Documenting Python Code

Christian Heitzmann | SimplexaCode AG

SimplexaCode

SimplexaCode AG | Grimselweg 11/501 | 6005 Lucerne | Switzerland
+41 43 810 06 03 | *info@simplexacode.ch* | *www.simplexacode.ch*

PYCON - PYCON - 24

PyCon Italia | Grand Hotel Mediterraneo
Florence | Italy | May 23, 2024

Typical Work Environment of a (New) Software Developer

1. **appliation phase**: promise the stars
2. **first day:** setup the machine
   - best (but not optimal) case: disk image
   - worst case: ask coworkers for help
   - typical case: outdated and incomplete wiki pages
3. **first month:** try to gain initial understanding of the “architecture”
   - no API or source code comments
   - outdated or missing documentation
   - *“We don’t do that here.”* or *“&$!#%”*
4. **following months and years:** reverse engineering, day in and day out

Typical Excuses (and Undeniable Reasons)

- ***“We are agile!”***
  - ⟹ Agile does *not* mean no documentation.
- ***“No time!” / “No money!”***
  - ⟹ Let’s do the math.
- ***“We do Clean Code / TDD / XP / [you name it], so the source code is the documentation!”***
  - ⟹ Dream on!

⟹ many undeniable reasons for software documentation: [1] [2]

[1] C. Heitzmann, *Javadoc with Style*, SimplexaCode Blog, 2022–2023, *link.simplexacode.ch/gh32*

[2] C. Heitzmann, *Markup Languages in Software Documentation*, SimplexaCode Blog, 2023–2024, *link.simplexacode.ch/4gwr*

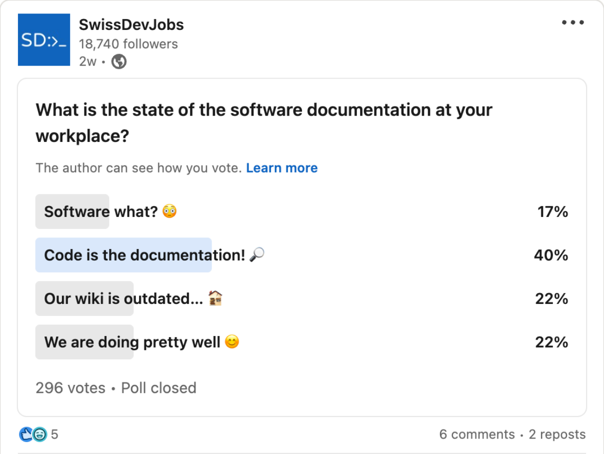
About You

- Welcome!
- shortlink to these slides
  - ***link.simplexacode.ch/s7ir***
- target audience
  - everyone working on professional or non-small Python projects
  - not relevant for personal hobby Python development, e.g., on Raspberry Pi

About Me

- MSc ETH in Computational Science and Engineering
- Certifications in Java, Python, and Spring
- CAS in Machine Learning
- > 20 years software development
- Teaching Diplomas in Computer Science and Mathematics
- > 12 years teaching classes and courses
  - Java and Python programming
  - mathematics and algorithms
- > 5 years technical writing
  - software documentation
  - articles for IT journals

LinkedIn Survey



SD:>_ **SwissDevJobs**
18,740 followers
2w •

What is the state of the software documentation at your workplace?

The author can see how you vote. **Learn more**

Software what? 😳	**17%**
Code is the documentation! 🔎	**40%**
Our wiki is outdated... 🏚	**22%**
We are doing pretty well 😊	**22%**

296 votes • Poll closed

5   6 comments • 2 reposts [3]

[3] *www.linkedin.com/posts/swissdev-jobs_activity-7112353654023553024-BQyI*

Documentation Debts

1. Every day you do not document, you increase the software's debts.
2. The debts show up more and more every day.
   If not with you, then with your employees, especially the new ones.
3. Until you reach the point where reverse engineering takes up most of the time.
4. Documentation debts are **technical debts**[4] – with all their consequences.

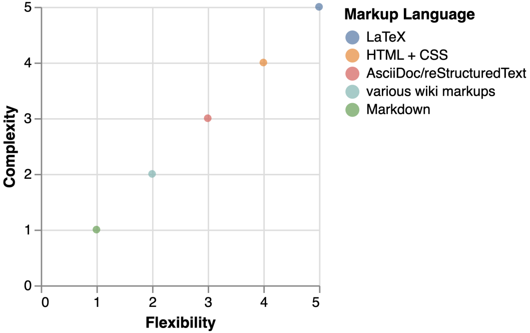
[4] C. Lilienthal, *Langlebige Software-Architekturen*, 3. Auflage, dpunkt.verlag, 2020

Audiences and Altitudes

- **Who** do you want to address **(audience)**?
  - external developers, end users, ...
  - product owners, managers, ...
  - internal developers
  - ...
- **What level of detail** do you want to document **(“altitude”)**?
  - overall software architecture → external documents and diagrams
  - classes, interfaces, APIs → docstrings, Javadoc, JSDoc, ...
  - function internals → source code comments
  - ...

(Lightweight) Markup Languages

- **markup language:**
  machine- and human-readable language for structuring and formatting text and other data
  - e.g., *HTML + CSS*, *LaTeX*, ...
- **lightweight markup language:**
  simpler syntax without disturbing the writing and reading flow
  - e.g., *Markdown*, *reStructuredText*, *AsciiDoc*, various wiki markups, ...



Docs as Code

- **Documentation is treated like source code.**
  - same IDE
  - same version control
  - same process integration
  - same CI/CD pipeline integration

- Documentation needs to be in text format (~~binary format~~).

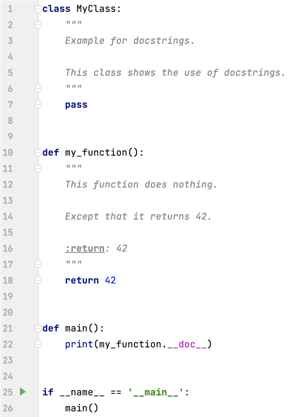
Near Source Code Documentation

- Java: **Javadoc** using HTML and defined Javadoc tags
- Python: **docstrings** usually using reStructuredText syntax
- C#/.NET: **XML documentation comments** with defined XML tags
- JavaScript: **JSDoc** using HTML and defined tags for JavaScript

- close **proximity between source code and documentation**

Docstrings

- Python uses **docstrings** for documentation.
    - They appear at the **beginning of** a class or function **body**.
    - Cf. **Javadocs** which appear **above** a class or method **definition.**
- Docstrings are data that can be **accessed at runtime**.
- Docstrings can and should contain **reStructuredText syntax.**[5] [6]



```python
class MyClass:
    """
    Example for docstrings.

    This class shows the use of docstrings.
    """
    pass


def my_function():
    """
    This function does nothing.

    Except that it returns 42.

    :return: 42
    """
    return 42


def main():
    print(my_function.__doc__)


if __name__ == '__main__':
    main()
```

[5] *docutils.sourceforge.io/rst.html*
[6] *www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html*

ReStructuredText (for Python Documentation Purposes)

devguide.python.org/documentation/markup/

Generating Python Documentation

- ~~`help(…)`~~
- ~~*pydoc*~~
- *Sphinx*[7]
  - `sphinx-apidoc`[8] using the `autodoc`[9] extension
  - *PEP 287 – reStructuredText Docstring Format*[10] or
  - NumPy docstrings format[11] or
  - Google docstrings format[12]

[7] *www.sphinx-doc.org*

[8] *www.sphinx-doc.org/en/master/man/sphinx-apidoc.html*

[9] *www.sphinx-doc.org/en/master/usage/extensions/autodoc.html*

[10] *peps.python.org/pep-0287/*

[11] *numpydoc.readthedocs.io/en/latest/format.html*

[12] *google.github.io/styleguide/pyguide.html#38-comments-and-docstrings*

Google Docstrings Format

google.github.io/styleguide/pyguide.html#38-comments-and-docstrings

Demo | Sphinx with Pure ReStructuredText 1

1. PyCharm terminal
   1. `pip install --upgrade pip`
   2. `pip install setuptools`
   3. `pip install sphinx`
   4. `pip install sphinxcontrib-kroki`
   5. `pip install sphinx_rtd_theme`
   6. `exit`
2. PyCharm terminal
   1. `mkdir docs`
   2. `cd docs`
   3. `sphinx-quickstart`
      - Separate source and build directories: y
3. `make html`
4. web browser

Demo | Sphinx with Pure ReStructuredText 2

5. *docs/source/conf.py*
   1. Change `copyright`.
   2. `html_theme = 'classic'`
   3. Add `'sphinxcontrib.kroki'` to `extensions`.
   4. `make html`
6. *docs/source/index.rst*
   1. Add content.
7. `make html`

Demo | Sphinx with sphinx-apidoc 1

1. *docs/source/conf.py*
    1. Add `sys.path.insert(0, os.path.abspath('../../`*`[path-to-module-root]`*`'))`.
    2. Add `'sphinx.ext.autodoc'` and `'sphinx.ext.napoleon'` to `extensions`.
    3. `html_theme = 'sphinx_rtd_theme'`
2. *docs/source/index.rst*
    1. Add modules below `toctree` / `maxdepth` / `caption`.

Demo | Sphinx with sphinx-apidoc 2

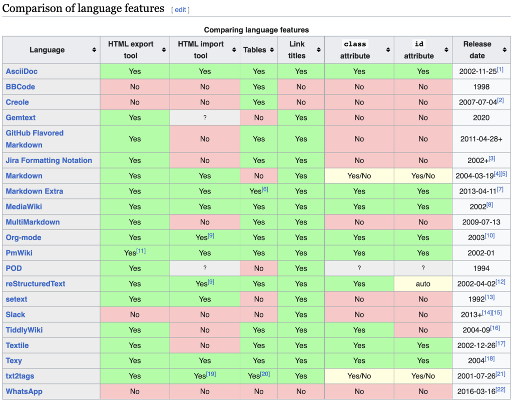
3. PyCharm terminal
   1. `cd docs`
   2. `sphinx-apidoc -o source/ ../`*`[path-to-module-root]`*
4. Check *docs/source/[module-name].rst* and *docs/source/modules.rst*.
5. Delete *docs/build/*
6. `make html`

Sphinx Themes and Examples

- Themes
  - Gallery: *sphinx-themes.org*
  - Projects: *www.sphinx-doc.org/en/master/examples.html*
- Examples
  - Python: *docs.python.org/3.12/library/csv.html*
  - NumPy: *numpy.org/doc/stable/reference/arrays.ndarray.html*
  - pandas: *pandas.pydata.org/docs/reference/api/pandas.read_pickle.html*
- Learn the documentation style from other modules' source code.
  - NumPy Example: *github.com/numpy/numpy/blob/main/doc/source/reference/arrays.ndarray.rst*

AsciiDoc as Lightweight Markup Language

- 21 lightweight markup languages[13]
- exclude
  - product or application focus
  - outdated or shadowy existence
- top 4
  1. **AsciiDoc**
  2. **reStructuredText**
  3. MediaWiki
  4. Textile



Comparison of language features [edit]

Comparing language features

Language	HTML export tool	HTML import tool	Tables	Link titles	`class` attribute	`id` attribute	Release date
AsciiDoc	Yes	Yes	Yes	Yes	Yes	Yes	2002-11-25[1]
BBCode	No	No	Yes	No	No	No	1998
Creole	No	No	Yes	No	No	No	2007-07-04[2]
Gemtext	Yes	?	No	Yes	No	No	2020
GitHub Flavored Markdown	Yes	No	Yes	Yes	No	No	2011-04-28+
Jira Formatting Notation	Yes	No	Yes	Yes	No	No	2002+[3]
Markdown	Yes	Yes	No	Yes	Yes/No	Yes/No	2004-03-19[4][5]
Markdown Extra	Yes	Yes	Yes[6]	Yes	Yes	Yes	2013-04-11[7]
MediaWiki	Yes	Yes	Yes	Yes	Yes	Yes	2002[8]
MultiMarkdown	Yes	No	Yes	Yes	No	No	2009-07-13
Org-mode	Yes	Yes[9]	Yes	Yes	Yes	Yes	2003[10]
PmWiki	Yes[11]	Yes	Yes	Yes	Yes	Yes	2002-01
POD	Yes	?	No	Yes	?	?	1994
reStructuredText	Yes	Yes[9]	Yes	Yes	Yes	auto	2002-04-02[12]
setext	Yes	Yes	No	Yes	No	No	1992[13]
Slack	No	No	No	Yes	No	No	2013+[14][15]
TiddlyWiki	Yes	No	Yes	Yes	Yes	No	2004-09[16]
Textile	Yes	No	Yes	Yes	Yes	Yes	2002-12-26[17]
Texy	Yes	Yes	Yes	Yes	Yes	Yes	2004[18]
txt2tags	Yes	Yes[19]	Yes[20]	Yes	Yes/No	Yes/No	2001-07-26[21]
WhatsApp	No	No	No	No	No	No	2016-03-16[22]

[13] *en.wikipedia.org/wiki/Lightweight_markup_language*

Markdown

- 👍 most popular lightweight markup language
- 👎 no standard
- 👎 many flavors
- 👎 plug-ins or HTML code necessary
- 👎 no tables, cross references, footnotes, YouTube videos, ...

⟹ **Don’t use Markdown!**[14]

⟹ AsciiDoc is an ideal drop-in replacement.

[14] E. Holscher, *Why You Shouldn’t Use “Markdown” for Documentation*, *www.ericholscher.com/blog/2016/mar/15/dont-use-markdown-for-technical-docs/*

AsciiDoc for Python Docstrings?

- I’m not aware of any well-established (Sphinx) converter/extension/plug-in.
- Within Python, **stick to Python standards** (here: reStructuredText).
- Compare to **Javadoc for Java** or **JSDoc for JavaScript**.
- **Keep docstrings simple.**

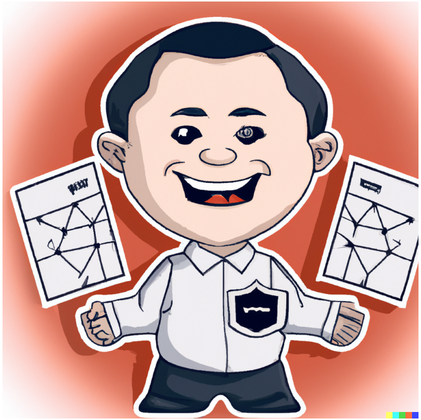
reStructuredText for *All* Software Documentation?

- technologically fine
- **AsciiDoc**
  - 👍 more generic, esp. for non-techies
  - 👎 non-Pythonic in Python
- **reStructuredText**
  - 👍 covers Python docstrings (but not Javadoc or JSDoc)
  - 👎 more nerdy, clumsy, and aimed at tech-savvy people
- compromise: **tagged source code includes in AsciiDoc**?
  - 👍 works similar across all programming languages
  - 👎 increases distance between source code and documentation

⟹ *Carefully* discuss with all developers and stakeholders.

Three Ingredients for Successful Software Documentation

1. **docs as code infrastructure**
   - reStructuredText/Sphinx
   - AsciiDoc/Antora
   - Javadoc/HTML
   - ~~Markdown~~
2. **few, but responsible people**
   - «Technical Writer»
   - «Doctator»
3. **documentation guide**[15]



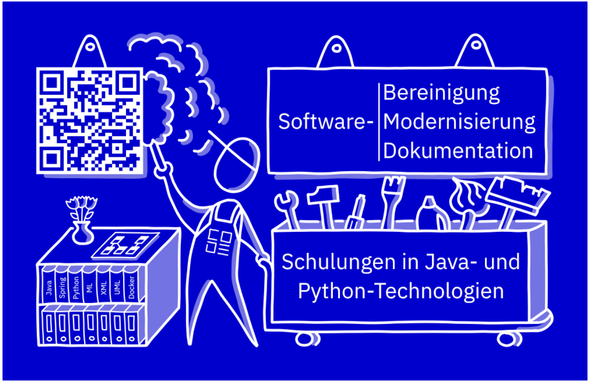
[15] specimen: GitLab, *Documentation Style Guide*, *docs.gitlab.com/ee/development/documentation/styleguide/*

Summary

- general insights
  - Documenting can be **learned**.
  - It all depends on a **systematic approach**.
  - Get **external help** if you want to avoid reinventing the wheel.
- specific next steps
  - Read ***link.simplexacode.ch/4gwr*** about lightweight markup languages.
  - Take my **Software Training Program** and **business card**.
  - Follow me on **LinkedIn** to stay up-to-date.

Questions?

SimplexaCode



link.simplexacode.ch/s7ir