Vectors

In this lesson, we will learn about various methods to create vectors in Python.

WE'LL COVER THE FOLLOWING

In Python, **vectors** are *one-dimensional arrays* and are the most commonly used data structure in NumPy.

Do not confuse NumPy vectors with mathematical vectors.

Let's see how they're created:

Creation

There are many ways to create 1-D arrays and we can create them according to our needs. Let's discuss these different ways below:

Method 1

We can create an array by entering the individual elements of an array. See the example below:

```
import numpy as np

x = np.array([1, 3, 5, 7, 9])
print(x)
```

In the code above, we are actually converting a Python `list` to a vector using the `np.array()` function with its input argument being a `list`.

Method 2

Another function to create an array is `np.ones(size)`, which creates an array of the specified `size` filled with the value 1.

There is an analogous function `np.zeros(size)` to create an array filled with the value 0.

```
import numpy as np

v1 = np.ones(5)
v0 = np.zeros(5)
print(v1)
print(v0)
```

> **Note:** Data type of values inside the vectors generated from `ones()` and `zeros()` functions are floating points.

Method 3

We can initialize an array using the `arange()` function. This function can take up to 3 arguments.

```
np.arange(start, end, step)
```

The first argument is the *start point*, second argument is the *end point* and third argument is the *step size*.

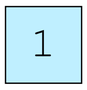
Let's look at the possible argument configurations of the `arange()` function in the `numpy` module:

```
import numpy as np

print(np.arange(1, 7))       # Takes default steps of 1 and doesn't include 7
print(np.arange(5))      # Starts at 0 by defualt and ends at 4, giving 5 numbers
print(np.arange(1, 10, 3))      # Starts at 1 and ends at less than 10,
                                # with a step size of 3
```

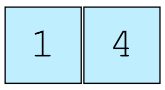
In line 5, the array will be generated according to the sequence: $1, 4, 7, 10, \ldots$ and so on. But since 10 is the upper limit, the sequence stops at 7.

Below is an illustration of this concept.



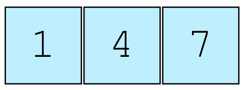
np.arrange(1, 10, 3)

1 of 5



np.arrange(1, 10, 3)

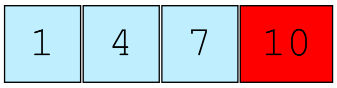
2 of 5



np.arrange(1, 10, 3)

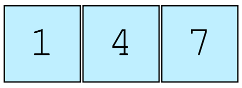
3 of 5

10 is the end point and cannot be included!



np.arrange(1, 10, 3)

4 of 5



Output vector

5 of 5

Method 4

We can also use the `linspace()` function to define an array with equally spaced numeric elements and both endpoints **included**.

```
np.linspace(start, end, size)
```

Run the code below to see the implementation of `linspace()`:

```
import numpy as np

print(np.linspace(1, 12, 12))
print(np.linspace(1, 12, 5))
print(np.linspace(1, 12, 3))
```

In the next lesson, let’s learn about multidimensional arrays.