OAK RIDGE National Laboratory

Best practices for documenting a scientific Python package

US-RSE conference in Chicago, IL

October 16-18, 2023

Gavin Wiggins, Gregory Cage, Robert Smith, Seth Hitefield, Marshall McDonnell, Lance Drane, Jesse McGaha, Michael Brim, Mark Abraham, Richard Archibald, and Addi Malviya-Thakur

ORNL is managed by UT-Battelle LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Utilize style guides, linting, and formatting tools

- Style guides, linters, and formatters help developers maintain a coherent code base that can be easily documented
- **PEP 8** style guide that defines naming conventions, line length, indentation, etc. for Python code quality and readability
- Linters such as **flake8** enforce coding styles defined by PEP 8 and other best practices like warning about unused imports
- Formatters such as **black** ensure consistent code formatting throughout the project
- Linter and formatter tools can be implemented in continuous integration (CI) workflows to enforce well written code

Use docstrings to document Python code

- Docstrings are comments in Python code that help users and developers with documentation
- Common docstring styles are the NumPy and Google styles

```python
def add_numbers(x, y):
    """Add two numbers.

    Parameters
    ----------
    x : float
        The first number.
    y : float
        The second number.

    Returns
    -------
    float
        The result of adding two numbers.
    """
    result = x + y

    return result
```

NumPy docstring style

```python
def add_numbers(x, y):
    """Add two numbers.

    Args:
        x (float): The first number.
        y (float): The second number.

    Returns:
        float: The result of adding two numbers.
    """
    result = x + y

    return result
```

Google docstring style

Generate documentation with Sphinx

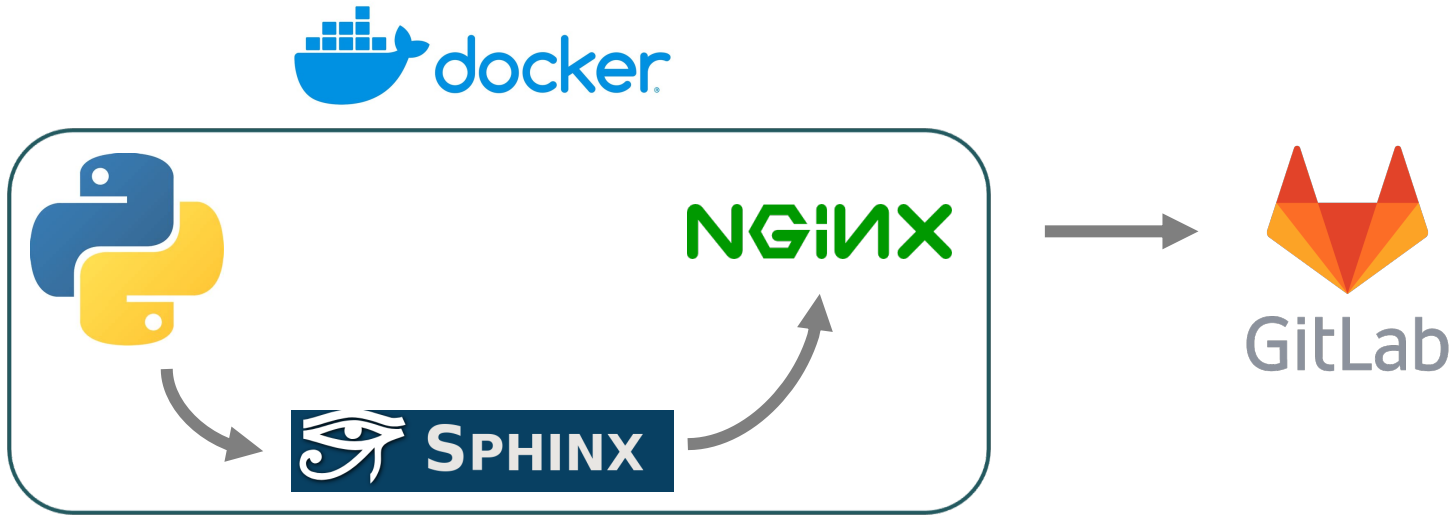
- **Sphinx** generates documentation for Python projects using docstrings and reStructuredText (rst) files
- Supports Markdown files via extensions
- The generated documentation can be hosted online (HTML) or offline (PDF)
- Renders LaTeX math equations
- Other tools for generating project documentation are Doxygen, pdoc, and MkDocs but Sphinx is preferred for Python projects

Basic structure of a Python project with Sphinx documentation.

```
my_project/
|
|-- docs/
|   |-- index.rst
|   |-- conf.py
|   |-- func.rst
|
|-- src/
|   |-- my_package/
|       |-- __init__.py
|       |-- func.py
|
|-- tests/
|-- examples/
|-- pyproject.toml
|-- LICENSE.md
|-- README.md
```

Deploy documentation for users

- **Read the Docs** provides free documentation hosting for open-source projects
- Can also use a **Docker** container to build the HTML documentation with Sphinx and host it with Nginx web server
- Continuous deployment can be achieved via **GitLab** container registry and pipeline triggers



More information about these tools and services

PEP 8 style guide	https://peps.python.org/pep-0008
NumPy docstrings	https://numpydoc.readthedocs.io/en/latest/format.html
Google docstrings	https://google.github.io/styleguide/pyguide.html
flake8 linter	https://github.com/pycqa/flake8
black formatter	https://github.com/psf/black
Sphinx documentation	https://www.sphinx-doc.org
Read the Docs hosting	https://about.readthedocs.com
Docker containers	https://www.docker.com
GitLab platform	https://about.gitlab.com