Numpy:

Documentação : https://numpy.org/doc/stable/user/absolute_beginners.html#numpy-the-absolute-basics-for-beginners (https://numpy.org/doc/stable/user/absolute_beginners.html#numpy-the-absolute-basics-for-beginners)

Instalação:

Insira o código abaixo no Anaconda Prompt: **pip install numpy**

https://numpy.org/install/ (https://numpy.org/install/)

Importação:

https://numpy.org/doc/stable/user/absolute_beginners.html#how-to-import-numpy (https://numpy.org/doc/stable/user/absolute_beginners.html#how-to-import-numpy)

In [1]:

```
import numpy as np
```

O que é um array?

Tipos de arrays: ndarrays -> significam arrays com N dimensões
1-D array-> Possui apenas uma dimensão. Será comumente chamado de **vetor ou vector**
2-D array -> Possui 2 dimensões. Será comumente chamado de **matriz ou matrix**
3-D ou Mais array -> Possui 3 ou mais dimensões. Será comumente chamado de **tensor**

https://numpy.org/doc/stable/reference/arrays.html#arrays (https://numpy.org/doc/stable/reference/arrays.html#arrays)

Criando um Array:

np.array()

https://numpy.org/doc/stable/reference/generated/numpy.array.html?highlight=numpy%20array#numpy-array (https://numpy.org/doc/stable/reference/generated/numpy.array.html?highlight=numpy%20array#numpy-array)

In [2]:

```
a = np.array([1,2,3,4,5,6])
print(a)
print(type(a))
```

```
[1 2 3 4 5 6]
<class 'numpy.ndarray'>
```

np.zeros()

https://numpy.org/doc/stable/reference/generated/numpy.zeros.html (https://numpy.org/doc/stable/reference/generated/numpy.zeros.html)

In [3]:

```
zero_array = np.zeros(shape = (5,3,6))
print(zero_array)
```

```
[[[0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]]

 [[0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]]

 [[0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]]

 [[0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]]

 [[0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]]]
```

np.ones()

https://numpy.org/doc/stable/reference/generated/numpy.ones.html (https://numpy.org/doc/stable/reference/generated/numpy.ones.html)

In [4]:

```
um_array = np.ones(2)
print(um_array)
```

```
[1. 1.]
```

no.empty()

https://numpy.org/doc/stable/reference/generated/numpy.empty.html
(https://numpy.org/doc/stable/reference/generated/numpy.empty.html)

In [5]:

```
vazio_array = np.empty(3)
print(vazio_array)
```

```
[1.0635808e-311 0.0000000e+000 4.9406565e-324]
```

np.arange()

https://numpy.org/doc/stable/reference/generated/numpy.arange.html
(https://numpy.org/doc/stable/reference/generated/numpy.arange.html)

In [6]:

```
# Dá uma sequência de números:
zero_dez = np.arange(10)
print(zero_dez)
pula_dois = np.arange(3,15,2)
print(pula_dois)
```

```
[0 1 2 3 4 5 6 7 8 9]
[ 3  5  7  9 11 13]
```

np.linspace()

https://numpy.org/doc/stable/reference/generated/numpy.linspace.html
(https://numpy.org/doc/stable/reference/generated/numpy.linspace.html)

In [7]:

```
array_linear = np.linspace(0, 100 , num = 20, endpoint = False, retstep = True)
print(array_linear)
```

```
(array([ 0.,  5., 10., 15., 20., 25., 30., 35., 40., 45., 50., 55., 60.,
       65., 70., 75., 80., 85., 90., 95.]), 5.0)
```

Descobrindo o tamanho de um array:

Número de dimensões : https://numpy.org/doc/stable/reference/generated/numpy.ndarray.ndim.html
(https://numpy.org/doc/stable/reference/generated/numpy.ndarray.ndim.html)
Número de items: https://numpy.org/doc/stable/reference/generated/numpy.ndarray.size.html
(https://numpy.org/doc/stable/reference/generated/numpy.ndarray.size.html) Formato :
https://numpy.org/doc/stable/reference/generated/numpy.ndarray.shape.html
(https://numpy.org/doc/stable/reference/generated/numpy.ndarray.shape.html)

In [8]:

```
zero_array = np.zeros(shape = (5,3,6))
print(zero_array)
```

```
[[[0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]]

 [[0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]]

 [[0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]]

 [[0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]]

 [[0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0.]]]
```

In [9]:

```
print(zero_array.shape)
print(zero_array.size)
print(zero_array.ndim)
```

```
(5, 3, 6)
90
3
```

Mudando o tamanho de um array:

https://numpy.org/doc/stable/reference/generated/numpy.reshape.html (https://numpy.org/doc/stable/reference/generated/numpy.reshape.html)

Rankeando um array:

https://numpy.org/doc/stable/reference/generated/numpy.sort.html (https://numpy.org/doc/stable/reference/generated/numpy.sort.html)

Transformando um Vetor (1-D) em uma matrix(2-D)

.newaxis:https://numpy.org/doc/stable/reference/constants.html?#numpy.newaxis (https://numpy.org/doc/stable/reference/constants.html#numpy.newaxis)
.expand_dims:https://numpy.org/doc/stable/reference/generated/numpy.expand_dims.html#numpy.expand_dims (https://numpy.org/doc/stable/reference/generated/numpy.expand_dims.html#numpy.expand_dims)

In [10]:

```
a = np.array( [ 1, 2, 3])
print(a.shape)
```

```
(3,)
```

In [11]:

```
a2 = a[np.newaxis,:]
print(a2.shape)
print(a2)
```

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

In [12]:

```
a2 = a[:,np.newaxis]
print(a2.shape)
print(a2)
```

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

In [13]:

```
a2[2][0] = 2
print(a2)
```

```
[[1]
 [2]
 [2]]
```

Concatenando arrays:

https://numpy.org/doc/stable/reference/generated/numpy.concatenate.html (https://numpy.org/doc/stable/reference/generated/numpy.concatenate.html)

In [14]:

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

c=np.concatenate((a,b))
d=np.concatenate((b,a))

print(c)
print(d)
```

```
[1 2 3 4 5 6]
[4 5 6 1 2 3]
```

Consultando itens de uma array:

https://numpy.org/doc/stable/user/absolute_beginners.html#indexing-and-slicing (https://numpy.org/doc/stable/user/absolute_beginners.html#indexing-and-slicing)

In [15]:

```
a = np.array([[1 , 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
print(a)
print('------------')
print(a[a<8])
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
------------
[1 2 3 4 5 6 7]
```

Operações com Arrays:

Soma : https://numpy.org/doc/stable/reference/generated/numpy.sum.html#numpy.sum (https://numpy.org/doc/stable/reference/generated/numpy.sum.html#numpy.sum)
Valor mínimo : https://numpy.org/doc/stable/reference/generated/numpy.ndarray.min.html (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.min.html)
Valor máximo : https://numpy.org/doc/stable/reference/generated/numpy.ndarray.max.html (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.max.html)
Média : https://numpy.org/doc/stable/reference/generated/numpy.ndarray.mean.html (https://numpy.org/doc/stable/reference/generated/numpy.ndarray.mean.html)

In [16]:

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

print(a.max())
print(a.min())
print(a.mean())
print(a.sum())
```

```
3
1
2.0
6
```

Gerando amostras aleatórias:

In [17]:

```
from numpy.random import default_rng
```

In [18]:

```
rng= default_rng()
aleatorio = rng.integers(10, size=(2,4))
print(aleatorio)
```

```
[[0 2 0 4]
 [8 5 3 2]]
```

Diferença entre Arrays e Listas:

In [19]:

```
a = np.array([1,3,4,5,6,5,7,8])
print("Essa é o array 'a':",a)
print("Esse é tipo de 'a':",type(a))
print('-------------------------------------')
lista_a=[1,3,4,5,6,5,7,8]
print("Essa é a 'lista_a':", lista_a)
print("Esse é tipo de 'lista_a':",type(lista_a))
```

```
Essa é o array 'a': [1 3 4 5 6 5 7 8]
Esse é tipo de 'a': <class 'numpy.ndarray'>
-------------------------------------
Essa é a 'lista_a': [1, 3, 4, 5, 6, 5, 7, 8]
Esse é tipo de 'lista_a': <class 'list'>
```

In [20]:

```
#arrays não permitem tipos de dados distintos:
a = np.array([1,'Daniel',2,3,4,5,6,7,8])
print(a)
print(type(a[0]))
```

```
['1' 'Daniel' '2' '3' '4' '5' '6' '7' '8']
<class 'numpy.str_'>
```

In [20]:

```
#já as listas sim:
lista_a = [1,'Daniel',2,3,4,5,6,7,8]
print(a)
print(type(lista_a[0]))
```

```
[1 3 4 5 6 5 7 8]
<class 'int'>
```

Comparando o processamento:

In [28]:

```
from time import process_time
lista_a = list(rng.integers(10, 100, 10000000))
print(type(lista_a))
lista_b = list(rng.integers(10, 100, 10000000))
c = lista_a*lista_b
```

```
<class 'list'>


---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-28-520a902e403f> in <module>
      3 print(type(lista_a))
      4 lista_b = list(rng.integers(10, 100, 10000000))
----> 5 c = lista_a*lista_b

TypeError: can't multiply sequence by non-int of type 'list'
```

In [29]:

```
print(type(lista_a))
print(len(lista_a))
```

```
<class 'list'>
10000000
```

In [30]:

```
c=[]
t1 = process_time()
for i in range(len(lista_a)):
    c.append(lista_a[i] * lista_b[i])
t2 = process_time()

print(t2-t1)
```

```
2.375
```

In [31]:

```
a = rng.integers(10, 100, 10000000)
b = rng.integers(10, 100, 10000000)
print(type(a))
print(a)
t1a=process_time()
c=a*b
t2a=process_time()
print(t2a-t1a)
```

```
<class 'numpy.ndarray'>
[59 36 80 ... 42 51 72]
0.578125
```

In [32]:

```
import matplotlib.pyplot as plt

dados_x = rng.integers(20, size = 30)
dados_y = rng.integers(12, size = 30)

plt.scatter(x = dados_x, y = dados_y)
plt.show()
```