Subhead

Python packaging, Unit Tests and Documentation

Hello Packaging World!

20240313 – Sven Giese

Ground Rules for Today

Template available here: https://github.com/gieses/aichemist-copier-template

- Highly biased
- Highly selective = incomplete
- Let’s all appreciate that ”scientific programming” & good engineering is **hard**

"The next painting you do is going to be a little bit better, you're going to learn a little more."

Bob Ross

Agenda

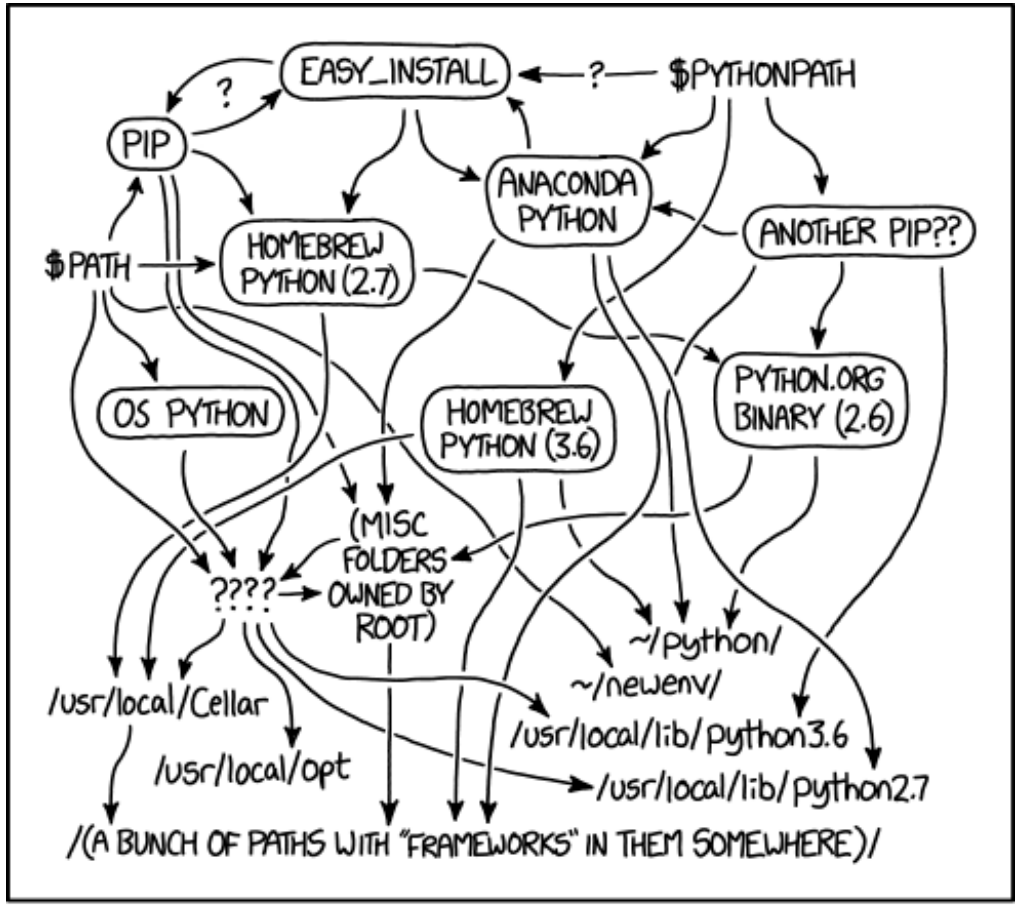
What does it take to build a good practice python package

- **Packaging**
  - => make your code reusable, more structured, installable
  - environment management
  - => provide means to your user, peers AND YOURSELF to reproduce your environment

- **Testing**
  - => allow yourself to develop code without worrying too much about breaking something

- **Documentation** (code & package docs)
  - => let your IDE help you writing code and easily generate beautiful docs

Python Packaging overlaps with (awful) environment handling

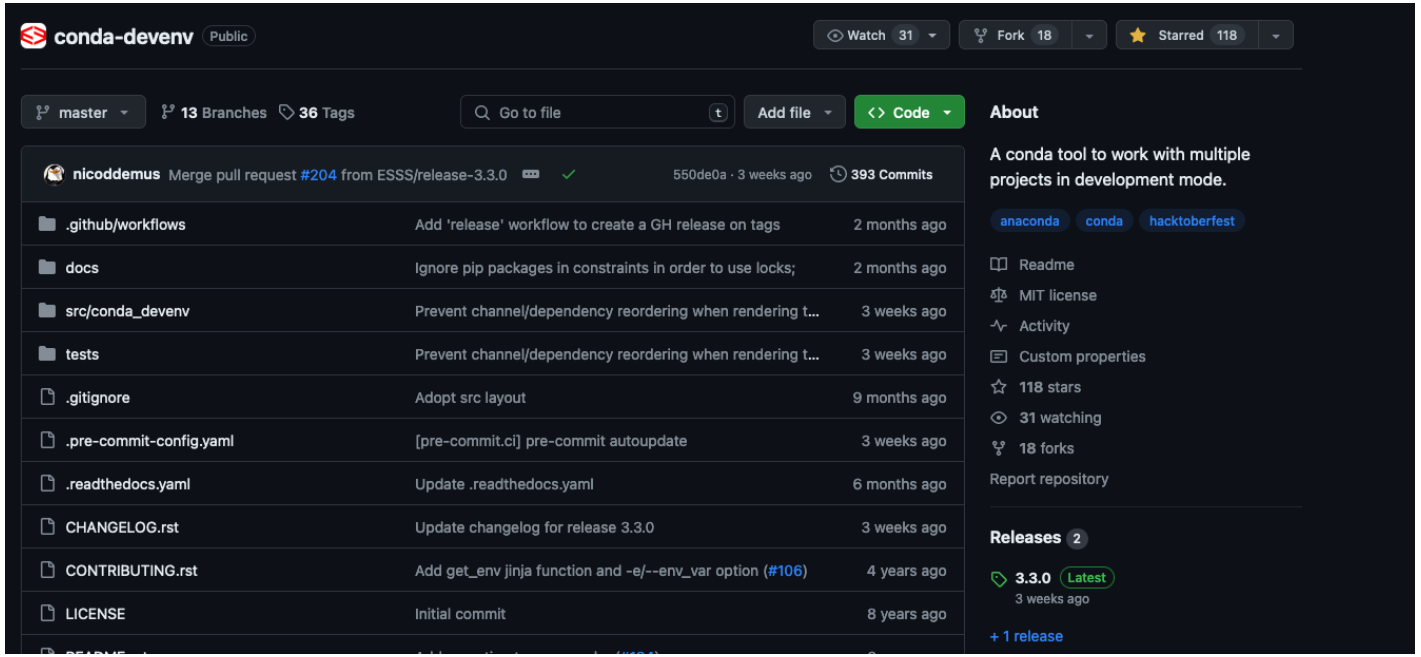


MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Environment Handling

We use:

- Conda / mamba
- Conda-forge
- Conda devenv
  - => handle modular environments



https://github.com/ESSS/conda-devenv

Here's a simple `environment.devenv.yml` file:

```
{% set conda_py = os.environ.get('CONDA_PY', '35') %}
name: web-ui-py{{ conda_py }}

includes:
  - {{ root }}/../core-business/environment.devenv.yml

dependencies:
  - gcc  # [linux]

environment:
  PYTHONPATH:
    - {{ root }}/src
  STAGE: DEVELOPMENT
```

Python Packaging Goal:

Motivation (be selfish):

- Make it easy to install *your* app / model
- Make it easy to run *your* code
- Reduce mental load when working with your code
  - Where is my module
  - Am I working locally or on the server?
  - Sys.append is wrong?

Side-Effects:

- Relatively simple to make code available via pypi, conda afterwards
- Share code with co-workers
- Re-use code across projects
- Easy entry points to use notebooks (for people that like notebooks..)

https://pypi.org/project/scikit-learn/

Python Packaging: Ingredients

- Python Package
  - Layout
  - Config (pyproject.toml!)
- Templating (for data science)
  - copier
- Environment management
  - conda

```
packaging_tutorial/
├── LICENSE
├── pyproject.toml
├── README.md
├── src/
│   └── example_package
│       ├── __init__.py
│       └── example.py
└── tests/
```

Src vs. Flat- Layout

```
packaging_tutorial/
├── LICENSE
├── pyproject.toml
├── README.md
├── src/
│   └── example_package
│       ├── __init__.py
│       └── example.py
└── tests/
```

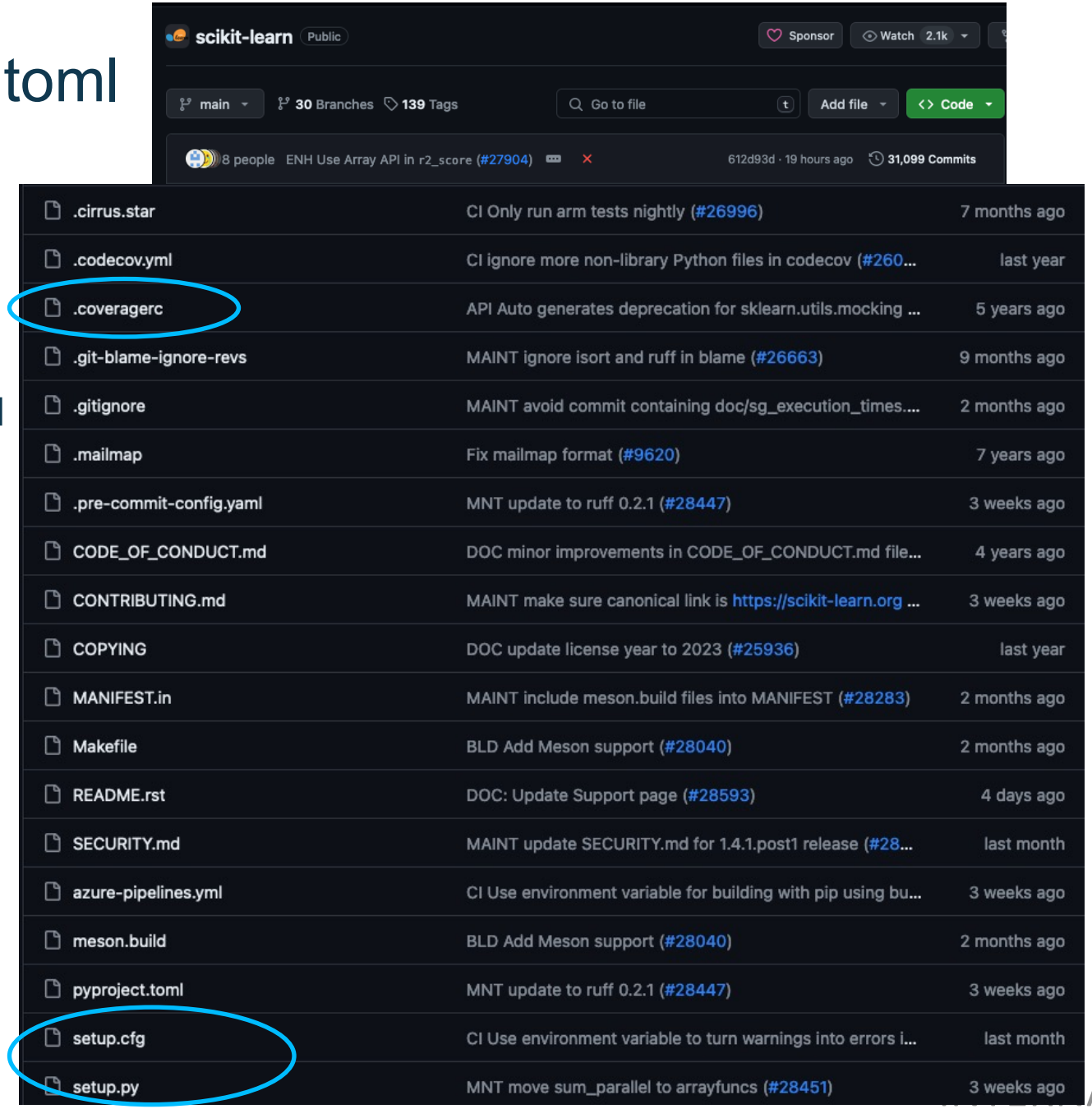
```
packaging_tutorial/
├── LICENSE
├── pyproject.toml
├── README.md
├── example_package
│   ├── __init__.py
│   └── example.py
│
└── tests/
```

- I *used to* prefer the flat-layout but it has some caveats
  - „sometimes“ – „odd“ behavior (namespace packages, setuptools scm)
  - possibly hard to debug errors in –e installs (project's root added to sys.path)
- Practically, in many circumstances negligible

https://blog.ionelmc.ro/2014/05/25/python-packaging/

Packaging Config: pyproject.toml

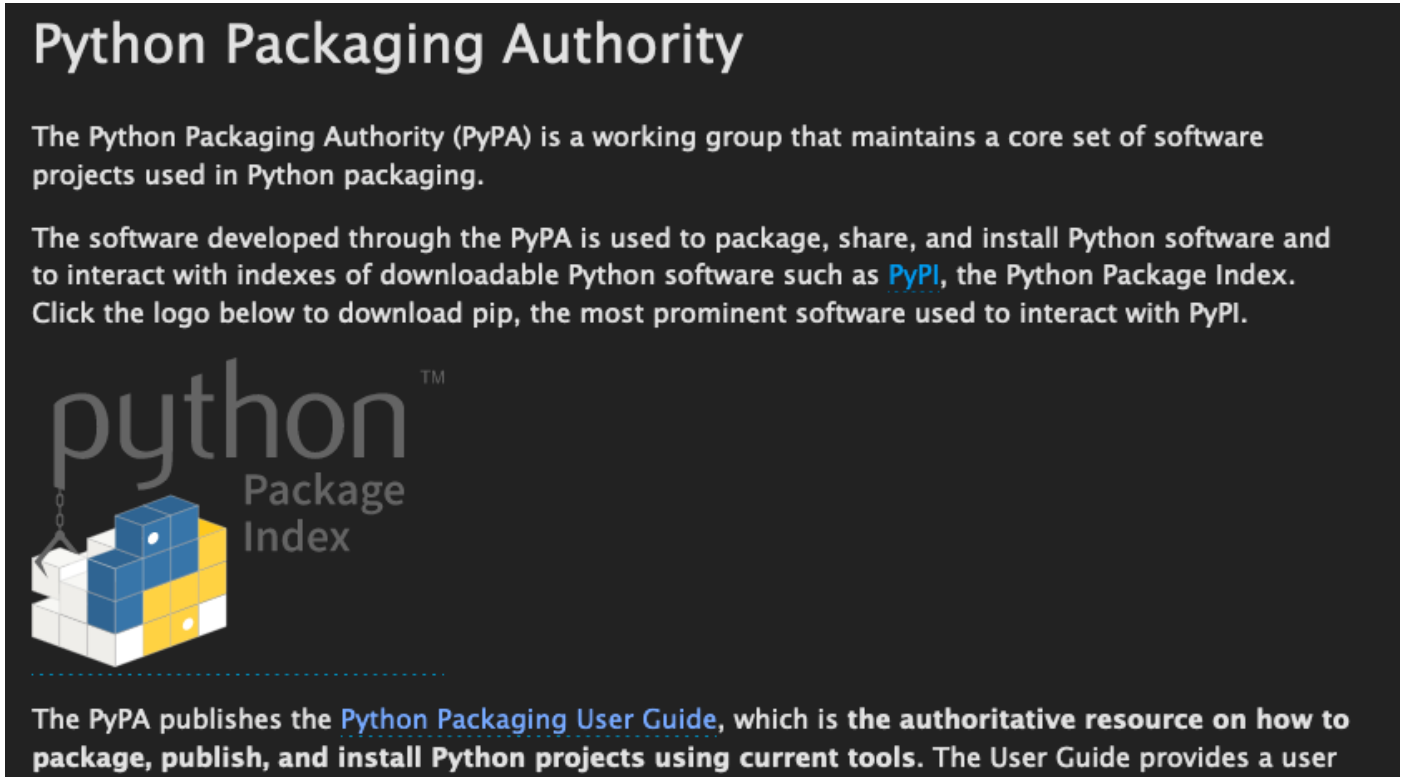
- PEP 518 – Build System Requirements
- Kind of new / growing
  - Replace the various config files needed
- Integrates with:
  - Setup-tools
  - Poetry
  - pdm
  - Testing / Ruff ecosystem
- Can almost entirely replace setup.py



Python Package

Readme:

- https://www.pypa.io/en/latest/
- https://the-hitchhikers-guide-to-packaging.readthedocs.io/en/latest/history.html



Tooling

- Copier (https://copier.readthedocs.io/en/stable/) = template engine
- Pytest (https://docs.pytest.org/en/8.0.x/)
- Ruff (https://pypi.org/project/pytest-ruff/)
- Makefiles (https://makefiletutorial.com/)
- Conda- / Mamba-forge (https://mamba.readthedocs.io/en/latest/installation/mamba-installation.html)
- IDE: PyCharm

Copier / Cookiecutter Components

Jinja templating to create standardize python package configurations

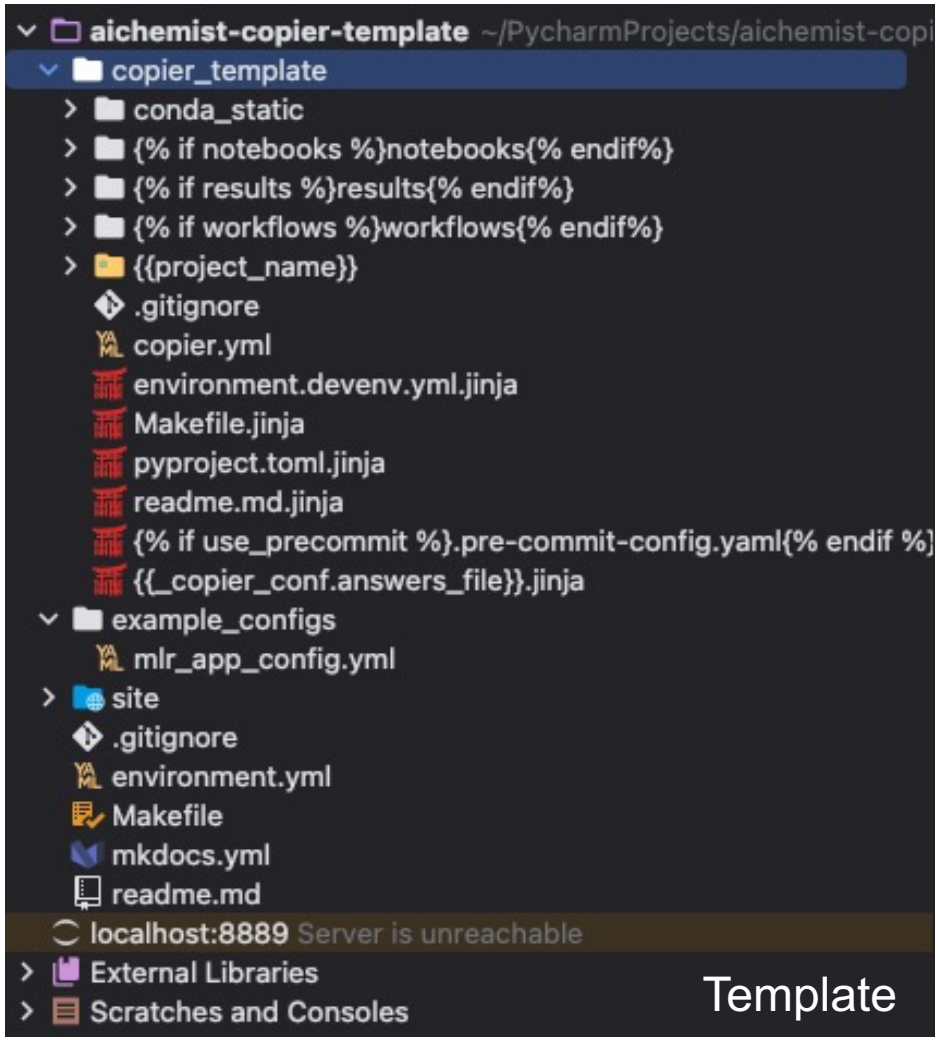
```
(base) ~/P/aichemist-copier-template ❯❯❯ copier copy copier_template aichemist --trust
🎤 What is your project name?
   aichemist
🎤 What is your preferred conda environment name?
   aichemist_env
🎤 What is your Python module name?
   example
🎤 What is your name?
   Sven
🎤 What is your email?
```

Q & A Config

environment.yml | copier.yml ×

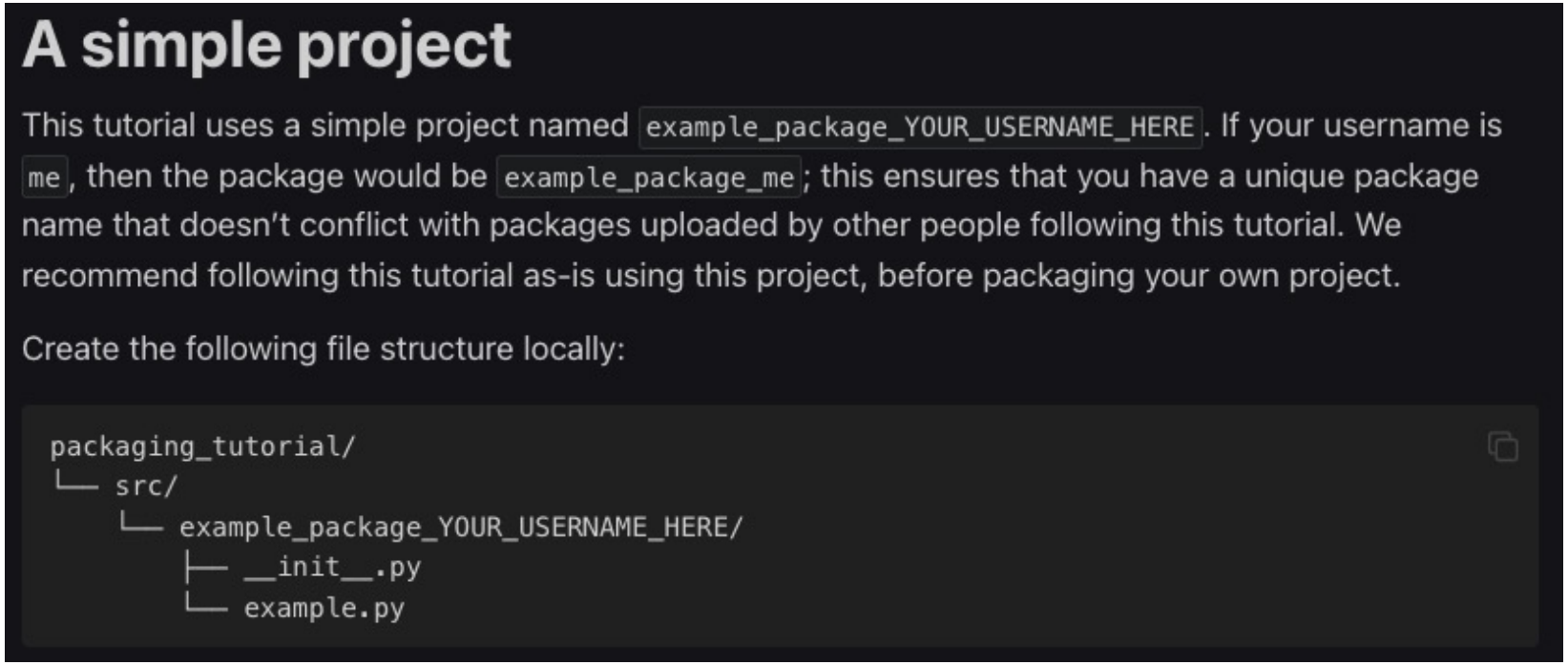
```yaml
# questions
project_name:
    type: str
    help: What is your project name?

environment_name:
    type: str
    help: What is your preferred conda environment name?

module_name: <2 keys>

author: <2 keys>
```

Define Q & A



Template

How to get started?



* https://packaging.python.org/en/latest/tutorials/packaging-projects/

Agenda

Unit Tests

Motivation (be selfish):

- Make sure your code does what it is supposed to do
- Lower maintenance because changes can be easier verified

Side-Effects:

- Write better structured code
  - “no-side effects” / “single responsibility principle”
- Become a better software engineer
- Easily debug small parts of your pipeline
- Look! My code has tests! (=> quality assumption)

Disclaimer: I am not familiar with alternatives to test driven development. However, there are some! In the most simplest case, one can go away from function-based testing to ingeration testing. Buzzwords: **Acceptance Test-Driven Development (ATDD), Integration Testing, Exploratory Testing, …**

Practical: Writing tests in Pytest

- Many flavours
  - “standard” tests for functions and classes
  - Extensions
    - Linting (PEP8, pyflakes, etc.)
    - Typing (mypy, Pyright)
    - Formatting (black)
    - Specific conventions
      - Numpy / pandas / …
  - Ruff, > 700 rules (expanding)
  - Machine learning?

https://docs.astral.sh/ruff/rules/

pep8-naming (N)

For more, see pep8-naming on PyPI.

Code	Name	Message
N801	invalid-class-name	Class name `{name}` should use CapWords convention
N802	invalid-function-name	Function name `{name}` should be lowercase
N803	invalid-argument-name	Argument name `{name}` should be lowercase
N804	invalid-first-argument-name-for-class-method	First argument of a class method should be named `cls`
N805	invalid-first-argument-name-for-method	First argument of a method should be named `self`
N806	non-lowercase-variable-in-function	Variable `{name}` in function should be lowercase
N807	dunder-function-name	Function name should not start and end with `__`
N811	constant-imported-as-non-constant	Constant `{name}` imported as non-constant `{asname}`
N812	lowercase-imported-as-non-lowercase	Lowercase `{name}` imported as non-lowercase `{asname}`
N813	camelcase-imported-as-lowercase	Camelcase `{name}` imported as lowercase `{asname}`

pandas-vet (PD)

For more, see pandas-vet on PyPI.

Code	Name	Message
PD002	pandas-use-of-inplace-argument	`inplace=True` should be avoided; it has inconsistent behavior
PD003	pandas-use-of-dot-is-null	`.isna` is preferred to `.isnull`; functionality is equivalent

Example:

Examples

```
import os

os.path.join(os.path.join(ROOT_PATH, "folder"), "file.py")
```

Use instead:

```
from pathlib import Path

Path(ROOT_PATH) / "folder" / "file.py"
```

Why is this bad?

`pathlib` offers a high-level API for path manipulation, as compared to the lower-level API offered by `os`. When possible, using `Path` object methods such as `Path.joinpath()` or the `/` operator can improve readability over the `os` module's counterparts (e.g., `os.path.join()`).

https://docs.astral.sh/ruff/rules/os-path-join/

Make use of the brilliant python eco system and learn from them!

Simple Testing (fake data and classification outcome) vs. Specific Behavior (fake data for feature importance)

```python
from sklearn.utils.fixes import COO_CONTAINERS, CSC_CONTAINERS, CSR_CONTAINERS
from sklearn.utils.multiclass import type_of_target
from sklearn.utils.parallel import Parallel
from sklearn.utils.validation import check_random_state

# toy sample
X = [[-2, -1], [-1, -1], [-1, -2], [1, 1], [1, 2], [2, 1]]
y = [-1, -1, -1, 1, 1, 1]
T = [[-1, -1], [2, 2], [3, 2]]
true_result = [-1, 1, 1]
```

```python
@pytest.mark.parametrize("name", FOREST_CLASSIFIERS)
def test_classification_toy(name):
    """Check classification on a toy dataset."""
    ForestClassifier = FOREST_CLASSIFIERS[name]

    clf = ForestClassifier(n_estimators=10, random_state=1)
    clf.fit(X, y)
    assert_array_equal(clf.predict(T), true_result)
    assert 10 == len(clf)

    clf = ForestClassifier(n_estimators=10, max_features=1, random_state=1)
    clf.fit(X, y)
    assert_array_equal(clf.predict(T), true_result)
    assert 10 == len(clf)

    # also test apply
    leaf_indices = clf.apply(X)
    assert leaf_indices.shape == (len(X), clf.n_estimators)
```

```python
# Larger classification sample used for testing feature importances
X_large, y_large = datasets.make_classification(
    n_samples=500,
    n_features=10,
    n_informative=3,
    n_redundant=0,
    n_repeated=0,
    shuffle=False,
    random_state=0,
)
```

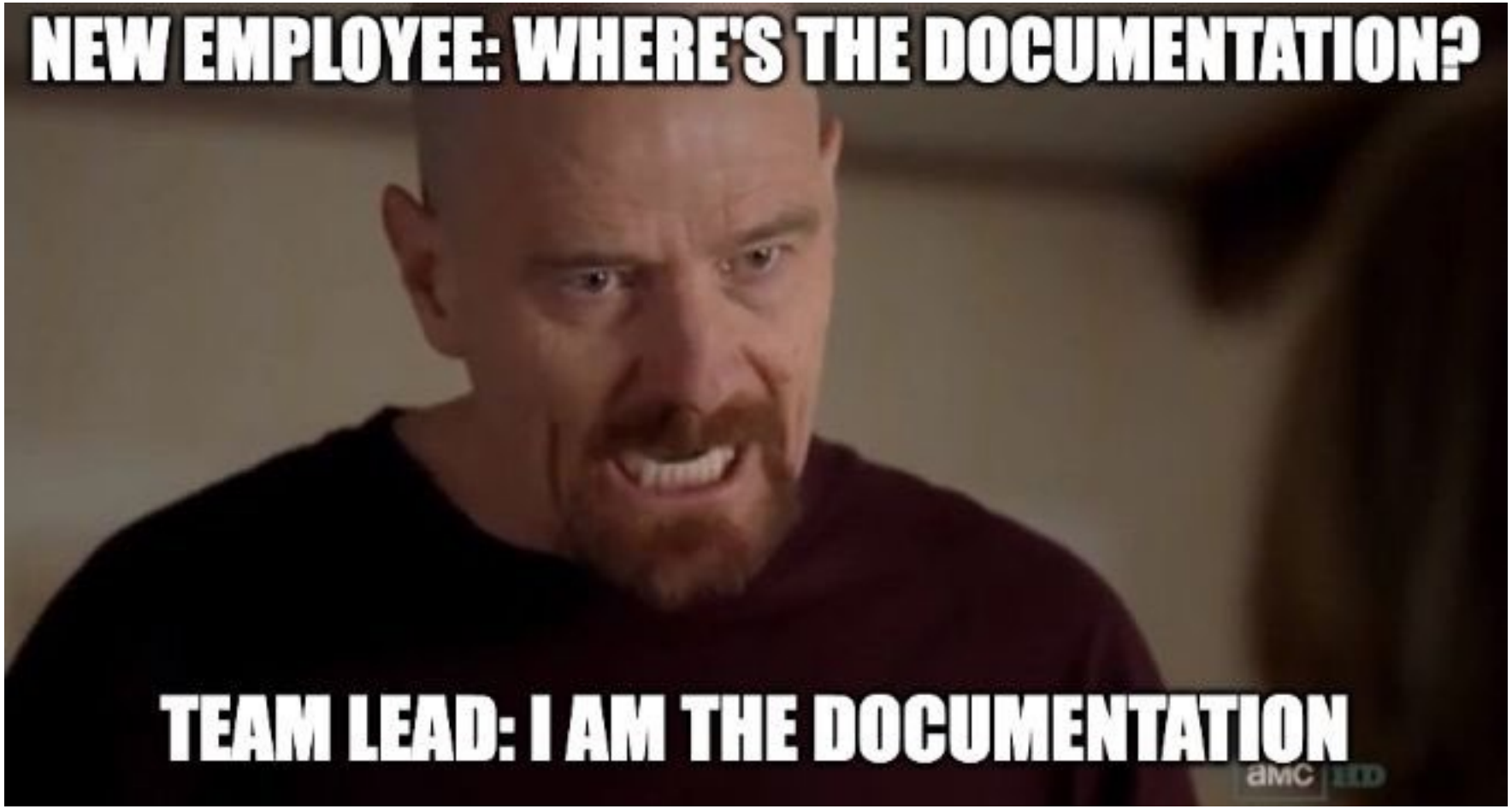
```python
def test_importances(dtype, name, criterion):
    tolerance = 0.01
    if name in FOREST_REGRESSORS and criterion == "absolute_error":
        tolerance = 0.05

    # cast as dtype
    X = X_large.astype(dtype, copy=False)
    y = y_large.astype(dtype, copy=False)

    ForestEstimator = FOREST_ESTIMATORS[name]

    est = ForestEstimator(n_estimators=10, criterion=criterion, random_state=0)
    est.fit(X, y)
    importances = est.feature_importances_

    # The forest estimator can detect that only the first 3 features of the
    # dataset are informative:
    n_important = np.sum(importances > 0.1)
    assert importances.shape[0] == 10
    assert n_important == 3
    assert np.all(importances[:3] > 0.1)
```

https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/ensemble/tests/test_forest.py

https://medium.com/analytics-vidhya/testing-ml-code-how-scikit-learn-does-it-97e45180e834

Agenda

Documentation



https://twitter.com/petergyang/status/1575289122364608513

Defining Documentation: scikit-learn

```python
class KMeans(_BaseKMeans):
    """K-Means clustering.

    Read more in the :ref:`User Guide <k_means>`.

    Parameters
    ----------

    n_clusters : int, default=8
        The number of clusters to form as well as the number of
        centroids to generate.

        For an example of how to choose an optimal value for `n_clusters` refer to
        :ref:`sphx_glr_auto_examples_cluster_plot_kmeans_silhouette_analysis.py`.

    init : {'k-means++', 'random'}, callable or array-like of shape \
            (n_clusters, n_features), default='k-means++'
        Method for initialization:

        * 'k-means++' : selects initial cluster centroids using sampling \
            based on an empirical probability distribution of the points' \
            contribution to the overall inertia. This technique speeds up \
            convergence. The algorithm implemented is "greedy k-means++". It \
            differs from the vanilla k-means++ by making several trials at \
            each sampling step and choosing the best centroid among them.

        * 'random': choose `n_clusters` observations (rows) at random from \
        data for the initial centroids.

        * If an array is passed, it should be of shape (n_clusters, n_features)\
        and gives the initial centers.

        * If a callable is passed, it should take arguments X, n_clusters and a\
        random state and return an initialization.

        For an example of how to use the different `init` strategy, see the example
        entitled :ref:`sphx_glr_auto_examples_cluster_plot_kmeans_digits.py`.
```

```
Examples
--------

>>> from sklearn.cluster import
>>> import numpy as np
>>> X = np.array([[1, 2], [1, 4]
...               [10, 2], [10,
>>> kmeans = KMeans(n_clusters=2
>>> kmeans.labels_
array([1, 1, 1, 0, 0, 0], dtype=
>>> kmeans.predict([[0, 0], [12,
array([1, 0], dtype=int32)
>>> kmeans.cluster_centers_
array([[10.,  2.],
       [ 1.,  2.]])
```

1.1. Linear Models

The following are a set of methods intended for regression in which the target value is expected to be a linear combination of the features. In mathematical notation, if $\hat{y}$ is the predicted value.

$$\hat{y}(w, x) = w_0 + w_1 x_1 + \ldots + w_p x_p$$

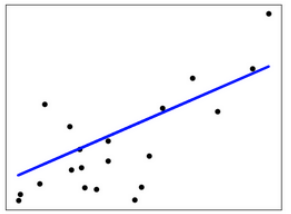
Across the module, we designate the vector $w = (w_1, \ldots, w_p)$ as `coef_` and w_0 as `intercept_`.

To perform classification with generalized linear models, see Logistic regression.

1.1.1. Ordinary Least Squares

`LinearRegression` fits a linear model with coefficients $w = (w_1, \ldots, w_p)$ to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. Mathematically it solves a problem of the form:

$$\min_w ||Xw - y||_2^2$$



Prev | Up | Next

scikit-learn 1.4.1
Other versions

Please cite us if you use the software.

API Reference
- `sklearn`: Settings and information tools
- `sklearn.base`: Base classes and utility functions
- `sklearn.calibration`: Probability Calibration
- `sklearn.cluster`: Clustering
- `sklearn.compose`: Composite Estimators
- `sklearn.covariance`: Covariance

API Reference

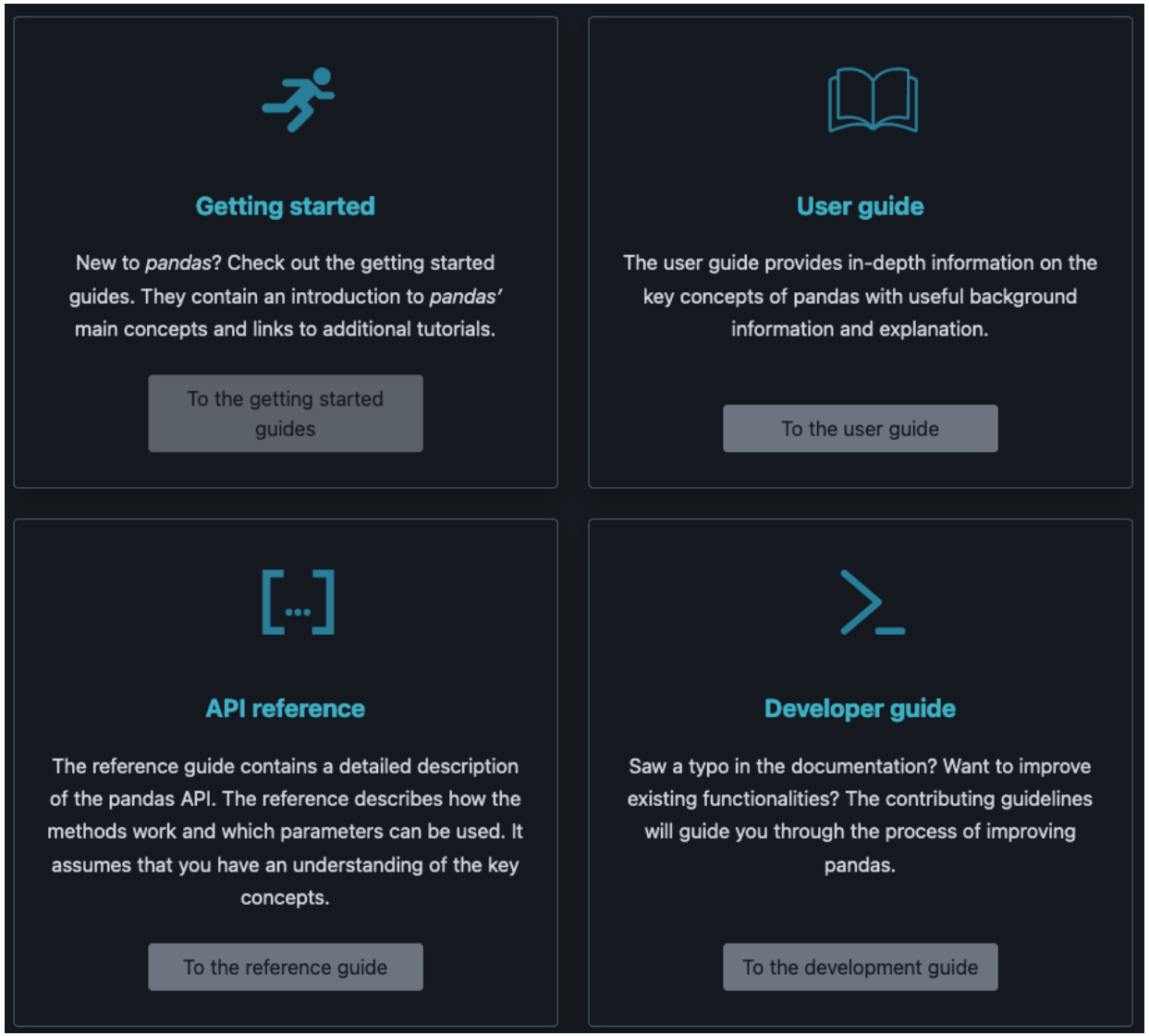
This is the class and function reference of scikit-learn. Please refer to the full user guide for further details, as the class and function raw specifications may not be enough to give full guidelines on their uses. For reference on concepts repeated across the API, see Glossary of Common Terms and API Elements.

`sklearn`: Settings and information tools

The `sklearn` module includes functions to configure global settings and get information about the working environment.

`config_context`(*[, assume_finite, ...])	Context manager for global scikit-learn configuration.
`get_config`()	Retrieve current values for configuration set by `set_config`.
`set_config`([assume_finite, working_memory, ...])	Set global scikit-learn configuration.
`show_versions`()	Print useful debugging information"

Defining Documentation: pandas




```
class DataFrame(NDFrame, OpsMixin):
    """
    Two-dimensional, size-mutable, potentially heterogeneous tabular data.

    Data structure also contains labeled axes (rows and columns).
    Arithmetic operations align on both row and column labels. Can be
    thought of as a dict-like container for Series objects. The primary
    pandas data structure.

    Parameters
    ----------
    data : ndarray (structured or homogeneous), Iterable, dict, or DataFrame
        Dict can contain Series, arrays, constants, dataclass or list-like objects. If
        data is a dict, column order follows insertion-order. If a dict contains Series
        which have an index defined, it is aligned by its index. This alignment also
        occurs if data is a Series or a DataFrame itself. Alignment is done on
        Series/DataFrame inputs.

        If data is a list of dicts, column order follows insertion-order.

    index : Index or array-like
        Index to use for resulting frame. Will default to RangeIndex if
        no indexing information part of input data and no index provided.
```

DataFrame

Constructor

`DataFrame`([data, index, columns, dtype, copy])	Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Attributes and underlying data

Axes

`DataFrame.index`	The index (row labels) of the DataFrame.
`DataFrame.columns`	The column labels of the DataFrame.

Doc String Formats – dont really matter for IDEs but humans =)

Google

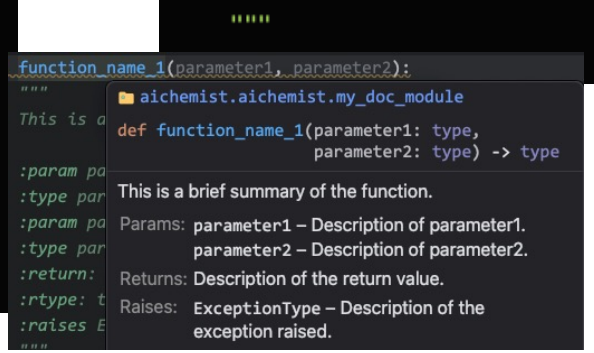
```
def function_name(parameter1, parameter2):
    """
    This is a brief summary of the function.

    Args:
        parameter1 (type): Description of parameter1.
        parameter2 (type): Description of parameter2.

    Returns:
        type: Description of the return value.

    Raises:
        ExceptionType: Description of the exception raised.
    """
    pass
```

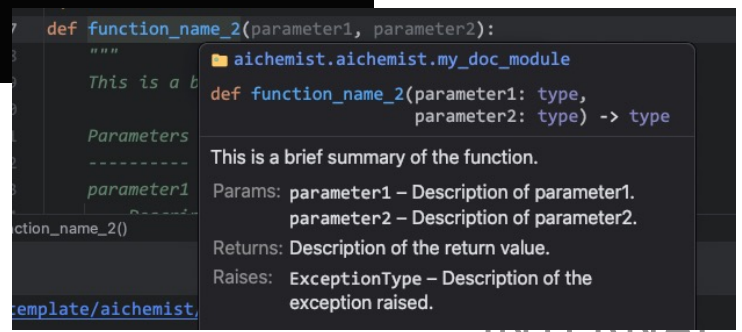
Numpy

```
def function_name(parameter1, parameter2):
    """
    This is a brief summary of the function.

    Parameters
    ----------
    parameter1 : type
        Description of parameter1.
    parameter2 : type
        Description of parameter2.

    Returns
    -------
    type
        Description of the return value.

    Raises
    ------
    ExceptionType
        Description of the exception raised.
    """
```

Sphinx

```
def function_name(parameter1, parameter2):
    """
    This is a brief summary of the function.

    :param parameter1: Description of parameter1.
    :type parameter1: type
    :param parameter2: Description of parameter2.
    :type parameter2: type
    :return: Description of the return value.
    :rtype: type
    :raises ExceptionType: Description of the exception raised.
    """
    pass
```





https://joshdimella.com/blog/python-docstring-formats-best-practices

Mkdocs

- Docs folder
- Mkdocs.yml (config)

Usage:

```
## build the docs using mkdocs
docs:
    mkdocs build

## serve docs locally
docs_serve: docs        ## make & serve the docs locally
    mkdocs serve

## deploy docs to github-pages
docs_deploy:        ## deploy the docs to github-pages
    mkdocs gh-deploy
```

(Makefile shortcuts)

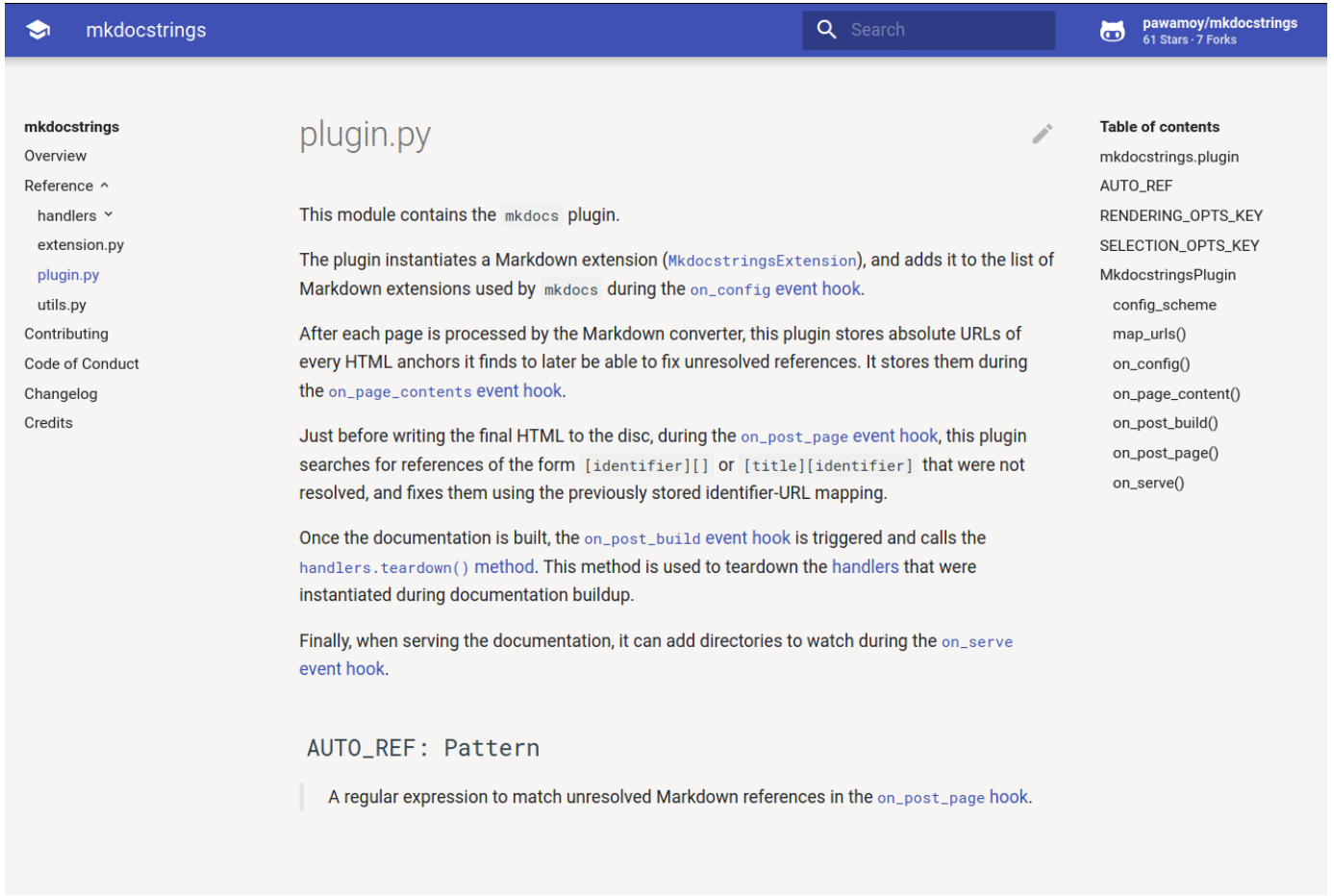
```
# nice tutorial: https://realpython.com/python-project-documentation-with-mkdocs/
site_name: "Advanced machine learning for Innovative Drug Discovery"
repo_name: my_super_repository
repo_url: https://github.com/my_super_repository
copyright: Copyright &copy; 2016 - 2024 Bayer - Machine Learning Research
theme:
  name: "material"
  logo: static/logo.png
  icon:
    repo: fontawesome/brands/git-alt
  highlightjs: true
  hljs_languages:
    - yaml
    - rust
    - python
  features:
    - tabs
    - content.code.select
    - content.code.copy
    - navigation.footer
```

```
plugins:
- search
- numkdoc
- autorefs
- mkdocstrings: <1 key>


nav:
  - Home: index.md
  - Installation: install.md
  - API documentation: aichemist_api.md

```

(Mkdocs.yml)

Tooling

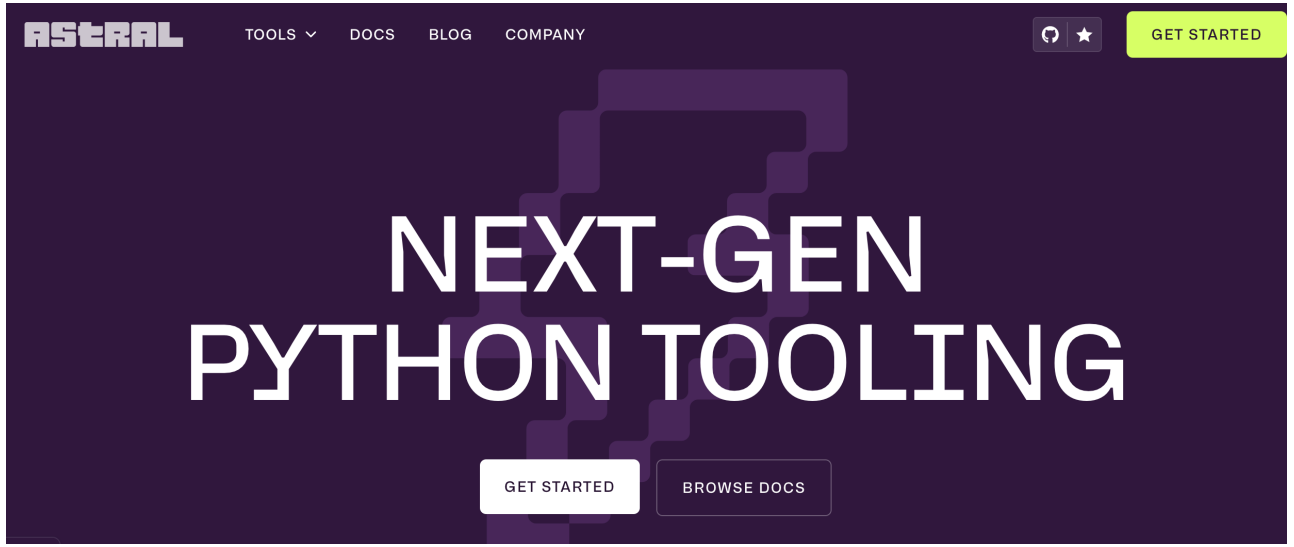
- Mkdocs + plugins
- Alternatives:
  - Sphinx
- Hosting:
  - gh-pages
  - readthedocs



https://github.com/mkdocstrings/mkdocstrings

Outlook

- ”surprisingly” a lot of movement in the packaging, environment ecosystem
  - Ruff
  - Vu
- No real standard for environment handling
- Packages in different places
  - Pypi
  - Conda / + different channels (!)
    - Nvidia
    - pytorch



https://astral.sh/

Thanks! Any questions?

"The next *python project* you do is going to be a little bit better, you're going to learn a little more."

Bob Ross

(maybe)

Copier Template

https://github.com/gieses/aichemist-copier-template