developer.skatelescope.org Documentation

Release 0.1.0-beta

Marco Bartolini

Sep 14, 2023

Home

Documentation Status

Briefly describe your project here

CHAPTER 1

Requirements

The system used for development needs to have Python 3, `pip` and Poetry installed.

CHAPTER 2

Install pip

Always use a virtual environment. Pipenv is now Python's officially recommended method, but we are not using it for installing requirements when building on the CI Pipeline. You are encouraged to use your preferred environment isolation (i.e. `pip`, `conda` or `pipenv` while developing locally.

For working with `Pipenv`, follow these steps at the project root:

First, ensure that `~/.local/bin` is in your `PATH` with:

```
> echo $PATH
```

In case `~/.local/bin` is not part of your `PATH` variable, under Linux add it with:

```
> export PATH=~/.local/bin:$PATH
```

or the equivalent in your particular OS.

Then proceed to install pipenv and the required environment packages:

```
> pip install pipenv # if you don't have pipenv already installed on your system
> pipenv install
> pipenv shell
```

You will now be inside a pipenv shell with your virtual environment ready.

Use `exit` to exit the pipenv environment.

CHAPTER 3

Install Poetry

First we need to install Poetry: curl -sSL https://raw.githubusercontent.com/python-poetry/poetry/master/get-poetry.py | python3 - The get-poetry.py script described here will be replaced in Poetry 1.2 by install-poetry.py. From Poetry 1.1.7 onwards, you can already use this script as described as below: curl -sSL https://install.python-poetry.org | python3 - To manage application dependencies Poetry supports pyproject.toml config file. This .toml file have three sections:

- [tool.poetry] — fields that describe our application, some of them are required,
- [tool.poetry.dependencies] —a list of all the required packages with version numbers,
- [tool.poetry.dev-dependencies] — a list of the required packages for development purposes: pytest for running unit tests, black for code linting and mypy for static type check.To install all those dependencies simply run: "poetry install". The dependencies will be installed to the virtual environment created and managed by Poetry by creating poetry.lock file which will resolve and install all the dependencies that are listed in pyproject.toml file. In this way, Poetry handles both the dependencies of our application in one go.

CHAPTER 4

Testing

- Put tests into the `tests` folder
- Use PyTest as the testing framework
  - Reference: PyTest introduction
- Run tests with `python setup.py test`
  - Configure PyTest in `setup.py` and `setup.cfg`
- Running the test creates the `htmlcov` folder
  - Inside this folder a rundown of the issues found will be accessible using the `index.html` file
- All the tests should pass before merging the code

CHAPTER 5

Code analysis

- Use Pylint as the code analysis framework
- By default it uses the PEP8 style guide
- Use the provided `code-analysis.sh` script in order to run the code analysis in the `module` and `tests`
- Code analysis should be run by calling `pylint ska_python_skeleton`. All pertaining options reside under the `.pylintrc` file.
- Code analysis should only raise document related warnings (i.e. `#FIXME` comments) before merging the code

CHAPTER 6

Writing documentation

- The documentation generator for this project is derived from SKA's SKA Developer Portal repository
- The documentation can be edited under `./docs/src`
- If you want to include only your README.md file, create a symbolic link inside the `./docs/src` directory if the existing one does not work:

```
$ cd docs/src
$ ln -s ../../README.md README.md
```

- In order to build the documentation for this specific project, execute the following under `./docs`:

```
$ make html
```

- The documentation can then be consulted by opening the file `./docs/build/html/index.html`

CHAPTER 7

Development

7.1 PyCharm

As this project uses a `src` folder structure, under *Preferences* > *Project Structure*, the `src` folder needs to be marked as "Sources". That will allow the interpreter to be aware of the package from folders like `tests` that are outside of `src`. When adding Run/Debug configurations, make sure "Add content roots to PYTHONPATH" and "Add source roots to PYTHONPATH" are checked.

Todo:

- Insert todo's here

CHAPTER 8

Package-name documentation

This section describes requirements and guidelines.

8.1 Subtitle

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

8.1.1 Public API Documentation

Functions

Classes

CHAPTER 9

Project-name documentation HEADING

These are all the packages, functions and scripts that form part of the project.