Numpy

> NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

https://numpy.org/doc/stable/index.html (https://numpy.org/doc/stable/index.html)

Basics:

https://numpy.org/doc/stable/user/absolute_beginners.html (https://numpy.org/doc/stable/user/absolute_beginners.html)

Installation:

In [1]:
```
import numpy as np
```

Vectors: One-dimensional Arrays

Create a vector `v` that contains the numbers `7, 8, 9, 10`. Use the `np.array` function (Link to numpy documentation (https://numpy.org/doc/stable/reference/generated/numpy.array.html)).

In [2]:
```
v = np.array([7, 8, 9, 10])
```

In [3]:
```
type(v)
```

Out[3]:
```
numpy.ndarray
```

In [4]:
```
v
```

Out[4]:
```
array([ 7,  8,  9, 10])
```

In [5]:
```
print(v)
```

```
[ 7  8  9 10]
```

NumPy vectors might look like lists, at first sight, but they are a different data type

In [31]:
```
l = [7, 8, 9, 10]
```

In [32]:
```
l
```

Out[32]:
```
[7, 8, 9, 10]
```

In [8]:
```
type(v) == type(l)
```

Out[8]:
```
False
```

Indexing and Slicing is quite similar

In [9]:
```
v[1:3]
```

Out[9]:
```
array([8, 9])
```

In [10]:
```
len(v)
```

Out[10]:
```
4
```

In [11]:
```
l[1:3]
```

Out[11]:
```
[8, 9]
```

In [12]:
```
len(l)
```

Out[12]:
```
4
```

Create a vector `v` that contains the values 14 to 47 with the help of the function `np.arange` (Link to numpy documentation (https://numpy.org/doc/stable/reference/generated/numpy.arange.html)).

In [13]:
```
v = np.arange(14, 48)
```

In [14]:
```
v
```

Out[14]:
```
array([14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
       31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47])
```

What is the data type of v?

In [15]:
```
v.dtype
```

Out[15]:
```
dtype('int32')
```

What is the dimension of vector v?

In [16]:
```
v.ndim
```

Out[16]:
```
1
```

In [17]:
```
v.shape
```

Out[17]:
```
(34,)
```

Replace the 9. value with the number 12

In [18]:
```
v[8] = 12
v
```

Out[18]:
```
array([14, 15, 16, 17, 18, 19, 20, 21, 12, 23, 24, 25, 26, 27, 28, 29, 30,
       31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47])
```

Reverse the array (via *slicing*), to get `[47,46,....,14]` .

In [20]:
```
v[::-1]
```

Out[20]:
```
array([47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31,
       30, 29, 28, 27, 26, 25, 24, 23, 12, 21, 20, 19, 18, 17, 16, 15, 14])
```

In [21]:
```
v
```

Out[21]:
```
array([14, 15, 16, 17, 18, 19, 20, 21, 12, 23, 24, 25, 26, 27, 28, 29, 30,
       31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47])
```

In [22]:
```
v[:-1]
```

Out[22]:
```
array([14, 15, 16, 17, 18, 19, 20, 21, 12, 23, 24, 25, 26, 27, 28, 29, 30,
       31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46])
```

NumPy Arrays handle operators differently that what it was for lists.

Instead of applying the operator on the whole list, the operations are done on each element of the vector.

In [23]:
```
v + 1
```

Out[23]:
```
array([15, 16, 17, 18, 19, 20, 21, 22, 13, 24, 25, 26, 27, 28, 29, 30, 31,
       32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48])
```

In [24]:
```
v * 2
```

Out[24]:
```
array([28, 30, 32, 34, 36, 38, 40, 42, 24, 46, 48, 50, 52, 54, 56, 58, 60,
       62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94])
```

In [25]:
```
v[:3] * 3
```

Out[25]:
```
array([42, 45, 48])
```

In [33]:
```
l * 2
```

Out[33]:
```
[7, 8, 9, 10, 7, 8, 9, 10]
```

Matrices: Two-dimensional Arrays

Create a NumPy Array x with the help of the `np.array` function:

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

In [34]:
```
x = np.array(
    [[1,  7, 13],
     [2,  8, 14],
     [3,  9, 15],
     [4, 10, 16],
     [5, 11, 17],
     [6, 12, 18]]
)
```

In [35]:
```
print(x)
```

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

What is the dimension of matrix x?

In [36]:
```
x.shape
```

Out[36]:
```
(6, 3)
```

Select the 3rd column of the Array

In [38]:
```
x[:,2]
```

Out[38]:
```
array([13, 14, 15, 16, 17, 18])
```

Select the 3rd row

In [41]:
```
x[2,:]
```

Out[41]:
```
array([ 3,  9, 15])
```

Select all the elements up until element 2 of the array

In [42]:
```
x[:2]
```

Out[42]:
```
array([[ 1,  7, 13],
       [ 2,  8, 14]])
```

Select all the elements starting from Index 2

In [43]:
```
x[2:]
```

Out[43]:
```
array([[ 3,  9, 15],
       [ 4, 10, 16],
       [ 5, 11, 17],
       [ 6, 12, 18]])
```

Diagonal matrics

Create a diagonal matrix `m_diag` , in which the elements of the diagonal are the first 6 numbers of `v = np.arange(1,12)` .

Use the function `np.diag` (Link to numpy documentation (https://numpy.org/doc/stable/reference/generated/numpy.diag.html)).

In [48]:
```
a = np.arange(1,12)[:6]

m_diag = np.diag(a)
m_diag
```

Out[48]:
```
array([[1, 0, 0, 0, 0, 0],
       [0, 2, 0, 0, 0, 0],
       [0, 0, 3, 0, 0, 0],
       [0, 0, 0, 4, 0, 0],
       [0, 0, 0, 0, 5, 0],
       [0, 0, 0, 0, 0, 6]])
```

In [49]:
```
a
```

Out[49]:
```
array([1, 2, 3, 4, 5, 6])
```

In [50]:
```
m_diag.shape
```

Out[50]:
```
(6, 6)
```