

# Documentación y herramientas de documentación automática

Cecilia Jarne

[cecilia.jarne@unq.edu.ar](mailto:cecilia.jarne@unq.edu.ar)



# Documentación

**Documentar significa comunicar**

**Es necesaria en todos los niveles:**

- 1) Anotaciones en el código: de formato, comentarios, de estructuración de funciones de clases.
- 2) Manual de usuario o de referencia
- 3) Introducción para nuevos usuarios o desarrolladores.

# Documentación

**El código que escribimos puede ser reutilizado y leído varias veces.**

- Es importante la claridad, mas que la astucia al escribir.
- Elegir un estilo y respetarlo: estructura de bloques, indentación, ser cuidadoso con el nombre de las variables, la longitud de linea
- Respetar las convenciones de lenguaje la comunidad en la que uno esta trabajando (en Python, ver PEP 8)
- Ser consistente (no cambiar a la mitad!!)

# Documentación

Explicar la intención más que el trabajo hecho

```
def tripleTuple(x):  
    # assign x to y  
    y = x  
    # assign x to z  
    z = x  
    # double y  
    y *= 2  
    # triple z  
    z *= 3  
    # create tuple  
    t = (x, y, z)  
    # return the tuple  
    return t
```

```
def tripleTuple(x):  
    y = z = x  
    # apply foo scaling, see [34] eq (2.3)  
    y *= 2  
    z *= 3  
    return (x, y, z)
```

También cualquier desviación de respecto de los estándares

# Documentación

```
/******  
*.comentarios..  
*****/
```

o bien

```
////////////////////////////////////  
//////comentarios..  
////////////////////////////////////
```

# Documentación

Existen algunas herramientas que automáticamente extraen la documentación escrita en el código

**Pydoc**

**Doxygen**

**Sphinx**

## Doxygen

- Soporte para C++, C, Obj-C, C#, PHP, Java, Python, IDL, Fortran, VHDL, Tcl
- Puede ser estructurada desde files sin documentar.
- Visualización gráfica de las dependencias
- Permite escribir también paginas generales

# Documentación

<http://www.sphinx-doc.org/es/stable/tutorial.html>

## Install Sphinx

Install Sphinx, either from a distribution package or from [PyPI](#) with

```
$ pip install Sphinx
```

## Setting up the documentation sources

The root directory of a Sphinx collection of reStructuredText document sources is called the [source directory](#). This directory also contains the Sphinx configuration file `conf.py`, where you can configure all aspects of how Sphinx reads your sources and builds your documentation. [\[1\]](#)

Sphinx comes with a script called **sphinx-quickstart** that sets up a source directory and creates a default `conf.py` with the most useful configuration values from a few questions it asks you. Just run

```
$ sphinx-quickstart
```

and answer its questions. (Be sure to say yes to the “autodoc” extension.)



# Documentación

<http://www.sphinx-doc.org/es/stable/tutorial.html>

## Install Sphinx

Install Sphinx, either from a distribution package or from [PyPI](#) with

```
$ pip install Sphinx
```

## Setting up the documentation sources

The root directory of a Sphinx collection of reStructuredText document sources is called the [source directory](#). This directory also contains the Sphinx configuration file `conf.py`, where you can configure all aspects of how Sphinx reads your sources and builds your documentation. [1]

Sphinx comes with a script called **sphinx-quickstart** that sets up a source directory and creates a default `conf.py` with the most useful configuration values from a few questions it asks you. Just run

```
$ sphinx-quickstart
```

and answer its questions. (Be sure to say yes to the “autodoc” extension.)

There is also an automatic “API documentation” generator called **sphinx-apidoc**; see [Invocation of sphinx-apidoc](#) for details.

## Defining document structure

Let’s assume you’ve run **sphinx-quickstart**. It created a source directory with `conf.py` and a master document, `index.rst` (if you accepted the defaults). The main function of the [master document](#) is to serve as a welcome page, and to contain the root of the “table of contents tree” (or *toctree*). This is one of the main things that Sphinx adds to reStructuredText, a way to connect multiple files to a single hierarchy of documents.

The *toctree* directive initially is empty, and looks like this:

```
.. toctree::  
   :maxdepth: 2
```

You add documents listing them in the *content* of the directive:

```
.. toctree::  
   :maxdepth: 2  
  
   intro  
   tutorial  
   ...
```

This is exactly how the *toctree* for this documentation looks. The documents to include are given as [document names](#), which in short

### reStructuredText directives

*toctree* is a reStructuredText *directive*, a very versatile piece of markup. Directives can have arguments, options and content.

*Arguments* are given directly after the double colon following the directive’s name. Each directive decides whether it can have arguments, and how many.

*Options* are given after the arguments, in form of a “field list”. The `maxdepth` is such an option for the *toctree* directive.

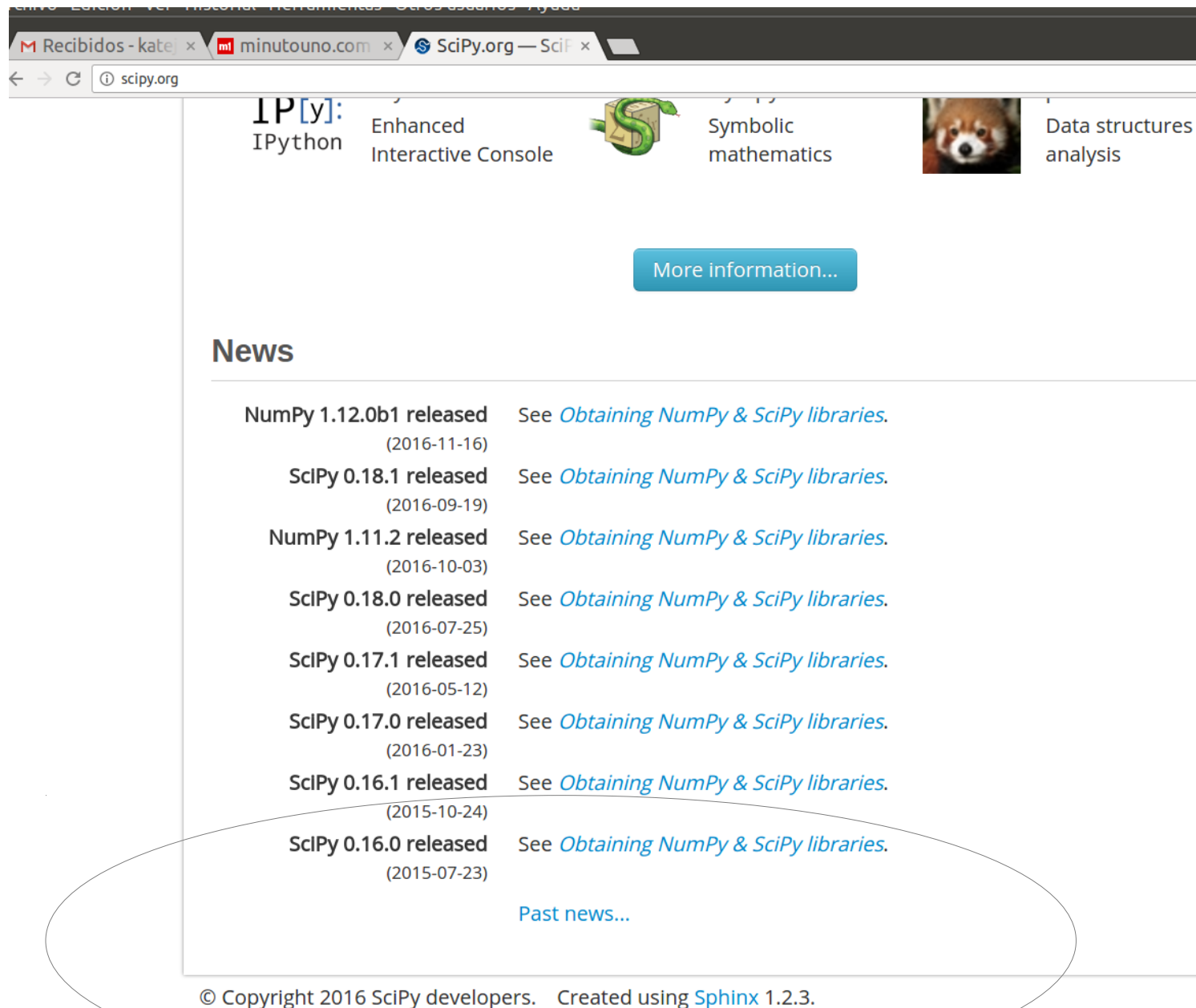
*Content* follows the options or arguments after a blank line. Each directive decides whether it

# Documentación



- Soporte muy bueno para los principiantes.
- Especial para generar documentación en python.

# Documentación



The screenshot shows the SciPy.org website. At the top, there are links for IPython (Enhanced Interactive Console), SymPy (Symbolic mathematics), and Data structures analysis. Below these is a 'More information...' button. The 'News' section lists recent releases of NumPy and SciPy, each with a link to 'Obtaining NumPy & SciPy libraries'. The releases listed are:

- NumPy 1.12.0b1 released (2016-11-16)
- SciPy 0.18.1 released (2016-09-19)
- NumPy 1.11.2 released (2016-10-03)
- SciPy 0.18.0 released (2016-07-25)
- SciPy 0.17.1 released (2016-05-12)
- SciPy 0.17.0 released (2016-01-23)
- SciPy 0.16.1 released (2015-10-24)
- SciPy 0.16.0 released (2015-07-23)

Below the list is a link for 'Past news...'. At the bottom, the copyright notice reads: '© Copyright 2016 SciPy developers. Created using Sphinx 1.2.3.'

# Documentación

## numpy.correlate

**numpy.correlate**(*a*, *v*, *mode*='valid')

[\[source\]](#)

Cross-correlation of two 1-dimensional sequences.

This function computes the correlation as generally defined in signal processing texts:

```
c_{av}[k] = sum_n a[n+k] * conj(v[n])
```

with *a* and *v* sequences being zero-padded where necessary and *conj* being the conjugate.

**Parameters:** *a, v* : *array\_like*

Input sequences.

**mode** : {'valid', 'same', 'full'}, optional

Refer to the [convolve](#) docstring. Note that the default is 'valid', unlike [convolve](#), which uses 'full'.

**old\_behavior** : *bool*

*old\_behavior* was removed in NumPy 1.10. If you need the old behavior, use [multiarray.correlate](#).

**Returns:** *out* : *ndarray*

Discrete cross-correlation of *a* and *v*.

### See also:

[convolve](#) Discrete, linear convolution of two one-dimensional sequences.

[multiarray.correlate](#) Old, no conjugate, version of `correlate`.

### Notes

The definition of correlation above is not unique and sometimes correlation may be defined differently. Another common definition is:

```
c'_{av}[k] = sum_n a[n] conj(v[n+k])
```

which is related to  $c_{av}[k] = \sum_n a[n+k] \text{conj}(v[n])$  by  $c'_{av}[k] = c_{av}[-k]$ .

Previous topic

[numpy.corrcoef](#)

Next topic

[numpy.cov](#)

# Documentación

## Notes

The definition of correlation above is not unique and sometimes correlation may be defined differently. Another common definition is:

$$c'_{\{av\}}[k] = \sum_n a[n] \text{ conj}(v[n+k])$$

which is related to  $c_{\{av\}}[k]$  by  $c'_{\{av\}}[k] = c_{\{av\}}[-k]$ .

## Examples

```
>>> np.correlate([1, 2, 3], [0, 1, 0.5])
array([ 3.5])
>>> np.correlate([1, 2, 3], [0, 1, 0.5], "same")
array([ 2. ,  3.5,  3. ])
>>> np.correlate([1, 2, 3], [0, 1, 0.5], "full")
array([ 0.5,  2. ,  3.5,  3. ,  0. ])
```

>>>

Using complex sequences:

```
>>> np.correlate([1+1j, 2, 3-1j], [0, 1, 0.5j], 'full')
array([ 0.5-0.5j,  1.0+0.j ,  1.5-1.5j,  3.0-1.j ,  0.0+0.j ])
```

>>>

Note that you get the time reversed, complex conjugated result when the two input sequences change places, i.e.,  $c_{\{va\}}[k] = c^{\{*\}}_{\{av\}}[-k]$ :

```
>>> np.correlate([0, 1, 0.5j], [1+1j, 2, 3-1j], 'full')
array([ 0.0+0.j ,  3.0+1.j ,  1.5+1.5j,  1.0+0.j ,  0.5+0.5j])
```

>>>

# Documentación

**Evitar el karma de la programación:**

**No le hagas a otros lo que no te gusta que te hagan!!!! Escribí todo lo necesario para que otro pueda usar tu código y programar usando tu código!!!**

# Documentación

Existen Siempre es buena idea leer el manual de usuario un poco al menos(ahorra tiempo!!)

También sirve hacer los tutoriales (se aprende mas rápido)