

# Software Deployment

## Documentation

---

Sergio Peignier

[sergio.peignier@insa-lyon.fr](mailto:sergio.peignier@insa-lyon.fr)

Associate Professor

INSA Lyon

Biosciences department

# Table of contents

1. Docstrings
2. Sphinx
3. Unity Tests
4. Basic deployment

# Docstrings

---

# Docstrings

- Documentation of Python modules/function/classes/methods
- String constant as 1st object statement
- Describe **what the code does**
- **Every** piece of software should have a **docstring**

```
Project666.py
1  """
2  This is a title
3  =====
4  fezioshgfdihgkjshqfkjlsd
5  """
6
7  def the_function_42():
8      """
9      The function 42 returns 42
10     """
11     return(42)
```

# help()

```
[In [2]: from Project666 import the_function_42  
[In [3]: help(the_function_42)
```

Help on function the\_function\_42 in module Project666:

**the\_function\_42()**

The function 42 returns 42

(END)

```
[In [4]: the_function_42.__doc__
```

```
Out[4]: '\n The function 42 returns 42\n '
```

# Main styles

- **reStructuredText** (the original one, not easily readable)
- **Google** (used by google, readable)
- **Numpy** (used in Numpy, readable)

## reStructuredText style (RST)

```
:param arg1: description  
:param arg2: description  
:type arg1: type description  
:type arg1: type description  
:return: return description  
:rtype: the return type description
```

```
def function_with_types_in_docstring(param1, param2):  
    """Example function with types documented in the docstring.  
  
    `PEP 484`_ type annotations are supported. If attribute, parameter, and  
    return types are annotated according to `PEP 484`_, they do not need to be  
    included in the docstring:  
  
    Args:  
        param1 (int): The first parameter.  
        param2 (str): The second parameter.  
  
    Returns:  
        bool: The return value. True for success, False otherwise.  
  
    .. _PEP 484:  
        https://www.python.org/dev/peps/pep-0484/  
  
    """
```

# Numpy style

```
def function_with_types_in_docstring(param1, param2):
    """Example function with types documented in the docstring.

    `PEP 484`_ type annotations are supported. If attribute, parameter, and
    return types are annotated according to `PEP 484`_, they do not need to be
    included in the docstring:

    Parameters
    -----
    param1 : int
        The first parameter.
    param2 : str
        The second parameter.

    Returns
    -----
    bool
        True if successful, False otherwise.

    .. _PEP 484:
       https://www.python.org/dev/peps/pep-0484/

    """
```

# Potentials

- Write math **equations**
- Add **links**
- Add **examples**
- ...

# Sphinx

---



# SPHINX

Python Documentation Generator

[Home](#)

[Get it](#)

## Welcome

Sphinx is a tool that makes it easy to create intelligent and beautiful documentation, written by Georg Brandl and licensed under the BSD license.

*What users say:*

“Cheers for a great tool that actually makes programmers **want** to write documentation!”

# Features

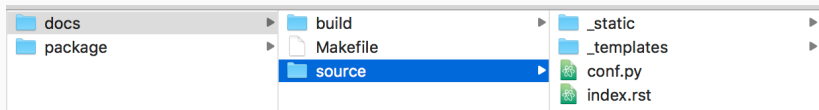
- Includes docstrings from python code
- **Output formats:** HTML, PDF (via LaTeX), ePub, ...
- Build **cross-references**
- Build **index**
- **Test code snippets**

# sphinx-quickstart

Run the sphinx-quickstart script, and say yes to :

- "> autodoc: automatically insert docstrings ..."
- "> doctest: automatically test code snippets ..."
- "> todo: write "todo" entries that ..."
- "> coverage: checks for documentation coverage ..."
- "> mathjax: include math, rendered"
- "> Create Makefile? (y/n) [y]:"

This creates some files and folders:



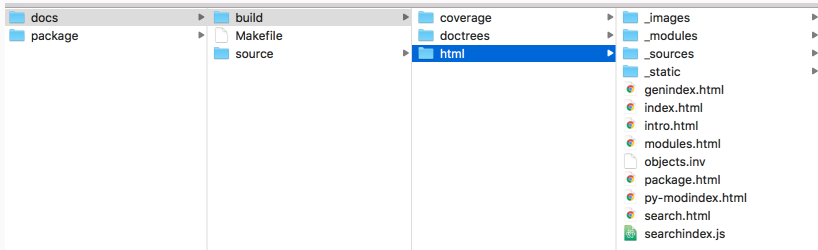
# index.rst

```
1 |.. Project_666_yeah documentation master file, created by
2 |   sphinx-quickstart on Mon Mar 25 11:28:47 2019.
3 |   You can adapt this file completely to your liking, but it should at least
4 |   contain the root `toctree` directive.
5 |
6 | Welcome to Project_666_yeah's documentation!
7 | =====
8 |
9 | .. module:: package
10 |
11 | .. toctree::
12 |    :maxdepth: 2
13 |    :caption: Contents:
14 |
15 |     intro
16 |     modules
17 |     package
18 |
19 | Indices and tables
20 | =====
21 |
22 | * :ref:`genindex`
23 | * :ref:`modindex`
24 | * :ref:`search`
25 |
```

```
20 # -- Project information -----
21
22 project = 'Project_666_yeah'
23 copyright = '2019, bad dog'
24 author = 'bad dog'
25
26 # The short X.Y version
27 version = ''
28 # The full version, including alpha/beta/rc tags
29 release = '0'
30
31
32 # -- General configuration -----
33
34 # If your documentation needs a minimal Sphinx version, state it here.
35 #
36 # needs_sphinx = '1.0'
37
38 # Add any Sphinx extension module names here, as strings. They can be
39 # extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
40 # ones.
41 extensions = [
42     'sphinx.ext.autodoc',
43     'sphinx.ext.doctest',
44     'sphinx.ext.todo',
45     'sphinx.ext.coverage',
46     'sphinx.ext.mathjax',
47     'sphinx.ext.ifconfig',
48     'sphinx.ext.viewcode',
49     'sphinx.ext.githubpages',
50     'sphinx.ext.napoleon',
51     'recommonmark',
52 ]
```

# makefile

Run: "make html" to create the documentation in HTML



## Project\_666\_yeah 0 documentation

WELCOME TO PROJECT\_666\_YEAH'S DOCUMENTATION

### Welcome to Project\_666\_yeah's documentation!

---

Contents:

- **Welcome to the crazy introduction**
- **Indices and tables**
- **package**
  - **package package**
- **package package**
  - **Submodules**
  - **package.Project666 module**
  - **Module contents**

# sphinx-apidoc

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

Run:

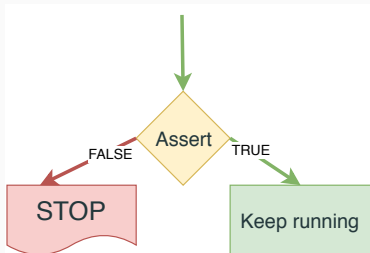
```
> sphinx-apidoc -f -o <output folder> <package path>
```

# Unity Tests

---

# Assert

- Boolean expressions
- Checks if a condition is true or false
- Debugging tool / run-time checks
- `assert(<condition>,<error message>)`
- Check also unittest (<https://docs.python.org/2/library/unittest.html>)



# Assert | Assertions list

Method	Checks that
<code>assertEqual(a, b)</code>	<code>a == b</code>
<code>assertNotEqual(a, b)</code>	<code>a != b</code>
<code>assertTrue(x)</code>	<code>bool(x) is True</code>
<code>assertFalse(x)</code>	<code>bool(x) is False</code>
<code>assertIs(a, b)</code>	<code>a is b</code>
<code>assertIsNot(a, b)</code>	<code>a is not b</code>
<code>assertIsNone(x)</code>	<code>x is None</code>
<code>assertIsNotNone(x)</code>	<code>x is not None</code>
<code>assertIn(a, b)</code>	<code>a in b</code>
<code>assertNotIn(a, b)</code>	<code>a not in b</code>
<code>assertIsInstance(a, b)</code>	<code>isinstance(a, b)</code>
<code>assertNotIsInstance(a, b)</code>	<code>not isinstance(a, b)</code>

# Doctest

- Parses `doctring`
- Detects text looking like `interactive Python sessions`
- Executes the code
- "Literate testing" / "executable documentation"
- `https://docs.python.org/3/library/doctest.html`

## Doctest | Example

```
In [1]: def add_2_nbs(a,b):
51...:     """
52...:     this function adds 2 nbs
53...:     Args:
54...:     a (float): nb1
55...:     b (float): nb2
56...:     Returns:
57...:     float: sum
58...:     """
59...:     >>> add_2_nbs(1,2)
60...:     3
61...:     >>> add_2_nbs(40,2)
62...:     42
63...:     """
64...:     return(a+b)

g with:
[In [2]: import doctest
```

## Doctest | Example

```
[In [4]: doctest.testmod(verbose=True)
Trying:
    add_2_nbs(1,2)
Expecting:
    3
ok
Trying:
    add_2_nbs(40,2)
Expecting:
    42
ok
1 items had no tests:
  __main__
1 items passed all tests:
  2 tests in __main__.add_2_nbs
2 tests in 2 items.
2 passed and 0 failed.
```

# Basic deployment

---

# Github Repository

- clean **README.md**:
  - **Describe** the Project
  - Two audiences: **developers** and **users**
  - What does the program **solves**?
  - **Installation** instructions
  - **FAQ** section
  - **Contribute** section (issue tracker link, source code link)
  - **TODO** section (functionalities to be added)
- Add some **examples/tutorials** (script, notebook)
- Add **License**

## Deploy a webpage

- Try github pages <https://pages.github.com/>
- Try Read the docs <https://readthedocs.org>