



Subhead

Python packaging, Unit Tests and Documentation



Hello Packaging World!

20240313 – Sven Giese



INTERNAL



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



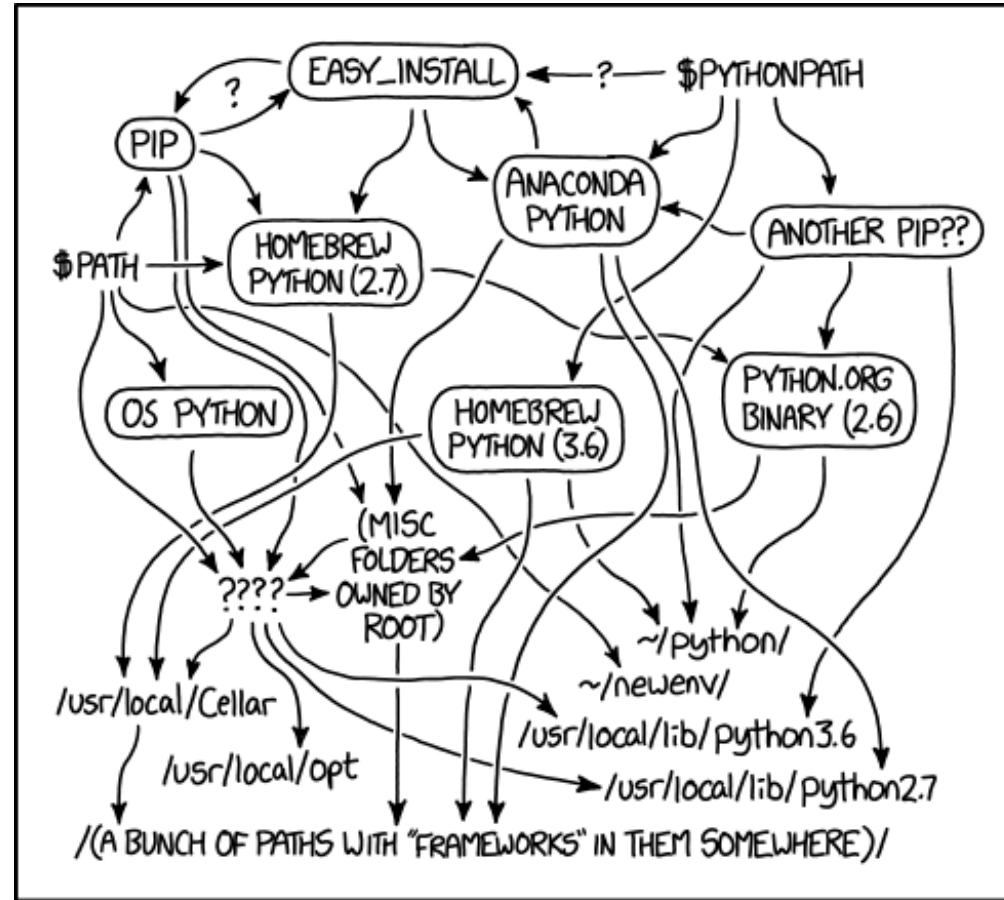


What does it take to build a good practice python package

- **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



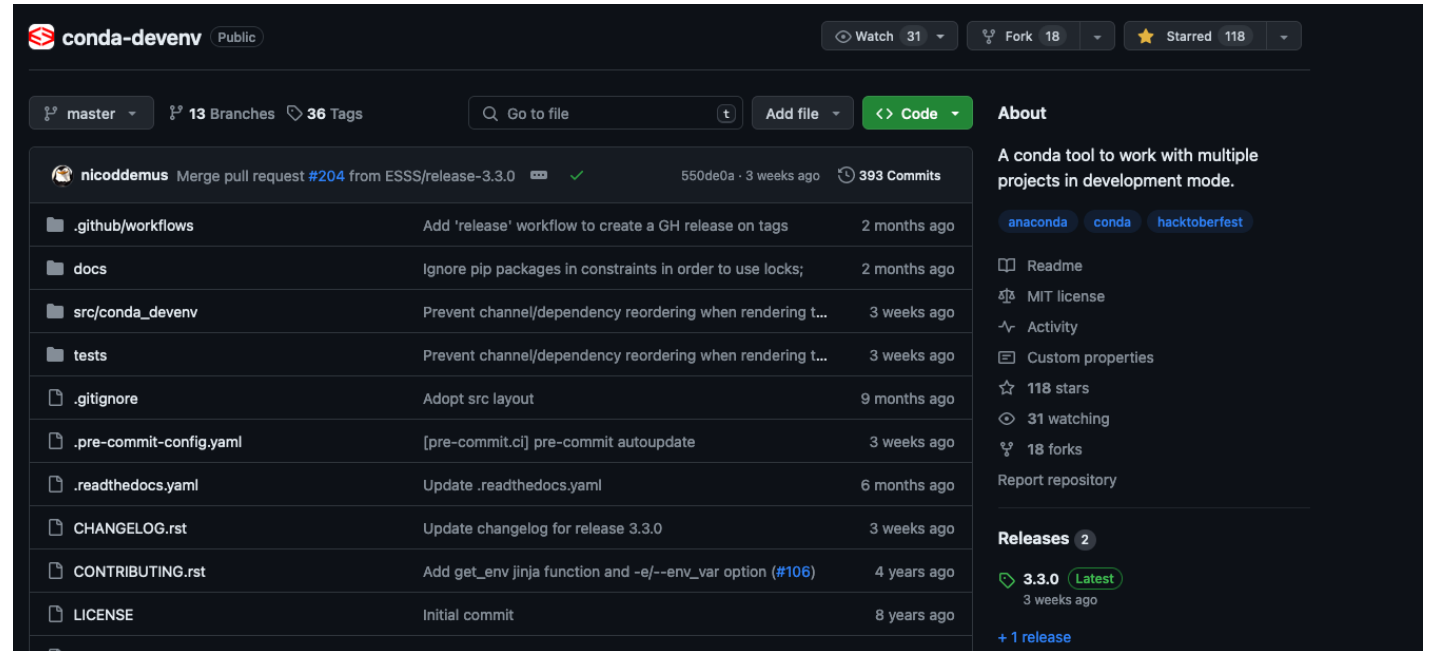
MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.



Environment Handling

We use:

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



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
```

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



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..)

scikit-learn 1.4.1.post1

```
pip install scikit-learn
```



Released: Feb 16, 2024

<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

File Name	Commit Message	Time Ago
.cirrus.star	CI Only run arm tests nightly (#26996)	7 months ago
.codecov.yml	CI ignore more non-library Python files in codecov (#260...	last year
.coveragerc	API Auto generates deprecation for sklearn.utils.mocking ...	5 years ago
.git-blame-ignore-revs	MAINT ignore isort and ruff in blame (#26663)	9 months ago
.gitignore	MAINT avoid commit containing doc/sg_execution_times....	2 months ago
.mailmap	Fix mailmap format (#9620)	7 years ago
.pre-commit-config.yaml	MNT update to ruff 0.2.1 (#28447)	3 weeks ago
CODE_OF_CONDUCT.md	DOC minor improvements in CODE_OF_CONDUCT.md file...	4 years ago
CONTRIBUTING.md	MAINT make sure canonical link is https://scikit-learn.org ...	3 weeks ago
COPYING	DOC update license year to 2023 (#25936)	last year
MANIFEST.in	MAINT include meson.build files into MANIFEST (#28283)	2 months ago
Makefile	BLD Add Meson support (#28040)	2 months ago
README.rst	DOC: Update Support page (#28593)	4 days ago
SECURITY.md	MAINT update SECURITY.md for 1.4.1.post1 release (#28...	last month
azure-pipelines.yml	CI Use environment variable for building with pip using bu...	3 weeks ago
meson.build	BLD Add Meson support (#28040)	2 months ago
pyproject.toml	MNT update to ruff 0.2.1 (#28447)	3 weeks ago
setup.cfg	CI Use environment variable to turn warnings into errors i...	last month
setup.py	MNT move sum_parallel to arrayfuncs (#28451)	3 weeks ago



Python Package

Readme:

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

Python Packaging Authority

The Python Packaging Authority (PyPA) is a working group that maintains a core set of software projects used in Python packaging.

The software developed through the PyPA is used to package, share, and install Python software and to interact with indexes of downloadable Python software such as [PyPI](#), the Python Package Index. Click the logo below to download pip, the most prominent software used to interact with PyPI.



The PyPA publishes the [Python Packaging User Guide](#), which is the authoritative resource on how to package, publish, and install Python projects using current tools. The User Guide provides a user



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 x
1 # questions
2 project_name:
3     type: str
4     help: What is your project name?
5
6 environment_name:
7     type: str
8     help: What is your preferred conda environment name?
9
10 > module_name: <2 keys>
13
14 > author: <2 keys>
```

Define Q & A

```

v aichemist-copier-template ~/PycharmProjects/aichemist-copi
v copier_template
  > conda_static
  > {% if notebooks %}notebooks{% endif%}
  > {% if results %}results{% endif%}
  > {% if workflows %}workflows{% endif%}
  > {{project_name}}
  .gitignore
  copier.yml
  environment.devenv.yml.jinja
  Makefile.jinja
  pyproject.toml.jinja
  readme.md.jinja
  {% if use_precommit %}.pre-commit-config.yaml{% endif %}
  {{_copier_conf.answers_file}}.jinja
v example_configs
  mlr_app_config.yml
> site
  .gitignore
  environment.yml
  Makefile
  mkdocs.yml
  readme.md
localhost:8889 Server is unreachable
> External Libraries
> Scratches and Consoles
```

Template



How to get started?

A simple project

This tutorial uses a simple project named `example_package_YOUR_USERNAME_HERE`. If your username is `me`, then the package would be `example_package_me`; this ensures that you have a unique package name that doesn't conflict with packages uploaded by other people following this tutorial. We recommend following this tutorial as-is using this project, before packaging your own project.

Create the following file structure locally:

```
packaging_tutorial/  
└─ src/  
    └─ example_package_YOUR_USERNAME_HERE/  
        ├── __init__.py  
        └─ example.py
```

* <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:

```
27 +  
28 +  
29 + def [redacted] (config: Configuration) -> Iterator[bytes]:  
30 +     dataset_dir = os.path.join(config.dataset_location, "train")
```

[redacted] minutes ago ...

Didn't you use pathlib in the other files? Is it possibly cleaner to stick to either pathlib or os.path throughout the whole project?

😊

gieses 2 minutes ago ...
cough cough [ruff](#)

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)

```
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]
```

```
@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)
```

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>

Python Packaging, Sven Giese

```
# 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,
)
```

```
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)
```

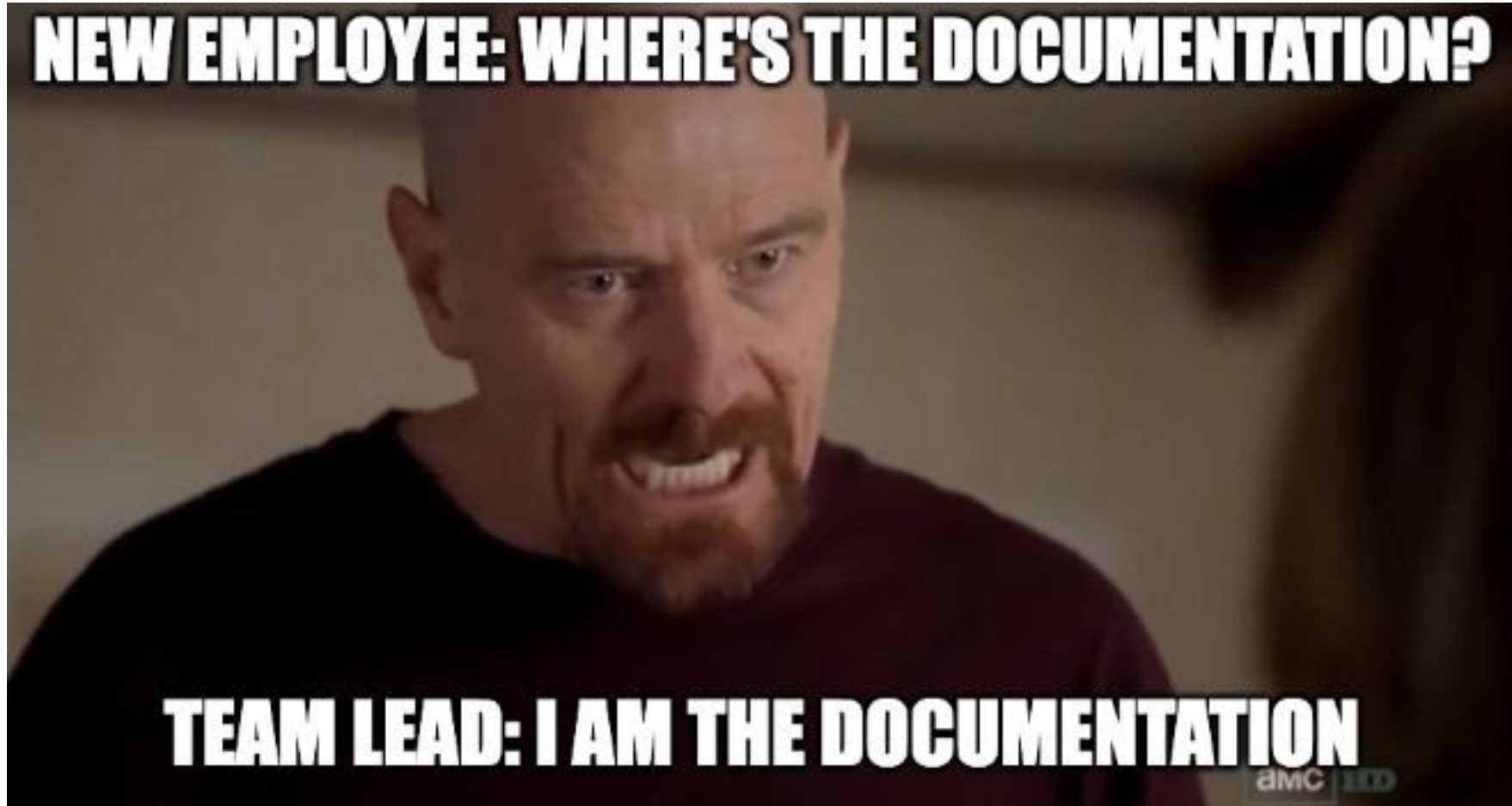


Agenda





Documentation



<https://twitter.com/peteryang/status/1575289122364608513>



Defining Documentation: scikit-learn

```

1213 class KMeans(_BaseKMeans):
1214     """K-Means clustering.
1215
1216     Read more in the :ref:`User Guide <k_means>`.
1217
1218     Parameters
1219     -----
1220
1221     n_clusters : int, default=8
1222         The number of clusters to form as well as the number of
1223         centroids to generate.
1224
1225         For an example of how to choose an optimal value for `n_clusters` refer to
1226         :ref:`sphx_glr_auto_examples_cluster_plot_kmeans_silhouette_analysis.py`.
1227
1228     init : {'k-means++', 'random'}, callable or array-like of shape \
1229           (n_clusters, n_features), default='k-means++'
1230         Method for initialization:
1231
1232         * 'k-means++' : selects initial cluster centroids using sampling \
1233           based on an empirical probability distribution of the points \
1234           contribution to the overall inertia. This technique speeds up \
1235           convergence. The algorithm implemented is "greedy k-means++". It \
1236           differs from the vanilla k-means++ by making several trials at \
1237           each sampling step and choosing the best centroid among them.
1238
1239         * 'random': choose `n_clusters` observations (rows) at random from \
1240           data for the initial centroids.
1241
1242         * If an array is passed, it should be of shape (n_clusters, n_features)\
1243           and gives the initial centers.
1244
1245         * If a callable is passed, it should take arguments X, n_clusters and a \
1246           random state and return an initialization.
1247
1248         For an example of how to use the different `init` strategy, see the example
1249         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.

$$\hat{y}(w, x) = w_0 + w_1x_1 + \dots + w_px_p$$

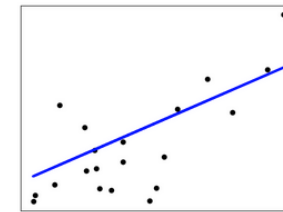
Across the module, we designate the vector $w = (w_1, \dots, w_p)$ as `coef_` and w_0 as `intercept_`.

To perform classification with generalized linear models, see [Logistic regression](#).

1.1.1. Ordinary Least Squares

[LinearRegression](#) fits a linear model with coefficients $w = (w_1, \dots, w_p)$ 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 Estimators

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.

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



Defining Documentation: pandas



Getting started

New to *pandas*? Check out the getting started guides. They contain an introduction to *pandas*' main concepts and links to additional tutorials.

To the getting started guides



User guide

The user guide provides in-depth information on the key concepts of *pandas* with useful background information and explanation.

To the user guide



API reference

The reference guide contains a detailed description of the *pandas* API. The reference describes how the methods work and which parameters can be used. It assumes that you have an understanding of the key concepts.

To the reference guide



Developer guide

Saw a typo in the documentation? Want to improve existing functionalities? The contributing guidelines will guide you through the process of improving *pandas*.

To the development guide

```
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
```

```
function_name_1(parameter1, parameter2):
    """
    This is a
    :param pa
    :type par
    :param pa
    :type par
    :return:
    :rtype: t
    :raises E
    """
    def function_name_1(parameter1: type,
                        parameter2: type) -> type
        This is a brief summary of the function.
        Params: parameter1 – Description of parameter1.
               parameter2 – Description of parameter2.
        Returns: Description of the return value.
        Raises: ExceptionType – Description of the
               exception raised.
```

```
def function_name_2(parameter1, parameter2):
    """
    This is a b
    Parameters
    -----
    parameter1
    parameter2
    """
    def function_name_2(parameter1: type,
                        parameter2: type) -> type
        This is a brief summary of the function.
        Params: parameter1 – Description of parameter1.
               parameter2 – Description of parameter2.
        Returns: Description of the return value.
        Raises: ExceptionType – Description of the
               exception raised.
```




Mkdocs

- Docs folder
- Mkdocs.yml (config)

Usage:

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

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

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

(Makefile shortcuts)

```
1 # nice tutorial: https://realpython.com/python-project-documentation-with-mkdocs/
2 site_name: "Advanced machine learning for Innovative Drug Discovery"
3 repo_name: my_super_repository
4 repo_url: https://github.com/my\_super\_repository
5 copyright: Copyright &copy; 2016 - 2024 Bayer - Machine Learning Research
6 v theme:
7   name: "material"
8   logo: static/logo.png
9   v icon:
10     repo: fontawesome/brands/git-alt
11   highlightjs: true
12   v hljs_languages:
13     - yaml
14     - rust
15     - python
16   v features:
17     - tabs
18     - content.code.select
19     - content.code.copy
20     - navigation.footer
```

```
67 plugins:
68 - search
69 - numkdoc
70 - autorefs
71 > - mkdocstrings: <1 key>
85
86
87 nav:
88 - Home: index.md
89 - Installation: install.md
90 - API documentation: aichemist\_api.md
91
```

(Mkdocs.yml)



Tooling

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

The screenshot shows the GitHub repository page for `mkdocstrings`. The repository name is `mkdocstrings` by `pawamoy/mkdocstrings`, which has 61 stars and 7 forks. The file `plugin.py` is selected, and its content is displayed. The file contains a Python module that implements a plugin for MkDocs. It instantiates a `MkdocstringsExtension` and adds it to the list of extensions used by MkDocs during the `on_config` event hook. The plugin also handles absolute URLs of HTML anchors, storing them during the `on_page_contents` event hook and fixing unresolved references during the `on_post_page` event hook. Finally, it adds directories to watch during the `on_serve` event hook. A table of contents on the right lists the various methods and attributes defined in the plugin.

Table of contents

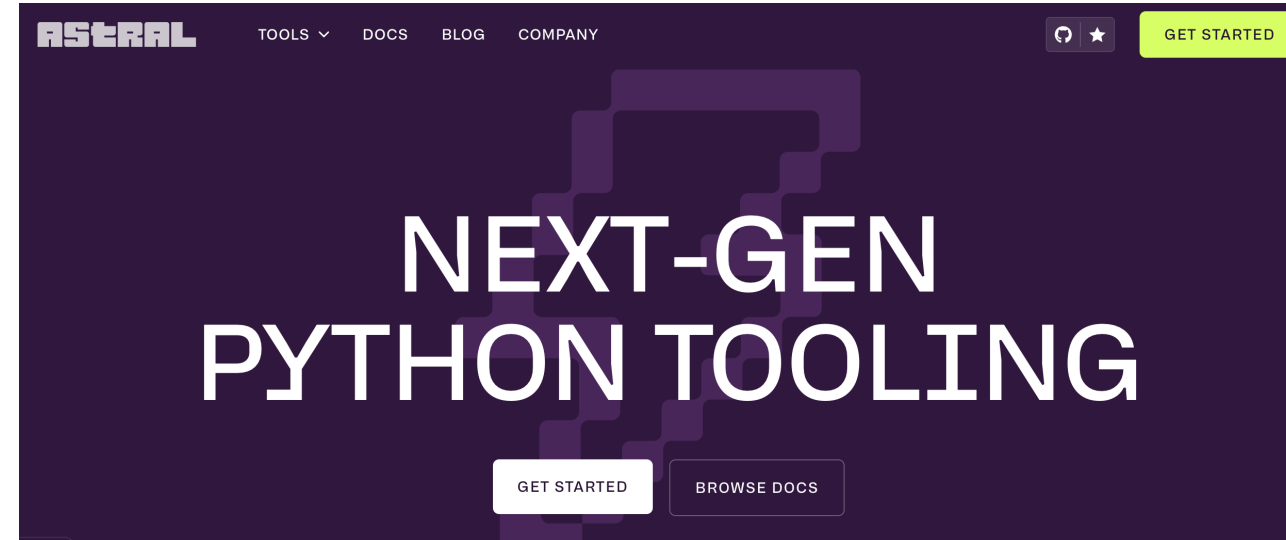
- mkdocstrings.plugin
- AUTO_REF
- RENDERING_OPTS_KEY
- SELECTION_OPTS_KEY
- MkdocstringsPlugin
- config_scheme
- map_urls()
- on_config()
- on_page_content()
- on_post_build()
- on_post_page()
- on_serve()

<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>