

Documentación en Sphinx

Documentar código

- **Se explican las funciones, clases, etc. en docstrings (comentarios)**
- **Se corre un programa que genera una web, pdf, etc.**
- **Algunos programas permiten además escribir texto aparte como una introducción, tutorial, etc.**

Documentar código

```
21
20 def _replace_extension(path, extension):
19     """
18     Reemplazar extensión del path dado.
17
16     No cambia la extensión del archivo, solamente modifica el string que
15     representa al path. Elimina todo lo que haya después del último ``.`` y
14     coloca la nueva extensión.
13
12     Args:
11         path (str): String que representa a un path.
10         extension (str): Extensión a colocar, no se debe proporcionar el ``.``,
9         ejemplo: ``mp3``.
8
7     Returns:
6         str: Nuevo path
5     """
4
3     name, _ = os.path.splitext(path)
2     return name + '.' + extension
1
```



batch_ffmpeg

0.1

Search docs

batch_ffmpeg

batch_ffmpeg.file

batch_ffmpeg.convert

batch_ffmpeg.command

batch_ffmpeg.ui.interactive

batch_ffmpeg.ui.shell

solamente se pueden obtener todos los paths de archivo desde una carpeta

batch_ffmpeg.file._replace_extension(*path*, *extension*) [\[fuente\]](#)

Reemplazar extensión del path dado.

No cambia la extensión del archivo, solamente modifica el string que representa al path. Elimina todo lo que haya después del último `.` y coloca la nueva extensión.

- Parámetros:
- *path* (*str*) – String que representa a un path.
 - *extension* (*str*) – Extensión a colocar, no se debe proporcionar el `.`, ej: `mp3`.

batch_ffmpeg.file.folder_exists(*path*) [\[fuente\]](#)

Indica si la carpeta dada existe.

Parámetros: *path* (*str*) – Path a la carpeta.

Devuelve: `True` si la carpeta existe y `False` de lo contrario.

batch_ffmpeg.file.get_paths_from_folder(*in_folder*, *in_extension*, *out_folder*, *out_extension*) [\[fuente\]](#)

Obtener paths de entrada y salida para todos los archivos presentes en la carpeta de entrada con la extensión de entrada

Documentar texto

```
1
2
3 Para trabajo con Dataframes
4 ~~~~~
5
6 .. code-block:: python
7
8     ss = pyspark.sql.Session.builder.config(conf=conf).getOrCreate()
9     sc = ss.sparkContext
10
11
12 SparkContext y SparkSession
13 ~~~~~
14
15 El *SparkContext* es la base de toda la aplicación, solamente se puede tener
16 uno, y una vez que se detiene no se puede reiniciar.
17
18 Es como que en las ultimas versiones está cambiando la API de Spark. Antes
19 *SparkContext* era la base de todo y trabajaba con RDDs, después se
20 necesitaban *StreamingContext*, *sqlContext*, etc. Ahora se quiere
21 usar el *Dataset* como la abstracción principal en vez de RDDs que van a quedar
22 como una API de mas bajo nivel.
23
24 Entonces en vez de usar *SparkContext* como base habria que usar a
25 *SparkSession*, pero no encuentro bien como sería, parece que por ahora
26 *SparkSession* está hecho para trabajar con Dataset. Creo que *SparkSession*
27 tiene su propio *SparkContext* y *StreamingContext*. (TODO)
28
29 Me parece que por ahora hay que usar:
30
31 * *SparkContext* si se trabaja con sólo RDDs.
32 * *SparkSession* si se trabaja con Dataframes.
```


Básico — documentación de martin_gidat - - Mozilla Firefox

batch_ffmpeg.file — document... x Básico — documentación de m... x Preferences x +

file:///media/mbernardi/da DuckDuckGo

CONTENIDOS:

Multiprocessing

▢ Apache Spark

Pruebas

▢ Programacion

▢ Básico

Dataset y SQL

Streaming

Machine Learning

Matrices

Notas

Configuracion

Problemas

Funcionamiento del cluster

Twisted

Herramientas para documentación

Configuraciones

Archivos extra

TODOS

Para trabajo con Streams

```
sc = SparkContext(conf=conf)
ssc = StreamingContext(sc, intervalo)
```

Para trabajo con Dataframes

```
ss = pyspark.sql.Session.builder.config(conf=conf).getOrCreate()
sc = ss.sparkContext
```

SparkContext y SparkSession

El *SparkContext* es la base de toda la aplicación, solamente se puede tener uno, y una vez que se detiene no se puede reiniciar.

Es como que en las ultimas versiones está cambiando la API de Spark. Antes *SparkContext* era la base de todo y trabajaba con RDDs, después se necesitaban *StreamingContext*, *sqlContext*, etc. Ahora se quiere usar el *Dataset* como la abstracción principal en vez de RDDs que van a quedar como una API de mas bajo nivel.

Entonces en vez de usar *SparkContext* como base habria que usar a *SparkSession*, pero no encuentro bien como sería, parece que por ahora *SparkSession* está hecho para trabajar con Dataset. Creo que

Programas

- **Sphinx**
- Doxygen
- Natural Docs
- **pydoctor**
- pdoc
- **pydoc**

Programas

- **Sphinx**
 - **El más completo y usado en Python**
 - **Permite documentar texto**
 - **Necesita bastante configuración**
- **pydoctor**
 - **Un poco más fácil de usar**
- **pydoc**
 - **Viene con Python**

Sphinx

🏠 batch_ffmpeg

0.1

Search docs

batch_ffmpeg

batch_ffmpeg.file

batch_ffmpeg.convert

batch_ffmpeg.command

batch_ffmpeg.ui.interactive

batch_ffmpeg.ui.shell

desde una carpeta

batch_ffmpeg.file._replace_extension(*path*, *extension*) [\[fuente\]](#)

Reemplazar extensión del path dado.

No cambia la extensión del archivo, solamente modifica el string que representa al path. Elimina todo lo que haya después del último `.` y coloca la nueva extensión.

- Parámetros:
- **path** (*str*) – String que representa a un path.
 - **extension** (*str*) – Extensión a colocar, no se debe proporcionar el `.`, ej: `mp3`.

batch_ffmpeg.file.folder_exists(*path*) [\[fuente\]](#)

Indica si la carpeta dada existe.

Parámetros: **path** (*str*) – Path a la carpeta.

Devuelve: `True` si la carpeta existe y `False` de lo contrario.

batch_ffmpeg.file.get_paths_from_folder(*in_folder*, *in_extension*, *out_folder*, *out_extension*) [\[fuente\]](#)

Obtener paths de entrada y salida para todos los archivos

pydoctor

```
def _replace_extension(path, extension):
```

Reemplazar extensión del path dado.

No cambia la extensión del archivo, solamente modifica el string que representa al path. Elimina todo lo que haya después del último ``.`` y coloca la nueva extensión.

Args:

path (str): String que representa a un path.

extension (str): Extensión a colocar, no se debe proporcionar el ``.``, ej: ``mp3``.

Returns:

str: Nuevo path

```
def get_paths_from_folder(in_folder, in_extension, out_folder, out_extension):
```

Obtener paths de entrada y salida para todos los archivos presentes en la carpeta de entrada con la extensión de entrada.

Devuelve una lista con los paths de entrada y otra con los paths de salida.

Args:

in_folder (str): Path a la carpeta de entrada, debe existir.

in_extension (str): Extensión de los archivos de entrada, no incluir el ``.``.

[Hide Private API](#)

pydoc

file

[Index](#)/media/mbernardi/datos/Documentos/repos/python/batch_ffmpeg/file.py

Herramientas para el manejo de archivos.

Todo:

- * Forma de obtener paths individualmente, por ahora solamente se pueden obtener todos los paths de archivo desde una carpeta

Modules

[errno](#)[os](#)

Functions

folder_exists(path)

Indica si la carpeta dada existe.

Args:

path (str): Path a la carpeta.

Returns:

bool: ``True`` si la carpeta existe y ``False`` de lo contrario.

get_paths_from_folder(in_folder, in_extension, out_folder, out_extension)

Obtener paths de entrada y salida para todos los archivos presentes en la carpeta de entrada con la extensión de entrada.

Devuelve una lista con los paths de entrada y otra con los paths de salida.

Args:

in_folder (str): Path a la carpeta de entrada, debe existir.

in_extension (str): Extensión de los archivos de entrada, no incluir el

``.``. ej: ``mp3``.

out_folder (str): Path a la carpeta de salida, debe existir.

out_extension (str): Extensión de los archivos de salida, no incluir el

Sintaxis para docstrings

- **No hay un estándar.**
- **Google Style**
 - No tiene símbolos.
 - Tiene palabras clave fáciles de recordar, como “Args:”, “Returns:”, “Examples:”.
 - Es muy legible en texto plano.
- **NumPy style:**
 - Casi lo mismo. Usemos Google Style.


```

29 def reemplazar_ext(path, extension):
28     """
27     Reemplazar extensión del path dado.
26
25     epytext para pydoctor.
24
23     @type path: str
22     @param path: String que representa a un path.
21
20     @type extension: str
19     @param extension: Extensión a colocar, no se debe proporcionar ., ej: mp3.
18
17     @rtype: str
16     @return: Nuevo path
15     """
14
13     nombre, _ = os.path.splitext(path)
12     return nombre + '.' + extension
11
10 def reemplazar_ext(path, extension):
9     """
8     Reemplazar extensión del path dado.
7
6     Google style para Sphinx.
5
4     Args:
3     path (str): String que representa a un path.
2     extension (str): Extensión a colocar, no se debe proporcionar el ., ej: ``mp3``.
1
2     Returns:
1     str: Nuevo path.
0     """
0
0     nombre, _ = os.path.splitext(path)
0     return nombre + '.' + extension
0

```

Estructura de archivos

Común:

```
miproyecto
├── docs
│   ├── build
│   │   ├── ...
│   │   └── html
│   │       ├── ...
│   │       └── index.html
│   └── Makefile
├── source
│   ├── ...
│   ├── conf.py
│   └── index.rst
├── README
├── __init__.py
├── main.py
└── ...
```

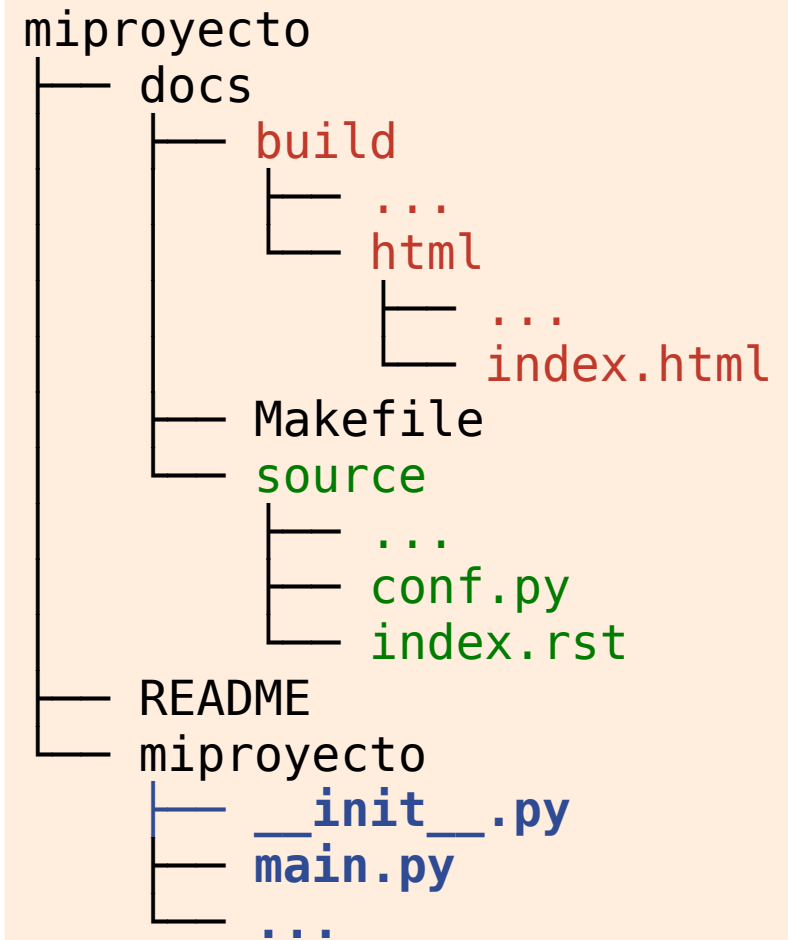
Recomendado:

```
miproyecto
├── docs
│   ├── build
│   │   ├── ...
│   │   └── html
│   │       ├── ...
│   │       └── index.html
│   └── Makefile
├── source
│   ├── ...
│   ├── conf.py
│   └── index.rst
├── README
├── miproyecto
│   ├── __init__.py
│   └── main.py
└── ...
```

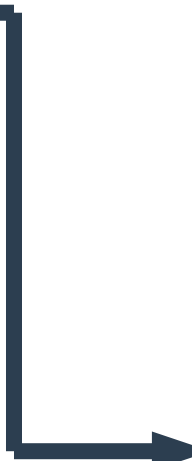

Funcionamiento de Sphinx

- Se escribe en “source”.
- Se genera en “build”.
- Cada página de la documentación se escribe en un “.rst”.

Estructura de archivos:



Funcionamiento de Sphinx

- Las páginas son texto, títulos, listas, imágenes, links, etc.
 - La extensión “autodoc” reemplaza directivas por texto obtenido desde *docstrings*.
- 

Ejemplo de un “.rst”:

```
Mi Programa
=====

Esta es la documentación
para mi programa.

Este programa sirve para:

* Hacer algo
* Hacer otra cosa
* ...

Documentación:
-----

.. automodule:: prog.py
   :members:
```

Funcionamiento de Sphinx

- El “index.rst” debe llevar un índice con los demás “.rst”
- Los demás “.rst” también pueden llevar índices.

index.rst

Mi Programa

=====

```
.. toctree::  
    :maxdepth: 2  
    :caption: Contents:
```

introduccion
otros/autores
otros/sobre

Este programa hace...

otros/autores.rst

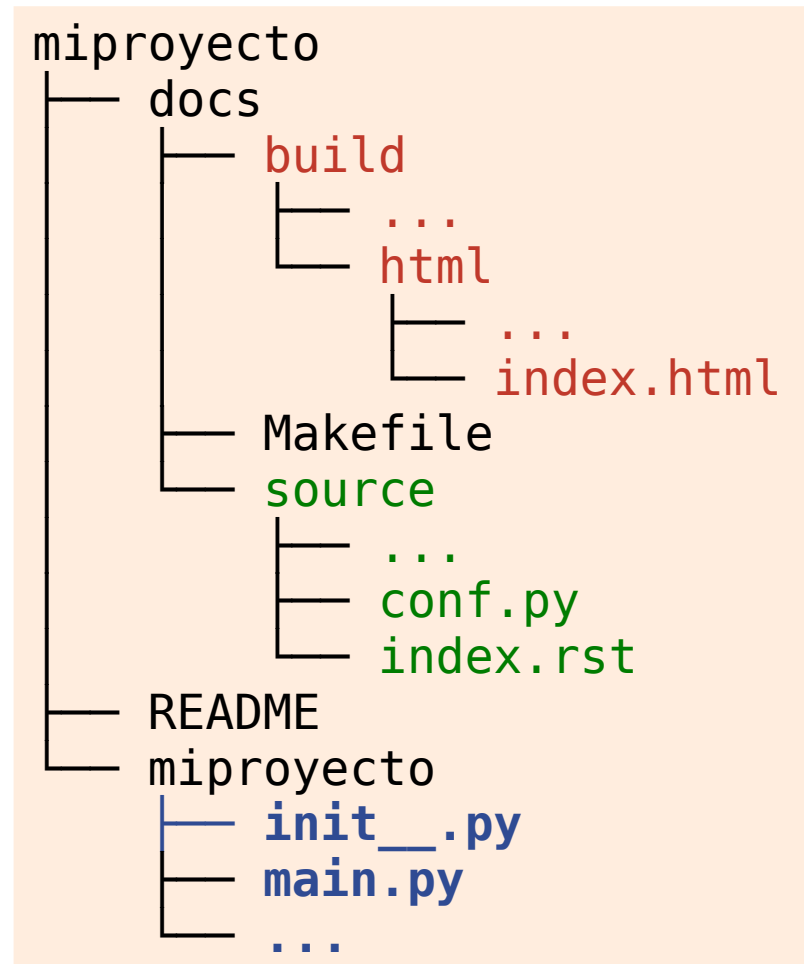
Autores

=====

- * Juan
- * Pablo
- * Pedro

Funcionamiento de Sphinx

- Al final uno se mueve a la carpeta con el “Makefile” y genera todo en “build” con:
`make html`
- En “index.html” está la página principal.



Formas de trabajar

- De alguna forma, en los “.rst” hay que poner una directiva para cada cosa a documentar desde docstrings.

`.. automodule::` `.. autoclass::` `.. autofunction::`

- Prefiero trabajar con módulos.
- Hay varias formas de organizarse:
 - 1) Poner todos los módulos en el “index.rst”.
 - 2) Hacer un “.rst” para cada módulo.
 - 3) Generar un “.rst” para cada módulo a partir de una plantilla.

Formas de trabajar

1

- **Si son pocos módulos ponerlos todos en “index.rst”.**
- **Va a terminar todo en una única página.**

Mi Programa

=====

```
.. toctree::  
    :maxdepth: 2  
    :caption: Contents:
```

Este programa hace...

Documentación:

```
.. automodule:: main.py  
    :members:
```

```
.. automodule:: color.py  
    :members:
```

```
.. automodule:: socket.py  
    :members:
```


Formas de trabajar

2

- **Escribir a mano un “.rst” para cada archivo.**

index.rst

```
Mi Programa
=====

.. toctree::
   :maxdepth: 2
   :caption: Contents:

   main
   color
   socket

Este programa hace...
```

color.rst

```
color
=====

.. automodule:: color.py
   :members:
```

main.rst

```
main
=====

.. automodule:: main.py
   :members:
```

Formas de trabajar

3

- Usar “autosummary” para generar los “.rst” desde una plantilla.
- Uno debe darle una lista de todos los módulos/clases/funciones a documentar.
- A partir de esa lista y de unas “templates”, genera una página para cada cosa.

Ejemplo de “autosummary”

index.rst

Mi Programa

=====

```
.. toctree::
    :maxdepth: 2
    :caption: Contents:
```

Este programa hace...

```
.. autosummary::
    :toctree: _autosummary
```

main
color
socket

_templates/autosummary/module.rst

```
{{ fullname }}
{{ underline }}
```

```
.. automodule:: {{fullname}}
    :members:
```

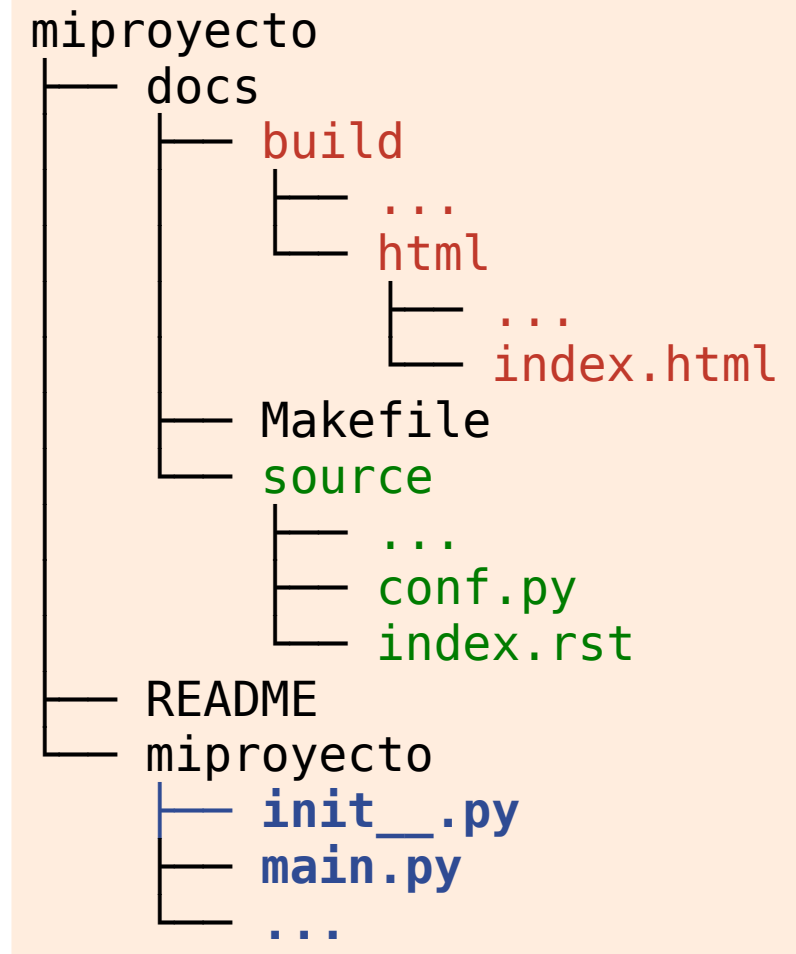
_autosummary/color.rst (generado automáticamente)

```
color
=====
```

```
.. automodule:: color.py
    :members:
```

Configuración de Sphinx

- Se generan las carpetas, configuraciones por defecto, etc. con `sphinx-quickstart`
- Se configura en “`conf.py`”



Receta simple

- **sphinx-quickstart**

- Root path for documentation: docs
- Separate source and build directories: y
- (Dejar demás opciones por defecto)
- Extensiones: intersphinx, todo, mathjax, githubpages

- **conf.py**

- Importar “os”, “sys” y agregar “../..” al path
- Agregar extensión “sphinx.ext.napoleon”
- En cualquier lado poner:
 - ```
html_sidebars = { '**': ['globaltoc.html', 'relations.html', 'sourcelink.html', 'searchbox.html'], }
```
- Cambiar tema a “nature”.

- **Modificar index.rst**

- **make html**

```
» sphinx-quickstart
```

```
Welcome to the Sphinx 1.5.5 quickstart utility.
```

Please enter values for the following settings (just press Enter to accept a default value, if one is given in brackets).

Enter the root path for documentation.

```
> Root path for the documentation [.]: docs
```

You have two options for placing the build directory for Sphinx output. Either, you use a directory "\_build" within the root path, or you separate "source" and "build" directories within the root path.

```
> Separate source and build directories (y/n) [n]: y
```

Inside the root directory, two more directories will be created; "\_templates" for custom HTML templates and "\_static" for custom stylesheets and other static files. You can enter another prefix (such as ".") to replace the underscore.

```
> Name prefix for templates and static dir [_]:
```

The project name will occur in several places in the built documentation.

```
> Project name: mi proyecto
```

```
> Author name(s): Martin Bernardi
```

Sphinx has the notion of a "version" and a "release" for the software. Each version can have multiple releases. For example, for Python the version is something like 2.5 or 3.0, while the release is something like 2.5.1 or 3.0a1. If you don't need this dual structure, just set both to the same value.

```
> Project version []: 0.1
```

```
> Project release [0.1]:
```

If the documents are to be written in a language other than English, you can select a language here by its language code. Sphinx will then translate text that it generates into that language.

For a list of supported codes, see

<http://sphinx-doc.org/config.html#confval-language>.

```
> Project language [en]: es
```



The file name suffix for source files. Commonly, this is either ".txt" or ".rst". Only files with this suffix are considered documents.

> Source file suffix [.rst]:

One document is special in that it is considered the top node of the "contents tree", that is, it is the root of the hierarchical structure of the documents. Normally, this is "index", but if your "index" document is a custom template, you can also set this to another filename.

> Name of your master document (without suffix) [index]:

Sphinx can also add configuration for epub output:

> Do you want to use the epub builder (y/n) [n]:

Please indicate if you want to use one of the following Sphinx extensions:

> autodoc: automatically insert docstrings from modules (y/n) [n]: y

> doctest: automatically test code snippets in doctest blocks (y/n) [n]:

> intersphinx: link between Sphinx documentation of different projects (y/n) [n]: y

> todo: write "todo" entries that can be shown or hidden on build (y/n) [n]: y

> coverage: checks for documentation coverage (y/n) [n]:

> imgmath: include math, rendered as PNG or SVG images (y/n) [n]:

> mathjax: include math, rendered in the browser by MathJax (y/n) [n]:

> ifconfig: conditional inclusion of content based on config values (y/n) [n]:

> viewcode: include links to the source code of documented Python objects (y/n) [n]: y

> githubpages: create .nojekyll file to publish the document on GitHub pages (y/n) [n]:

A Makefile and a Windows command file can be generated for you so that you only have to run e.g. `make html` instead of invoking sphinx-build directly.

> Create Makefile? (y/n) [y]: y

> Create Windows command file? (y/n) [y]: y

Creating file docs/source/conf.py.

Creating file docs/source/index.rst.

Creating file docs/Makefile.

Creating file docs/make.bat.

**Finished: An initial directory structure has been created.**

```
importar nuestro código
import os
import sys
sys.path.insert(0, os.path.abspath('../..'))
```

```
extensiones
extensions = ['sphinx.ext.autodoc',
 'sphinx.ext.intersphinx',
 'sphinx.ext.napoleon',
 'sphinx.ext.todo',
 'sphinx.ext.imgmath',
 'sphinx.ext.viewcode',
 'sphinx.ext.githubpages']
```

```
cosas que preguntó sphinx-quickstart
source_suffix = '.rst'
master_doc = 'index'
project = 'mi programa'
language = 'es'
...
```

```
opciones varias
templates_path = ['_templates']
todo_include_todos = True
pygments_style = 'sphinx'
```

```
importar nuestro código
import os
import sys
sys.path.insert(0, os.path.abspath('../..'))
```

```
extensiones
extensions = ['sphinx.ext.autodoc',
 'sphinx.ext.intersphinx',
 'sphinx.ext.napoleon',
 'sphinx.ext.todo',
 'sphinx.ext.imgmath',
 'sphinx.ext.viewcode',
 'sphinx.ext.githubpages']
```

```
cosas que preguntó sphinx-quickstart
source_suffix = '.rst'
master_doc = 'index'
project = 'mi programa'
language = 'es'
...
```

```
opciones varias
templates_path = ['_templates']
todo_include_todos = True
pygments_style = 'sphinx'
```

```
opciones para la exportación
html_theme = "sphinx_rtd_theme"
```

```
import os
import sys
sys.path.insert(0, os.path.abspath('../..'))

extensiones
extensions = ['sphinx.ext.autodoc',
 'sphinx.ext.intersphinx',
 'sphinx.ext.napoleon',
 'sphinx.ext.todo',
 'sphinx.ext.imgmath',
 'sphinx.ext.viewcode',
 'sphinx.ext.githubpages']

cosas que preguntó sphinx-quickstart
source_suffix = '.rst'
master_doc = 'index'
project = 'mi programa'
language = 'es'
...

opciones varias
templates_path = ['_templates']
todo_include_todos = True
pygments_style = 'sphinx'

opciones para la exportación
html_theme = "sphinx_rtd_theme"
html_theme_path = [sphinx_rtd_theme.get_html_theme_path()]
```

```
sys.path.insert(0, os.path.abspath('../..'))
```

```
extensiones
```

```
extensions = ['sphinx.ext.autodoc',
 'sphinx.ext.intersphinx',
 'sphinx.ext.napoleon',
 'sphinx.ext.todo',
 'sphinx.ext.imgmath',
 'sphinx.ext.viewcode',
 'sphinx.ext.githubpages']
```

```
cosas que preguntó sphinx-quickstart
```

```
source_suffix = '.rst'
master_doc = 'index'
project = 'mi programa'
language = 'es'
...
```

```
opciones varias
```

```
templates_path = ['_templates']
todo_include_todos = True
pygments_style = 'sphinx'
```

```
opciones para la exportación
```

```
html_theme = "sphinx_rtd_theme"
html_theme_path = [sphinx_rtd_theme.get_html_theme_path()]
```

```
latex_elements = {
#
```

```
extensiones
extensions = ['sphinx.ext.autodoc',
 'sphinx.ext.intersphinx',
 'sphinx.ext.napoleon',
 'sphinx.ext.todo',
 'sphinx.ext.imgmath',
 'sphinx.ext.viewcode',
 'sphinx.ext.githubpages']

cosas que preguntó sphinx-quickstart
source_suffix = '.rst'
master_doc = 'index'
project = 'mi programa'
language = 'es'
...

opciones varias
templates_path = ['_templates']
todo_include_todos = True
pygments_style = 'sphinx'

opciones para la exportación
html_theme = "sphinx_rtd_theme"
html_theme_path = [sphinx_rtd_theme.get_html_theme_path()]

latex_elements = {
...
}
```



```
extensions = ['sphinx.ext.autodoc',
 'sphinx.ext.intersphinx',
 'sphinx.ext.napoleon',
 'sphinx.ext.todo',
 'sphinx.ext.imgmath',
 'sphinx.ext.viewcode',
 'sphinx.ext.githubpages']

cosas que preguntó sphinx-quickstart
source_suffix = '.rst'
master_doc = 'index'
project = 'mi programa'
language = 'es'
...

opciones varias
templates_path = ['_templates']
todo_include_todos = True
pygments_style = 'sphinx'

opciones para la exportación
html_theme = "sphinx_rtd_theme"
html_theme_path = [sphinx_rtd_theme.get_html_theme_path()]

latex_elements = {
...
}
```