exact

Giuseppe Piero Brandino

Documentation

Outline

- Why documenting?
- Documentation guidelines
- Documentation example
  - Compiled code: Doxygen and the format
  - Python: docstrings
- Docs as code

Why documenting?

- Documentation: written specifications, instructions, logic, proof of concept of source code so it is readable by:
  - humans
  - machines

Why documenting?

- A contract between callers and implementors.
  - We continuously stumble upon it!
  - It may be annoying.
  - We argue about it!

Why documenting?

- Running out of time!
- Not my responsibility..
- The others do not document. Why should I do it?
- I remember what I am coding!
- We are not Google! (or Microsoft, Pivotal etc.)
- Who is going to read it?
- If I do not document, I shall be irreplaceable.

Why documenting?

- Reminds you of what you code
- Helps in IDE autocompletion functionality.
- Creates a pleasant mood to your code editor / reviewer / colleague.
- Reduces time spent on helping colleagues understand your code. – Res ipsa loquitir (the thing itself speaks)

Why documenting?

- Advertises your work (and consequently yourself)!
- Makes your product more competitive to other products.
- Makes your code suitable for contributing in Open Source projects.

Documentation guidelines

- Documentation is part of the code!
- Document while writing
- Self-documenting code
  - Variable, method, class, package etc. names must be meaningful
  - For very complex code sections, include short comment lines
- Each function/class should come with
  - a description of what it does
  - the descrption and type of its arguments and return value

Documentation guidelines

- Documentation is a live process. Update or produce new version of it until the end of development process
- Documentation is the story of your code
- Documentation reminds you that you write code first for humans, then for machines

Doxygen

Documentation Generator tools

- Tons of Languages
- Relationships
- Diagrams

http://www.doxygen.nl/

Doxygen

`doxygen -g`

Important variables

- PROJECT_NAME
- PROJECT_BRIEF
- OUTPUT_DIRECTORY
- GENERATE_HTML

Doxygen – methods

```
/** Registers the text to display in a tool tip.
 *   The text displays when the cursor lingers
 *   over the component.
 *  \param char* text the string to display.
 *  \return int      return 0 on success
 */
int setToolTipText(char* text) {
    // your code
    return 0;
    }
```

Doxygen – files

```
/** \file awesome_module.h
*
*
*  \authors B.Gates
*  \copyright 2019 eXact-lab s.r.l. GPLv2
*/
<your code here>
```

Doxygen – mainpage

```
/**
 * \mainpage My wonderful program
*  \section intro_sec Introduction
*
*  \authors B.Gates
*  \copyright 2019 eXact-lab s.r.l. GPLv2
*/
<your code here>
```

Running doxygen

- doxygen

  or
- doxygen <configuration.file>

- Produces an html version and a latex version
- Latex version currently broken on some linux distributions... (November 2019)
- Example http://www.netlib.org/lapack/explore-html/

Python - docstrings

- Documentation string which is string literal, and it occurs in the class, module, function or method definition,
- Docstrings are accessible from the *doc* attribute for any of the Python object
- Accessible also with the built-in help() function can come in handy

Python - docstrings

```
def some_function(argument1):
    """Summary or Description of the Function
    Parameters:
       argument1 (int): Description of arg1
    Returns:
       int:Returning value
   """

    return argument1
```

```
print(some_function.__doc__)
help(some_function)
```

```
Summary or Description of the Function
    Parameters:
    argument1 (int): Description of arg1
    Returns:
    int:Returning value
```

docstrings and doxygen

- Doxygen is able to read docstrings, and generate documentation from them
- The rendering is not as fancy as using doxygen tags
- Alternatively, you can use standard doxygen stynthax, but you loose the *help()* and *.__doc__*

Docs as code

Writing, testing, publishing, and maintaining documentation using the same tools developers use for software code

Goals	Tools
❑ Content authoring	→ Text editors
❑ Formatting, styling	→ Markup languages
❑ Version control	→ Git, SVN,...
❑ Issue tracking	→ JIRA, BugZilla,...
❑ Testing, validation	→ Scripts, linters, spell-checks,...
❑ Publishing	→ Site builders, CMS, ...