1

✕ Pip Stage 1

Topic is required for:

Array creation and indexing

Data types in NumPy

Operations with an array

Operations with several arrays

Table of contents:

↑ Intro to NumPy

§1. Installation

§2. NumPy arrays

§3. NumPy arrays vs Python lists

§4. Learning sizes

§5. Recap

Feedback & Comments

Let's take a look at arithmetic operations that can be applied both to arrays and to lists. All differences in them can be explained by the fact that Numpy arrays are created for computing and treated as vectors and matrices, while Python lists are a datatype made just to store data.

To illustrate it, we'll create two lists and two arrays containing the same elements:

```
list_a = [1, 2, 3, 4]
array_a = np.array(list_a)

list_b = [11, 22, 33, 44]
array_b = np.array(list_b)
```

First, let's find their sum. The addition of two arrays returns their sum as when we add two vectors.

```
print(array_a + array_b)  # [12 24 36 48]
```

For this reason, we can't add up arrays of different lengths, a `ValueError` will appear.

```
array_c = np.array([111, 222])
print(array_a + array_c)        # ValueError
```

When we try to add a list and an array, the former is converted to an array, so the result is also a sum of vectors.

```
print(list_a + array_a)   # [2 4 6 8]
```

However, when applied to lists, addition just merges them together.

```
print(list_a + list_b)    # [1, 2, 3, 4, 11, 22, 33, 44]
```

Similarly, if we try to multiply a list by `n`, we'll get the list repeated ***n*** times, while with an array, each element will be multiplied by ***n***:

```
print(list_a * 3)   # [1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
print(array_a * 3)  # [3 6 9 12]
```

## §4. Learning sizes

There're a number of ways to learn more about an array without printing it.

```
first = np.array([1, 2, 3, 4, 5])
second = np.array([[1, 1, 1],
                   [2, 2, 2]])
```

To find out the dimensions size, use `shape`. The first number of the output indicates the number of rows and the second — the number of columns.

```
print(second.shape)  # (2, 3)
```

If the NumPy array has only one dimension, the result will be a bit different:

```
print(first.shape)   # (5,)
```

In this case, the first number is not the number of rows but rather the number of elements in the only dimension, and the empty place after the comma means that there's no second dimension.

The length of the `shape` tuple is the number of axes, `ndim`.

```
print(first.ndim)   # 1
print(second.ndim)  # 2
```

The function `len()` returns the array's length, and `size` gives us the number of elements in the array.

```
print(len(first), first.size)    # 5 5
print(len(second), second.size)  # 2 6
```

Note that in the first case they return the same value, while in the second case the numbers differ. The thing is, `len()` works as it would work with regular lists, so if we regard the two-dimensional array above as a list containing two nested lists, it becomes clear that for finding its length only the nested lists are counted. Size, on the contrary, counts each single element in all nested lists.

Another thing to point out is that both length and size can also be got from its shape: length is actually the length of the first dimension, so it equals `shape[0]`, and size is the total number of elements, which equals the product of all elements in the shape tuple.

## §5. Recap

In this topic, we've learned the basics of NumPy:

- what is NumPy and what it can be used for,
- how to install and import the library,
- arrays, the basic datatype of Numpy,
- the difference between NumPy arrays and Python lists,
- ways to get information about an array's content.

Now, let's practice the acquired knowledge so that you'll be able to use it in the future.

Report a typo

**48** users liked this theory. **2** didn't like it. **What about you?**

😍 🙂 😐 ☹️ 😡

Start practicing

Comments (1) Hints (0) Useful links (1) Show discussion