





BOOTCAMP CIÊNCIA DE DADOS

# Numpy

por Thais Viana



---



**NumPy, abreviação de  
Numerical Python, é um dos  
mais importantes e  
fundamentais pacotes de  
computação numérica em  
Python.**

PYTHON FOR DATA ANALYSIS, WES MCKINNEY



---





---

**A maioria dos pacotes  
voltados para  
funcionalidades científicas  
usam NumPy's array objects  
como lingua franca para  
troca de dados.**

PYTHON FOR DATA ANALYSIS, WES MCKINNEY

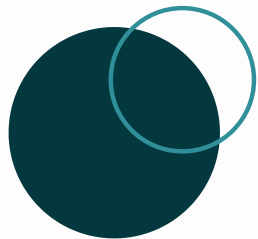


---



# NOSSA MOTIVAÇÃO

---



## PORQUE NÓS ESTUDAMOS NUMPY?

Biblioteca otimizada para funções  
matemáticas básicas para tudo que  
faremos!



# PONTOS PRINCIPAIS

ndarray

Funções Matemáticas

Entrada e saída de dados do array

Algebra Linear

C API

# ndarray

An array object represents a multidimensional, homogeneous array of fixed-size items. An associated data-type object describes the format of each element in the array (its byte-order, how many bytes it occupies in memory, whether it is an integer, a floating point number, or something else, etc.)

# COMPARANDO

```
In [7]: import numpy as np
```

```
In [8]: my_arr = np.arange(1000000)
```

```
In [9]: my_list = list(range(1000000))
```

```
In [10]: %time for _ in range(10): my_arr2 = my_arr * 2
```

```
CPU times: user 20 ms, sys: 10 ms, total: 30 ms
```

```
Wall time: 31.3 ms
```

```
In [11]: %time for _ in range(10): my_list2 = [x * 2 for x in my_list]
```

```
CPU times: user 680 ms, sys: 180 ms, total: 860 ms
```

```
Wall time: 861 ms
```

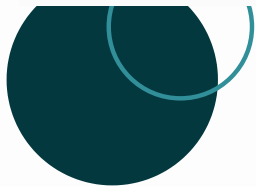
# GERAÇÃO DE NÚMEROS ALEATÓRIO

---

```
In [12]: import numpy as np

# Generate some random data
In [13]: data = np.random.randn(2, 3)

In [14]: data
Out[14]:
array([[ -0.2047,  0.4789, -0.5194],
       [-0.5557,  1.9658,  1.3934]])
```





# OPERAÇÕES MATEMÁTICAS

---

```
In [15]: data * 10  
Out[15]:  
array([[ -2.0471,   4.7894,  -5.1944],  
       [ -5.5573,  19.6578,  13.9341]])
```

```
In [16]: data + data  
Out[16]:  
array([[ -0.4094,   0.9579,  -1.0389],  
       [ -1.1115,   3.9316,   2.7868]])
```



# DATATYPES

---

```
In [33]: arr1 = np.array([1, 2, 3], dtype=np.float64)
```

```
In [34]: arr2 = np.array([1, 2, 3], dtype=np.int32)
```

```
In [35]: arr1.dtype
```

```
Out[35]: dtype('float64')
```

```
In [36]: arr2.dtype
```

```
Out[36]: dtype('int32')
```





# ATENÇÃO

## ARRAY INDEXING

```
>>> a[0]
0
>>> a[0] = 10
>>> a
array([10, 1, 2, 3])
```

## FILL

```
# set all values in an array
>>> a.fill(0)
>>> a
array([0, 0, 0, 0])

# this also works, but may
# be slower
>>> a[:] = 1
>>> a
array([1, 1, 1, 1])
```



## BEWARE OF TYPE COERCION

```
>>> a.dtype
dtype('int32')
```

```
# assigning a float into
# an int32 array truncates
# the decimal part
```

```
>>> a[0] = 10.6
>>> a
array([10, 1, 2, 3])
```

```
# fill has the same behavior
```

```
>>> a.fill(-4.8)
>>> a
array([-4, -4, -4, -4])
```

# SLICING

```
>>> a[0, 3:5]  
array([3, 4])
```

```
>>> a[4:, 4:]  
array([[44, 55],  
       [54, 55]])
```

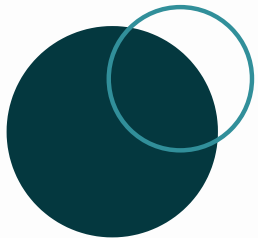
```
>>> a[:, 2]  
a([2, 12, 22, 32, 42, 52])
```

```
>>> a[2::2, ::2]  
array([[20, 22, 24],  
       [40, 42, 44]])
```

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 0  | 1  | 2  | 3  | 4  | 5  |
| 10 | 11 | 12 | 13 | 14 | 15 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 30 | 31 | 32 | 33 | 34 | 35 |
| 40 | 41 | 42 | 43 | 44 | 45 |
| 50 | 51 | 52 | 53 | 54 | 55 |

# SLICING

---



# INSTALAÇÃO

PYTHON + MINICONDA

<https://docs.conda.io/en/latest/miniconda.html>



## #1 DESAFIO

There are  $N$  bulbs, numbered from 1 to  $N$ , arranged in a row.

The first bulb is plugged into the power socket and each successive bulb is connected to the previous one (the second bulb to the first, the third bulb to the second, etc.).

\*(Wow, whoever wrote this doesn't know crap about wiring up lighting!)



## #1 DESAFIO

Initially, all the bulbs are turned off. At moment  $K$  (for  $K$  from 0 to  $N-1$ ), we turn on the  $A[K]$ -th bulb.

A bulb shines if it is on and all the previous bulbs are turned on too. Write a function solution that, given an array  $A$  of  $N$  different integers from 1 to  $N$ , returns the number of moments for which every turned on bulb shines.



# #1 DESAFIO

Examples:

Given [2, 1, 3, 5, 4], the function should return 3.

- At the 0th moment only the 2nd bulb is turned on, but it does not shine because the previous one is not on.
- At the 1st moment two bulbs are turned on (1st and 2nd) and both of them shine.
- At the 2nd moment three bulbs are turned on (1st, 2nd and 3rd) and all of them shine.
- At the 3rd moment four bulbs are turned on (1st, 2nd, 3rd and 5th), but the 5th bulb does not shine because the previous one is not turned on.
- At the 4th moment five bulbs are turned on (1st, 2nd, 3rd, 4th and 5th) and all five of them shine. There are three moments (1st, 2nd and 4th) when every turned on bulb shines.





## #1 DESAFIO

More sample cases:

$A=[2, 3, 4, 1, 5]$ , the function should return 2 (at the 3rd and 4th moment every turned on bulb shines).

$A=[1, 3, 4, 2, 5]$ , the function should return 3 (at the 0th, 3rd and 4th moment every turned on bulb shines).

ASSUME THAT:

- $N$  is an integer within the range  $[1..100]$ ;
- the elements of  $A$  are all distinct;
- each element of array  $A$  is an integer within the range  $[1..N]$ .



## # 2 DESAFIO

Contar o número de ocorrências de cada elemento em uma matriz.

Exemplo

$$A = \begin{bmatrix} 1 & 2 & 7 & 4 & 5 \\ 1 & 6 & 3 & 4 & 9 \\ 5 & 0 & 7 & 4 & 8 \end{bmatrix}$$

## # 2 DESAFIO

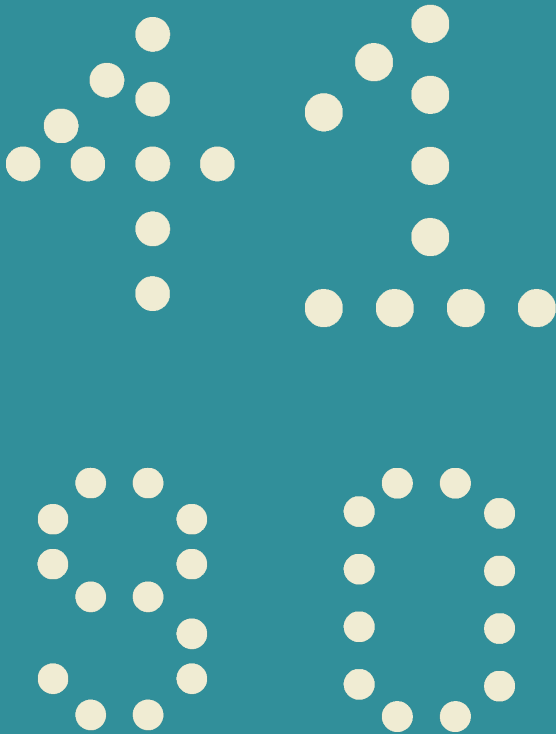
Contar o número de ocorrências de cada elemento em uma matriz.

Exemplo

```
A = [ [ 1 2 7 4 5]
      [ 1 6 3 4 9]
      [ 5 0 7 4 8]]
```

Resultado:

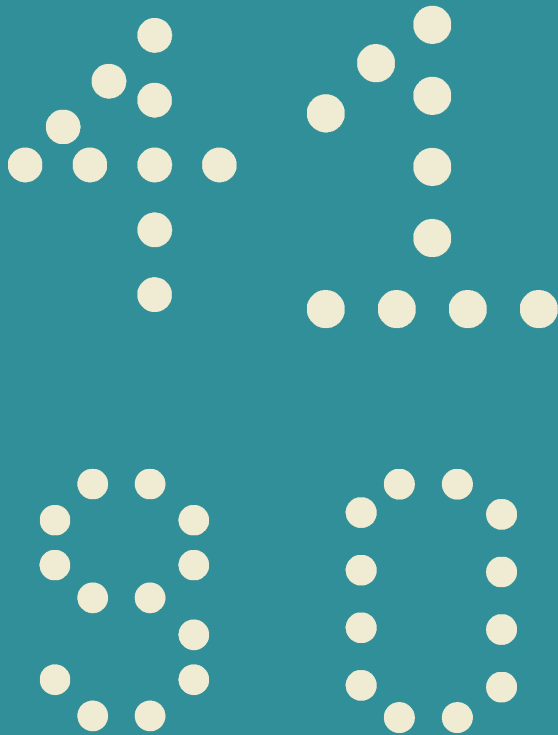
```
{0: 1, 1: 2, 2: 1, 3: 1, 4: 3, 5: 2, 6: 1, 7: 2, 8: 1, 9: 1}
```





## # 3 DESAFIO

Dada uma seqüência de  $n$  números inteiros, determinar um segmento de soma máxima.



## # 3 DESAFIO

Dada uma seqüência de  $n$  números inteiros, determinar um segmento de soma máxima.

Exemplo:

Na seqüência

5, 2, -2, -7, **3, 14, 10, -3, 9**, -6, 4, 1,  
a soma do segmento é 33.