MkDocs & reuse

Isac Pasianotto

"Code is read more often than it is written"

— Guido van Rossum

What's the point of documenting code?

- Helps you remember the design decisions you made...
  - Now you know what you are doing and why. But are you sure you'll remember everything in a few months? And in the next years?
  - If you need to update your code in the future, you'll have to spend (a lot of) time relearning how it works; unless you want to rewrite everything from scratch (please do not waste your time).
- I (probably) know what I am doing, but I am working with a team. My colleagues are not inside my head and to do their job they have to know what I'm doing.
- Someone might be interested in your work, and the documentation can help them understand whether the software is right for them or not.
  If it's not documented, the chances of someone giving it a chance are drastically reduced.

How to document *properly*?

There is no single way to do this. But there are standards.

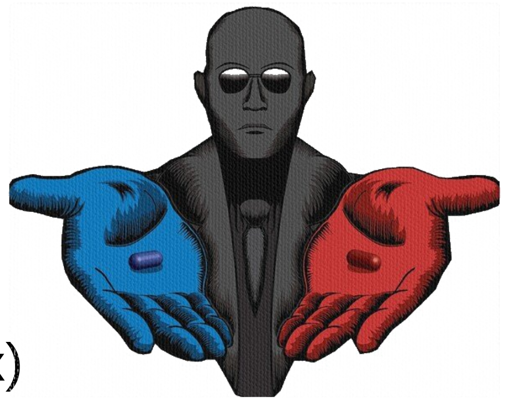
Blue pill: Add comments in the code

Pros:

- No need to use external tool (which my require different languages)
- The doc is inside the code
- Can integrate better with `help()` or use docs. Generator (e.g., Sphynx)

Cons:

- The doc is inside the code
- Not so flexible



- ***Red pill***: Use an external tool

Pros:

- More flexible, and usually easier

Cons:

- May require knowledge of others languages or tool

By the way, we are not in matrix! We can take both pills :)

Blue pill: Comments

- Requires to add comments formatted in a certain way, which my be not unique.

- Require a lot of effort to adapt the explanation of your code into the strict standard that you are following

That is a good reference to read something more!

```python
def get_spreadsheet_cols(file_loc,
  print_cols=False):
    """
    Gets and prints the spreadsheet's  header
    :param file_loc: The file location of the
         spreadsheet
    :type file_loc: str
    :param print_cols: A flag used to print
        the columns to the console (default is
         False)
    :type print_cols: bool
    :returns: a list of strings representing
         the header columns
    :rtype: list
    """
    file_data = pd.read_excel(file_loc)
    col_headers = list(file_data.columns.values)

    if print_cols:
        print("\n".join(col_headers))
    return col_headers
```

Red pill: external tools

- Instead of writing comments inside your code, use a tool to generate some [html/pdf] resources and make it available alongside with the code.

- Usually requires the knowledge of some additional languages (commonly Markdown).

- They are more flexible and have usually more feature that can be useful in the documentation process. E.g., :
  - Inserting images/gif/videos.
  - Links and references to other docs pages.
  - Modify your text appearance, to highlight very important stuffs, hide less important and so on.

MkDocs

Project documentation with Markdown.

Read the Docs

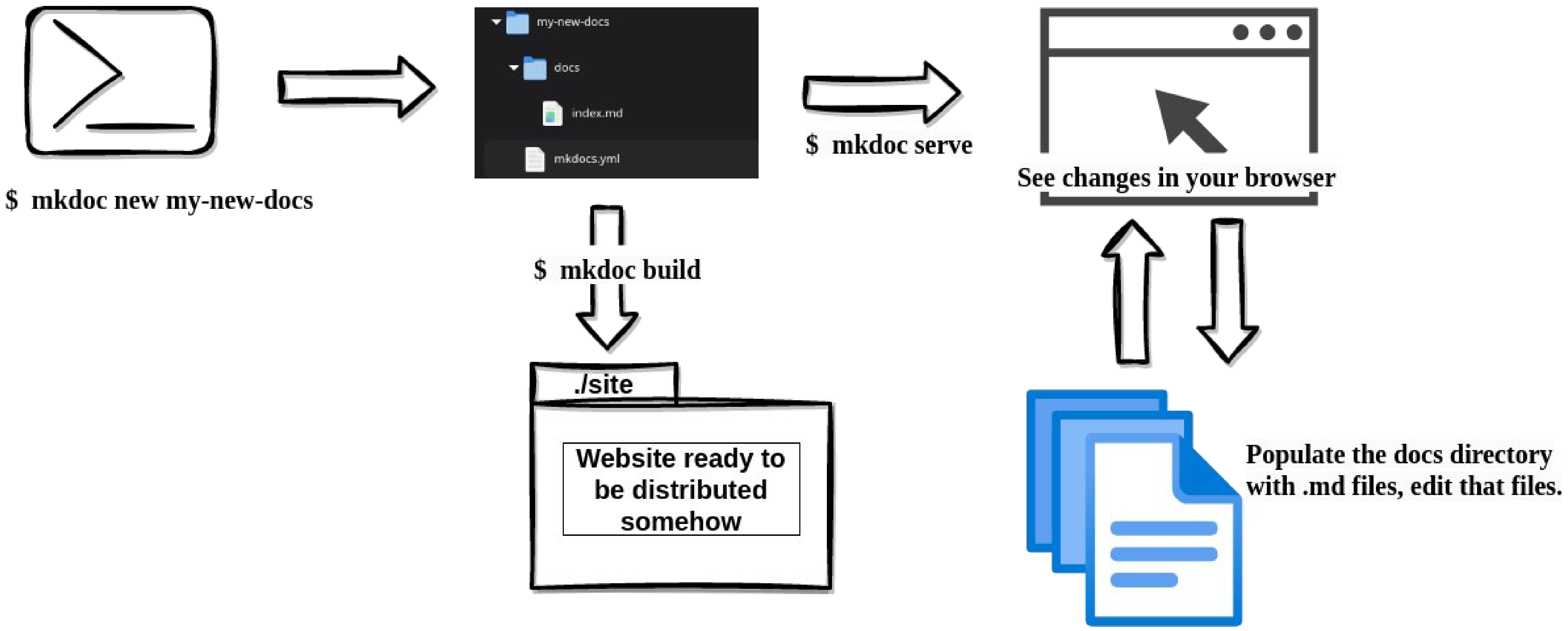
The LaTeX Project

MkDocs

- Very popular way to write communication
- Languages required:
  - `yaml`: very basic, only to edit `mkdocs.yml`
  - `Markdown`: to write the content of your doc
- Create the html-based documentation, which you can inspect through the browser.
- Comes with a web-server to see live the changes you did
- Build for you a entire web side which can be served with a web server (e.g., in a container)

An example: https://orfeo-doc.areasciencepark.it/

MkDocs (cont'd)

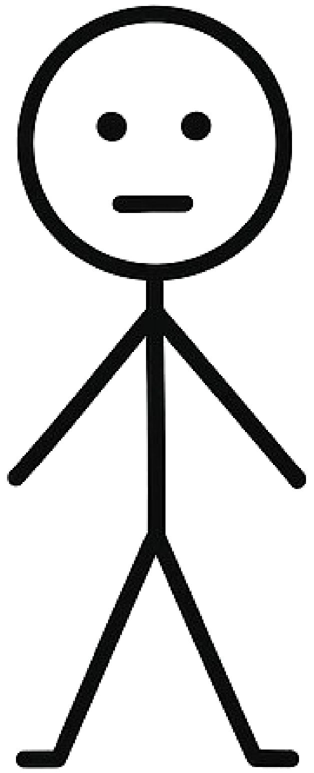


$ mkdoc new my-new-docs
my-new-docs
docs
index.md
mkdocs.yml
$ mkdoc serve
See changes in your browser
$ mkdoc build
./site
Website ready to be distributed somehow
Populate the docs directory with .md files, edit that files.

MkDocs live demo

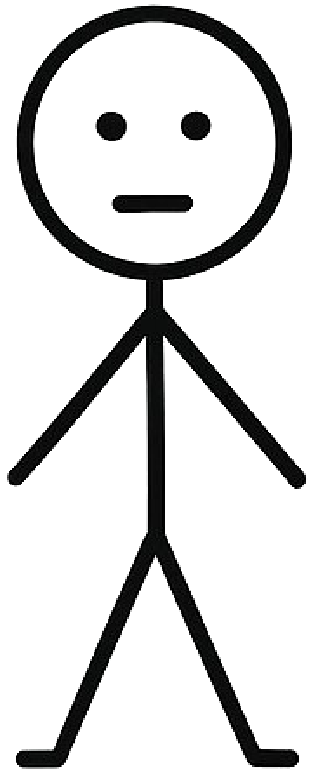
Meet Bill

- This is Bill.
- Bill has been working very hard on a project and wants to share it with the world because he truly believes it can be useful to other people.
- Bill have just published his brand-new amazing new open-source project on github



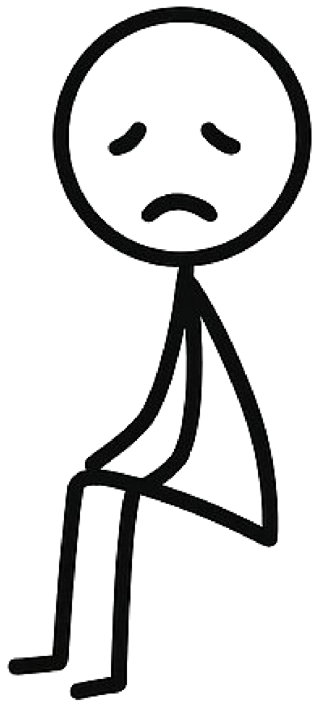
Meet Bill

- This is Bill.
- Bill has been working very hard on a project and wants to share it with the world because he truly believes it can be useful to other people.
- Bill have just published his brand-new amazing new open-source project on github
- Bill waits patiently...



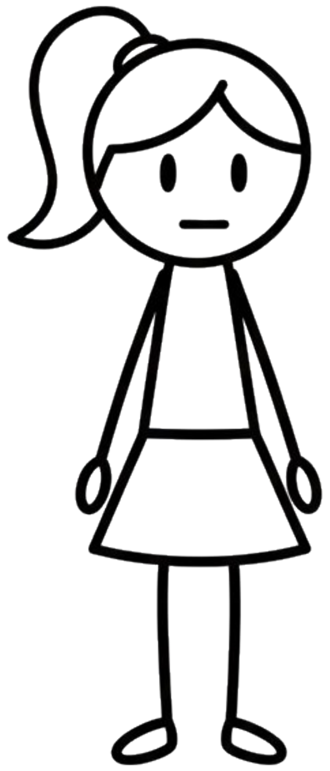
Meet Bill

- This is Bill.
- Bill has been working very hard on a project and wants to share it with the world because he truly believes it can be useful to other people.
- Bill have just published his brand-new amazing new open-source project on github.
- Bill waits patiently...
- No one has ever integrated Bill's project into their codebase or used it in any way.



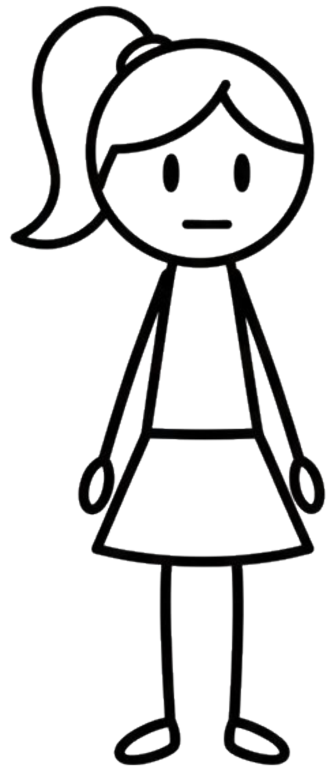
Meet Alice

- This is Alice
- Alice has also been working hard on a project and wants to share it with the world because she truly believes it can be useful to other people.



Meet Alice

- This is Alice
- Alice has also been working hard on a project and wants to share it with the world because she truly believes it can be useful to other people.
- Alice knows that if she doesn't indicate one (or more) licenses to the code on github, the default means "All rights reserved".
- If all rights are reserved, many may choose not to adopt Alice's fantastic project, even if they really like it, in order not to risk problems and licensing conflicts.

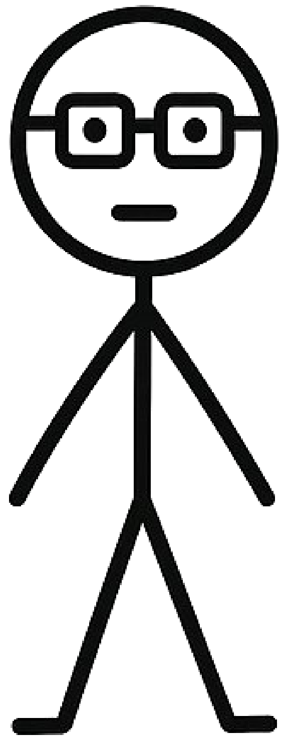


Meet Alice

- This is Alice
- Alice has also been working hard on a project and wants to share it with the world because she truly believes it can be useful to other people.
- Alice knows that if she doesn't indicate one (or more) licenses to the code on github, the default means "All rights reserved".
- If all rights are reserved, many may choose not to adopt Alice's fantastic project, even if they really like it, in order not to risk problems and licensing conflicts.
- Alice would like to license different parts of the code under different, more or less restrictive licenses.
- Alice spends countless hours manually marking down every single file with the right license, with the constant fear of having forgotten something or making a mistake in how she is licensing her project.

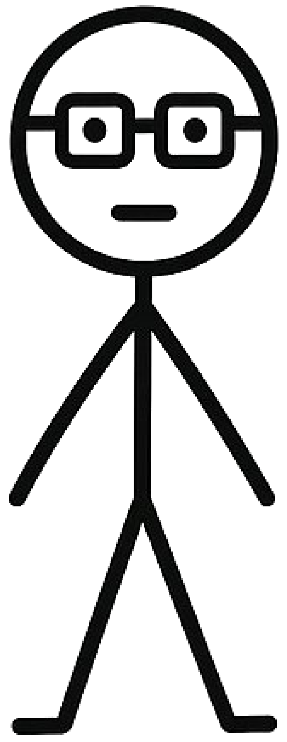
Meet Bob

- This is Bob
- Bob has also been working hard on a project and wants to share it with the world because she truly believes it can be useful to other people.
- Bob knows that if she doesn't indicate one (or more) licenses to the code on github, the default means "All rights reserved". He want to put (many) licenses to solve this issue.



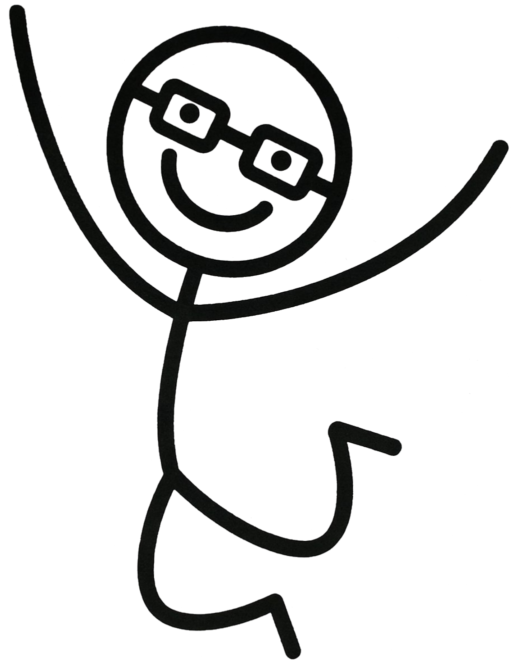
Meet Bob

- This is Bob
- Bob has also been working hard on a project and wants to share it with the world because she truly believes it can be useful to other people.
- Bob knows that if she doesn't indicate one (or more) licenses to the code on github, the default means "All rights reserved". He want to put (many) licenses to solve this issue.
- Bob knows that tools like `reuse` exists.



Meet Bob

- This is Bob
- Bob has also been working hard on a project and wants to share it with the world because she truly believes it can be useful to other people.
- Bob knows that if she doesn't indicate one (or more) licenses to the code on github, the default means "All rights reserved". He want to put (many) licenses to solve this issue.
- Bob knows that tools like `reuse` exists.
- Bob releases all his code with reuse, and is confident that everything is standards compliant.
- Bob's code is really awesome, many companies start using it and Bob gets hired by some big tech to maintain his project.



Reuse

Reuse is a tool and specification developed by the Free Software Foundation Europe (FSFE) that helps manage copyright and licensing information in software projects. It provides a standardized way to declare licenses for each file in a project using SPDX[1] identifiers and copyright notices, making it easier to ensure compliance and clarity about licensing terms across the entire codebase.

How to install? → It is a python package:

```
$  pip install reuse
```

1. SPDX identifiers are short, unique, standardized codes that identify software licenses in a clear, machine-readable manner. E.g., `MIT`, `GPL-3.0-or-later`, ...

Reuse in a nutshell

```
$ reuse download  <LICENSE>
```
[2]

This will download the license and place it in a LICENSES folder. Remember to always include this folder in every repository you publish.

```
$  reuse annotate  --license <LICENSE> \
     --copyright “<NAME SURNAME>  <EMAIL>”  <FILE(S)>
```

This will add some standardized-formatted comments to properly indicate the license the given file.

```
$ reuse lint
```

Check that everything is ok, and indicate if there are files that have not yet been released.

2. List of the possible licenses: https://spdx.org/licenses/

reuse live demo