

# 18 – API-documentation

Bálint Aradi

**Scientific Programming in Python (2024)**

<https://atticlectures.net/scipro/python-2024/>

# Prerequisites

- Sphinx (documentation system)
- Sphinx-book-theme (documentation style template)
- make (build tool)

```
conda install sphinx sphinx-book-theme make
```

## Application Programming Interface (API)

- All **public routines** of your project
- Called by other projects by importing modules from your project (reusability!)

## API-documentation

- Description of the **purpose** and **input/output** arguments of the API
- In Python the module/function **doc-strings** should be used to contain the API-documentation

## Extracting API-documentation

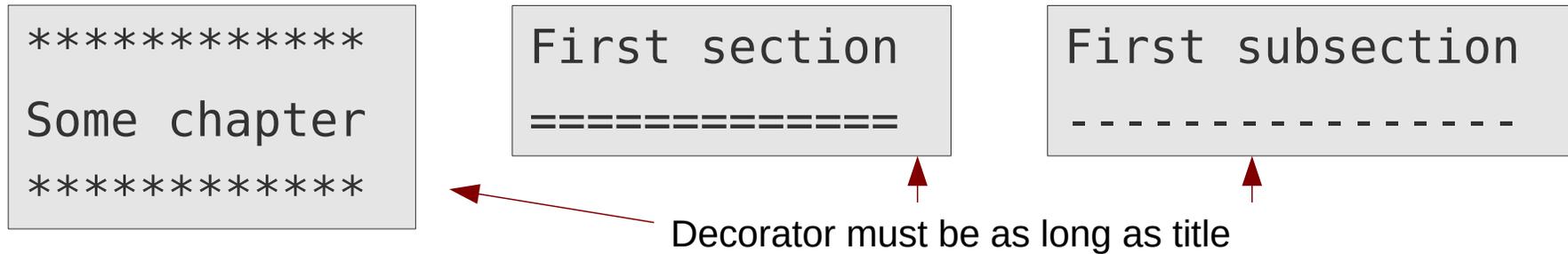
- Documentation is **extracted from** the **source** code
- Generated documentation independent from source code (e.g. HTML-pages)
- Modules can be (re)used without knowing the internal code details

## Sphinx documentation system

- Suitable for simple **code related documents** (e.g. user manual, reference manual, etc.)
- Can be used to **extract API-documentation from doc-strings**
- De-facto **standard tool** in the Python-world (all documentation on python.org is written using Sphinx)
- It uses the **reStructured Text** (RST) format

# ReStructured Text in a nutshell (1)

- HTML/TeX-like formatting language using mostly picturesque notation



```
This is emphasized (italic) and bold.
```

```
Here we use a TeX equation:  $E = mc^2$ 
```

```
Bulleted list:
```

```
* First bullet item
```

```
* Second bullet item
```

```
Enumerated list:
```

```
1. First enumerated item
```

```
2. Second enumerated item
```

## ReStructured Text in a nutshell (2)

- Similar to Python, **indentation** is part of the ReST-language semantics

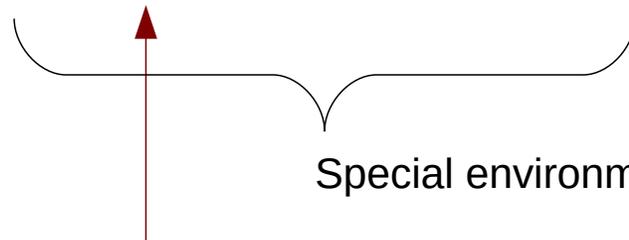
```
.. toctree::  
   :maxdepth: 2  
  
   api
```

Watch out for  
correct  
indentation!

```
We include a code example::
```

```
print("Hello, World!")
```

```
Snippet above will be rendered as code
```



Includes **api.rst** into the document and lists its sections in the toc

- Read the documentation for all available feature of ReST (very powerful)

### See also

- [Quick reStructuredText](#)
- [The reStructuredText Cheat Sheet](#)
- [A ReStructuredText Primer](#)

# Extracting API documentation

- Create a subfolder **docs/** in the project directory
- Set up a sphinx documentation project in it
- Edit generated **conf.py** file

```
mkdir docs
cd docs
sphinx-quickstart
```

Take default value  
wherever possible

```
from pathlib import Path
import sys
sys.path.insert(0, str(Path.cwd().parent))
```

List of folders to look up  
for Python modules

Ensures that sphinx finds Python  
module files in parent folder when  
extracting API-documentation

```
extensions = [
    'sphinx.ext.autodoc',
    'sphinx.ext.napoleon',
    'sphinx.ext.mathjax',
]
```

Automated API-extraction

Doc-strings in Google/Numpy-format

Render TeX in HTML with MathJax

Activate some  
extensions

```
html_theme = 'sphinx_book_theme'
```

Use the more modern [sphinx-book-theme](#)

# Extracting API documentation

- Edit generated file **index.rst** and create new file **api.rst** in the docs folder:

```
##### index.rst
Linsolver
#####

.. toctree::
   :maxdepth: 2

   api
```

```
***** api.rst
Linsolver API
*****

.. automodule:: solvers
   :members:
```

 Generates automatic documentation for all members of the solvers module

- Extract documentation and convert to HTML-format

```
make html Linux
```

```
./make.bat html Windows
```

Build finished. The HTML pages are in `_build/html`.

# Visualizing API documentation

- Open the `_build/index.html` file in a web-browser

```
firefox _build/html/index.html Linux
```

Welcome to solvers's documentation!

Contents:

[API-documentation](#)

## API-documentation

Routines for solving a linear system of equations.

```
solvers.backward_substitute(lu: ndarray[Any, dtype[float64]], bb: ndarray[Any, dtype[float64]]) → ndarray[Any, dtype[float64]]
```

Solves  $Ux = b$  for an upper triangle matrix via backward substitution.

### Parameters:

- **lu** – LU-decomposed matrix (as returned by `lu_decompose()`) containig upper triangle matrix U.
- **bb** – Right hand side of the equation.

### Returns:

The solution vector  $x$ .

```
solvers.forward_substitute(lu: ndarray[Any, dtype[float64]], bb: ndarray[Any, dtype[float64]]) → ndarray[Any, dtype[float64]]
```

Solves  $Ly = b$  for a lower triangle matrix via forward substitution.

### Parameters:

- **lu** – LU-decomposed matrix (as returned by `lu_decompose()`) containig lower triangle matrix L.
- **bb** – Right hand side of the equation.

### Returns:

The solution vector  $y$ .

# Some Sphinx-notes

- Sphinx is optimal for **small and middle size** documents, where type setting is not too complicated
- Sphinx has several output format beside html (LaTeX, PDF, etc.)
- Put the Sphinx source and configuration files of your project under **version control**, but **not the `_build` folder**

```
cd docs
git add api.rst conf.py index.rst make.bat Makefile _static/
_templates/
```

- Add the Sphinx build folder to the projects **.gitignore** file

```
__pycache__          .gitignore
docs/_build
```