Sharing is Caring

sharing pytest fixtures

-

Brian Okken

Slides + Code

pythontest.com/pycascades-2023

Brian Okken

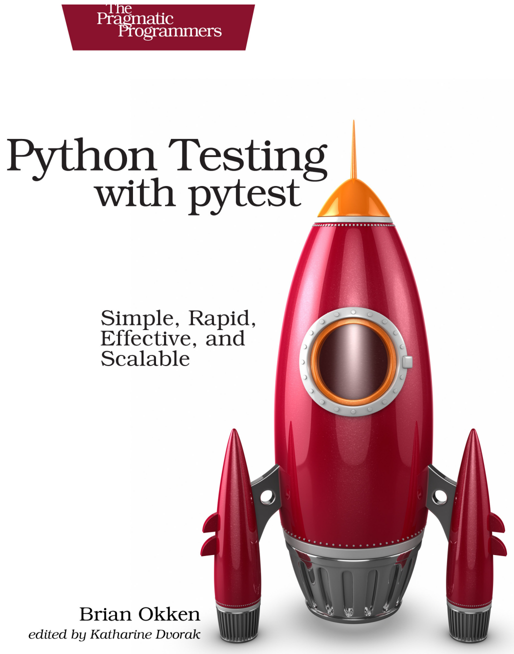
- Podcasts
  Python Bytes / Test & Code

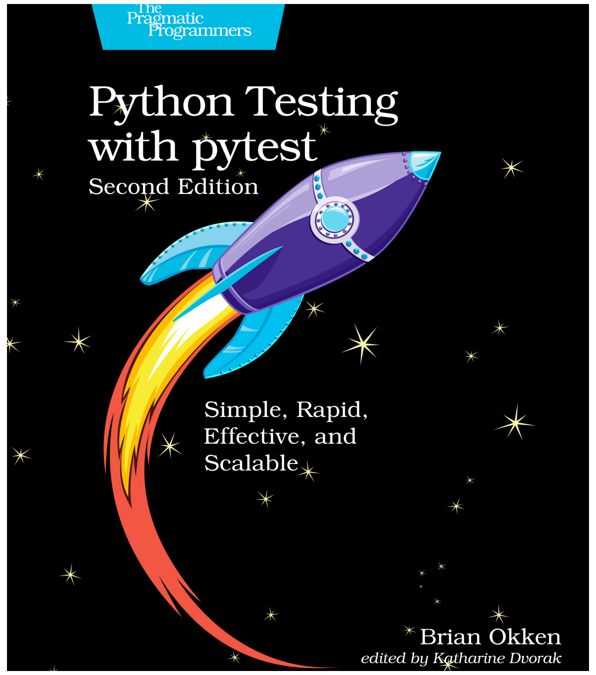
PYTHON BYTES
Python headlines delivered directly to your earbuds

TEST & CODE

Brian Okken

- Podcasts
  Python Bytes / Test & Code
- Books
  Python Testing with pytest



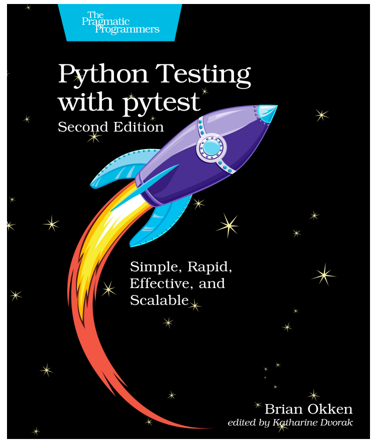


Brian Okken

- Podcasts
  Python Bytes / Test & Code
- Books
  Python Testing with pytest
- Training
  pythontest.com/courses
  pythontest.com/training

PYTHON BYTES
Python headlines delivered directly to your earbuds

TEST & CODE



Getting Started with pytest

pytest

Python's most powerful test framework

Brian Okken

- Podcasts
  Python Bytes / Test & Code

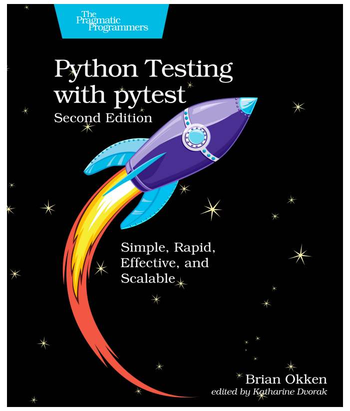
- Books
  Python Testing with pytest

- Training
  pythontest.com/courses
  pythontest.com/training

- Lead Software Engineer

PYTHON BYTES
Python headlines delivered directly to your earbuds

TEST & CODE



System level testing

Is what led me to pytest

Why?

multi-level pytest fixtures

- Setup: connect, reset, configure, …
- Teardown: reset switches, check logs, …
- Multiple Levels: session, module, function, …


pytest fixtures are awesome

pytest is awesome

you are awesome

you're here, right?

you will make

fixtures so awesome

you want to share them

Fixture crash course

typical example

```python
import pytest

@pytest.fixture()
def db():
    ... # connect to db
    yield _db
    ... # disconnect
```

typical example

```python
import pytest

@pytest.fixture()
def db():
    ... # connect to db
    yield _db
    ... # disconnect
```

```python
def test_foo(db):
    ...
    result = db.action()
    ...

def test_bar(db):
    ...
    result = db.action()
    ...
```

but frankly

that's too useful for this talk

How about

colorful print statements

```
(venv) $ pytest -s test_example.py
==================== test session starts =====================
collected 1 item

test_example.py
test_colors: this should be red
test_colors: this should be green
test_colors: this should be yellow
test_colors: this should be blue
test_colors: this should be magenta
test_colors: this should be cyan
.

===================== 1 passed in 0.00s ======================
```

let's start with just one

```
(venv) $ pytest -s test_example_two.py
====================== test session starts =======================
collected 1 item

test_example_two.py
test_magenta: this should be magenta
.

======================= 1 passed in 0.01s ========================
```

Using the fixture

```
def test_magenta(magenta):
    print("")
    magenta("this should be magenta")
```

The fixture

```python
import pytest
from functools import partial

MAGENTA = "\x1b[35m"
RESET = "\x1b[0m"
BOLD = "\x1b[1m"

def _color_print(color: str, func_name: str, text: str):
    print(f"{BOLD}{color}{func_name}: {text}{RESET}")

@pytest.fixture()
def magenta(request):
    return partial(_color_print, MAGENTA, request.node.name)
```

More colors

```
@pytest.fixture()
def red(request):
    return partial(_color_print, RED, request.node.name)

@pytest.fixture()
def green(request):
    return partial(_color_print, GREEN, request.node.name)

@pytest.fixture()
def yellow(request):
    return partial(_color_print, YELLOW, request.node.name)

@pytest.fixture()
def blue(request):
    return partial(_color_print, BLUE, request.node.name)

@pytest.fixture()
def magenta(request):
    return partial(_color_print, MAGENTA, request.node.name)

@pytest.fixture()
def cyan(request):
    return partial(_color_print, CYAN, request.node.name)
```

We can use them all!

```
def test_colors(red, green, yellow, blue, magenta, cyan):
    print("")
    red("this should be red")
    green("this should be green")
    yellow("this should be yellow")
    blue("this should be blue")
    magenta("this should be magenta")
    cyan("this should be cyan")
```

Several colors are nice

```
(venv) $ pytest -s test_example.py
===================== test session starts ======================
collected 1 item

test_example.py
test_colors: this should be red
test_colors: this should be green
test_colors: this should be yellow
test_colors: this should be blue
test_colors: this should be magenta
test_colors: this should be cyan
.

====================== 1 passed in 0.00s =======================
```

That was fun

But they can't be shared

Yet

They're in the test file

```python
import pytest
from functools import partial

# Terminal color codes
RED = "\x1b[31m"
GREEN = "\x1b[32m"
YELLOW = "\x1b[33m"
BLUE = "\x1b[34m"
MAGENTA = "\x1b[35m"
CYAN = "\x1b[36m"
RESET = "\x1b[0m"
BOLD = "\x1b[1m"

def _color_print(color: str, func_name: str, text: str):
    print(f"{BOLD}{color}{func_name}: {text}{RESET}")

@pytest.fixture()
def red(request):
    return partial(_color_print, RED, request.node.name)

@pytest.fixture()
def green(request):
    return partial(_color_print, GREEN, request.node.name)
```

```python
@pytest.fixture()
def yellow(request):
    return partial(_color_print, YELLOW, request.node.name)

@pytest.fixture()
def blue(request):
    return partial(_color_print, BLUE, request.node.name)

@pytest.fixture()
def magenta(request):
    return partial(_color_print, MAGENTA, request.node.name)

@pytest.fixture()
def cyan(request):
    return partial(_color_print, CYAN, request.node.name)

def test_colors(red, green, yellow, blue, magenta, cyan):
    print("")
    red("this should be red")
    green("this should be green")
    yellow("this should be yellow")
    blue("this should be blue")
    magenta("this should be magenta")
    cyan("this should be cyan")

def test_magenta(magenta):
    print("")
    magenta("this should be magenta")
```

We need to set them free

And put them in a conftest.py file

```
test_directory
├── conftest.py
├── test_example.py
└── test_example_two.py
```

conftest.py

```python
import pytest
from functools import partial

# Terminal color codes
RED = "\x1b[31m"
GREEN = "\x1b[32m"
YELLOW = "\x1b[33m"
BLUE = "\x1b[34m"
MAGENTA = "\x1b[35m"
CYAN = "\x1b[36m"
RESET = "\x1b[0m"
BOLD = "\x1b[1m"

def _color_print(color: str, func_name: str, text: str):
    print(f"{BOLD}{color}{func_name}: {text}{RESET}")

@pytest.fixture()
def red(request):
    return partial(_color_print, RED, request.node.name)


@pytest.fixture()
def green(request):
    return partial(_color_print, GREEN, request.node.name)
```

```python
@pytest.fixture()
def yellow(request):
    return partial(_color_print, YELLOW, request.node.name)


@pytest.fixture()
def blue(request):
    return partial(_color_print, BLUE, request.node.name)


@pytest.fixture()
def magenta(request):
    return partial(_color_print, MAGENTA, request.node.name)


@pytest.fixture()
def cyan(request):
    return partial(_color_print, CYAN, request.node.name)
```

test_example.py

```python
def test_colors(red, green, yellow, blue, magenta, cyan):
    print("")
    red("this should be red")
    green("this should be green")
    yellow("this should be yellow")
    blue("this should be blue")
    magenta("this should be magenta")
    cyan("this should be cyan")
```

test_example_two.py

```python
def test_magenta(magenta):
    print("")
    magenta("this should be magenta")
```

Now sharing works

In the same directory

```
test_directory
├── conftest.py
├── test_example.py
└── test_example_two.py
```

How about different directories?

For tests in subdirectories

It's still fine

```
test_directory
├── conftest.py
├── one
│   └── test_example.py
└── two
    └── test_example_two.py
```

Multiple conftest.py files

Still fine

```
test_directory
├── conftest.py
├── one
│   ├── conftest.py
│   └── test_example.py
└── two
    ├── conftest.py
    └── test_example_two.py
```

How about sharing between projects?

This is where it gets fun

We're going to start here

```
test_directory
├── conftest.py
├── test_example.py
└── test_example_two.py
```

And move to

```
test_directory
├── conftest.py
├── test_example.py
└── test_example_two.py
```

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

Yes. We're going to talk about ...

Packaging

and

building a

pytest plugin

packaging pytest plugins

Trust me

it's not that bad

So how do we get from here to there?

```
test_directory
├── conftest.py
├── test_example.py
└── test_example_two.py
```

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

test_directory -> pytest-crayons

```
test_directory
├── conftest.py
├── test_example.py
└── test_example_two.py
```

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

Fixtures: conftest.py -> pytest_crayons.py

```
test_directory
├── conftest.py
├── test_example.py
└── test_example_two.py
```

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

src is also an option

```
test_directory
├── conftest.py
├── test_example.py
└── test_example_two.py
```

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── src
│   └── pytest_colors.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

or you can get fancy

```
test_directory
├── conftest.py
├── test_example.py
└── test_example_two.py
```

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── src
│   └── pytest_colors
│       ├── __init__.py
│       └── plugin.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

But let's stick with pytest_crayons.py

```
test_directory
├── conftest.py
├── test_example.py
└── test_example_two.py
```

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

examples -> examples

```
test_directory
├── conftest.py
├── test_example.py
└── test_example_two.py
```

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

existing stuff

```
test_directory
├── conftest.py
├── test_example.py
└── test_example_two.py
```

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

New stuff

```
test_directory
├── conftest.py
├── test_example.py
└── test_example_two.py
```

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

LICENSE

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

```
The MIT License (MIT)

Copyright (c) 2023, Brian Okken

Permission is hereby granted, ...
...
```

README.md

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

```
# pytest-crayons

... < what it does >
... < why you should use it >

## Installation

... < how to install it >

## Usage

... < code sample >
... < how to use it >
```

pyproject.toml

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

Let's come back to that

more tests and tox

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

pyproject.toml

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

Now might be

a good time for a

deep breath

pyproject.toml

```toml
[project]
name = "pytest-crayons"
description = "A pytest plugin for colorful print statements"
version = "0.0.4"
authors = [{name = "Brian Okken"}]
requires-python = ">=3.7"
readme = "README.md"
license = {file = "LICENSE"}
dependencies = ["pytest"]
classifiers = [
    "License :: OSI Approved :: MIT License",
    "Framework :: Pytest"
    ]

[project.urls]
Home = "https://github.com/okken/pytest-crayons"

[project.entry-points.pytest11]
pytest_crayons = "pytest_crayons"

[build-system]
requires = ["flit_core >=3.2,<4"]
build-backend = "flit_core.buildapi"

[tool.flit.module]
name = "pytest_crayons"
```

Project

```
[project]
name = "pytest-crayons"
description = "A pytest plugin for colorful print statements"
version = "0.0.4"
authors = [{name = "Brian Okken"}]
requires-python = ">=3.7"
readme = "README.md"
license = {file = "LICENSE"}
dependencies = ["pytest"]
classifiers = [
    "License :: OSI Approved :: MIT License",
    "Framework :: Pytest"
    ]

[project.urls]
Home = "https://github.com/okken/pytest-crayons"

[project.entry-points.pytest11]
pytest_crayons = "pytest_crayons"

[build-system]
requires = ["flit_core >=3.2,<4"]
build-backend = "flit_core.buildapi"

[tool.flit.module]
name = "pytest_crayons"
```

Project

```
[project]
name = "pytest-crayons"
description = "A pytest plugin for colorful print statements"
version = "0.0.4"
authors = [{name = "Brian Okken"}]
requires-python = ">=3.7"
readme = "README.md"
license = {file = "LICENSE"}
dependencies = ["pytest"]
classifiers = [
    "License :: OSI Approved :: MIT License",
    "Framework :: Pytest"
    ]
```

Classifiers

```
classifiers = [
    "License :: OSI Approved :: MIT License",
    "Framework :: Pytest"
    ]
```

- These need to be exact matches
- Look them up at pypi.org/classifiers/

Meta

License: MIT License

Author: Brian Okken

Requires: Python >=3.7

Maintainers

okken

Classifiers

Framework

- Pytest

License

- OSI Approved :: MIT License

Shows up on left sidebar of PyPI

URL

```toml
[project]
name = "pytest-crayons"
description = "A pytest plugin for colorful print statements"
version = "0.0.4"
authors = [{name = "Brian Okken"}]
requires-python = ">=3.7"
readme = "README.md"
license = {file = "LICENSE"}
dependencies = ["pytest"]
classifiers = [
    "License :: OSI Approved :: MIT License",
    "Framework :: Pytest"
    ]

[project.urls]
Home = "https://github.com/okken/pytest-crayons"

[project.entry-points.pytest11]
pytest_crayons = "pytest_crayons"

[build-system]
requires = ["flit_core >=3.2,<4"]
build-backend = "flit_core.buildapi"

[tool.flit.module]
name = "pytest_crayons"
```

URL

```
[project.urls]
Home = "https://github.com/okken/pytest-crayons"
```

pytest entry point

```toml
[project]
name = "pytest-crayons"
description = "A pytest plugin for colorful print statements"
version = "0.0.4"
authors = [{name = "Brian Okken"}]
requires-python = ">=3.7"
readme = "README.md"
license = {file = "LICENSE"}
dependencies = ["pytest"]
classifiers = [
    "License :: OSI Approved :: MIT License",
    "Framework :: Pytest"
    ]

[project.urls]
Home = "https://github.com/okken/pytest-crayons"

[project.entry-points.pytest11]
pytest_crayons = "pytest_crayons"

[build-system]
requires = ["flit_core >=3.2,<4"]
build-backend = "flit_core.buildapi"

[tool.flit.module]
name = "pytest_crayons"
```

pytest entry point

```
[project.entry-points.pytest11]
pytest_crayons = "pytest_crayons"
```

pytest entry point

```
[project.entry-points.pytest11]
pytest_crayons = "pytest_crayons"
```

Supposedly it's:

```
plugin_name = "module"
```

pytest entry point

```
[project.entry-points.pytest11]
pytest_crayons = "pytest_crayons"
```

Plugin name

- `pytest_crayons` *or*
- `pytest-crayons` *or*
- `crayons`

Module name

```
...
├── pytest_crayons.py
...
```

If we have package.module

```
[project.entry-points.pytest11]
pytest_crayons = "pytest_crayons.plugin"

...
├── src
    └── pytest_crayons
        ├── __init__.py
        └── plugin.py
│
...
```

But we didn't

```
[project.entry-points.pytest11]
pytest_crayons = "pytest_crayons"

...
├── pytest_crayons.py
...
```

Build system

```
[project]
name = "pytest-crayons"
description = "A pytest plugin for colorful print statements"
version = "0.0.4"
authors = [{name = "Brian Okken"}]
requires-python = ">=3.7"
readme = "README.md"
license = {file = "LICENSE"}
dependencies = ["pytest"]
classifiers = [
    "License :: OSI Approved :: MIT License",
    "Framework :: Pytest"
    ]

[project.urls]
Home = "https://github.com/okken/pytest-crayons"

[project.entry-points.pytest11]
pytest_crayons = "pytest_crayons"

[build-system]
requires = ["flit_core >=3.2,<4"]
build-backend = "flit_core.buildapi"

[tool.flit.module]
name = "pytest_crayons"
```

Build system

```
[build-system]
requires = ["flit_core >=3.2,<4"]
build-backend = "flit_core.buildapi"
```

Module

```
[project]
name = "pytest-crayons"
description = "A pytest plugin for colorful print statements"
version = "0.0.4"
authors = [{name = "Brian Okken"}]
requires-python = ">=3.7"
readme = "README.md"
license = {file = "LICENSE"}
dependencies = ["pytest"]
classifiers = [
    "License :: OSI Approved :: MIT License",
    "Framework :: Pytest"
    ]

[project.urls]
Home = "https://github.com/okken/pytest-crayons"

[project.entry-points.pytest11]
pytest_crayons = "pytest_crayons"

[build-system]
requires = ["flit_core >=3.2,<4"]
build-backend = "flit_core.buildapi"

[tool.flit.module]
name = "pytest_crayons"
```

Module

```
[tool.flit.module]
name = "pytest_crayons"
```

Module

```
[tool.flit.module]
name = "pytest_crayons"
```

```
[project]
name = "pytest-crayons"
```

Module

```
[tool.flit.module]
name = "pytest_crayons"

...
├── pytest_crayons.py
...
```

```
[project]
name = "pytest-crayons"
```

Flit based pyproject.toml

```
[project]
name = "pytest-crayons"
description = "A pytest plugin for colorful print statements"
version = "0.0.4"
authors = [{name = "Brian Okken"}]
requires-python = ">=3.7"
readme = "README.md"
license = {file = "LICENSE"}
dependencies = ["pytest"]
classifiers = [
    "License :: OSI Approved :: MIT License",
    "Framework :: Pytest"
    ]

[project.urls]
Home = "https://github.com/okken/pytest-crayons"

[project.entry-points.pytest11]
pytest_crayons = "pytest_crayons"

[build-system]
requires = ["flit_core >=3.2,<4"]
build-backend = "flit_core.buildapi"

[tool.flit.module]
name = "pytest_crayons"
```

Flit specific

```toml
[project]
name = "pytest-crayons"
description = "A pytest plugin for colorful print statements"
version = "0.0.4"
authors = [{name = "Brian Okken"}]
requires-python = ">=3.7"
readme = "README.md"
license = {file = "LICENSE"}
dependencies = ["pytest"]
classifiers = [
    "License :: OSI Approved :: MIT License",
    "Framework :: Pytest"
    ]

[project.urls]
Home = "https://github.com/okken/pytest-crayons"

[project.entry-points.pytest11]
pytest_crayons = "pytest_crayons"

[build-system]
requires = ["flit_core >=3.2,<4"]
build-backend = "flit_core.buildapi"

[tool.flit.module]
name = "pytest_crayons"
```

flit

```
[build-system]
requires = ["flit_core >=3.2,<4"]
build-backend = "flit_core.buildapi"

[tool.flit.module]
name = "pytest_crayons"
```

hatch

```
[build-system]
requires = ["hatchling"]
build-backend = "hatchling.build"
```

setuptools

```
[build-system]
requires = ["setuptools"]
build-backend = "setuptools.build_meta"
```

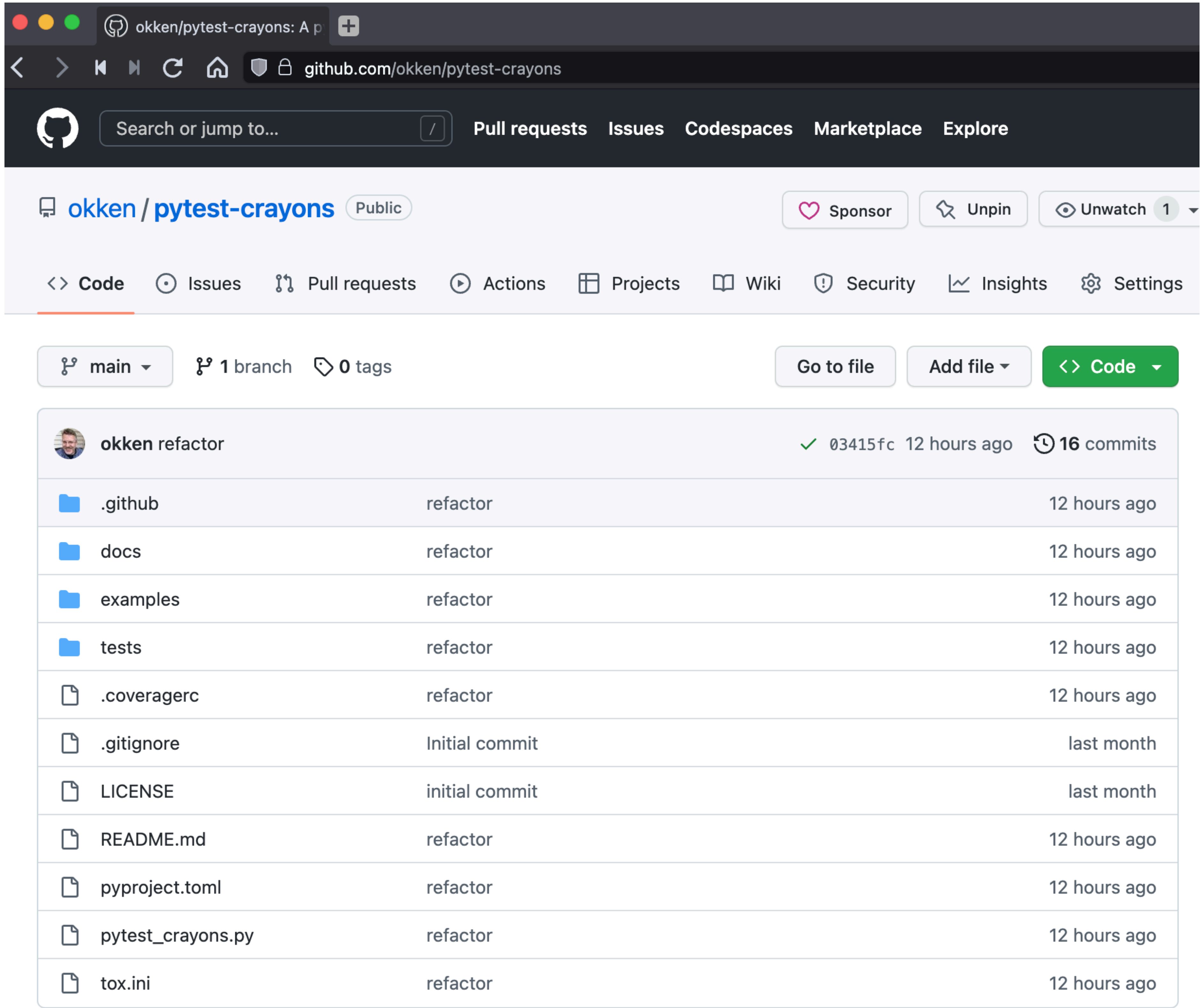
That wasn't too bad, was it?

Guess what?

We can share our code now

Seriously

We could just push to a git repo now


okken/pytest-crayons: A p
github.com/okken/pytest-crayons
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
okken / pytest-crayons
Public
Sponsor
Unpin
Unwatch 1
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
okken refactor
03415fc 12 hours ago
16 commits
.github refactor 12 hours ago
docs refactor 12 hours ago
examples refactor 12 hours ago
tests refactor 12 hours ago
.coveragerc refactor 12 hours ago
.gitignore Initial commit last month
LICENSE initial commit last month
README.md refactor 12 hours ago
pyproject.toml refactor 12 hours ago
pytest_crayons.py refactor 12 hours ago
tox.ini refactor 12 hours ago

We can

install directly from git

```
$ pip install git+https://github.com/okken/pytest-crayons.git
```

Even usable with `requirements.txt`

```
...
pytest
git+https://github.com/okken/pytest-crayons.git
tox
...
```

We haven't even built the wheel yet

`pip install` builds the wheel

```
$ pip install git+https://github.com/okken/pytest-crayons.git
Collecting ...
  Cloning ...
  Running command git clone ...
  Resolved ...
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: pytest-crayons
  Building wheel for pytest-crayons (pyproject.toml) ... done
  Created wheel for pytest-crayons: ...
Successfully built pytest-crayons
Installing collected packages: pytest-crayons
Successfully installed pytest-crayons-0.0.4
```

So hopefully the build works

Maybe we should test packaging

Look! Some tests

```
pytest-crayons
├── LICENSE
├── README.md
├── examples
│   ├── test_example.py
│   └── test_example_two.py
├── pyproject.toml
├── pytest_crayons.py
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

- `tox` for testing packaging
  - and running our tests
  - on multiple platforms
- `pytester` for testing plugin functionality

Let's start with tox

```
...
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

tox.ini

```
[tox]
envlist = py37, py38, py39, py310, py311, py312
skip_missing_interpreters = true
isolated_build = True

[testenv]
commands = pytest
description = Run pytest

[pytest]
testpaths =
    examples
    tests
```

Running tox

For example

```
$ pip install tox
$ tox -e py310
```

or all environments

```
$ tox
```

- creates a virtual environment
- builds a wheel
- installs all dependencies
- installs your code
- runs the tests

tox example

```
(code) $ tox -q -e py311
========================= test session starts ==========================
collected 4 items

examples/test_example.py .
examples/test_example_two.py ..
tests/test_colors.py .

========================== 4 passed in 0.05s ===========================
  py311: OK (2.02=setup[1.74]+cmd[0.28] seconds)
  congratulations :) (2.07 seconds)
```

tests

```
...
├── examples
│   ├── test_example.py
│   └── test_example_two.py
...
├── tests
│   ├── conftest.py
│   └── test_colors.py
└── tox.ini
```

tests/conftest.py

```python
pytest_plugins = "pytester"
```

tests/test_colors.py

```python
def test_colors_get_printed(pytester):
    pytester.copy_example("examples/test_example.py")
    pytester.makepyfile(__init__ = "")
    result = pytester.runpytest("-s")
    result.assert_outcomes(passed=1)
    bold, reset  = '\x1b[1m', '\x1b[0m'
    result.stdout.fnmatch_lines(
        [
            f"*{bold}\x1b[31mtest_colors: this should be red{reset}",
            f"*{bold}\x1b[32mtest_colors: this should be green{reset}",
            f"*{bold}\x1b[33mtest_colors: this should be yellow{reset}",
            f"*{bold}\x1b[34mtest_colors: this should be blue{reset}",
            f"*{bold}\x1b[35mtest_colors: this should be magenta{reset}",
            f"*{bold}\x1b[36mtest_colors: this should be cyan{reset}",
        ],
    )
```

We're testing a lot

- We're testing
  - the examples using the fixtures
  - the actual output
  - including the color ASCII codes
  - the packaging: building, installing, and dependencies
  - on multiple versions of Python

ahhh. confidence.

Is it ready for PyPI?

Actually, yes.

We won't go through those details

But essentially it's:

- Register with both test.pypi.org / pypi.org
- Build
- Publish to TestPyPI / PyPI:

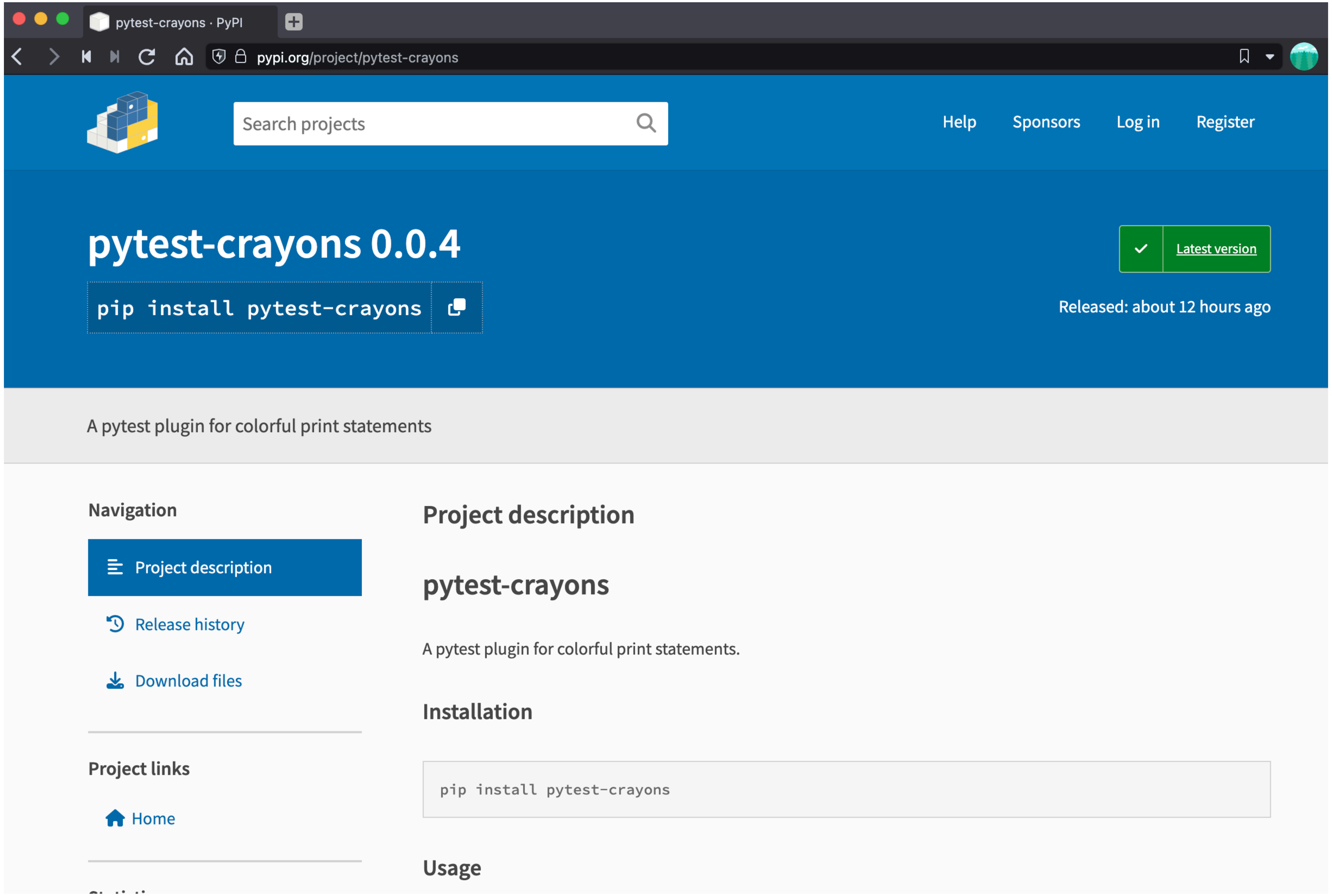
Build

- `flit build`
- `hatch build`
- `python -m build`
    - requires `pip install build`

Pubilish

First to TestPyPI, then PyPI

- `flit publish`
- `hatch publish`
- `twine upload dist/*`


pytest-crayons · PyPI
pypi.org/project/pytest-crayons
Search projects
Help
Sponsors
Log in
Register
pytest-crayons 0.0.4
pip install pytest-crayons
Latest version
Released: about 12 hours ago
A pytest plugin for colorful print statements
Navigation
Project description
Release history
Download files
Project links
Home
Project description
pytest-crayons
A pytest plugin for colorful print statements.
Installation
pip install pytest-crayons
Usage

Absolutely don't want PyPI?

Preventing PyPI upload

```
classifiers = [
    "License :: OSI Approved :: MIT License",
    "Framework :: Pytest",
    "Private :: Do Not Upload"
    ]
```

PyPI will always reject packages with classifiers beginning with `"Private ::"`. - pypi.org/classifiers

Learned this from Brett Cannon

We covered

- pytest fixtures
- conftest.py for sharing fixtures
- creating a pytest plugin
- packaging with flit, hatch, setuptools
- sharing via git repository
- tox for testing packaging
    - on multiple Python versions
- testing functionality with pytester
- uploading to PyPI

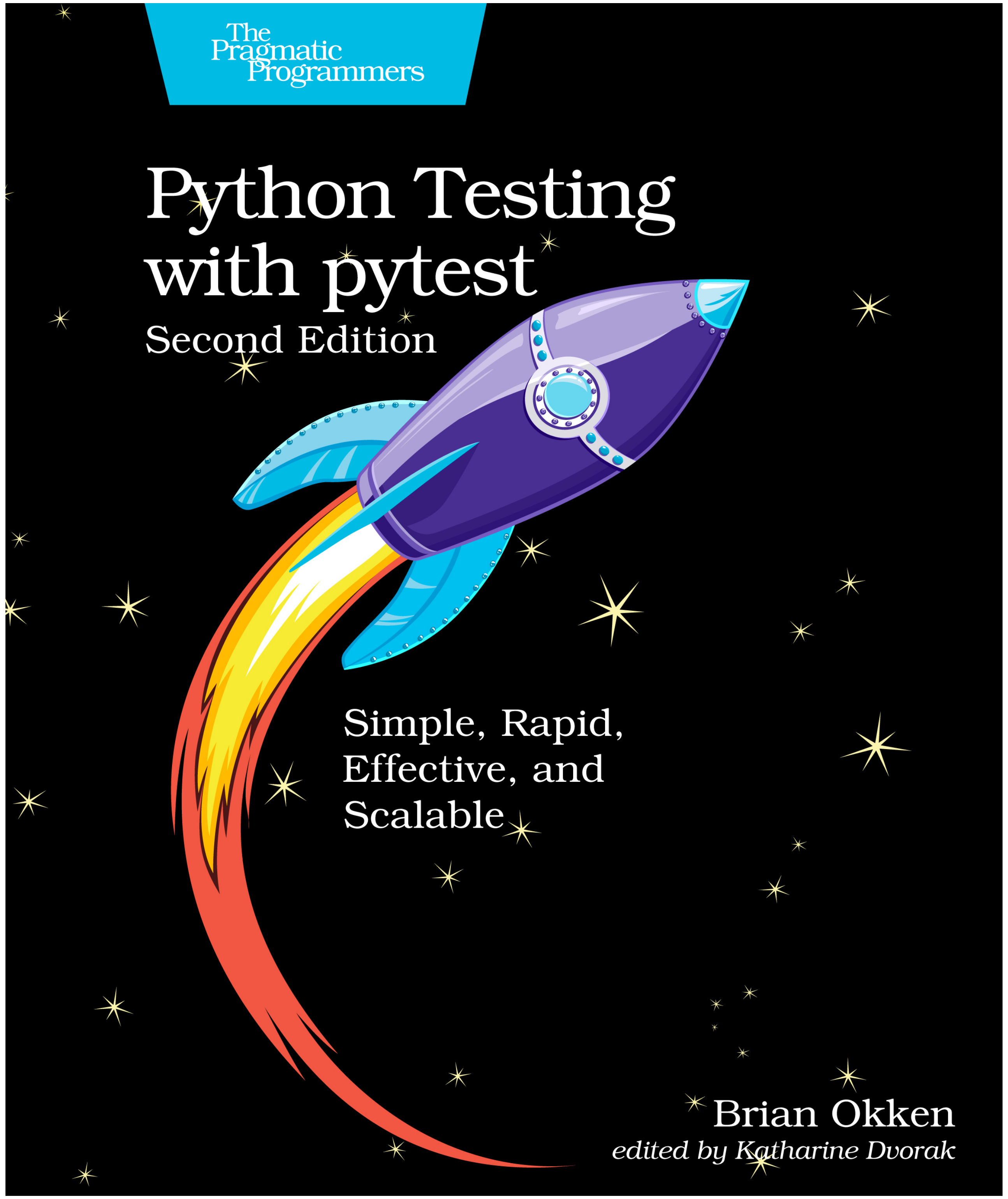
Not too bad, was it?

If you want to know more about

building and testing fixtures and plugins

See:

- Ch 3 for fixtures
- Ch 15 for building plugins
- Ch 11 includes
    - tox
    - GitHub Actions



Keep in touch

- pythontest.com
  training, courses, book
- pythonbytes.fm
  Python news and headlines directly to your earbuds
- testandcode.com
  Coding with automation
- testandcode.com/contact
  Email contact form
- @brianokken@fosstodon.org
  Mastodon