A brief introduction to Sphinx

Benjamin Audren

École Polytechnique Fédérale de Lausanne

October 24, 2013

Why writing doc is important

again...

Some reasons

- A code is **read more than it is written**

Why writing doc is important

again...

Some reasons

- A code is **read more than it is written**
- Time-saver when **expanding** your code (need good IDE)

Why writing doc is important

again...

Some reasons

- A code is **read more than it is written**
- Time-saver when **expanding** your code (need good IDE)
- **Essential** when sharing with others

Why writing doc is important

again...

Some reasons

- A code is **read more than it is written**
- Time-saver when **expanding** your code (need good IDE)
- **Essential** when sharing with others

See uses with the ipython notebook

Docstring convention

Reminder

How to write docstrings

- Text within **triple back-quotes**: `"""something"""`

Docstring convention

Reminder

How to write docstrings

- Text within **triple back-quotes**: `"""something"""`
- Write **below** statements

Docstring convention

Reminder

How to write docstrings

- Text within **triple back-quotes**: `"""something"""`
- Write **below** statements
- **First line** must contain **short summary**, followed by blanck line

Docstring convention

Reminder

How to write docstrings

- Text within **triple back-quotes**: `"""something"""`
- Write **below** statements
- **First line** must contain **short summary**, followed by blanck line
- Then, paragraph of text, and description of arguments, with possibly type, etc...

Docstring convention
Reminder

How to write docstrings

- Text within **triple back-quotes**: `"""something"""`
- Write **below** statements
- **First line** must contain **short summary**, followed by blanck line
- Then, paragraph of text, and description of arguments, with possibly type, etc...
- Enjoy using `pydoc -g`

Docstring convention

Reminder

How to write docstrings

- Text within **triple back-quotes**: `"""something"""`
- Write **below** statements
- **First line** must contain **short summary**, followed by blanck line
- Then, paragraph of text, and description of arguments, with possibly type, etc...
- Enjoy using `pydoc -g`
- more on PEP 257 http://www.python.org/dev/peps/pep-0257/

Docstring convention

Reminder

How to write docstrings

- Text within **triple back-quotes**: `"""something"""`
- Write **below** statements
- **First line** must contain **short summary**, followed by blanck line
- Then, paragraph of text, and description of arguments, with possibly type, etc...
- Enjoy using `pydoc -g`
- more on PEP 257 `http://www.python.org/dev/peps/pep-0257/`
- even more on **Google style-guide**

How can it become awesomer ?

Sphinx !

An **automated documentation generator** !
It reads your code, build a website/pdf/what you want !
`http://sphinx-doc.org/`

Sphinx by example

likelihood class.py I

```
"""
.. module:: likelihood_class
   :synopsis: Definition of the major likelihoods
.. moduleauthor:: Julien Lesgourgues <lesgourg@cern.ch>
.. moduleauthor:: Benjamin Audren <benjamin.audren@epfl.ch>

Contains the definition of the base likelihood class, with basic
    functions,
as well as more specific likelihood classes that may be reused to
    implement
new ones.

The most important one is :class:‘likelihood‘

"""
import os
```

...

Sphinx by example

likelihood class.py II

```
class likelihood(object):
    """
    General class that all likelihoods will inherit from.

    """

    def __init__(self, path, data, command_line):
        """
        It copies the content of self.path from the initialization routine of
        the :class:`data` class, and defines a handful of useful methods, that
        every likelihood might need.

        If the nuisance parameters required to compute this likelihood are not
        defined (either fixed or varying), the code will stop.

        :Parameters:
            - **data** (`class`) - initialized instance of :class:`data`
            - **command_line** (`dict`) - dictionary containing the command
              line arguments
```

...

Sphinx by example

mcmc I

```
"""
.. module:: mcmc
   :synopsis: Monte Carlo procedure
.. moduleauthor:: Benjamin Audren <benjamin.audren@epfl.ch>

This module defines one key function, :func:`chain`, that handles the Markov
chain. So far, the code uses only one chain, as no parallelization is done.

This function in turn calls several other routines. These are called just once:

* :func:`get_covariance_matrix`
* :func:`read_args_from_chain`
* :func:`read_args_from_bestfit`

Their usage is pretty straightforward, and detailled below anyway. On the
contrary, these routines are called at every step:

* :func:`compute_lkl` is called at every step in the Markov chain, returning
  the likelihood at the current point in the parameter space.
* :func:`get_new_position` returns a new point in the parameter space,
  depending on the proposal density.

The arguments of these functions will often contain **data** and/or **cosmo**.
They are both initialized instances of respectively :class:`data` and the
cosmological class. They will thus not be described for every function.
"""
```

...

How does this render ?

go to http://baudren.web.cern.ch/baudren/build/likelihood_class.html and http://baudren.web.cern.ch/baudren/build/mcmc.html

What is this new devilry ?

Source code for the html files

mcmc.rst:

```
Mcmc module
===========

.. automodule:: mcmc
    :members:
    :undoc-members:
    :show-inheritance:
```

analyze.rst:

```
Likelihood class module
=======================

.. automodule:: likelihood_class
    :members:
    :undoc-members:
    :show-inheritance:
```