Scientific Software Development

Inga Ulusoy, Scientific Software Center, Interdisciplinary Center for Scientific Computing, Heidelberg University

January 2024

Unit 4: Documentation

- A good documentation
- Documentation style guides
- Set up your documentation with sphinx (doxygen)
- Publish your documentation on readthedocs
- Publish your documentation on GitHub pages

The documentation tools for your software will be set up and structured.

A good documentation

You want people to use your project

- they need to know what it does

Users should be able to install your project

- they need to know requirements and how to set everything up

In five months from now, you still want to remember what you actually programmed there

- you need to tell which parameter and which function does what and why

You want to have more contributors

- they need to know what your project does and how

A good documentation

You want people to use your project

- they need to know what it does

- Description of the data transformations in your project
- Description of input and output
- Description of options that can be selected
- Targeted application and applicability range

Users should be able to install your project

- they need to know requirements and how to set everything up

In five months from now, you still want to remember what you actually programmed there

- you need to tell which parameter and which function does what and why

You want to have more contributors

- they need to know what your project does and how

A good documentation

You want people to use your project

- they need to know what it does

- Description of the data transformations in your project
- Description of input and output
- Description of options that can be selected
- Targeted application and applicability range

Users should be able to install your project

- they need to know requirements and how to set everything up

- Description of installation requirements
- Installation instructions

In five months from now, you still want to remember what you actually programmed there

- you need to tell which parameter and which function does what and why

You want to have more contributors

- they need to know what your project does and how

A good documentation

You want people to use your project

- they need to know what it does

- Description of the data transformations in your project
- Description of input and output
- Description of options that can be selected
- Targeted application and applicability range

Users should be able to install your project

- they need to know requirements and how to set everything up

- Description of installation requirements
- Installation instructions

In five months from now, you still want to remember what you actually programmed there

- you need to tell which parameter and which function does what and why

- Description of the methods used for the data transformations (functions, classes need to be described with their options)
- Reasoning for choice of method needs to be included

You want to have more contributors

- they need to know what your project does and how

A good documentation

You want people to use your project

- they need to know what it does

- Description of the data transformations in your project
- Description of input and output
- Description of options that can be selected
- Targeted application and applicability range

Users should be able to install your project

- they need to know requirements and how to set everything up

- Description of installation requirements
- Installation instructions

In five months from now, you still want to remember what you actually programmed there

- you need to tell which parameter and which function does what and why

- Description of the methods used for the data transformations (functions, classes need to be described with their options)
- Reasoning for choice of method needs to be included

You want to have more contributors

- they need to know what your project does and how

- Description of the methods used for the data transformations (functions, classes need to be described with their options)
- Licensing information

A good documentation

You want people to use your project

- they need to know what it does

- Description of the data transformations in your project
- Description of input and output
- Description of options that can be selected
- Targeted application and applicability range

Users should be able to install your project

- they need to know requirements and how to set everything up

- Description of installation requirements
- Installation instructions

In five months from now, you still want to remember what you actually programmed there

- you need to tell which parameter and which function does what and why

- Description of the methods used for the data transformations (functions, classes need to be described with their options)
- Reasoning for choice of method needs to be included

You want to have more contributors

- they need to know what your project does and how

- Description of the methods used for the data transformations (functions, classes need to be described with their options)
- Licensing information

Reproducibility
Impact
Sustainability

A good documentation

- Description of the data transformations in your project
- Description of input and output
- Description of options that can be selected
- Targeted application and applicability range

- Description of installation requirements
- Installation instructions

- Description of the methods used for the data transformations (functions, classes need to be described with their options)
- Licensing information

- Description of the methods used for the data transformations (functions, classes need to be described with their options)
- Reasoning for choice of method needs to be included

- Name and short description of the software, authors, date of initial development
- Main features
- Main requirements
  - Input examples and explanations, step-by-step tutorial
  - More detailed description of scientific approach and input variables reference
  - Validity range of the parameters
  - License information, bug tracker, references, citations
  - Source code description - functions and classes, modules, variables

A good documentation

- Name and short description of the software, authors, date of initial development
- Main features
- Main requirements
  - Input examples and explanations, step-by-step tutorial
  - More detailed description of scientific approach and input variables reference
  - Validity range of the parameters
  - License information, bug tracker, references, citations
  - Source code description - functions and classes, modules, variables

- Combine sphinx using your docstrings and mark-up files to put together a reasonably readable html
- You will learn in unit6 how to automatically push this to readthedocs – it will be updated whenever you make changes

A good documentation

- Read through this post on using documentation and comments for Python code: https://realpython.com/documenting-python-code/

Unit 4: Documentation

- *A good documentation*
- Documentation style guides
- Set up your documentation with sphinx (doxygen)
- Publish your documentation on readthedocs
- Publish your documentation on GitHub pages

The documentation tools for your software package will be set up and structured.

Documentation style guides

- PEP 257 - https://www.python.org/dev/peps/pep-0257/ - docstring conventions:
  - docstring = description that you add below the first line when defining a new function/`__init__` method of a class, using """
  - Becomes an attribute of that object – `__doc__`

1. Phrase ending with "."
2. Describe as command "Do this"
3. Does not reiterate the code

Summary line
blank line
More elaborate description

PEP 257

The docstring for a module should generally list the classes, exceptions and functions (and any other objects) that are exported by the module, with a one-line summary of each. (These summaries generally give less detail than the summary line in the object's docstring.) The docstring for a package (i.e., the docstring of the package's __init__.py module) should also list the modules and subpackages exported by the package.

The docstring for a function or method should summarize its behavior and document its arguments, return value(s), side effects, exceptions raised, and restrictions on when it can be called (all if applicable). Optional arguments should be indicated. It should be documented whether keyword arguments are part of the interface.

The docstring for a class should summarize its behavior and list the public methods and instance variables. If the class is intended to be subclassed, and has an additional interface for subclasses, this interface should be listed separately (in the docstring). The class constructor should be documented in the docstring for its __init__ method. Individual methods should be documented by their own docstring.

Google style docstrings

- We will use the napoleon extension for the formatting of the docstrings. An example can be found here: https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html
- An example how it is rendered is found here: https://tdci-analysis.readthedocs.io/en/latest/tdci-a.html

Unit 4: Documentation

- *A good documentation*
- *Documentation style guides*
- Set up your documentation with sphinx (doxygen)
- Publish your documentation on readthedocs
- Publish your documentation on GitHub pages

The documentation tools for your software package will be set up and structured.

Documentation using sphinx

- sphinx = documentation generator, specific to Python (but interfaces exist)
- Takes docstrings and puts them together in a linked html (latex, pdf, …)
- You can add additional files such as README, license, …
- reStructuredText (rst) but also markup language (recommended)

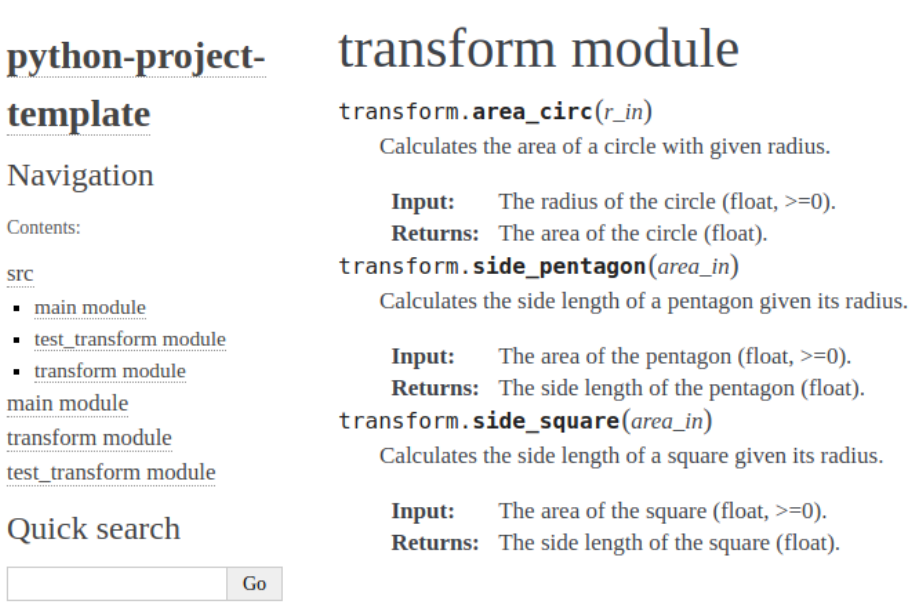
Documentation using `sphinx`

https://www.sphinx-doc.org/en/master/
https://pythonhosted.org/an_example_pypi_project/sphinx.html

```
def area_circ(r_in):
    """Calculate the area of a circle with given radius.

    :Input: The radius of the circle (float, >=0).
    :Returns: The area of the circle (float)."""
```

use documentation string in function head



python-project-template

Navigation

Contents:

src
- main module
- test_transform module
- transform module

main module
transform module
test_transform module

Quick search

Go

transform module

`transform.`**`area_circ`**(*r_in*)

Calculates the area of a circle with given radius.

Input: The radius of the circle (float, >=0).
Returns: The area of the circle (float).

`transform.`**`side_pentagon`**(*area_in*)

Calculates the side length of a pentagon given its radius.

Input: The area of the pentagon (float, >=0).
Returns: The side length of the pentagon (float).

`transform.`**`side_square`**(*area_in*)

Calculates the side length of a square given its radius.

Input: The area of the square (float, >=0).
Returns: The side length of the square (float).

Documentation using `sphinx`

- Navigate into your `doc` directory
- Type `sphinx-quickstart`
- Answer "y" - "your project name" - "Author names" - "release version" - I would choose 1.0 as that would be the first official release version; "project language"
- Open `conf.py` and uncomment `import os, import sys, sys.path.insert(0, os.path.abspath('.'))`
- Put the correct path – ie. `sys.path.insert(0, os.path.abspath('../src/'))`
- Add `extensions = ['sphinx.ext.autodoc']`
- For a selection of themes, visit https://www.sphinx-doc.org/en/master/usage/theming.html
- Type `make html`
- Open the index.html file in your build/html directory – it should open in your browser and display the initial documentation page
- Use autodoc to generate the modules.rst file: `sphinx-apidoc -o source/ ../src`
- Type `make html`
- Again check index.html – it should have added your source code docstrings in modules

Sphinx tips

- All code needs to be self-contained (i.e. in a function or class), otherwise sphinx will run your code!
- Use `myst-parser` to include mark-up type formatting and add `extensions = ['myst-parser']`
- Use napoleon extension for nicer highlighting on the html

```
extensions = ['sphinx.ext.napoleon']
```

Unit 4: Documentation

- *A good documentation*
- *Documentation style guides*
- *Set up your documentation with sphinx (doxygen)*
- Publish your documentation on readthedocs
- Publish your documentation on GitHub pages

The documentation tools for your software package will be set up and structured.

readthedocs - https://readthedocs.org/

- Free documentation hosting for open-source projects
- Uses the sphinx generator

Please create an account on readthedocs.

Unit 4: Documentation

- *A good documentation*
- *Documentation style guides*
- *Set up your documentation with sphinx (doxygen)*
- *Publish your documentation on readthedocs*
- Publish your documentation on GitHub pages

The documentation tools for your software package will be set up and structured.

GitHub pages https://pages.github.com/

- You can also use GitHub pages to host your documentation.

Please complete the learning lab to learn about GitHub pages.

https://github.com/skills/github-pages

You can use this for all kinds of things, for example to showcase your CV and accomplished research/software projects.

Unit 4: Documentation

- *A good documentation*
- *Documentation style guides*
- *Set up your documentation with sphinx (doxygen)*
- *Publish your documentation on readthedocs*
- *Publish your documentation on GitHub pages*

The documentation tools for your software package will be set up and structured.

Live lesson

- In the live lesson, we will set up sphinx together step by step. You will add README and additional files, and work on improving your documentation.

Live lesson - Demonstrations

- The following demonstrations will take place in the beginning of the live session:
  - How to set up sphinx for your software project