Numpy

Numeric Python

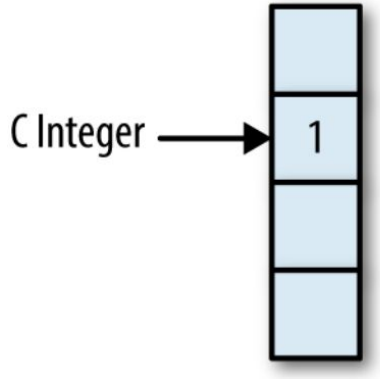
Técnicas de programación II
Ciclo 2022-II, UNALM
18/10/2022

Hablemos mal de Python

"En Python todo es un objeto" = costo en la memoría

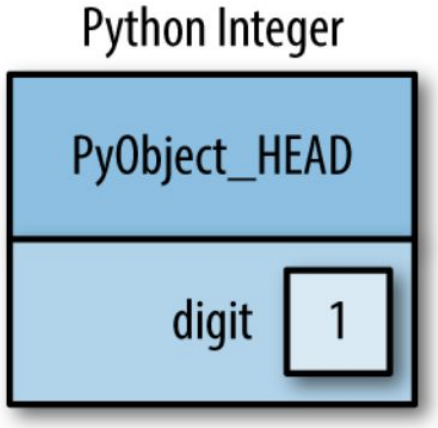
Definamos un entero...

```
int a = 1;
```



Bytes en la memoria = valor entero

```
a = 1
```



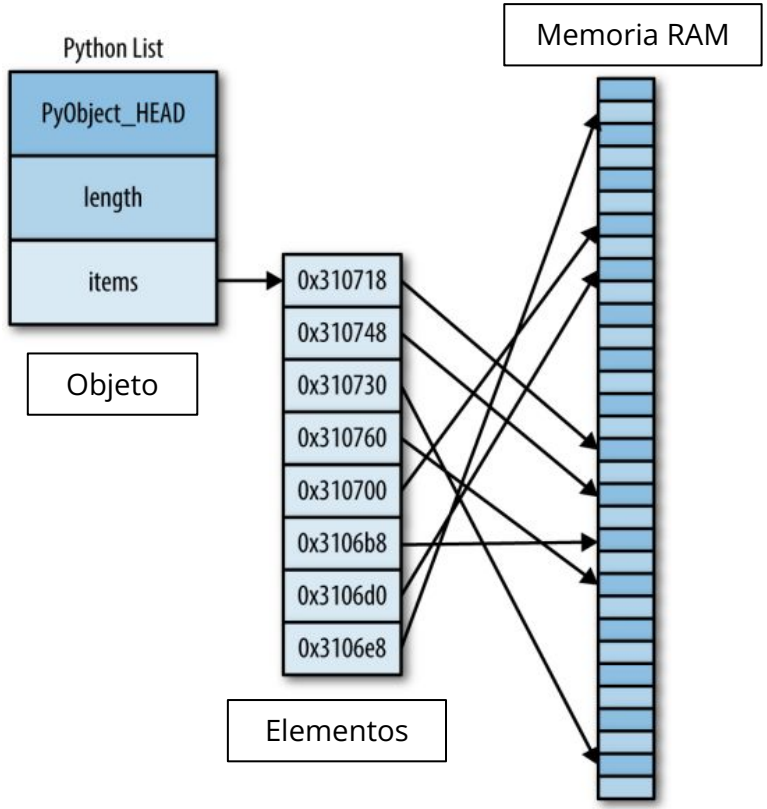
- Tipo de dato
- Posición
- Valor
- Tamaño

Un objeto con información (bytes)

Hablemos mal de Python

Tipado dinámico = mayor costo computacional

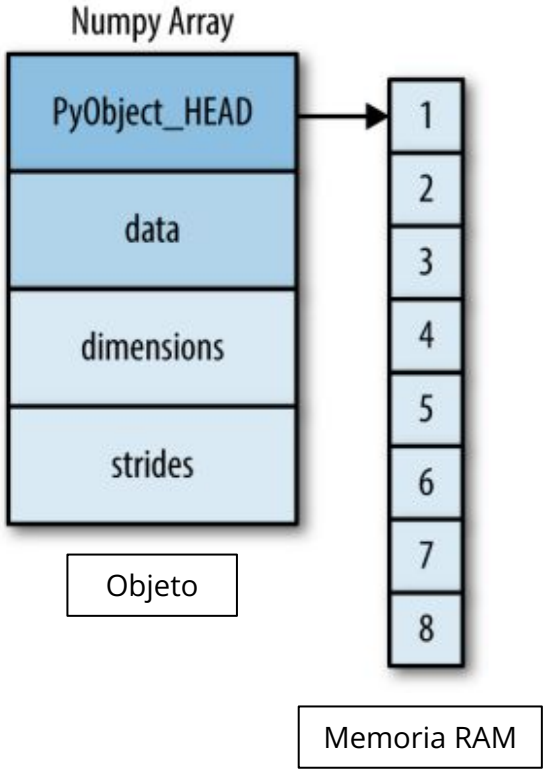
Ahora veamos una lista...




```
lista = [1, 2, 3, 4, 5, 6]
```

Lista (objeto) almacena otros objetos, con información propia cada una.

El héroe sin capa



Una de las librerías más importantes

NumPy

Almacenamiento eficiente

+

Operatividad matemática

+

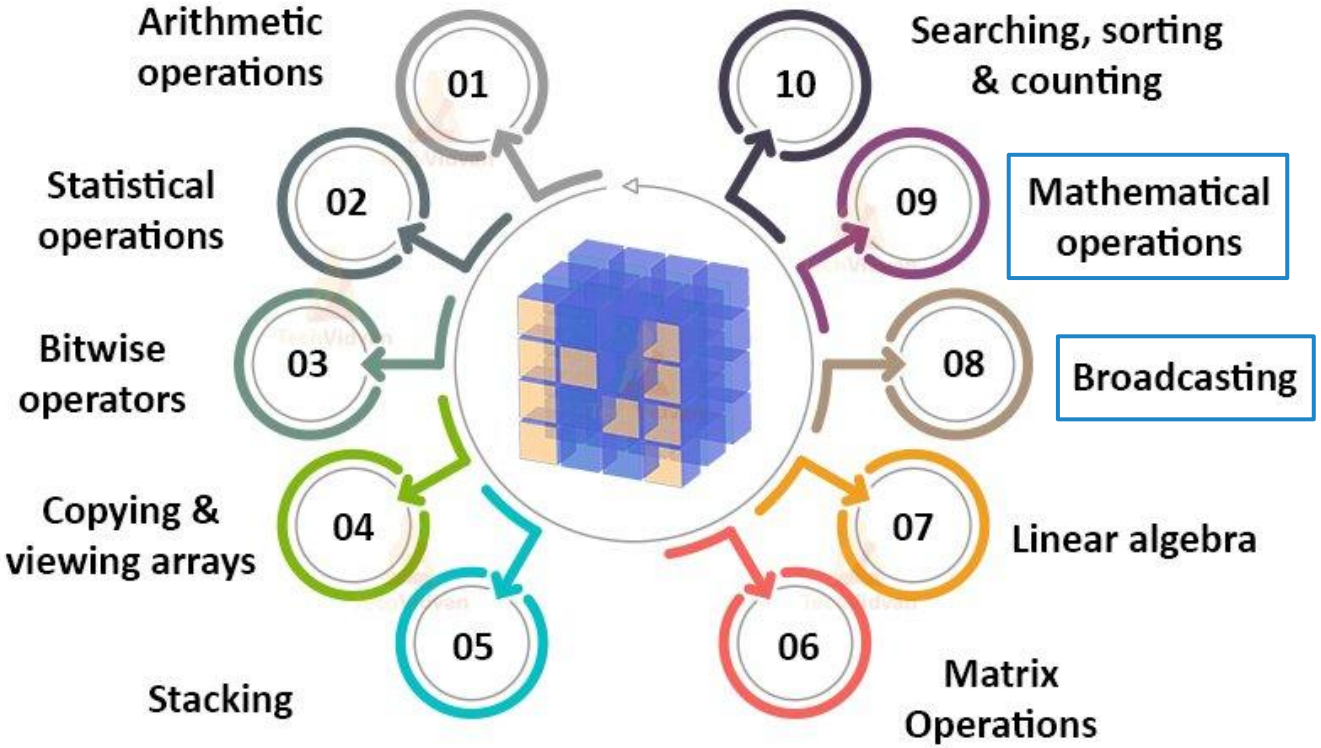
Rápido procesamiento

↓

Creación y manipulación de matrices

Numpy

Estructuras numéricas eficientes

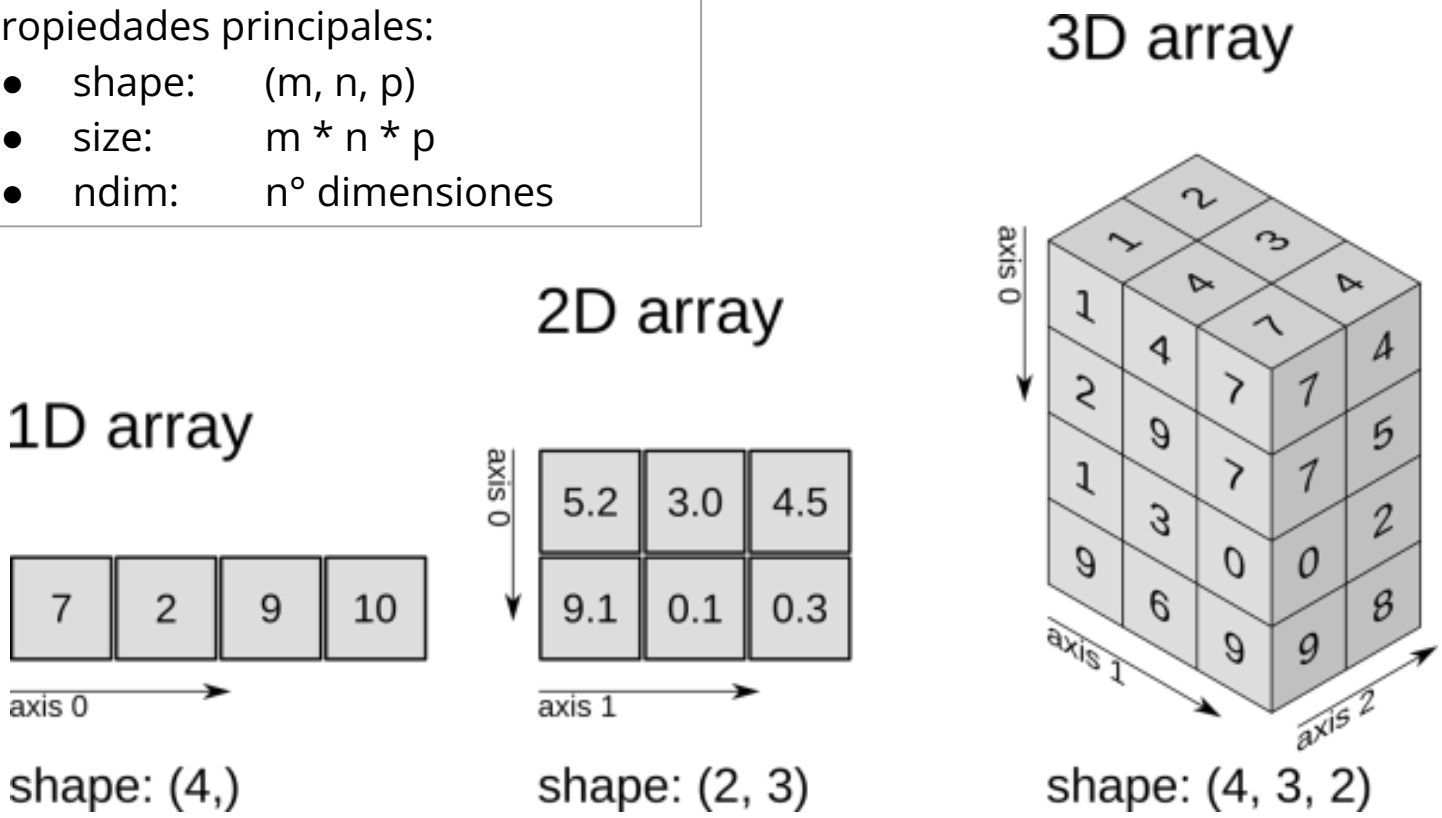


Arrays

Almacenan datos eficientemente en una variable

Propiedades principales:

- shape: (m, n, p)
- size: m * n * p
- ndim: n° dimensiones



Arrays

Almacenan datos eficientemente en una variable

1. A partir de estructuras de Python

```
import numpy as np

np.array( _____ , dtype = _____ )
```

Estructura de datos

Tipo de datos

- Lista
- Tupla
- ~~Diccionario~~

Table 2-1. Standard NumPy data types

Data type	Description
`bool_`	Boolean (True or False) stored as a byte
`int_`	Default integer type (same as C `long`; normally either `int64` or `int32`)
`intc`	Identical to C `int` (normally `int32` or `int64`)
`intp`	Integer used for indexing (same as C `ssize_t`; normally either `int32` or `int64`)
`int8`	Byte (–128 to 127)
`int16`	Integer (–32768 to 32767)
`int32`	Integer (–2147483648 to 2147483647)
`int64`	Integer (–9223372036854775808 to 9223372036854775807)
`uint8`	Unsigned integer (0 to 255)
`uint16`	Unsigned integer (0 to 65535)
`uint32`	Unsigned integer (0 to 4294967295)
`uint64`	Unsigned integer (0 to 18446744073709551615)
`float_`	Shorthand for `float64`
`float16`	Half-precision float: sign bit, 5 bits exponent, 10 bits mantissa
`float32`	Single-precision float: sign bit, 8 bits exponent, 23 bits mantissa
`float64`	Double-precision float: sign bit, 11 bits exponent, 52 bits mantissa
`complex_`	Shorthand for `complex128`
`complex64`	Complex number, represented by two 32-bit floats
`complex128`	Complex number, represented by two 64-bit floats

Fuente: VanderPlas J. (2017)

Arrays

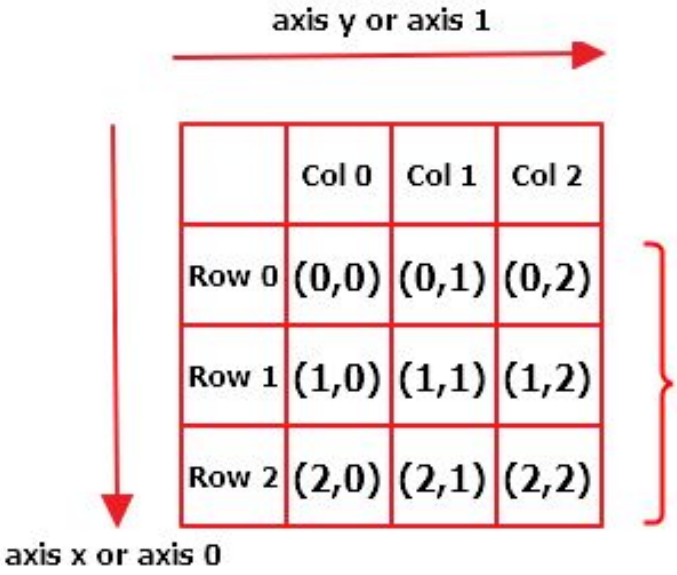
Almacenan datos eficientemente en una variable

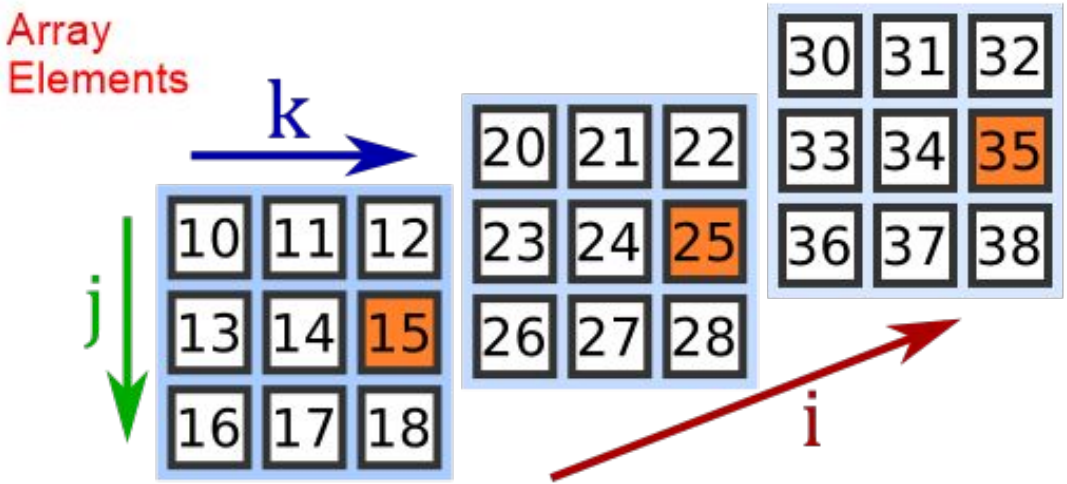
2. A partir de built-in functions

Code	Returns
`np.arange(6)`	`array([0, 1, 2, 3, 4, 5])`
`np.zeros(4)`	`array([ 0.,  0.,  0.,  0.])`
`np.ones((2, 3))`	`array([[ 1.,  1.,  1.],` `[ 1.,  1.,  1.]])`
`np.empty(4)`	`array([1.28506949e-316, 6.95226953e-310,` `8.30132260e-317, 6.95226842e-310])`
`np.linspace(1, 2, 5)`	`array([ 1.  ,  1.25,  1.5 ,  1.75,  2.])`
`np.logspace(1, -1, 3)`	`array([ 10. ,  1. ,  0.1])`

Adaptado de: VanderPlas J. (2017)

Indexing


axis y or axis 1
Col 0
Col 1
Col 2
Row 0
(0,0)
(0,1)
(0,2)
Row 1
(1,0)
(1,1)
(1,2)
Row 2
(2,0)
(2,1)
(2,2)
axis x or axis 0

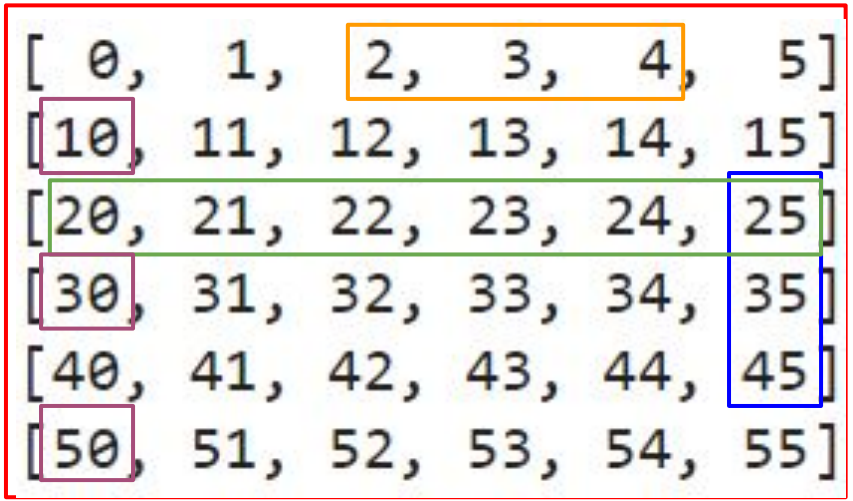

Array
Elements
k
j
i
10
11
12
13
14
15
16
17
18
20
21
22
23
24
25
26
27
28
30
31
32
33
34
35
36
37
38

Slicings / Masking

Manipulación de arrays

array[start:end:step , :]

- array[0 , 2:5]
- array[2:3 , 5]
- array[2 , 1:10]
- array[: , :]
- array[0:0 , :]
- array[1:5:2 , 0]



Slicings / Masking

Manipulación de arrays

array[boolean , :]

```
[  1,   2,   3,   4,   5,   6,   7,   8,   9]
[ 10,  20,  30,  40,  50,  60,  70,  80,  90]
[100, 200, 300, 400, 500, 600, 700, 800, 900]
```

+

```
[False, False, False,  True, False, False, False,  True, False]
[False,  True, False,  True, False,  True, False,  True, False]
[ True,  True,  True,  True,  True,  True,  True,  True,  True]
```

```
array([  4,   8,  20,  40,  60,  80, 100, 200, 300, 400, 500, 600, 700,
       800, 900])
```

Universal functions

Operaciones vectorizadas

Afectan a todos (universo) de elementos del array

Table 2-2. Arithmetic operators implemented in NumPy

Operator	Equivalent ufunc	Description
+	np.add	Addition (e.g., 1 + 1 = 2)
-	np.subtract	Subtraction (e.g., 3 - 2 = 1)
-	np.negative	Unary negation (e.g., -2)
*	np.multiply	Multiplication (e.g., 2 * 3 = 6)
/	np.divide	Division (e.g., 3 / 2 = 1.5)
//	np.floor_divide	Floor division (e.g., 3 // 2 = 1)
**	np.power	Exponentiation (e.g., 2 ** 3 = 8)
%	np.mod	Modulus/remainder (e.g., 9 % 4 = 1)

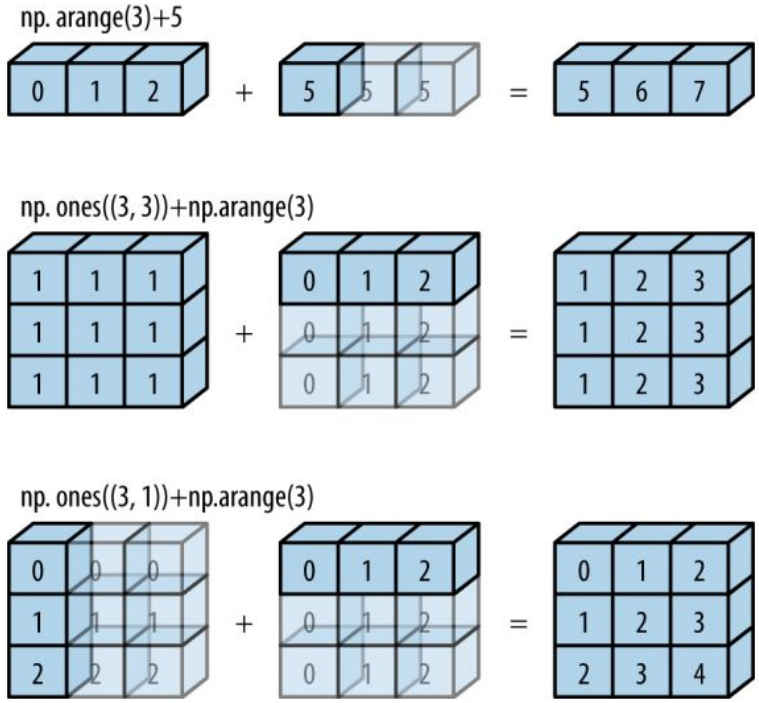
Operator	Equivalent ufunc
==	np.equal
!=	np.not_equal
<	np.less
<=	np.less_equal
>	np.greater
>=	np.greater_equal

Adaptado de: VanderPlas J. (2017)

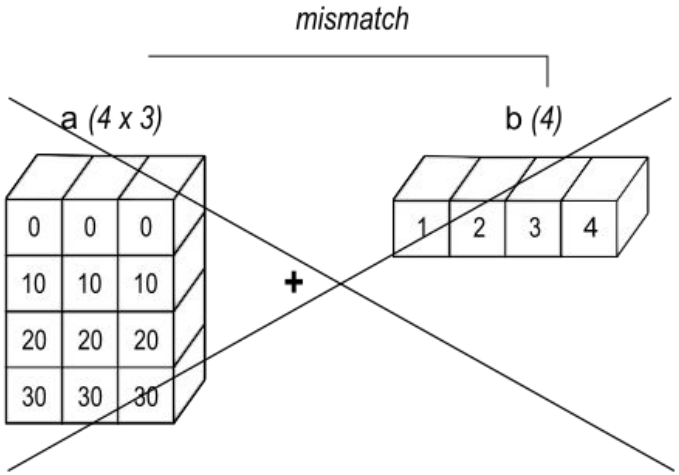
Broadcasting

Operaciones entre N-arrays

"Autocompleta" el N-array



Fuente: VanderPlas J. (2017)



Aggregations functions

Resumen los arrays

Table 2-3. Aggregation functions available in NumPy

Function Name	NaN-safe Version	Description
`np.sum`	`np.nansum`	Compute sum of elements
`np.prod`	`np.nanprod`	Compute product of elements
`np.mean`	`np.nanmean`	Compute median of elements
`np.std`	`np.nanstd`	Compute standard deviation
`np.var`	`np.nanvar`	Compute variance
`np.min`	`np.nanmin`	Find minimum value
`np.max`	`np.nanmax`	Find maximum value
`np.argmin`	`np.nanargmin`	Find index of minimum value
`np.argmax`	`np.nanargmax`	Find index of maximum value
`np.median`	`np.nanmedian`	Compute median of elements
`np.percentile`	`np.nanpercentile`	Compute rank-based statistics of elements
`np.any`	N/A	Evaluate whether any elements are true
`np.all`	N/A	Evaluate whether all elements are true

Fuente: VanderPlas J. (2017)

Aplicación

Coeficiente de correlación de Pearson

$$\rho_{xy} = \frac{Cov_{xy}}{\sigma_x \sigma_y} = \frac{\sum_{i=1}^{n} (Xi - \overline{X})(Yi - \overline{Y})}{\sqrt{\left[\sum_{i=1}^{n} (Xi - \overline{X})^2\right]\left[\sum_{i=1}^{n} (Yi - \overline{Y})^2\right]}}$$

Referencias recomendadas

- Tutoriales muy buenos y concisos.
- Ejemplos y ejercicios desde básicos a avanzados
- Comunidad, libros, etc.

- El “yahoo respuestas” para programadores
- Donde pasarán la mayor cantidad de tiempo cuando necesiten solucionar un problema.

- Biblioteca de repositorios para encontrar la “inspiración” (dile NO al plagio :v) que te falta.
- Recomendación: empieza a crear tu **portafolio**.