EPFL

Documentation, testing, typing

Practical Programming in Chemistry

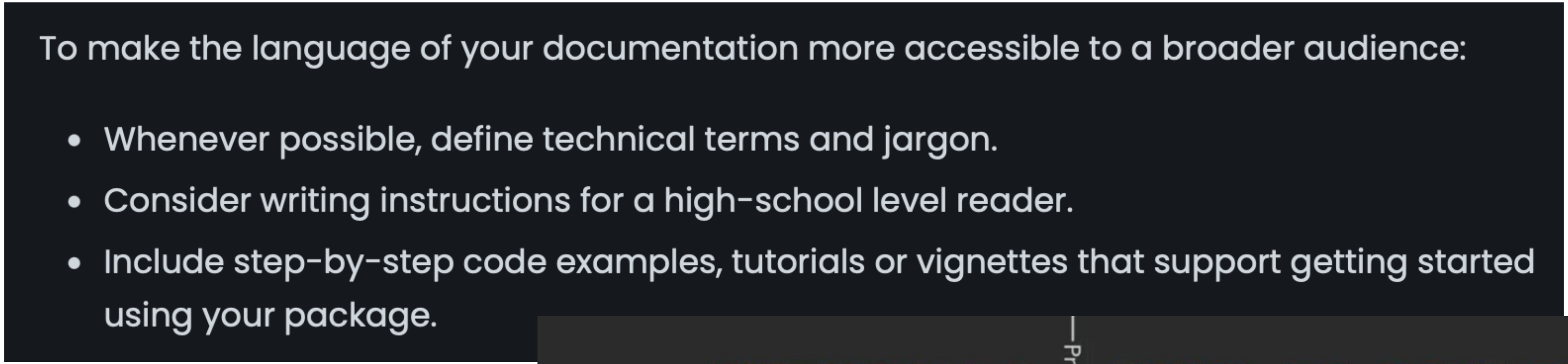
Prof. Philippe Schwaller

This lecture

- Documentation
  - Sphinx
- Typing
  - Mypy
- Automation
  - Tox
- Testing
  - pytest

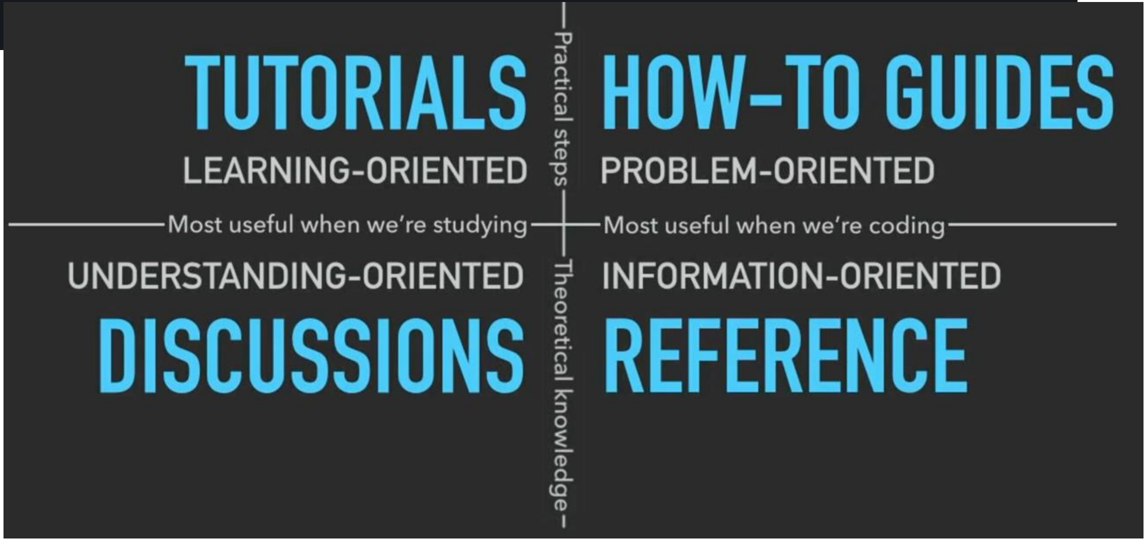
How to write beautiful and reusable code – with help of tools

User documentation

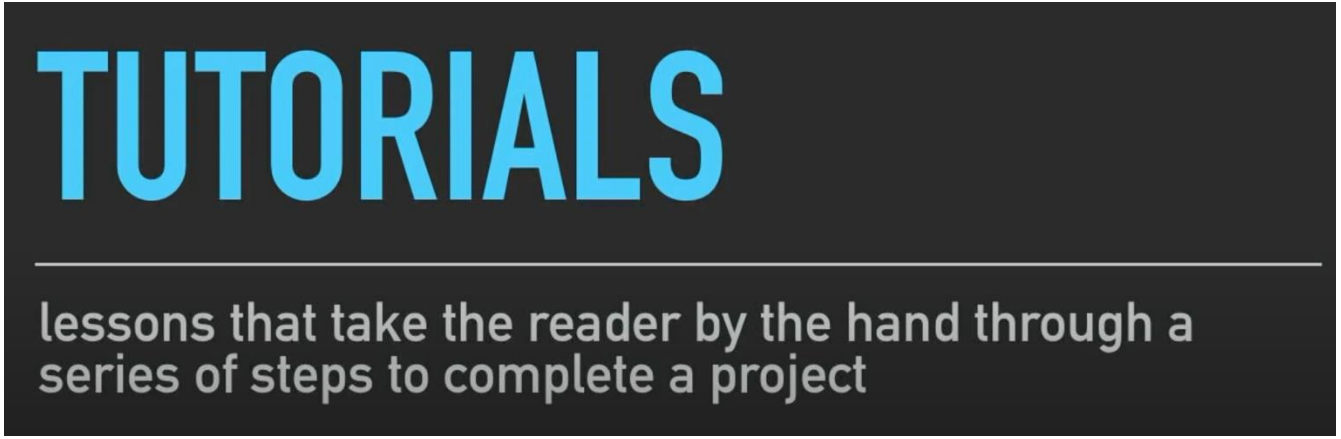


To make the language of your documentation more accessible to a broader audience:

- Whenever possible, define technical terms and jargon.
- Consider writing instructions for a high-school level reader.
- Include step-by-step code examples, tutorials or vignettes that support getting started using your package.

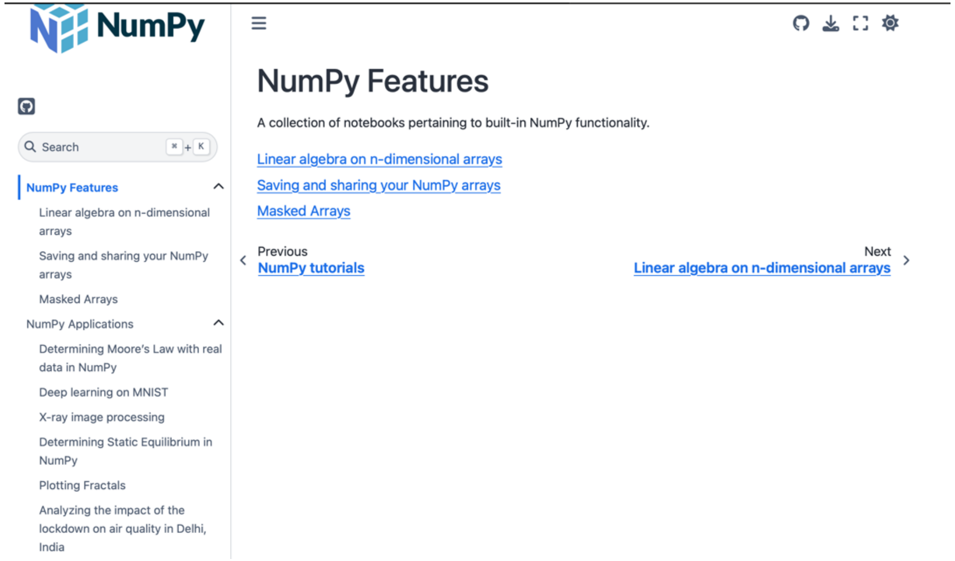


https://www.youtube.com/watch?v=t4vKPhjcMZg - **What nobody tells you about documentation**



TUTORIALS

lessons that take the reader by the hand through a series of steps to complete a project


NumPy
Search
NumPy Features
Linear algebra on n-dimensional arrays
Saving and sharing your NumPy arrays
Masked Arrays
NumPy Applications
Determining Moore's Law with real data in NumPy
Deep learning on MNIST
X-ray image processing
Determining Static Equilibrium in NumPy
Plotting Fractals
Analyzing the impact of the lockdown on air quality in Delhi, India

NumPy Features

A collection of notebooks pertaining to built-in NumPy functionality.

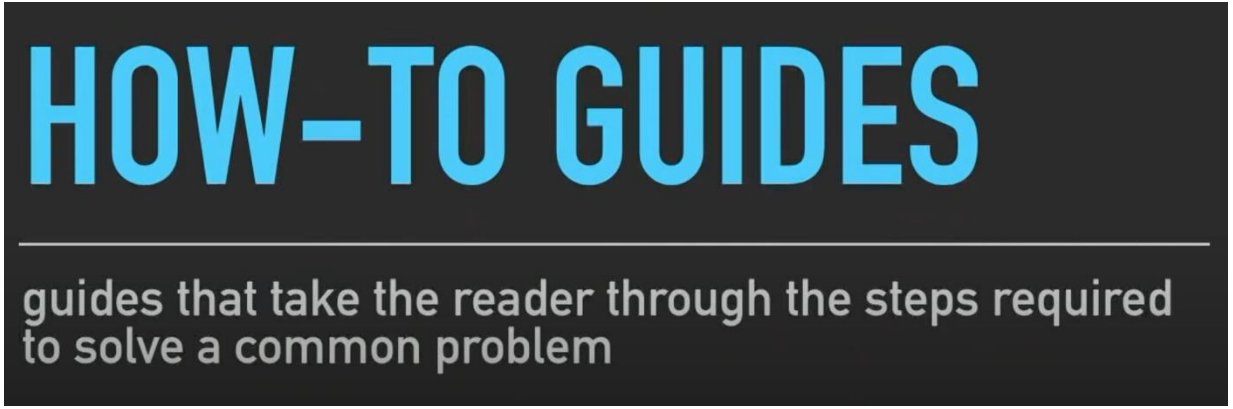
Linear algebra on n-dimensional arrays
Saving and sharing your NumPy arrays
Masked Arrays

Previous
NumPy tutorials

Next
Linear algebra on n-dimensional arrays

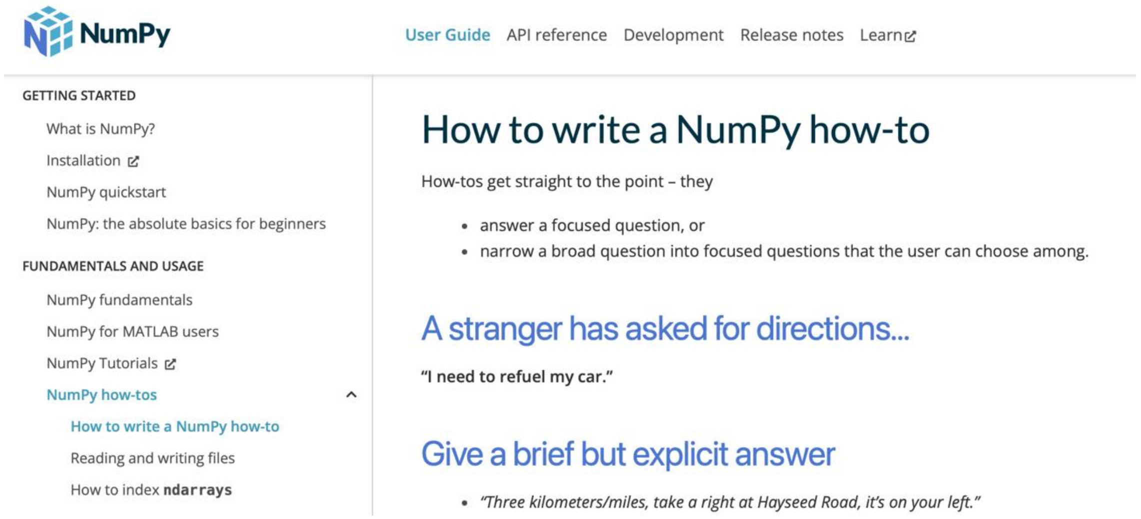
For example, Jupyter notebooks that use the code from your package.

The exercises in this course
→ Learning-oriented



HOW-TO GUIDES

guides that take the reader through the steps required to solve a common problem



NumPy

User Guide API reference Development Release notes Learn

GETTING STARTED

What is NumPy?

Installation

NumPy quickstart

NumPy: the absolute basics for beginners

FUNDAMENTALS AND USAGE

NumPy fundamentals

NumPy for MATLAB users

NumPy Tutorials

NumPy how-tos

How to write a NumPy how-to

Reading and writing files

How to index **ndarrays**

How to write a NumPy how-to

How-tos get straight to the point – they

- answer a focused question, or
- narrow a broad question into focused questions that the user can choose among.

A stranger has asked for directions...

"I need to refuel my car."

Give a brief but explicit answer

- *"Three kilometers/miles, take a right at Hayseed Road, it's on your left."*

Analogy:

A cooking recipe
or synthesis procedure.

How to make this meal/
compound.

Step 1: …
Step 2: ...
→ Problem-oriented

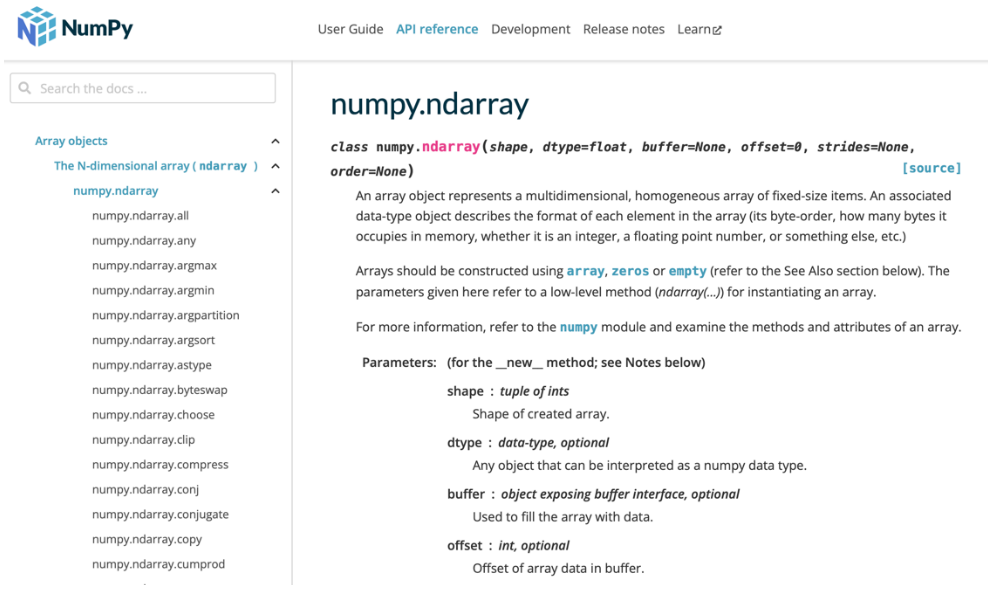
Speaker

REFERENCE

technical descriptions of the machinery and how to operate it

API documentation

This is something you should always do.



numpy.ndarray

```
class numpy.ndarray(shape, dtype=float, buffer=None, offset=0, strides=None,
order=None)                                                          [source]
```

An array object represents a multidimensional, homogeneous array of fixed-size items. An associated data-type object describes the format of each element in the array (its byte-order, how many bytes it occupies in memory, whether it is an integer, a floating point number, or something else, etc.)

Arrays should be constructed using `array`, `zeros` or `empty` (refer to the See Also section below). The parameters given here refer to a low-level method (*ndarray(...)*) for instantiating an array.

For more information, refer to the `numpy` module and examine the methods and attributes of an array.

Parameters: (for the __new__ method; see Notes below)

shape : *tuple of ints*
Shape of created array.

dtype : *data-type, optional*
Any object that can be interpreted as a numpy data type.

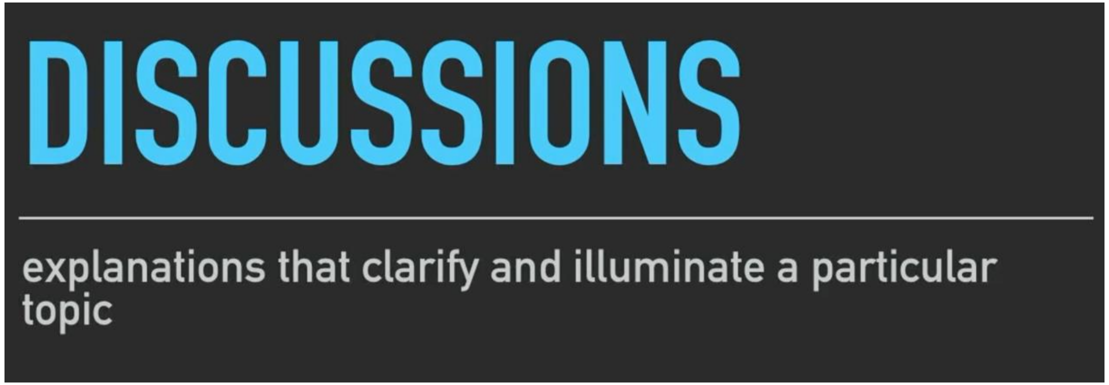
buffer : *object exposing buffer interface, optional*
Used to fill the array with data.

offset : *int, optional*
Offset of array data in buffer.

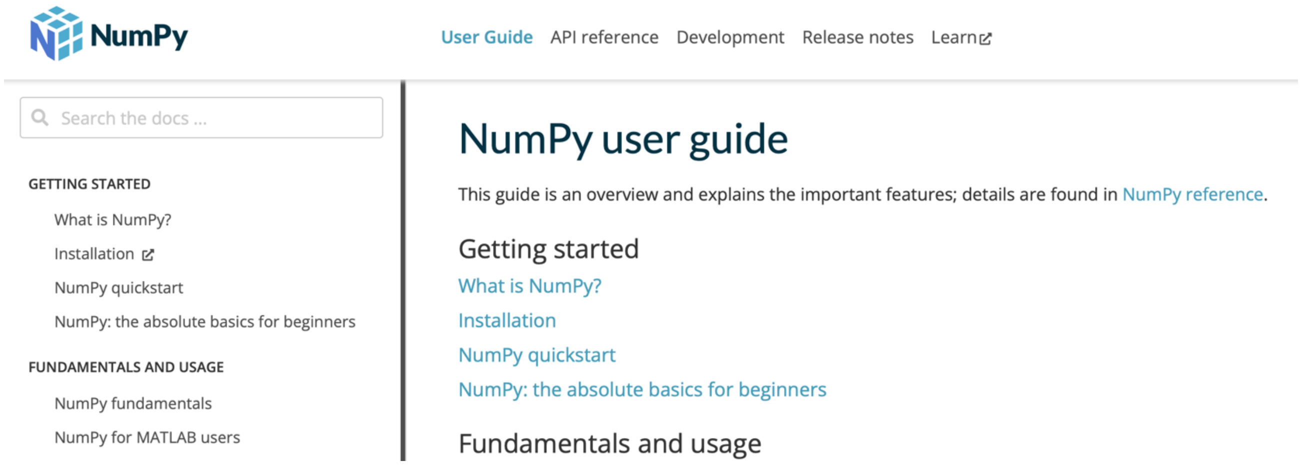
What you put in a function documentation string in your code.

- Give you the facts (inputs, outputs)
- Can include examples.

→ Information-oriented


DISCUSSIONS
explanations that clarify and illuminate a particular topic

Background information on the project.



GETTING STARTED

- What is NumPy?
- Installation
- NumPy quickstart
- NumPy: the absolute basics for beginners

FUNDAMENTALS AND USAGE

- NumPy fundamentals
- NumPy for MATLAB users

NumPy user guide

This guide is an overview and explains the important features; details are found in NumPy reference.

Getting started

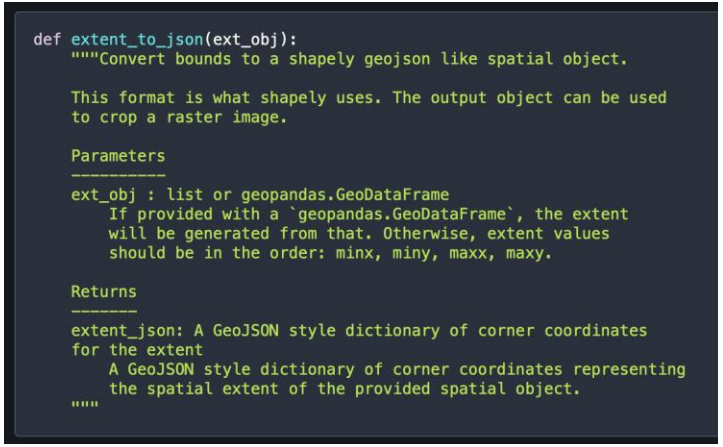
What is NumPy?
Installation
NumPy quickstart
NumPy: the absolute basics for beginners

Fundamentals and usage

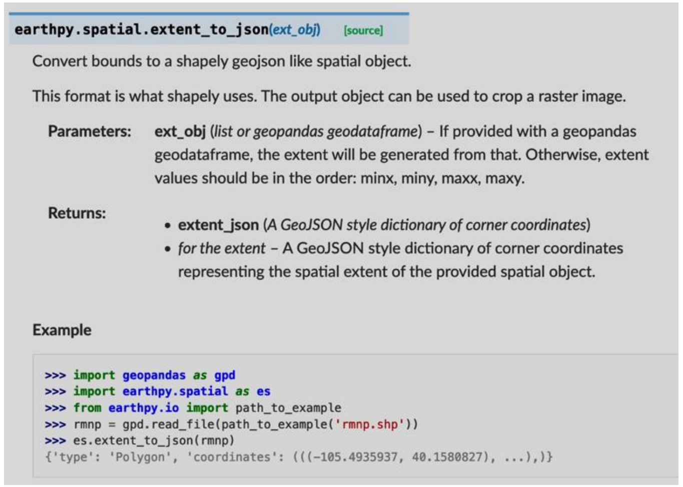
If you want your work/code to be used by others, make it easy for them to get started and use it.

→ Write a good documentation.

API documentation → docstrings for functions



```
def extent_to_json(ext_obj):
    """Convert bounds to a shapely geojson like spatial object.

    This format is what shapely uses. The output object can be used
    to crop a raster image.

    Parameters
    ----------
    ext_obj : list or geopandas.GeoDataFrame
        If provided with a `geopandas.GeoDataFrame`, the extent
        will be generated from that. Otherwise, extent values
        should be in the order: minx, miny, maxx, maxy.

    Returns
    -------
    extent_json: A GeoJSON style dictionary of corner coordinates
    for the extent
        A GeoJSON style dictionary of corner coordinates representing
        the spatial extent of the provided spatial object.
    """
```



earthpy.spatial.extent_to_json(*ext_obj*) [source]

Convert bounds to a shapely geojson like spatial object.

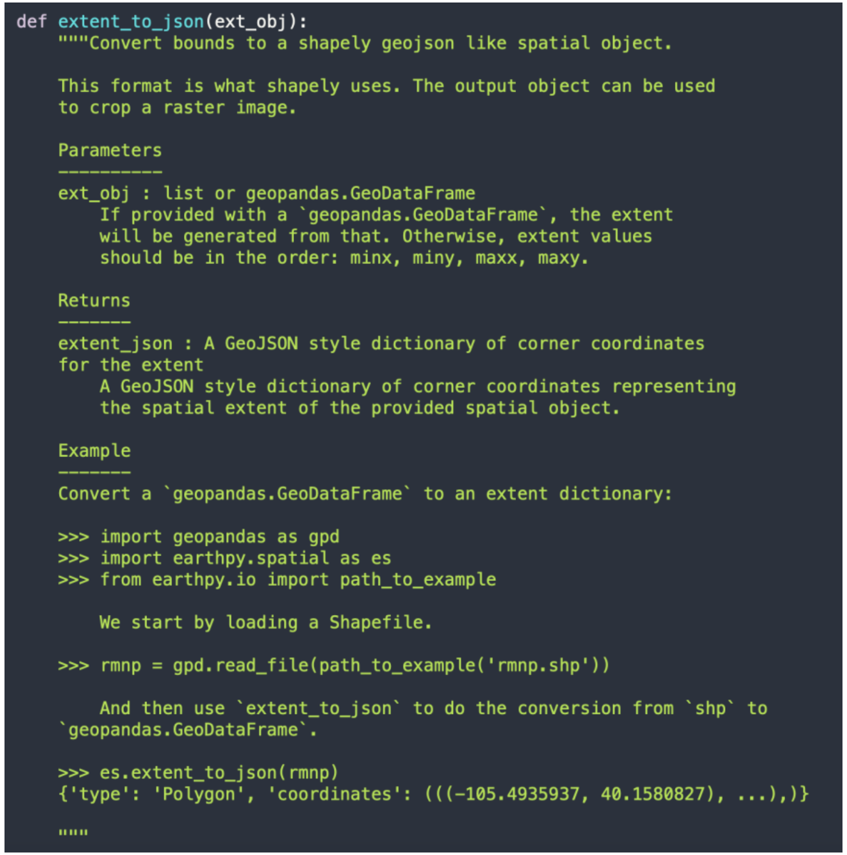
This format is what shapely uses. The output object can be used to crop a raster image.

Parameters: **ext_obj** (*list or geopandas geodataframe*) – If provided with a geopandas geodataframe, the extent will be generated from that. Otherwise, extent values should be in the order: minx, miny, maxx, maxy.

Returns:
- **extent_json** (*A GeoJSON style dictionary of corner coordinates*)
- *for the extent* – A GeoJSON style dictionary of corner coordinates representing the spatial extent of the provided spatial object.

Example

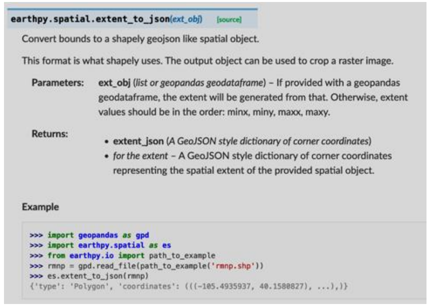
```
>>> import geopandas as gpd
>>> import earthpy.spatial as es
>>> from earthpy.io import path_to_example
>>> rmnp = gpd.read_file(path_to_example('rmnp.shp'))
>>> es.extent_to_json(rmnp)
{'type': 'Polygon', 'coordinates': (((-105.4935937, 40.1580827), ...),)}
```



```
def extent_to_json(ext_obj):
    """Convert bounds to a shapely geojson like spatial object.

    This format is what shapely uses. The output object can be used
    to crop a raster image.

    Parameters
    ----------
    ext_obj : list or geopandas.GeoDataFrame
        If provided with a `geopandas.GeoDataFrame`, the extent
        will be generated from that. Otherwise, extent values
        should be in the order: minx, miny, maxx, maxy.

    Returns
    -------
    extent_json : A GeoJSON style dictionary of corner coordinates
    for the extent
        A GeoJSON style dictionary of corner coordinates representing
        the spatial extent of the provided spatial object.

    Example
    -------
    Convert a `geopandas.GeoDataFrame` to an extent dictionary:

    >>> import geopandas as gpd
    >>> import earthpy.spatial as es
    >>> from earthpy.io import path_to_example

        We start by loading a Shapefile.

    >>> rmnp = gpd.read_file(path_to_example('rmnp.shp'))

        And then use `extent_to_json` to do the conversion from `shp` to
    `geopandas.GeoDataFrame`.

    >>> es.extent_to_json(rmnp)
    {'type': 'Polygon', 'coordinates': (((-105.4935937, 40.1580827), ...),)}

    """
```

Automatically, compiled using tools like Sphinx.

Sphinx – tool for documentation



`earthpy.spatial.extent_to_json`(*ext_obj*) [source]

Convert bounds to a shapely geojson like spatial object.

This format is what shapely uses. The output object can be used to crop a raster image.

Parameters: **ext_obj** (*list or geopandas geodataframe*) – If provided with a geopandas geodataframe, the extent will be generated from that. Otherwise, extent values should be in the order: minx, miny, maxx, maxy.

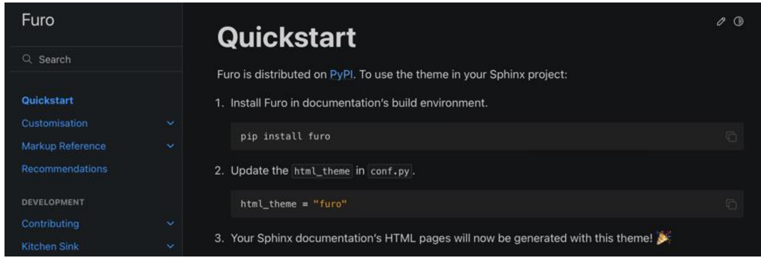
Returns:
- **extent_json** (*A GeoJSON style dictionary of corner coordinates*)
- *for the extent* – A GeoJSON style dictionary of corner coordinates representing the spatial extent of the provided spatial object.

Example

```
>>> import geopandas as gpd
>>> import earthpy.spatial as es
>>> from earthpy.io import path_to_example
>>> rmnp = gpd.read_file(path_to_example('rmnp.shp'))
>>> es.extent_to_json(rmnp)
{'type': 'Polygon', 'coordinates': (((-105.4935937, 40.1580827), ...),)}
```

- Sphinx is a **static-site generator**. A static site generator is a tool that creates html for a website based upon a set of templates. The html files are then served “statically” which means that there is no generation or modification of the files on the fly.
- **Automated documentation generation** from markdown files and docstrings
- It might be a bit challenging to set up at first, but we set it up for you in the copier-liac template (with the Furo theme).

https://www.sphinx-doc.org/en/master/



Quickstart

Furo is distributed on PyPI. To use the theme in your Sphinx project:

1. Install Furo in documentation’s build environment.

```
pip install furo
```

2. Update the `html_theme` in `conf.py`.

```
html_theme = "furo"
```

3. Your Sphinx documentation’s HTML pages will now be generated with this theme! 🎉

Speaker

NAME EVENT / NAME PRESENTATION

Docstring styles – Numpy vs Google

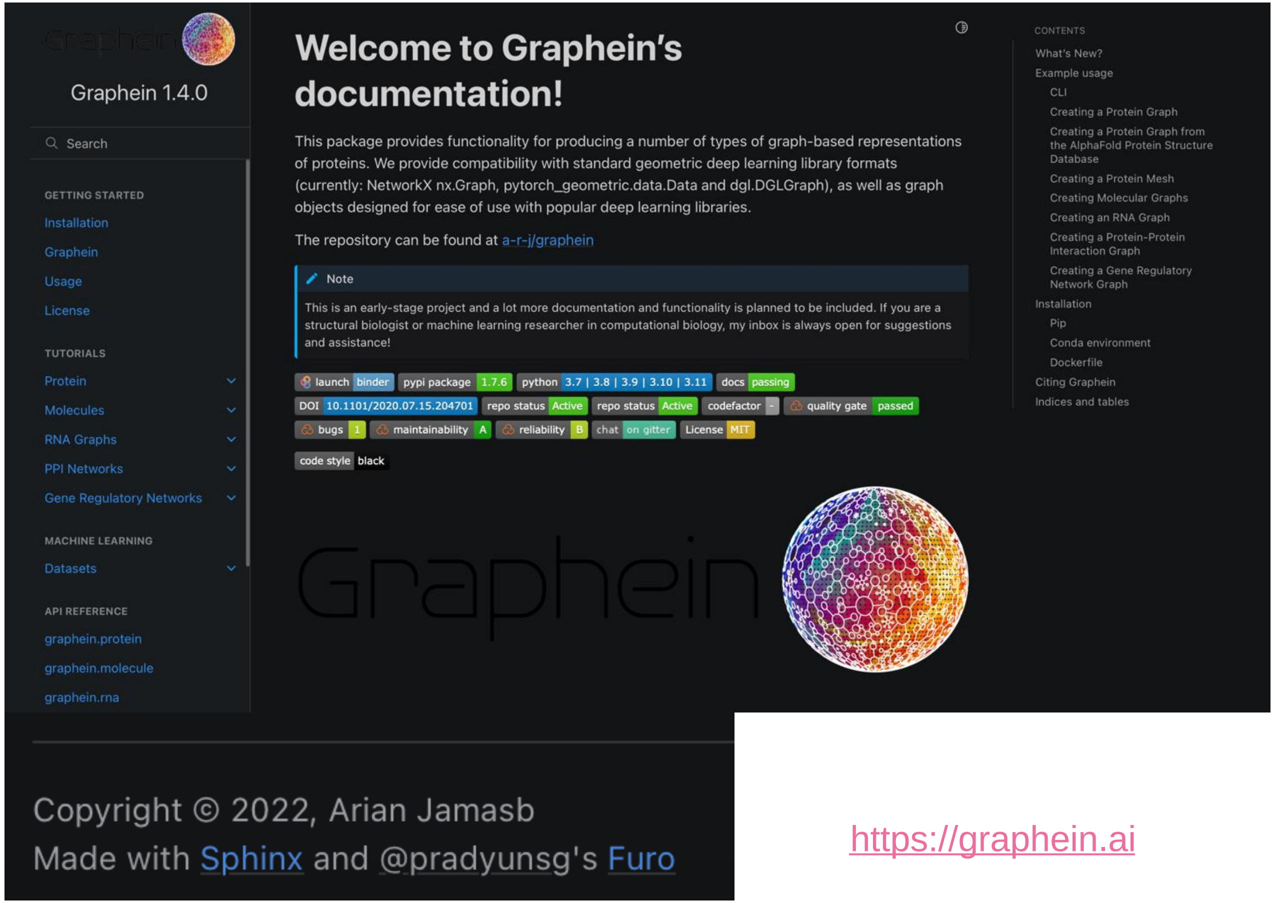
```python
def func(arg1, arg2):
    """Summary line.

    Extended description of function.

    Parameters
    ----------
    arg1 : int
        Description of arg1
    arg2 : str
        Description of arg2

    Returns
    -------
    bool
        Description of return value

    """
    return True
```

Numpy – more vertical space

```python
def func(arg1, arg2):
    """Summary line.

    Extended description of function.

    Args:
        arg1 (int): Description of arg1
        arg2 (str): Description of arg2

    Returns:
        bool: Description of return value

    """
    return True
```

Google – more horizontal space

Example of a beautiful documentation – Graphein



Graphein 1.4.0

Search

GETTING STARTED
Installation
Graphein
Usage
License

TUTORIALS
Protein
Molecules
RNA Graphs
PPI Networks
Gene Regulatory Networks

MACHINE LEARNING
Datasets

API REFERENCE
graphein.protein
graphein.molecule
graphein.rna

Welcome to Graphein's documentation!

This package provides functionality for producing a number of types of graph-based representations of proteins. We provide compatibility with standard geometric deep learning library formats (currently: NetworkX nx.Graph, pytorch_geometric.data.Data and dgl.DGLGraph), as well as graph objects designed for ease of use with popular deep learning libraries.

The repository can be found at a-r-j/graphein

Note

This is an early-stage project and a lot more documentation and functionality is planned to be included. If you are a structural biologist or machine learning researcher in computational biology, my inbox is always open for suggestions and assistance!

launch binder | pypi package 1.7.6 | python 3.7 | 3.8 | 3.9 | 3.10 | 3.11 | docs passing
DOI 10.1101/2020.07.15.204701 | repo status Active | repo status Active | codefactor - | quality gate passed
bugs 1 | maintainability A | reliability B | chat on gitter | License MIT
code style black

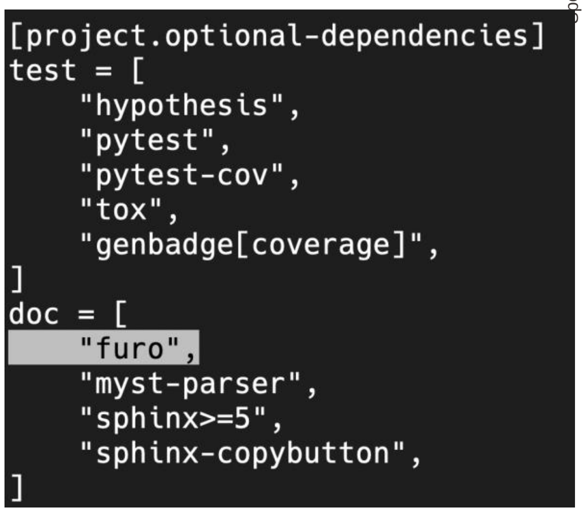
CONTENTS
- What's New?
- Example usage
  - CLI
  - Creating a Protein Graph
  - Creating a Protein Graph from the AlphaFold Protein Structure Database
  - Creating a Protein Mesh
  - Creating Molecular Graphs
  - Creating an RNA Graph
  - Creating a Protein-Protein Interaction Graph
  - Creating a Gene Regulatory Network Graph
- Installation
  - Pip
  - Conda environment
  - Dockerfile
- Citing Graphein
- Indices and tables

Copyright © 2022, Arian Jamasb
Made with Sphinx and @pradyunsg's Furo

https://graphein.ai

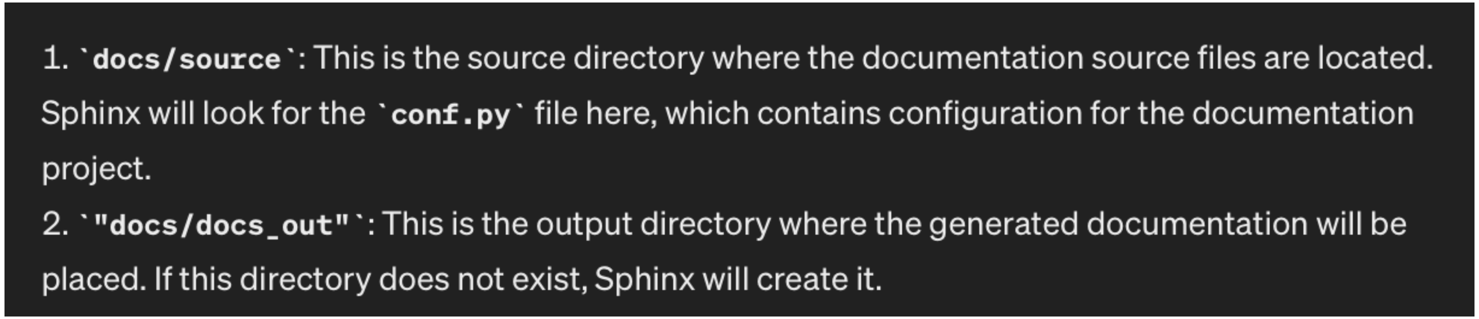
With copier-liac template

In the copier-liac-minimal pyproject.toml



```
[project.optional-dependencies]
test = [
    "hypothesis",
    "pytest",
    "pytest-cov",
    "tox",
    "genbadge[coverage]",
]
doc = [
    "furo",
    "myst-parser",
    "sphinx>=5",
    "sphinx-copybutton",
]
```

- Install it with *pip install -e ".[test,doc]"* (you need the optional dependencies)
- *sphinx-build docs/source "docs/docs_out"*
- Or just "*tox*" (which runs tests and builds docs)



1. `docs/source`: This is the source directory where the documentation source files are located. Sphinx will look for the `conf.py` file here, which contains configuration for the documentation project.
2. `"docs/docs_out"`: This is the output directory where the generated documentation will be placed. If this directory does not exist, Sphinx will create it.

EPFL

What is tox?

tox

standardise testing in Python

- **Automated Testing Across Environments** (e.g., multiple Python versions, not just the one you are using)
- **Virtual Environment Management**: Tox automatically creates separate virtual environments to isolate testes Python package from other dependencies/system-wide installed packages.
- **Dependency Management**: Tox can have separate dependencies for each test/docs environment.
- **Automation of Commands**: Tox gives you a single command to check code style, generate documentation, and tests.
- Configuration in ”tox.ini”, and run with “tox” in the project folder.

tox.ini

```
[tox]
env_list = py3{10,11,12}, docs, coverage
```

Test Python 10, 11, 12, and then,
some special stuff for docs and coverage

```
[testenv]
basepython =
    {py310,docs,coverage}: python3.10
    py311: python3.11
    py312: python3.12
setenv =
    PYTHONUNBUFFERED = yes
passenv =
    *
extras =
    test
commands =
    pytest
usedevelop = true
```

Basic test environment setup

```
[testenv:docs]
description = build HTML docs
setenv =
    READTHEDOCS_PROJECT = ch200
    READTHEDOCS_VERSION = latest
extras =
    doc
commands =
    sphinx-build -d "{toxworkdir}/docs_doctree" docs/source "docs/docs_out"

[testenv:coverage]
commands =
    pytest --cov=src/ch200 --cov-report xml:.tox/coverage.xml --cov-report term
    genbadge coverage -i .tox/coverage.xml -o assets/coverage-badge.svg
```

Special commands for docs and coverage.

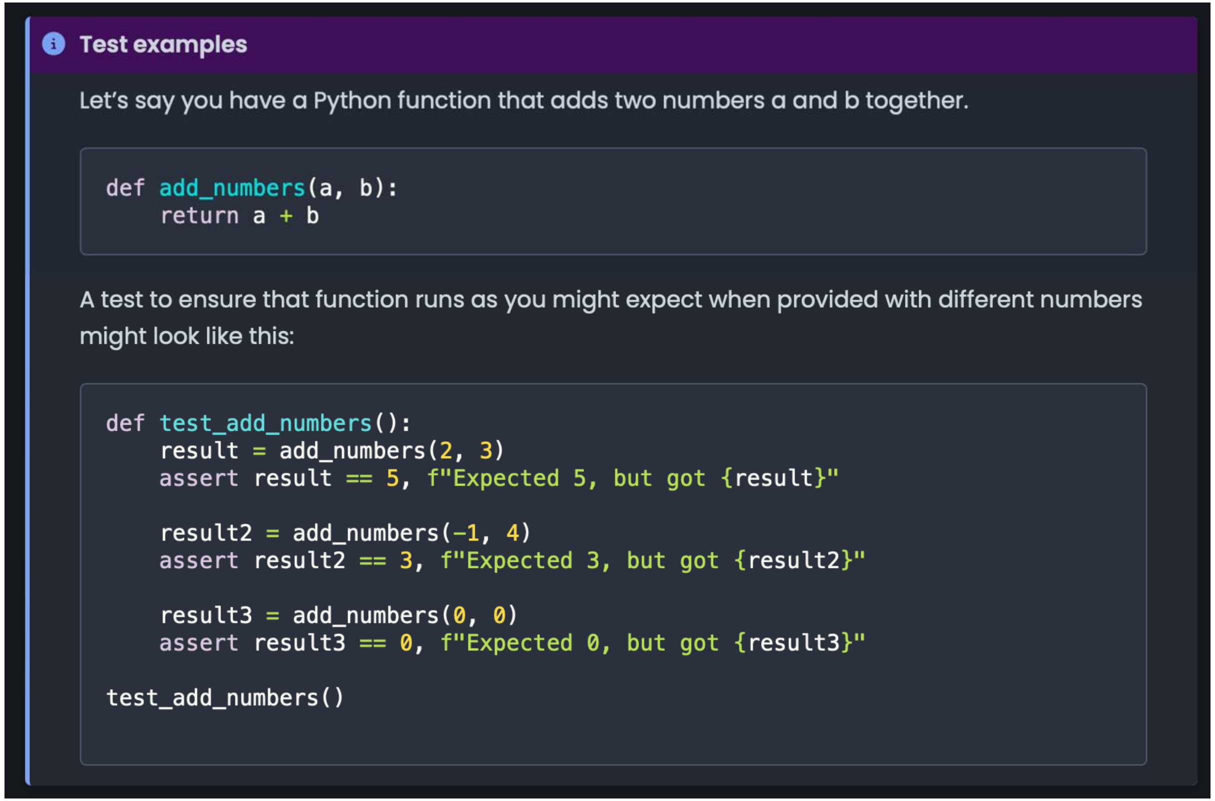
```
[gh-actions]
python =
    3.10: py310, docs
    3.11: py311
    3.12: py312
```

What tests to run using github actions
→ the workflows defined in .github

Tests for your code.

EPFL

Test your code! (important)



Test examples

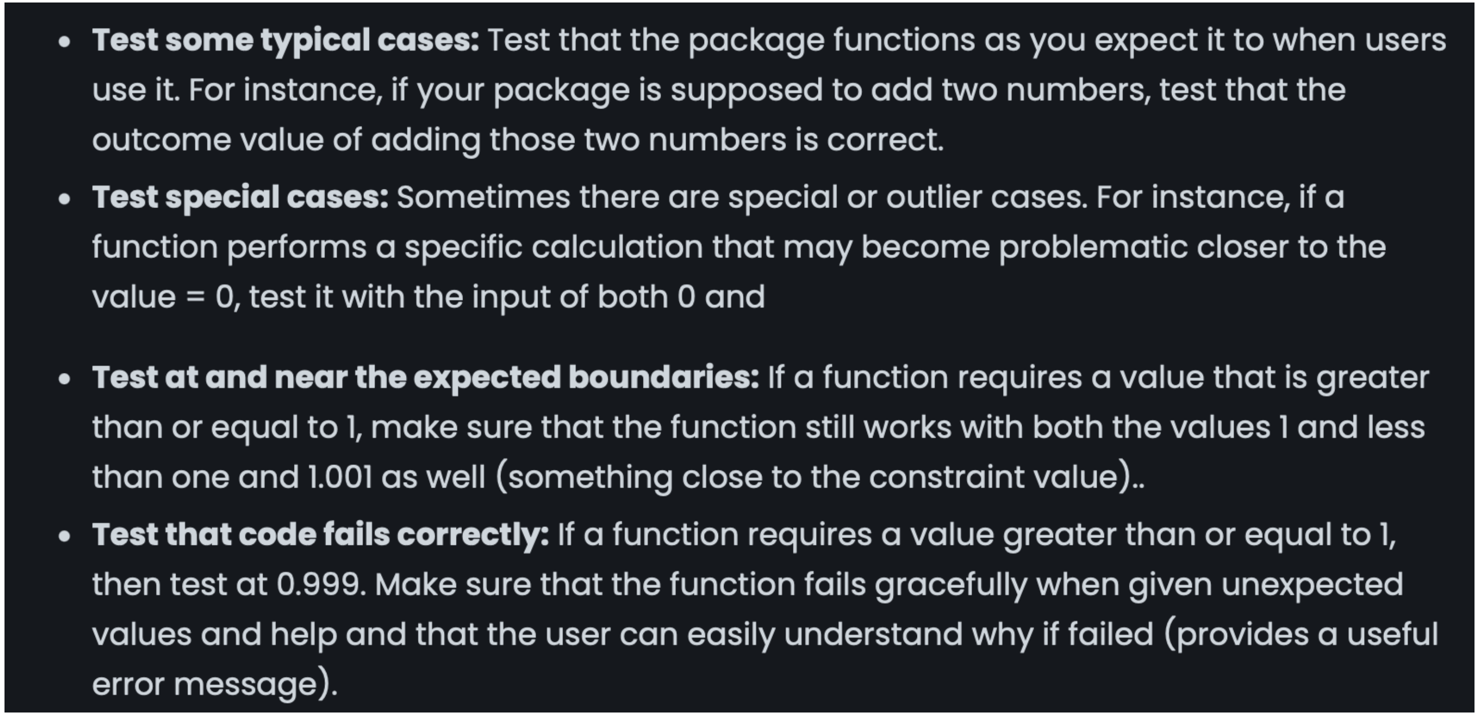
Let's say you have a Python function that adds two numbers a and b together.

```python
def add_numbers(a, b):
    return a + b
```

A test to ensure that function runs as you might expect when provided with different numbers might look like this:

```python
def test_add_numbers():
    result = add_numbers(2, 3)
    assert result == 5, f"Expected 5, but got {result}"

    result2 = add_numbers(-1, 4)
    assert result2 == 3, f"Expected 3, but got {result2}"

    result3 = add_numbers(0, 0)
    assert result3 == 0, f"Expected 0, but got {result3}"

test_add_numbers()
```

What to test?



- **Test some typical cases:** Test that the package functions as you expect it to when users use it. For instance, if your package is supposed to add two numbers, test that the outcome value of adding those two numbers is correct.
- **Test special cases:** Sometimes there are special or outlier cases. For instance, if a function performs a specific calculation that may become problematic closer to the value = 0, test it with the input of both 0 and
- **Test at and near the expected boundaries:** If a function requires a value that is greater than or equal to 1, make sure that the function still works with both the values 1 and less than one and 1.001 as well (something close to the constraint value)..
- **Test that code fails correctly:** If a function requires a value greater than or equal to 1, then test at 0.999. Make sure that the function fails gracefully when given unexpected values and help and that the user can easily understand why if failed (provides a useful error message).

https://www.pyopensci.org/python-package-guide/tests/write-tests.html

EPFL

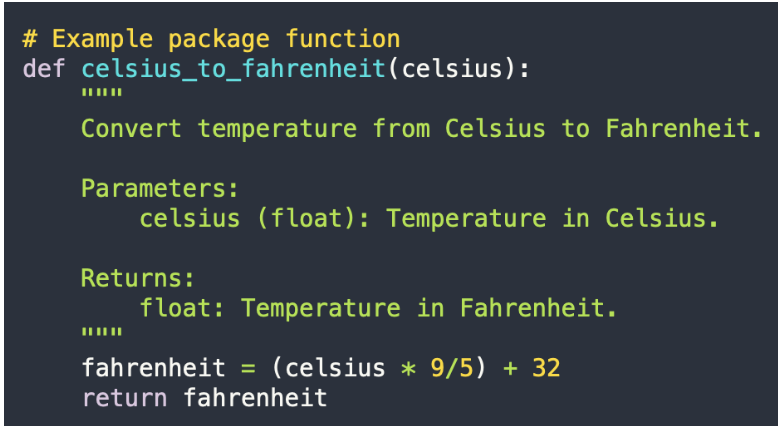
Types of tests

- **Unit Tests:** Verify the functionality of individual components in isolation.
- **Integration Tests:** Check the interactions between combined components to ensure they work together correctly.
- **End-to-End Tests:** Evaluate the entire system’s performance and functionality from start to finish in real-world scenarios.

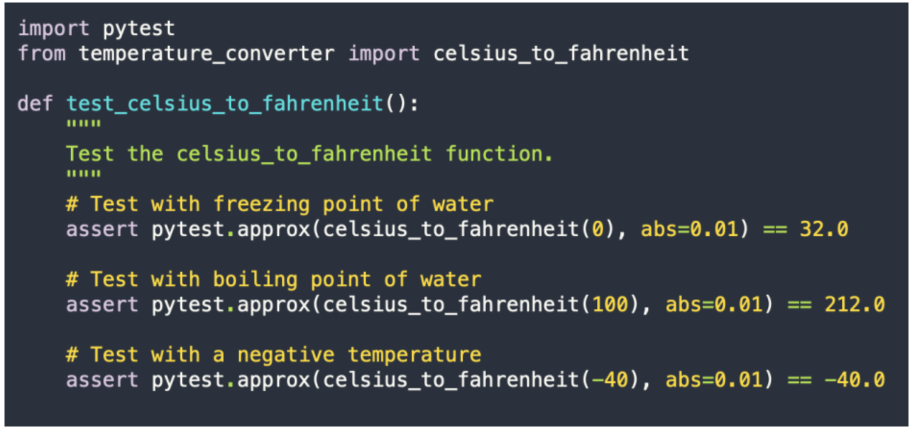
We will focus on Unit Tests…

EPFL

Unit tests → test your functions for expected behaviour



```python
# Example package function
def celsius_to_fahrenheit(celsius):
    """
    Convert temperature from Celsius to Fahrenheit.

    Parameters:
        celsius (float): Temperature in Celsius.

    Returns:
        float: Temperature in Fahrenheit.
    """
    fahrenheit = (celsius * 9/5) + 32
    return fahrenheit
```



```python
import pytest
from temperature_converter import celsius_to_fahrenheit

def test_celsius_to_fahrenheit():
    """
    Test the celsius_to_fahrenheit function.
    """
    # Test with freezing point of water
    assert pytest.approx(celsius_to_fahrenheit(0), abs=0.01) == 32.0

    # Test with boiling point of water
    assert pytest.approx(celsius_to_fahrenheit(100), abs=0.01) == 212.0

    # Test with a negative temperature
    assert pytest.approx(celsius_to_fahrenheit(-40), abs=0.01) == -40.0
```

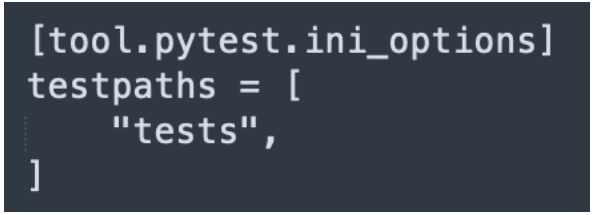
Why **pytest.approx**?
→ Floating-point numbers often have small rounding errors

https://www.pyopensci.org/python-package-guide/tests/run-tests.html

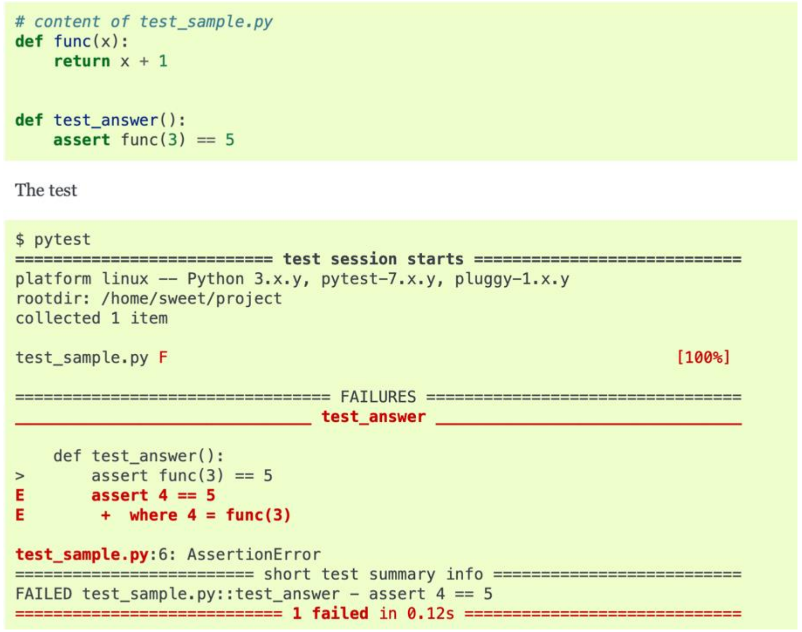
Running tests using pytest

- https://docs.pytest.org/en/7.1.x/getting-started.html
- pip install pytest
- Then run “pytest /path/to/testfolder”

- In copier-liac template
  with pip install –e ”.[test]”
  → just run “pytest”



```
[tool.pytest.ini_options]
testpaths = [
    "tests",
]
```



```
# content of test_sample.py
def func(x):
    return x + 1


def test_answer():
    assert func(3) == 5
```

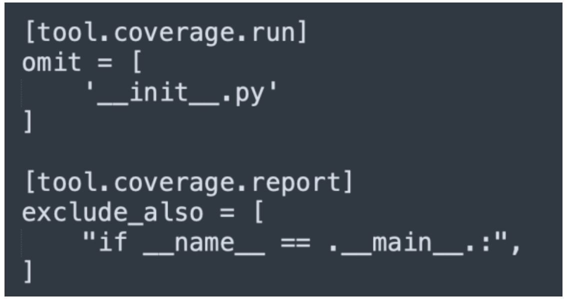
The test

```
$ pytest
=========================== test session starts ============================
platform linux -- Python 3.x.y, pytest-7.x.y, pluggy-1.x.y
rootdir: /home/sweet/project
collected 1 item

test_sample.py F                                                     [100%]

================================= FAILURES =================================
_______________________________ test_answer ________________________________

    def test_answer():
>       assert func(3) == 5
E       assert 4 == 5
E        +  where 4 = func(3)

test_sample.py:6: AssertionError
========================= short test summary info ==========================
FAILED test_sample.py::test_answer - assert 4 == 5
============================ 1 failed in 0.12s =============================
```

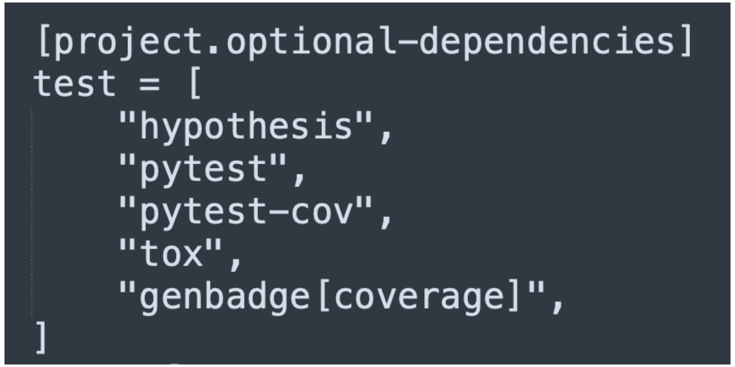
Speaker

Getting the coverage

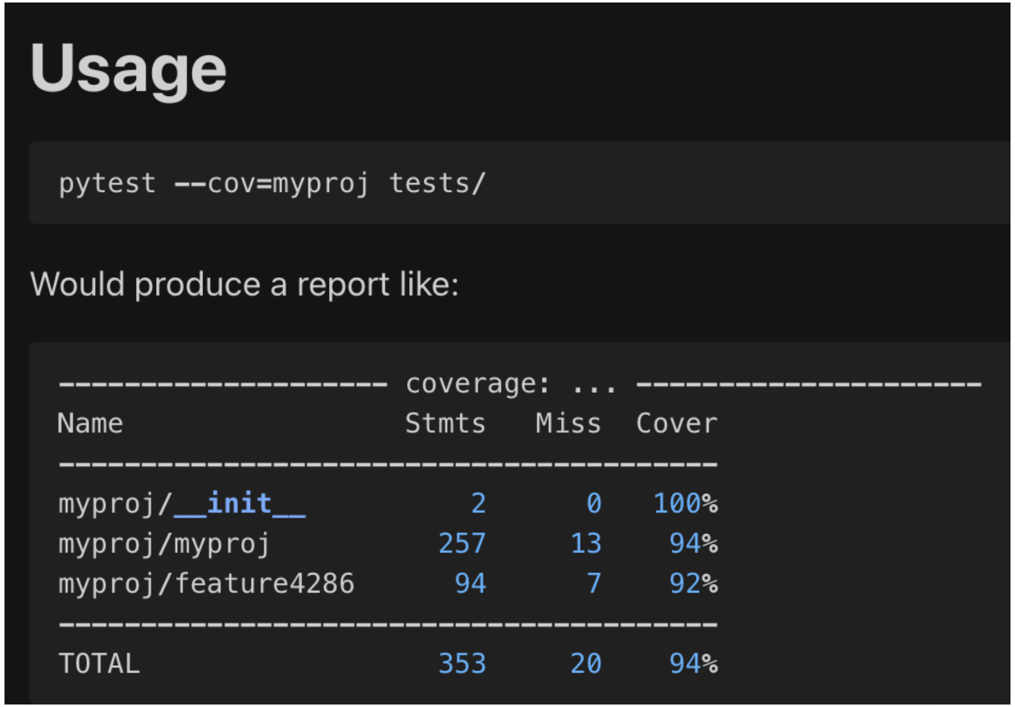
- What % is tested.
- pip install pytest-cov



```
[tool.coverage.run]
omit = [
    '__init__.py'
]

[tool.coverage.report]
exclude_also = [
    "if __name__ == .__main__.:",
]
```



```
[project.optional-dependencies]
test = [
    "hypothesis",
    "pytest",
    "pytest-cov",
    "tox",
    "genbadge[coverage]",
]
```



Usage

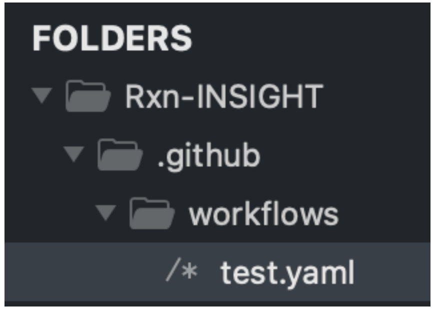
```
pytest --cov=myproj tests/
```

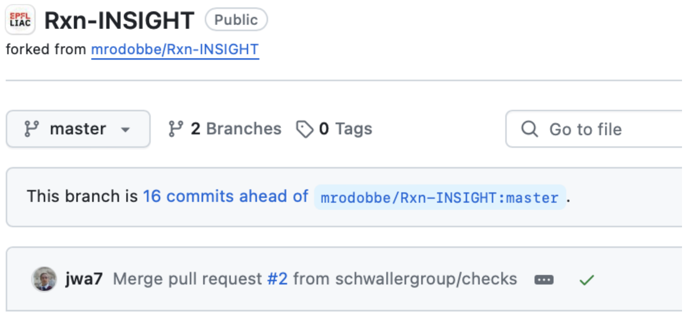
Would produce a report like:

```
-------------------- coverage: ... ---------------------
Name                 Stmts   Miss  Cover
----------------------------------------
myproj/__init__          2      0   100%
myproj/myproj          257     13    94%
myproj/feature4286      94      7    92%
----------------------------------------
TOTAL                  353     20    94%
```

More automation ☺

GitHub actions



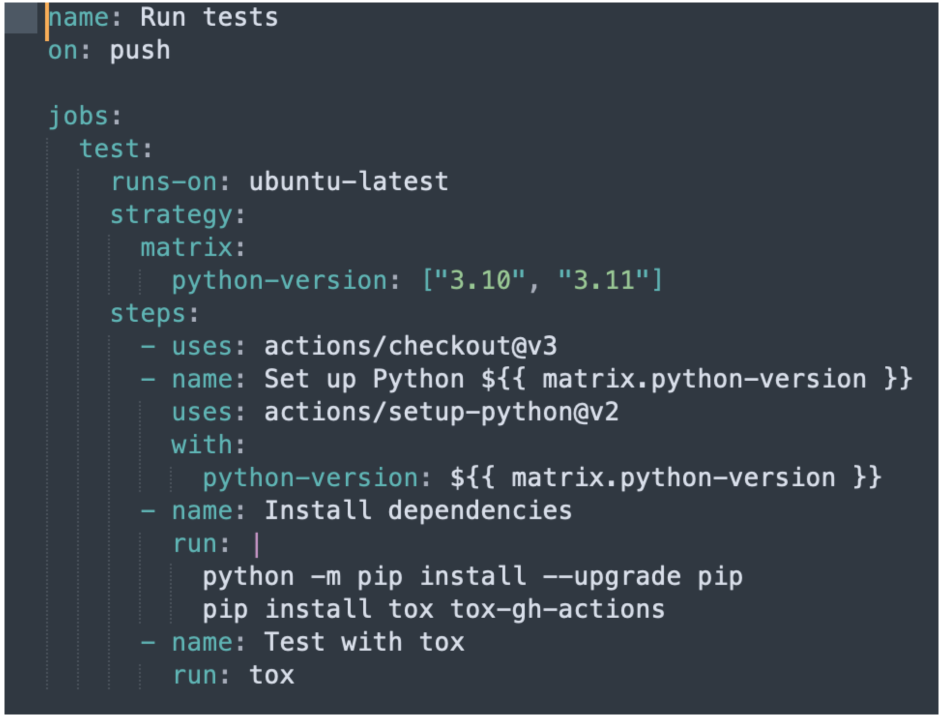

Rxn-INSIGHT Public
forked from mrodobbe/Rxn-INSIGHT

master 2 Branches 0 Tags Go to file

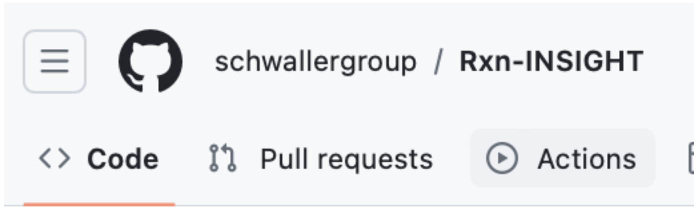
This branch is 16 commits ahead of mrodobbe/Rxn-INSIGHT:master .

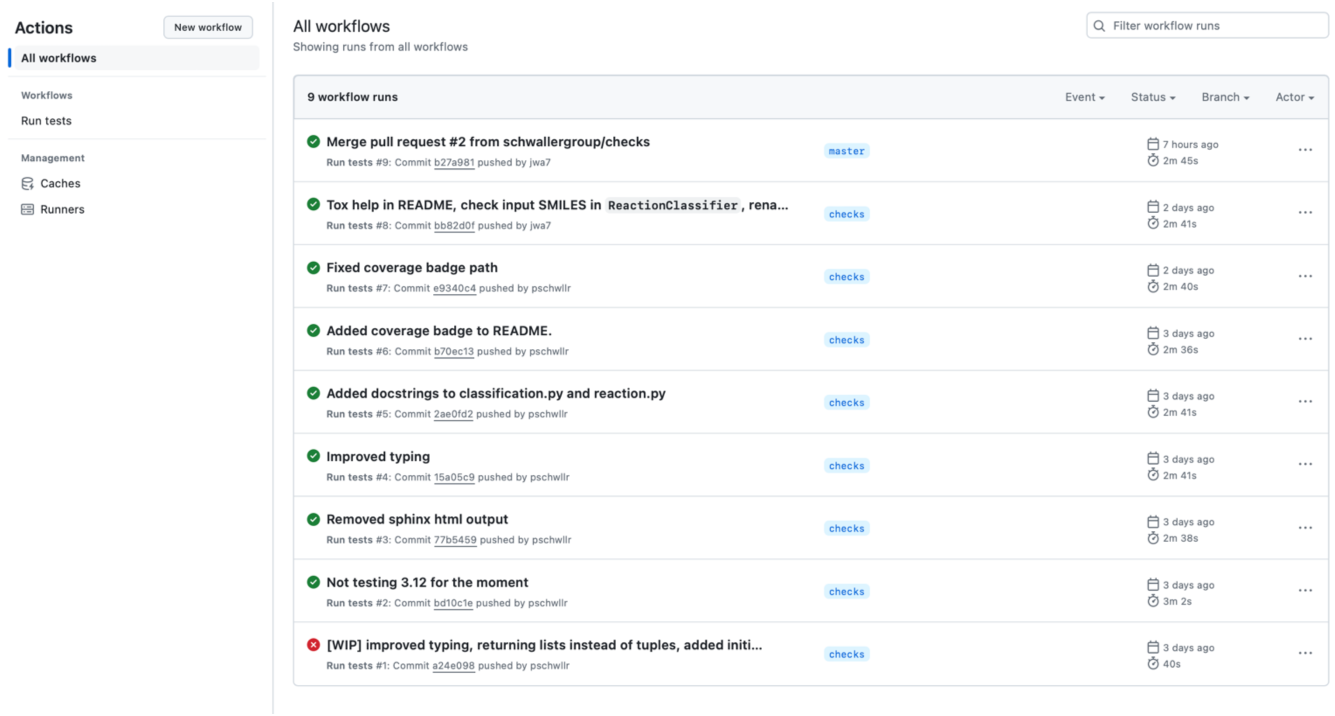
jwa7 Merge pull request #2 from schwallergroup/checks

Same in the liac copier template.


```yaml
name: Run tests
on: push

jobs:
  test:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        python-version: ["3.10", "3.11"]
    steps:
      - uses: actions/checkout@v3
      - name: Set up Python ${{ matrix.python-version }}
        uses: actions/setup-python@v2
        with:
          python-version: ${{ matrix.python-version }}
      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install tox tox-gh-actions
      - name: Test with tox
        run: tox
```

Github actions





Actions New workflow

All workflows

Workflows

Run tests

Management

Caches

Runners

All workflows

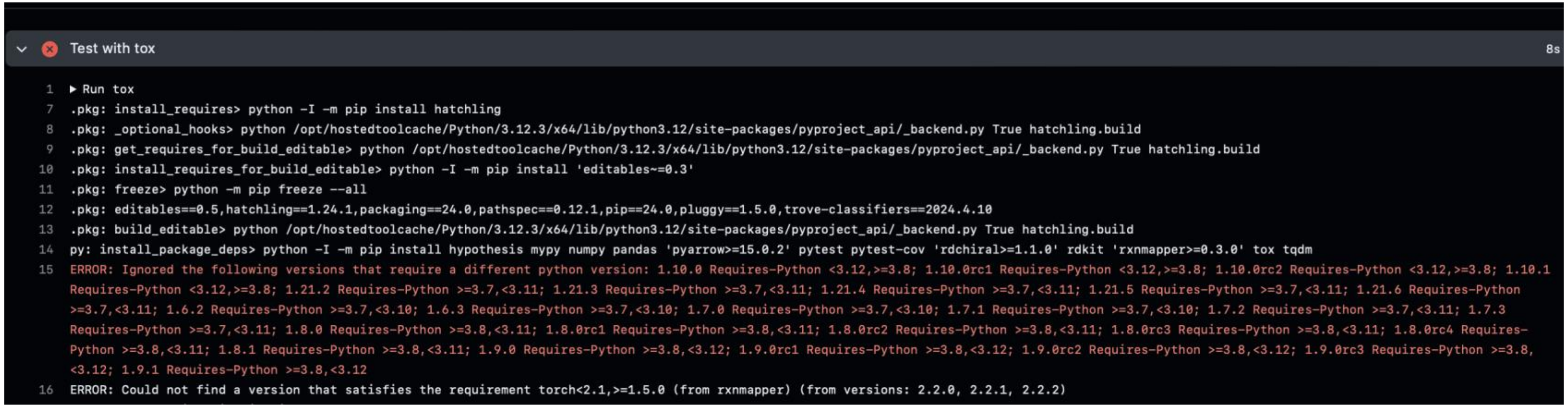
Showing runs from all workflows

Filter workflow runs

9 workflow runs Event Status Branch Actor

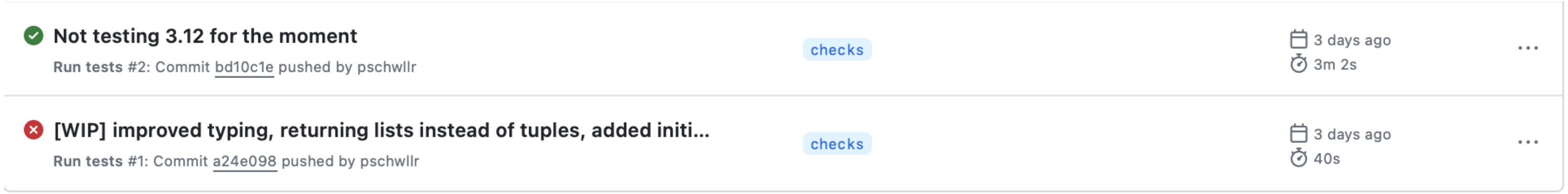
Workflow run	Branch	Time
✓ Merge pull request #2 from schwallergroup/checks Run tests #9: Commit b27a981 pushed by jwa7	master	7 hours ago 2m 45s
✓ Tox help in README, check input SMILES in `ReactionClassifier`, rena... Run tests #8: Commit bb82d0f pushed by jwa7	checks	2 days ago 2m 41s
✓ Fixed coverage badge path Run tests #7: Commit e9340c4 pushed by pschwllr	checks	2 days ago 2m 40s
✓ Added coverage badge to README. Run tests #6: Commit b70ec13 pushed by pschwllr	checks	3 days ago 2m 36s
✓ Added docstrings to classification.py and reaction.py Run tests #5: Commit 2ae0fd2 pushed by pschwllr	checks	3 days ago 2m 41s
✓ Improved typing Run tests #4: Commit 15a05c9 pushed by pschwllr	checks	3 days ago 2m 41s
✓ Removed sphinx html output Run tests #3: Commit 77b5459 pushed by pschwllr	checks	3 days ago 2m 38s
✓ Not testing 3.12 for the moment Run tests #2: Commit bd10c1e pushed by pschwllr	checks	3 days ago 3m 2s
✗ [WIP] improved typing, returning lists instead of tuples, added initi... Run tests #1: Commit a24e098 pushed by pschwllr	checks	3 days ago 40s

How did I find the error?



Test with tox 8s

```
▶ Run tox
.pkg: install_requires> python -I -m pip install hatchling
.pkg: _optional_hooks> python /opt/hostedtoolcache/Python/3.12.3/x64/lib/python3.12/site-packages/pyproject_api/_backend.py True hatchling.build
.pkg: get_requires_for_build_editable> python /opt/hostedtoolcache/Python/3.12.3/x64/lib/python3.12/site-packages/pyproject_api/_backend.py True hatchling.build
.pkg: install_requires_for_build_editable> python -I -m pip install 'editables~=0.3'
.pkg: freeze> python -m pip freeze --all
.pkg: editables==0.5,hatchling==1.24.1,packaging==24.0,pathspec==0.12.1,pip==24.0,pluggy==1.5.0,trove-classifiers==2024.4.10
.pkg: build_editable> python /opt/hostedtoolcache/Python/3.12.3/x64/lib/python3.12/site-packages/pyproject_api/_backend.py True hatchling.build
py: install_package_deps> python -I -m pip install hypothesis mypy numpy pandas 'pyarrow>=15.0.2' pytest pytest-cov 'rdchiral>=1.1.0' rdkit 'rxnmapper>=0.3.0' tox tqdm
ERROR: Ignored the following versions that require a different python version: 1.10.0 Requires-Python <3.12,>=3.8; 1.10.0rc1 Requires-Python <3.12,>=3.8; 1.10.0rc2 Requires-Python <3.12,>=3.8; 1.10.1
Requires-Python <3.12,>=3.8; 1.21.2 Requires-Python >=3.7,<3.11; 1.21.3 Requires-Python >=3.7,<3.11; 1.21.4 Requires-Python >=3.7,<3.11; 1.21.5 Requires-Python >=3.7,<3.11; 1.21.6 Requires-Python
>=3.7,<3.11; 1.6.2 Requires-Python >=3.7,<3.10; 1.6.3 Requires-Python >=3.7,<3.10; 1.7.0 Requires-Python >=3.7,<3.10; 1.7.1 Requires-Python >=3.7,<3.10; 1.7.2 Requires-Python >=3.7,<3.11; 1.7.3
Requires-Python >=3.7,<3.11; 1.8.0 Requires-Python >=3.8,<3.11; 1.8.0rc1 Requires-Python >=3.8,<3.11; 1.8.0rc2 Requires-Python >=3.8,<3.11; 1.8.0rc3 Requires-Python >=3.8,<3.11; 1.8.0rc4 Requires-
Python >=3.8,<3.11; 1.8.1 Requires-Python >=3.8,<3.11; 1.9.0 Requires-Python >=3.8,<3.12; 1.9.0rc1 Requires-Python >=3.8,<3.12; 1.9.0rc2 Requires-Python >=3.8,<3.12; 1.9.0rc3 Requires-Python >=3.8,
<3.12; 1.9.1 Requires-Python >=3.8,<3.12
ERROR: Could not find a version that satisfies the requirement torch<2.1,>=1.5.0 (from rxnmapper) (from versions: 2.2.0, 2.2.1, 2.2.2)
```



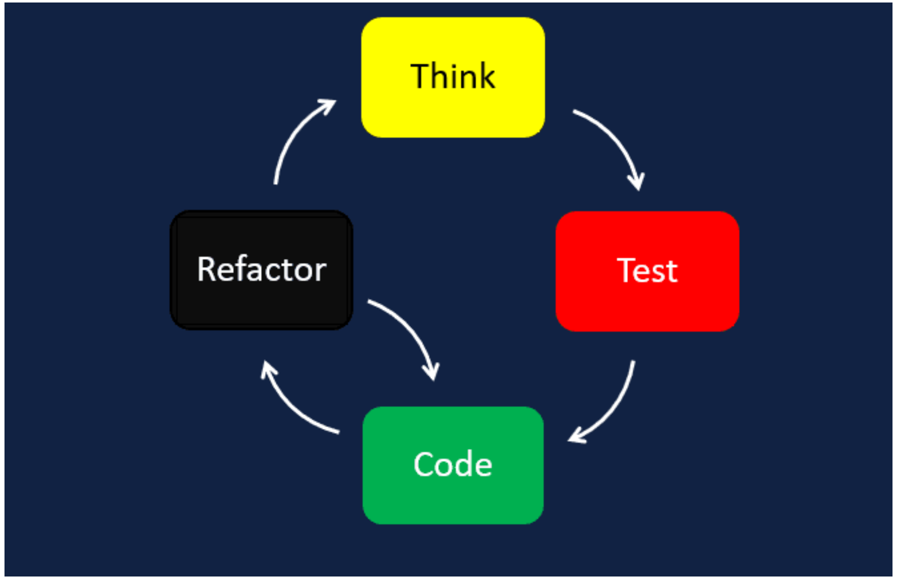
✓ **Not testing 3.12 for the moment**
Run tests #2: Commit bd10c1e pushed by pschwllr — checks — 3 days ago — 3m 2s

✗ **[WIP] improved typing, returning lists instead of tuples, added initi...**
Run tests #1: Commit a24e098 pushed by pschwllr — checks — 3 days ago — 40s

TDD – Test driven development (ideal, but can be slow)

What feature do I need? What should my function/program do?

Refactor = make the code cleaner, better, faster.
As you have a test, you can always evaluate that the code is correct.



Write a test with the expected behaviour.
→ Fails

Code until the test passes.

TDD = writing the tests before the code

Python Packaging Toolbox

Tests
→ pytest

pyproject.toml

Pre-commit hooks (advanced)
→ Ruff style checks
→ Check for too large files
→ Check for merge conflicts

.pre-commit-config.yaml

Documentation
→ sphinx

tox.ini

Tools for better code

Type hints / typing (advanced)
→ mypy

pyproject.toml

Automation
→ tox

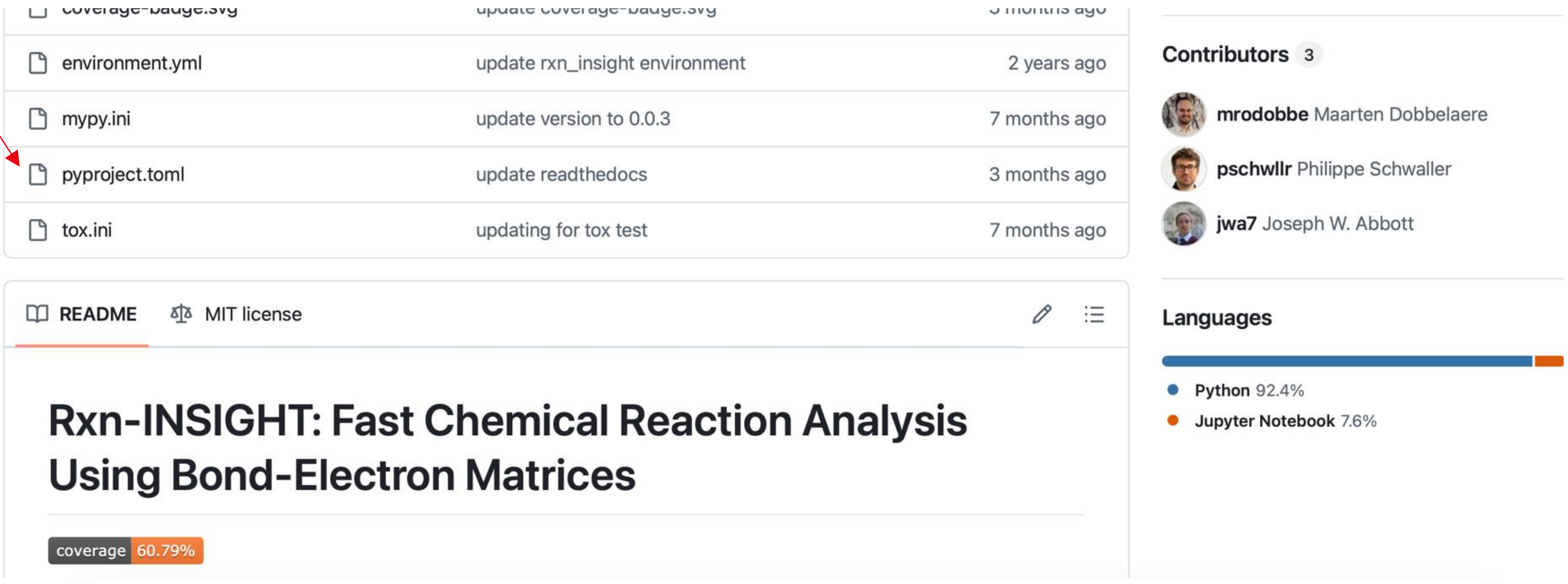
tox.ini

GitHub workflows
→ GitHub actions

.github/workflows/*

Example of us turning non-packaged code into a python package.

Last year this was not there yet.



environment.yml	update rxn_insight environment	2 years ago
mypy.ini	update version to 0.0.3	7 months ago
pyproject.toml	update readthedocs	3 months ago
tox.ini	updating for tox test	7 months ago

README MIT license

Rxn-INSIGHT: Fast Chemical Reaction Analysis Using Bond-Electron Matrices

coverage 60.79%

Contributors 3

mrodobbe Maarten Dobbelaere

pschwllr Philippe Schwaller

jwa7 Joseph W. Abbott

Languages

● Python 92.4%

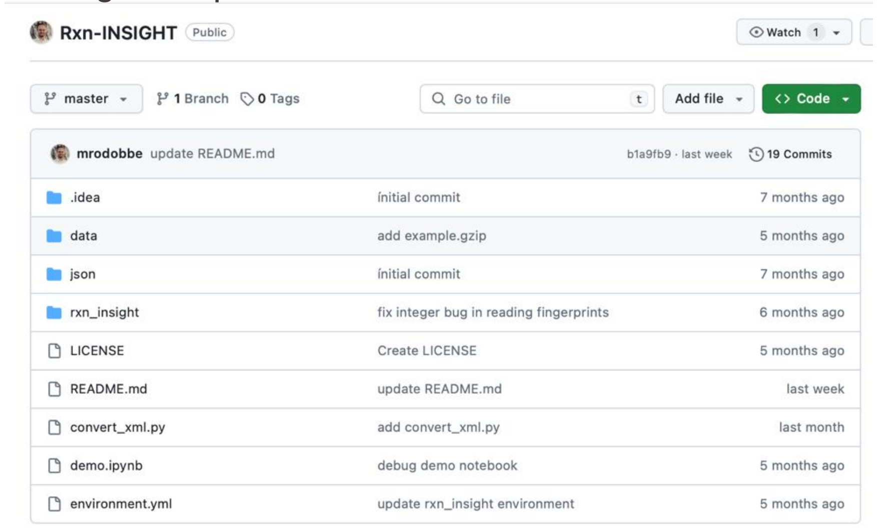
● Jupyter Notebook 7.6%

https://github.com/mrodobbe/Rxn-INSIGHT

Turning RXN-Insight into a package

- First goal, get it to pip install (simple)
- Extended goal (make code nicer, more challenging)

Original repo



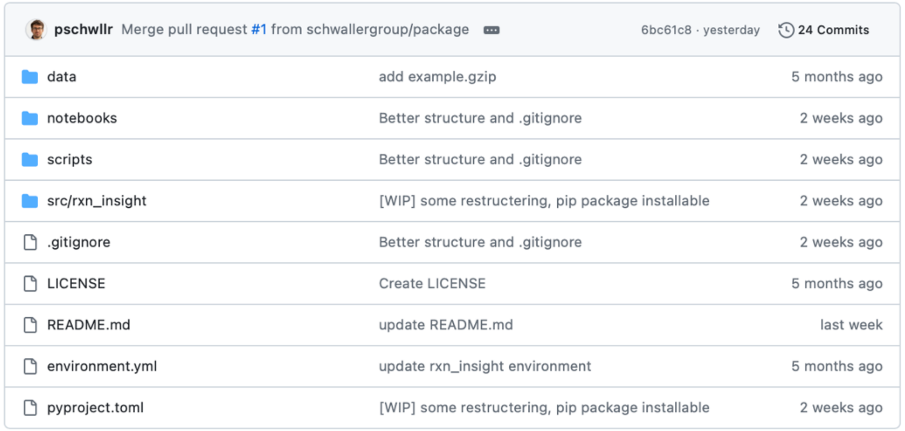
Rxn-INSIGHT Public — Watch 1

master · 1 Branch · 0 Tags · Go to file · Add file · Code

mrodobbe update README.md — b1a9fb9 · last week — 19 Commits

.idea	initial commit	7 months ago
data	add example.gzip	5 months ago
json	initial commit	7 months ago
rxn_insight	fix integer bug in reading fingerprints	6 months ago
LICENSE	Create LICENSE	5 months ago
README.md	update README.md	last week
convert_xml.py	add convert_xml.py	last month
demo.ipynb	debug demo notebook	5 months ago
environment.yml	update rxn_insight environment	5 months ago

SchwallerGroup fork

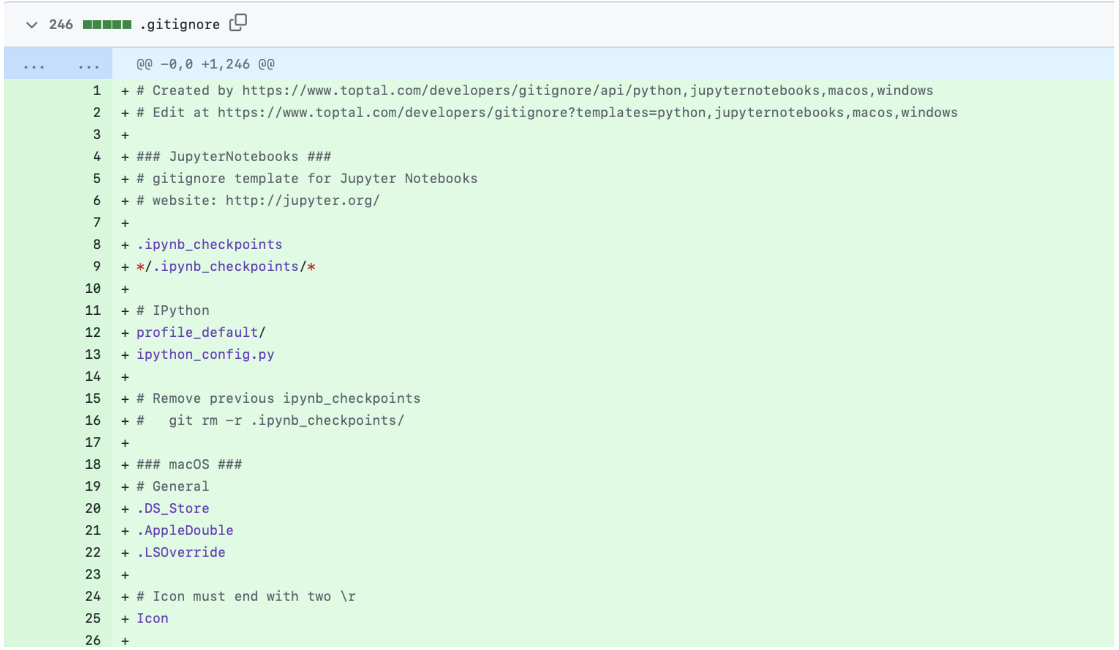


pschwllr Merge pull request #1 from schwallergroup/package — 6bc61c8 · yesterday — 24 Commits

data	add example.gzip	5 months ago
notebooks	Better structure and .gitignore	2 weeks ago
scripts	Better structure and .gitignore	2 weeks ago
src/rxn_insight	[WIP] some restructering, pip package installable	2 weeks ago
.gitignore	Better structure and .gitignore	2 weeks ago
LICENSE	Create LICENSE	5 months ago
README.md	update README.md	last week
environment.yml	update rxn_insight environment	5 months ago
pyproject.toml	[WIP] some restructering, pip package installable	2 weeks ago

1st goal – what did I do?

- https://github.com/schwallergroup/Rxn-INSIGHT/pull/1/files (check out if interested in details)
- Gitignore was missing, got one from **gitignore.io**

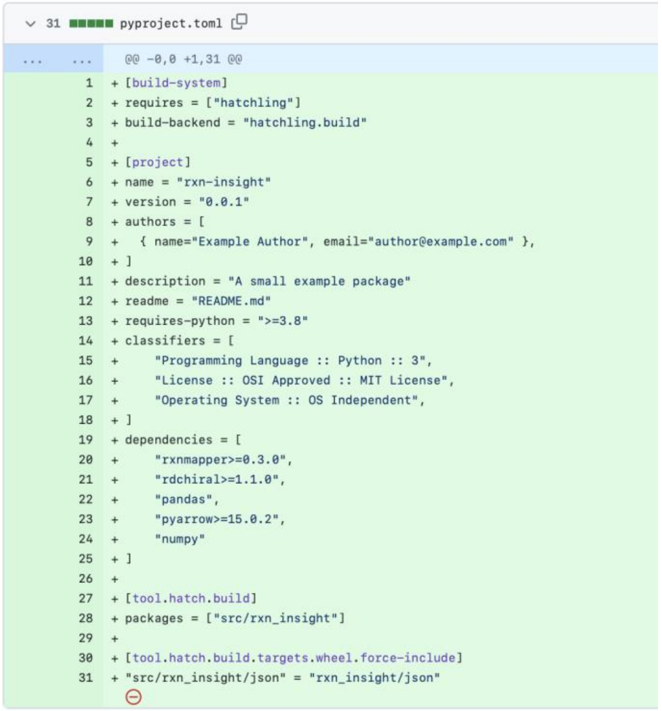


246 .gitignore

```
@@ -0,0 +1,246 @@
+ # Created by https://www.toptal.com/developers/gitignore/api/python,jupyternotebooks,macos,windows
+ # Edit at https://www.toptal.com/developers/gitignore?templates=python,jupyternotebooks,macos,windows
+
+ ### JupyterNotebooks ###
+ # gitignore template for Jupyter Notebooks
+ # website: http://jupyter.org/
+
+ .ipynb_checkpoints
+ */.ipynb_checkpoints/*
+
+ # IPython
+ profile_default/
+ ipython_config.py
+
+ # Remove previous ipynb_checkpoints
+ #   git rm -r .ipynb_checkpoints/
+
+ ### macOS ###
+ # General
+ .DS_Store
+ .AppleDouble
+ .LSOverride
+
+ # Icon must end with two \r
+ Icon
+
```

1st goal – Created a pyproject.toml file.

For the baseline, I included minimal information.



```
31 pyproject.toml
@@ -0,0 +1,31 @@
+ [build-system]
+ requires = ["hatchling"]
+ build-backend = "hatchling.build"
+
+ [project]
+ name = "rxn-insight"
+ version = "0.0.1"
+ authors = [
+   { name="Example Author", email="author@example.com" },
+ ]
+ description = "A small example package"
+ readme = "README.md"
+ requires-python = ">=3.8"
+ classifiers = [
+     "Programming Language :: Python :: 3",
+     "License :: OSI Approved :: MIT License",
+     "Operating System :: OS Independent",
+ ]
+ dependencies = [
+     "rxnmapper>=0.3.0",
+     "rdchiral>=1.1.0",
+     "pandas",
+     "pyarrow>=15.0.2",
+     "numpy"
+ ]
+
+ [tool.hatch.build]
+ packages = ["src/rxn_insight"]
+
+ [tool.hatch.build.targets.wheel.force-include]
+ "src/rxn_insight/json" = "rxn_insight/json"
```

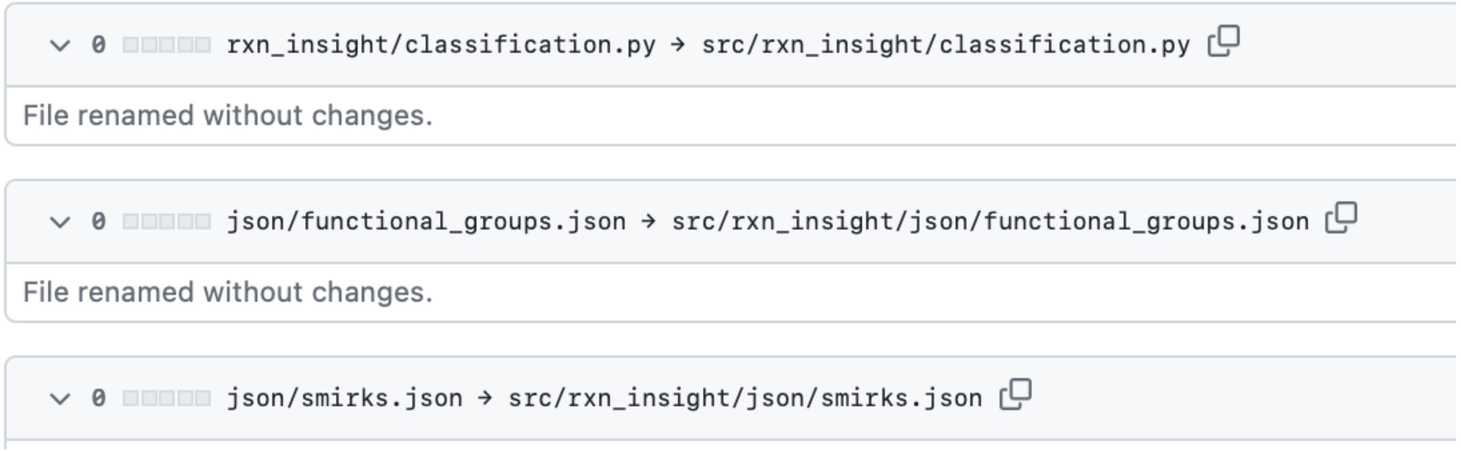
Hatchling build backend

Dependencies (note: I forgot rdkit)

Changed the package location to “src/rxn_insight”

Note: by default it will only include the .py files into the packages,
The json folder contains crucial information for the package,
so I had to force-include it.

1st goal – Moved rxn_insight and json into a src folder



```
0 rxn_insight/classification.py → src/rxn_insight/classification.py
```

File renamed without changes.

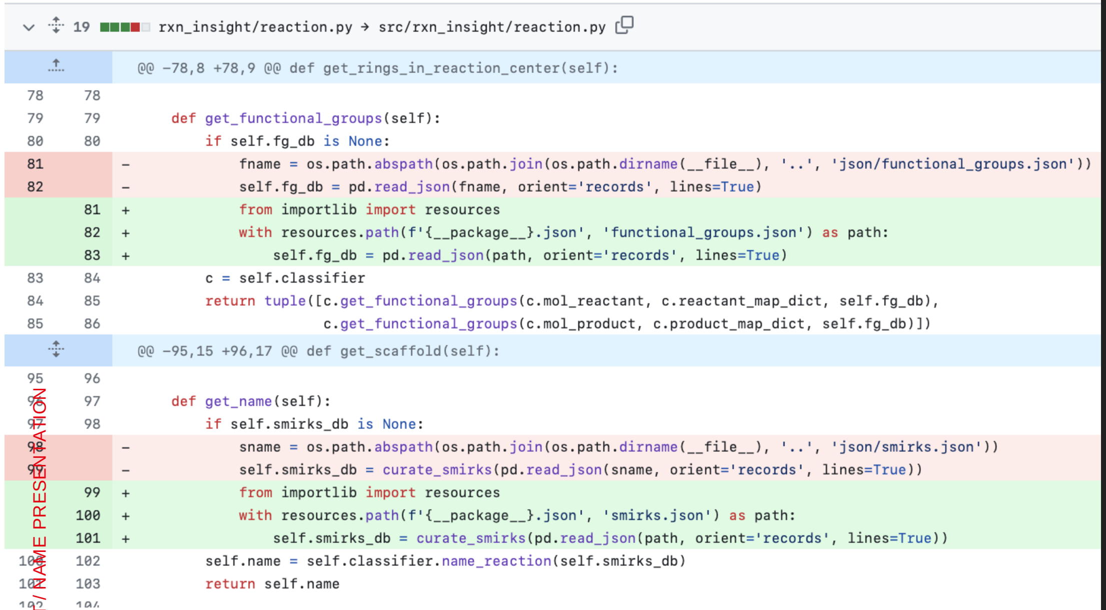
```
0 json/functional_groups.json → src/rxn_insight/json/functional_groups.json
```

File renamed without changes.

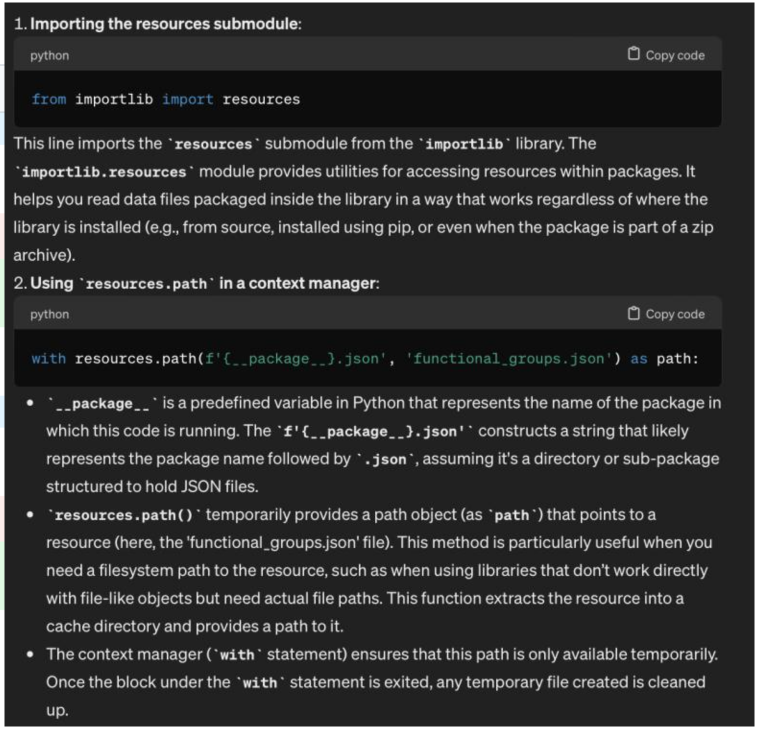
```
0 json/smirks.json → src/rxn_insight/json/smirks.json
```

Did the same with the json files, as I wanted them in the package (more later)

1st goal – access the json files in the installed package



```
19 rxn_insight/reaction.py → src/rxn_insight/reaction.py

@@ -78,8 +78,9 @@ def get_rings_in_reaction_center(self):

    def get_functional_groups(self):
        if self.fg_db is None:
-           fname = os.path.abspath(os.path.join(os.path.dirname(__file__), '..', 'json/functional_groups.json'))
-           self.fg_db = pd.read_json(fname, orient='records', lines=True)
+           from importlib import resources
+           with resources.path(f'{__package__}.json', 'functional_groups.json') as path:
+               self.fg_db = pd.read_json(path, orient='records', lines=True)
        c = self.classifier
        return tuple([c.get_functional_groups(c.mol_reactant, c.reactant_map_dict, self.fg_db),
                      c.get_functional_groups(c.mol_product, c.product_map_dict, self.fg_db)])
@@ -95,15 +96,17 @@ def get_scaffold(self):

    def get_name(self):
        if self.smirks_db is None:
-           sname = os.path.abspath(os.path.join(os.path.dirname(__file__), '..', 'json/smirks.json'))
-           self.smirks_db = curate_smirks(pd.read_json(sname, orient='records', lines=True))
+           from importlib import resources
+           with resources.path(f'{__package__}.json', 'smirks.json') as path:
+               self.smirks_db = curate_smirks(pd.read_json(path, orient='records', lines=True))
        self.name = self.classifier.name_reaction(self.smirks_db)
        return self.name
```



1. **Importing the resources submodule:**

```python
from importlib import resources
```

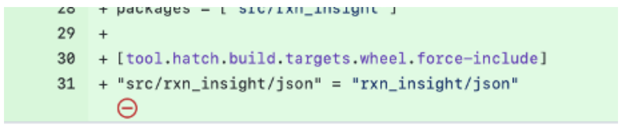
This line imports the `resources` submodule from the `importlib` library. The `importlib.resources` module provides utilities for accessing resources within packages. It helps you read data files packaged inside the library in a way that works regardless of where the library is installed (e.g., from source, installed using pip, or even when the package is part of a zip archive).

2. **Using `resources.path` in a context manager:**

```python
with resources.path(f'{__package__}.json', 'functional_groups.json') as path:
```

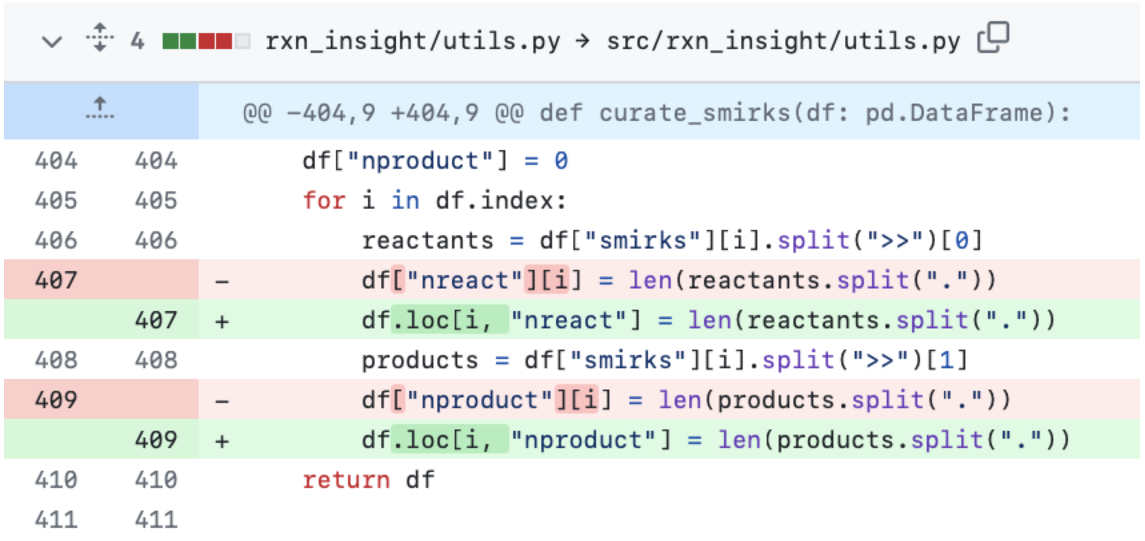
- `__package__` is a predefined variable in Python that represents the name of the package in which this code is running. The `f'{__package__}.json'` constructs a string that likely represents the package name followed by `.json`, assuming it's a directory or sub-package structured to hold JSON files.
- `resources.path()` temporarily provides a path object (as `path`) that points to a resource (here, the 'functional_groups.json' file). This method is particularly useful when you need a filesystem path to the resource, such as when using libraries that don't work directly with file-like objects but need actual file paths. This function extracts the resource into a cache directory and provides a path to it.
- The context manager (`with` statement) ensures that this path is only available temporarily. Once the block under the `with` statement is exited, any temporary file created is cleaned up.

This combined with the force-include in the pyproject.toml, makes the json files available when after you build the package.



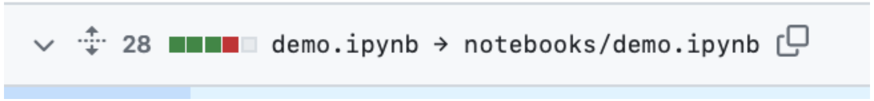
```
+ packages = [ "src/rxn_insight" ]
+
+ [tool.hatch.build.targets.wheel.force-include]
+ "src/rxn_insight/json" = "rxn_insight/json"
```

1st goal – Fixed some pandas code that will be depreciated in the future

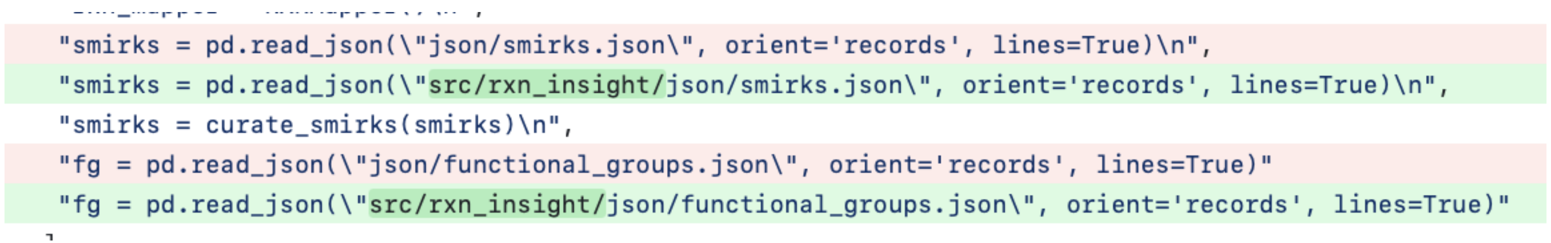


```
4  rxn_insight/utils.py → src/rxn_insight/utils.py

@@ -404,9 +404,9 @@ def curate_smirks(df: pd.DataFrame):
    df["nproduct"] = 0
    for i in df.index:
        reactants = df["smirks"][i].split(">>")[0]
-       df["nreact"][i] = len(reactants.split("."))
+       df.loc[i, "nreact"] = len(reactants.split("."))
        products = df["smirks"][i].split(">>")[1]
-       df["nproduct"][i] = len(products.split("."))
+       df.loc[i, "nproduct"] = len(products.split("."))
    return df

```

1st goal – moved demo.ipynb into notebooks folder



```
28 demo.ipynb → notebooks/demo.ipynb
```

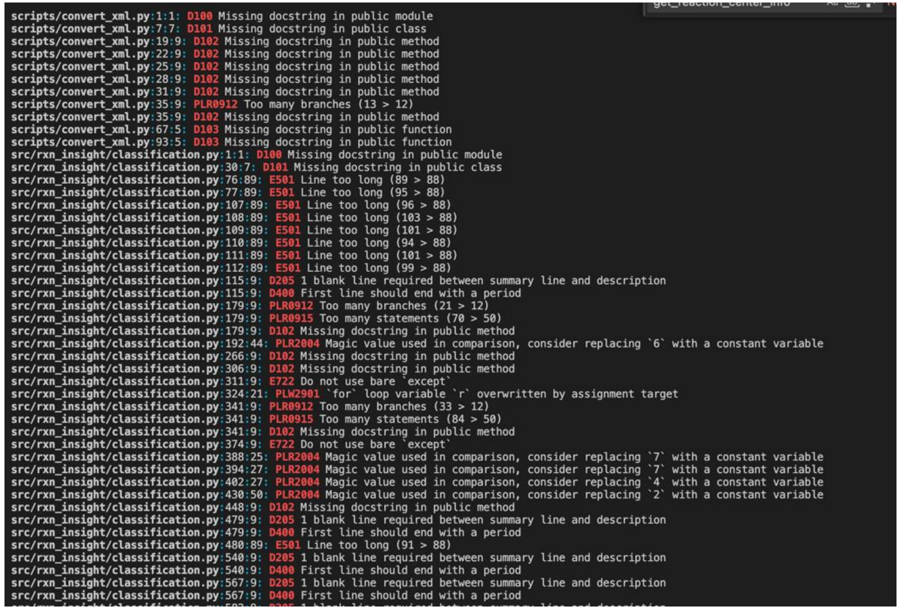


```
"smirks = pd.read_json(\"json/smirks.json\", orient='records', lines=True)\n",
"smirks = pd.read_json(\"src/rxn_insight/json/smirks.json\", orient='records', lines=True)\n",
"smirks = curate_smirks(smirks)\n",
"fg = pd.read_json(\"json/functional_groups.json\", orient='records', lines=True)"
"fg = pd.read_json(\"src/rxn_insight/json/functional_groups.json\", orient='records', lines=True)"
```

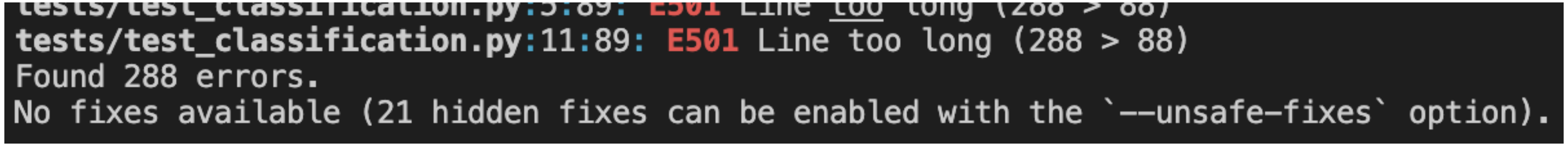
- Had to adapt some paths in the notebook.

Extended goal

- Fixed **>120 typing issues**…
  typing is nice,
  but if you want to do it,
  do it from the beginning
- Precommit checks with ruff
  → just too many to do the complete refactoring…



```
scripts/convert_xml.py:1:1: D100 Missing docstring in public module
scripts/convert_xml.py:7:7: D101 Missing docstring in public class
scripts/convert_xml.py:19:9: D102 Missing docstring in public method
scripts/convert_xml.py:22:9: D102 Missing docstring in public method
scripts/convert_xml.py:25:9: D102 Missing docstring in public method
scripts/convert_xml.py:28:9: D102 Missing docstring in public method
scripts/convert_xml.py:31:9: D102 Missing docstring in public method
scripts/convert_xml.py:35:9: PLR0912 Too many branches (13 > 12)
scripts/convert_xml.py:35:9: D102 Missing docstring in public method
scripts/convert_xml.py:67:5: D103 Missing docstring in public function
scripts/convert_xml.py:93:5: D103 Missing docstring in public function
src/rxn_insight/classification.py:1:1: D100 Missing docstring in public module
src/rxn_insight/classification.py:30:7: D101 Missing docstring in public class
src/rxn_insight/classification.py:76:89: E501 Line too long (89 > 88)
src/rxn_insight/classification.py:77:89: E501 Line too long (95 > 88)
src/rxn_insight/classification.py:107:89: E501 Line too long (96 > 88)
src/rxn_insight/classification.py:108:89: E501 Line too long (103 > 88)
src/rxn_insight/classification.py:109:89: E501 Line too long (101 > 88)
src/rxn_insight/classification.py:110:89: E501 Line too long (94 > 88)
src/rxn_insight/classification.py:111:89: E501 Line too long (101 > 88)
src/rxn_insight/classification.py:112:89: E501 Line too long (99 > 88)
src/rxn_insight/classification.py:115:9: D205 1 blank line required between summary line and description
src/rxn_insight/classification.py:115:9: D400 First line should end with a period
src/rxn_insight/classification.py:179:9: PLR0912 Too many branches (21 > 12)
src/rxn_insight/classification.py:179:9: PLR0915 Too many statements (70 > 50)
src/rxn_insight/classification.py:179:9: D102 Missing docstring in public method
src/rxn_insight/classification.py:192:44: PLR2004 Magic value used in comparison, consider replacing `6` with a constant variable
src/rxn_insight/classification.py:266:9: D102 Missing docstring in public method
src/rxn_insight/classification.py:306:9: D102 Missing docstring in public method
src/rxn_insight/classification.py:311:9: E722 Do not use bare `except`
src/rxn_insight/classification.py:324:21: PLW2901 `for` loop variable `r` overwritten by assignment target
src/rxn_insight/classification.py:341:9: PLR0912 Too many branches (33 > 12)
src/rxn_insight/classification.py:341:9: PLR0915 Too many statements (84 > 50)
src/rxn_insight/classification.py:341:9: D102 Missing docstring in public method
src/rxn_insight/classification.py:374:9: E722 Do not use bare `except`
src/rxn_insight/classification.py:388:25: PLR2004 Magic value used in comparison, consider replacing `7` with a constant variable
src/rxn_insight/classification.py:394:27: PLR2004 Magic value used in comparison, consider replacing `7` with a constant variable
src/rxn_insight/classification.py:402:27: PLR2004 Magic value used in comparison, consider replacing `4` with a constant variable
src/rxn_insight/classification.py:430:50: PLR2004 Magic value used in comparison, consider replacing `2` with a constant variable
src/rxn_insight/classification.py:448:9: D102 Missing docstring in public method
src/rxn_insight/classification.py:479:9: D205 1 blank line required between summary line and description
src/rxn_insight/classification.py:479:9: D400 First line should end with a period
src/rxn_insight/classification.py:480:89: E501 Line too long (91 > 88)
src/rxn_insight/classification.py:540:9: D205 1 blank line required between summary line and description
src/rxn_insight/classification.py:540:9: D400 First line should end with a period
src/rxn_insight/classification.py:567:9: D205 1 blank line required between summary line and description
src/rxn_insight/classification.py:567:9: D400 First line should end with a period
```

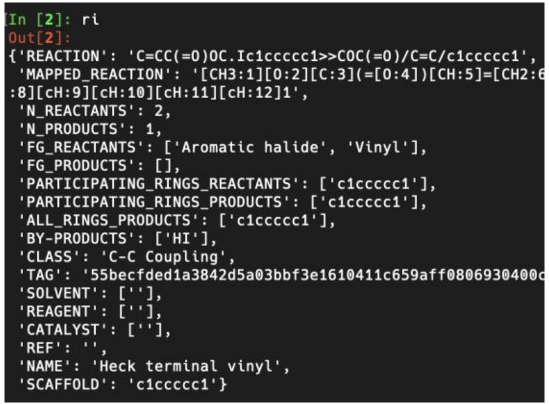


```
tests/test_classification.py:5:89: E501 Line too long (288 > 88)
tests/test_classification.py:11:89: E501 Line too long (288 > 88)
Found 288 errors.
No fixes available (21 hidden fixes can be enabled with the `--unsafe-fixes` option).
```

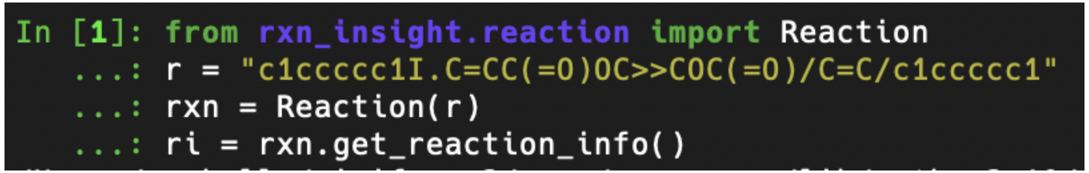
Speaker

What can I do with that?

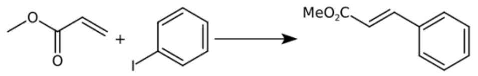
- Go to https://github.com/schwallergroup/Rxn-INSIGHT
- Get the address to clone
  - “git clone https://github.com/schwallergroup/Rxn-INSIGHT.git”
  - “cd Rxn-INSIGHT”
  - *Activate a conda environment*
  - “pip install –e .” or pip install –e “.[test,doc]” (for additional dependencies)
- And then I can use the installed package in another project, without being in the same folder.



```
In [2]: ri
Out[2]:
{'REACTION': 'C=CC(=O)OC.Ic1ccccc1>>COC(=O)/C=C/c1ccccc1',
 'MAPPED_REACTION': '[CH3:1][O:2][C:3](=[O:4])[CH:5]=[CH2:6
:8][cH:9][cH:10][cH:11][cH:12]1',
 'N_REACTANTS': 2,
 'N_PRODUCTS': 1,
 'FG_REACTANTS': ['Aromatic halide', 'Vinyl'],
 'FG_PRODUCTS': [],
 'PARTICIPATING_RINGS_REACTANTS': ['c1ccccc1'],
 'PARTICIPATING_RINGS_PRODUCTS': ['c1ccccc1'],
 'ALL_RINGS_PRODUCTS': ['c1ccccc1'],
 'BY-PRODUCTS': ['HI'],
 'CLASS': 'C-C Coupling',
 'TAG': '55becfded1a3842d5a03bbf3e1610411c659aff0806930400c
 'SOLVENT': [''],
 'REAGENT': [''],
 'CATALYST': [''],
 'REF': '',
 'NAME': 'Heck terminal vinyl',
 'SCAFFOLD': 'c1ccccc1'}
```

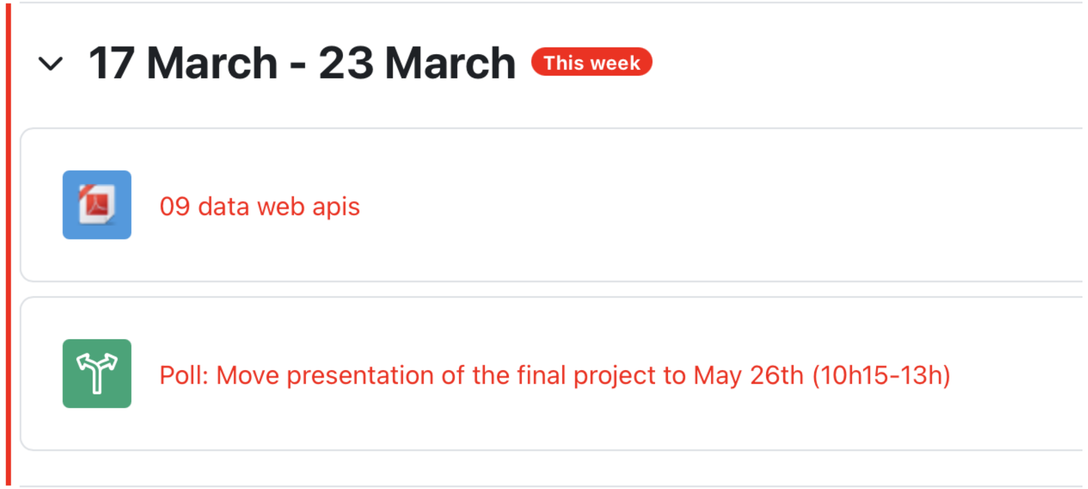


```
In [1]: from rxn_insight.reaction import Reaction
   ...: r = "c1ccccc1I.C=CC(=O)OC>>COC(=O)/C=C/c1ccccc1"
   ...: rxn = Reaction(r)
   ...: ri = rxn.get_reaction_info()
```



Project sign ups (deadline was yesterday!) - 71/89

Please all register individually if you have not yet done so...



17 March - 23 March This week

09 data web apis

Poll: Move presentation of the final project to May 26th (10h15-13h)