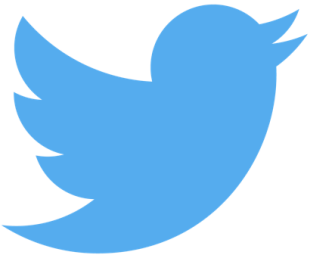
pytest

what's new in 3.0

EuroPython 2016 - July 21, 2016

Raphael Pierzina



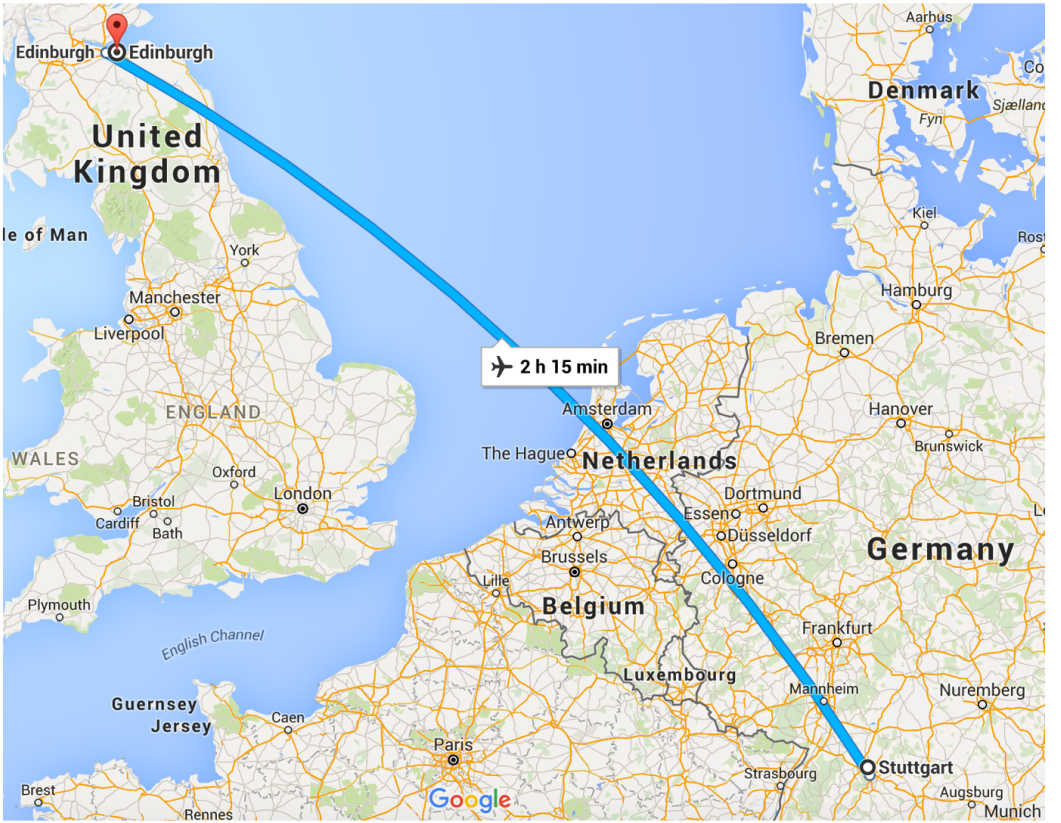
@hackebrot

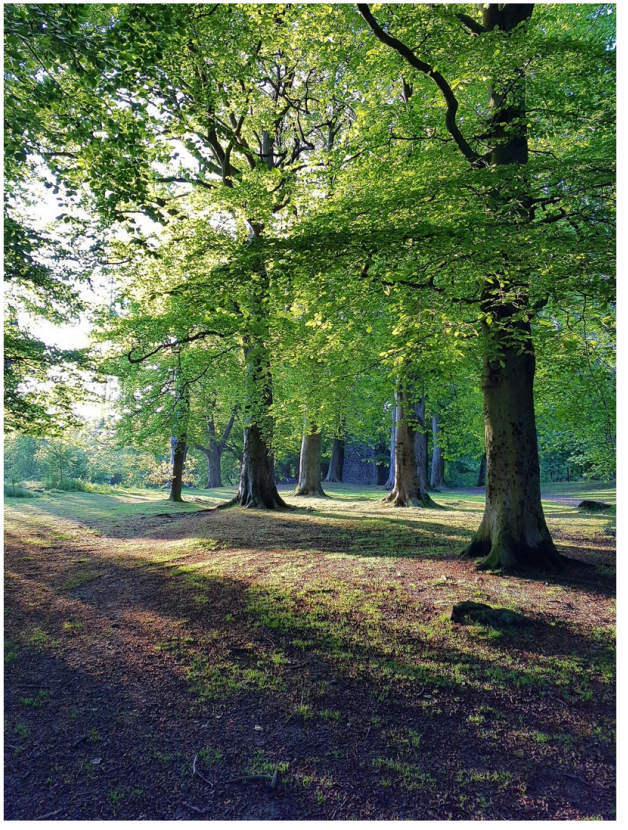
pytest

FanDuel

FanDuel

- We pioneered daily fantasy sports in the US, having spotted a gap in the market for a faster-paced, more exciting form of fantasy sports.
- Over 6 million registered users in the US and Canada
- Offices in Edinburgh, Glasgow, New York, Orlando and LA
- Approx 400 staff - split across UK and US


Edinburgh
Edinburgh
United Kingdom
le of Man
York
Manchester
Liverpool
ENGLAND
WALES
Oxford
Bristol
Cardiff
Bath
London
Plymouth
English Channel
Guernsey
Jersey
Caen
Brest
Rennes
Paris
2 h 15 min
Amsterdam
The Hague
Netherlands
Antwerp
Brussels
Lille
Belgium
Luxembourg
Aarhus
Denmark
Fyn
Kiel
Hamburg
Bremen
Hanover
Brunswick
Dortmund
Essen
Düsseldorf
Cologne
Germany
Frankfurt
Mannheim
Nuremberg
Strasbourg
Stuttgart
Augsburg
Munich
Google



www.fanduel.com/careers

cookiecutter

- command-line utility that helps you to start new projects following best practices of the community or simply based on your very own experience.
- renders templates based on user-input on the CLI
- project templates can be in any programming language or markup format

```
$ cookiecutter gh:pytest-dev/cookiecutter-pytest-plugin
Cloning into 'cookiecutter-pytest-plugin'...
remote: Counting objects: 659, done.
remote: Total 659 (delta 0), reused 0 (delta 0), pack-reused 659
Receiving objects: 100% (659/659), 124.03 KiB | 0 bytes/s, done.
Resolving deltas: 100% (382/382), done.
Checking connectivity... done.
```

```
full_name [Raphael Pierzina]:
email [raphael@hackebrot.de]:
github_username [hackebrot]:
plugin_name [foobar]: pokemon
module_name [pokemon]:
short_description [A simple plugin to use with Pytest]: Gotta test em all
version [0.1.0]:
pytest_version [2.9.1]: 2.9.2
Select docs_tool:
1 - mkdocs
2 - sphinx
3 - none
Choose from 1, 2, 3 [1]: 1
```

```
Select license:
1 - MIT
2 - BSD-3
3 - GNU GPL v3.0
4 - Apache Software License 2.0
5 - Mozilla Public License 2.0
Choose from 1, 2, 3, 4, 5 [1]: 2
INFO:post_gen_project:Moving files for mkdocs.
INFO:post_gen_project:Removing all temporary license files
```

```
$ tree pytest-pokemon/

pytest-pokemon/
├── LICENSE
├── README.rst
├── appveyor.yml
├── docs
|   └── index.md
├── mkdocs.yml
├── pytest_pokemon.py
├── setup.py
├── tests
|   ├── conftest.py
|   └── test_pokemon.py
└── tox.ini

2 directories, 10 files
```

```
setup(
    name='pytest-pokemon',
    version='0.1.0',
    author='Raphael Pierzina',
    author_email='raphael@hackebrot.de',
    maintainer='Raphael Pierzina',
    maintainer_email='raphael@hackebrot.de',
    license='BSD-3',
    url='https://github.com/hackebrot/pytest-pokemon',
    description='Gotta test em all',
    long_description=read('README.rst'),
    py_modules=['pytest_pokemon'],
    install_requires=['pytest>=2.9.2'],
    classifiers=[
        'Development Status :: 4 - Beta',
        'Framework :: Pytest',
        'Programming Language :: Python',
        'Programming Language :: Python :: 2',
        'Programming Language :: Python :: 3',
        'License :: OSI Approved :: BSD License',
    ],
    entry_points={
        'pytest11': [
            'pokemon = pytest_pokemon',
        ],
    },
)
```

github.com/audreyr/cookiecutter

Raise your hands... 

Do you use pytest?

Are you familiar with how pytest plugins work?

pytest

- mature testing framework for Python
- available on OS X, Linux and Windows
- compatible with CPython 2.6, 2.7, 3.3, 3.4, 3.5 and PyPy

pytest

- distributed under the terms of the MIT license
- free and open source software
- developed by a thriving community of volunteers

github.com/pytest-dev/pytest

```
$ pip install pytest

$ py.test --version
This is pytest version 2.9.2
```

wait what?! 😮

I need to install “pytest”,
but run “py.test”

MIND = BLOWN

History

- **pytest** came into existence as part of the **py library**, providing a tool called py.test
- even after **pytest** was moved to a separate project, the py.test name for the command-line tool was kept to preserve **backward compatibility** with existing scripts and tools.

SOON 

pytest 3.0 allows both
“pytest” and **“py.test”**

“py.test” will be kept for compatibility! 😌

pytest

- plain assert statements
- regular Python comparisons
- requires little to no boilerplate
- easy parametrization

Note:
Examples use Python 3 🐍

Example: Test a CLI app

- setup CLI runner using the ‘click’ library + teardown
- three commands: config, update, search
- two flags: --verbose, -v
- smoke test (exit code is expected to be 0)

unittest

```
import unittest

from click import testing
from cibopath import cli, utils


class TestCliCommands(unittest.TestCase):

    def setUp(self):
        self.runner = testing.CliRunner()

    def tearDown(self):
        utils.clean_up()

    def test_config_verbose(self):
        command = 'config'
        flags = ['--verbose']
        result = self.runner.invoke(cli.main, [*flags, command])
        self.assertEqual(result.exit_code, 0)

if __name__ == '__main__':
    unittest.main()
```

```
import unittest

from click import testing
from cibopath import cli, utils


class TestCliCommands(unittest.TestCase):

    def setUp(self):
        self.runner = testing.CliRunner()

    def tearDown(self):
        utils.clean_up()

    def test_config_v(self):
        command = 'config'
        flags = ['-v']
        result = self.runner.invoke(cli.main, [*flags, command])
        self.assertEqual(result.exit_code, 0)

    def test_config_verbose(self):
        command = 'config'
        flags = ['--verbose']
        result = self.runner.invoke(cli.main, [*flags, command])
        self.assertEqual(result.exit_code, 0)

    def test_update_v(self):
        command = 'update'
        flags = ['-v']
        result = self.runner.invoke(cli.main, [*flags, command])
        self.assertEqual(result.exit_code, 0)

    def test_update_verbose(self):
        command = 'update'
        flags = ['--verbose']
        result = self.runner.invoke(cli.main, [*flags, command])
        self.assertEqual(result.exit_code, 0)

    def test_search_v(self):
        command = 'search'
        flags = ['-v']
        result = self.runner.invoke(cli.main, [*flags, command])
        self.assertEqual(result.exit_code, 0)

    def test_search_verbose(self):
        command = 'search'
        flags = ['--verbose']
        result = self.runner.invoke(cli.main, [*flags, command])
        self.assertEqual(result.exit_code, 0)

if __name__ == '__main__':
    unittest.main()
```

```
import pytest

from click import testing
from cibopath import cli, utils


@pytest.yield_fixture
def runner():
    cli_runner = testing.CliRunner()
    yield cli_runner
    utils.clean_up()


def test_cli(runner):
    result = runner.invoke(cli.main, ['verbose', 'config'])
    assert result.exit_code == 0
```

```
import pytest

from click import testing
from cibopath import cli, utils


@pytest.yield_fixture
def runner():
    cli_runner = testing.CliRunner()
    yield cli_runner
    utils.clean_up()


@pytest.mark.parametrize('command', [
    'config',
    'update',
    'search',
])
@pytest.mark.parametrize('flags', [
    ['--verbose'],
    ['-v'],
])
def test_cli(runner, command, flags):
    result = runner.invoke(cli.main, [*flags, command])
    assert result.exit_code == 0
```

Fundamentals

Naming matters!

Test discovery, fixture system, hooks, ...

@pytest.mark.parametrize

```
import pytest

@pytest.mark.parametrize(
    'number, word', [
        (1, '1'),
        (3, 'Fizz'),
        (5, 'Buzz'),
        (10, 'Buzz'),
        (15, 'FizzBuzz'),
        (16, '16')
    ]
)
def test_fizzbuzz(number, word):
    assert fizzbuzz(number) == word
```

```
$ py.test -v test_parametrize.py
============================= test session starts ==============================
collected 6 items

test_parametrize.py::test_fizzbuzz[1-1] PASSED
test_parametrize.py::test_fizzbuzz[3-Fizz] PASSED
test_parametrize.py::test_fizzbuzz[5-Buzz] PASSED
test_parametrize.py::test_fizzbuzz[10-Buzz] PASSED
test_parametrize.py::test_fizzbuzz[15-FizzBuzz] PASSED
test_parametrize.py::test_fizzbuzz[16-16] PASSED

========================== 6 passed in 0.01 seconds ===========================
```

@pytest.fixture

```
import pytest

@pytest.fixture(params=[
    'apple',
    'banana',
    'plum',
])
def fruit(request):
    return request.param

def test_is_healthy(fruit):
    assert is_healthy(fruit)
```

```
$ pytest -v test_fruits.py
========================= test session starts ==========================
collected 3 items

test_fruits.py::test_is_healthy[apple] PASSED
test_fruits.py::test_is_healthy[banana] PASSED
test_fruits.py::test_is_healthy[plum] PASSED

======================== 3 passed in 0.01 seconds ======================
```

pytest

- extensible through plugins
- customizable through hooks
- intelligent test selection with markers
- powerful fixture system

```
def test_bake_project(cookies):
    """Create a project from our cookiecutter template."""

    result = cookies.bake(extra_context={
        'repo_name': 'helloworld',
    })

    assert result.exit_code == 0
    assert result.exception is None
    assert result.project.basename == 'helloworld'
```

github.com/hackebrot/pytest-cookies

github.com/pytest-dev/cookiecutter-pytest-plugin

Hooks

Run only tests that use fixture “new_fixture”

```
# conftest.py

def pytest_collection_modifyitems(items, config):
    selected_items = []
    deselected_items = []

    for item in items:
        if 'new_fixture' in getattr(item, 'fixturenames', ()):
            selected_items.append(item)
        else:
            deselected_items.append(item)
    config.hook.pytest_deselected(items=deselected_items)
    items[:] = selected_items
```

hackebrot.github.io/
pytest-tricks

New Features

approx()

```
import pytest

@pytest.fixture
def value():
    return 0.1

def test_approx(value):
    assert value + 0.2 == pytest.approx(0.3)
```

yield fixture

```
@pytest.fixture
def cookies(request, tmpdir, _cookiecutter_config_file):
    """Yield an instance of the Cookies helper class that
    can be used to generate a project from a template.
    """
    template_dir = request.config.option.template

    output_dir = tmpdir.mkdir('cookies')
    output_factory = output_dir.mkdir

    yield Cookies(
        template_dir,
        output_factory,
        _cookiecutter_config_file,
    )

    output_dir.remove()
```

doctest_namespace
(when using --doctest-modules)

```
# content of conftest.py

import numpy

@pytest.fixture(autouse=True)
def add_np(doctest_namespace):
    doctest_namespace['np'] = numpy
```

```
# content of numpy.py

def arange():
    """
    >>> a = np.arange(10)
    >>> len(a)
    10
    """
    pass
```

named fixtures

```
import pytest

from cookiecutter.main import cookiecutter


@pytest.fixture
def template():
    return 'https://github.com/pytest-dev/cookiecutter-pytest-plugin'


def generate_plugin_project(template, tmpdir):
    project_dir = cookiecutter(
        template,
        no_input=True,
        output_dir=str(tmpdir),
    )
    assert project_dir.endswith('pytest-foobar')
```

```
$ pylint test_fixturename.py
No config file found, using default configuration
************* Module test_fixturename
C:  1, 0: Missing module docstring (missing-docstring)
C:  7, 0: Missing function docstring (missing-docstring)
C: 11, 0: Missing function docstring (missing-docstring)
W: 11,33: Redefining name 'template' from outer scope (line 7)
(redefined-outer-name)
```

```
import pytest

from cookiecutter.main import cookiecutter


@pytest.fixture(name='template')
def plugin_template():
    return 'https://github.com/pytest-dev/cookiecutter-pytest-plugin'


def test_generate_plugin_project(template, tmpdir):
    project_dir = cookiecutter(
        template,
        no_input=True,
        output_dir=str(tmpdir),
    )
    assert project_dir.endswith('pytest-foobar')
```

pytest_make_parametrize_id

```
import pytest
from foobar import Package, Woman, Man

PACKAGES = [
    Package('requests', 'Apache 2.0'),
    Package('django', 'BSD'),
    Package('pytest', 'MIT'),
]

@pytest.fixture(params=PACKAGES)
def python_package(request):
    return request.param

@pytest.mark.parametrize('person', [
    Woman('Audrey'), Woman('Brianna'),
    Man('Daniel'), Woman('Ola'), Man('Jameson')
])
def test_become_a_programmer(person, python_package):
    person.learn(python_package.name)
    assert person.looks_like_a_programmer

def test_is_open_source(python_package):
    assert python_package.is_open_source
```

```
test_foobar.py::test_become_a_programmer[python_package0-person0] PASSED
test_foobar.py::test_become_a_programmer[python_package0-person1] PASSED
test_foobar.py::test_become_a_programmer[python_package0-person2] PASSED
test_foobar.py::test_become_a_programmer[python_package0-person3] PASSED
test_foobar.py::test_become_a_programmer[python_package0-person4] PASSED
test_foobar.py::test_become_a_programmer[python_package1-person0] PASSED
test_foobar.py::test_become_a_programmer[python_package1-person1] PASSED
test_foobar.py::test_become_a_programmer[python_package1-person2] PASSED
test_foobar.py::test_become_a_programmer[python_package1-person3] PASSED
test_foobar.py::test_become_a_programmer[python_package1-person4] PASSED
test_foobar.py::test_become_a_programmer[python_package2-person0] PASSED
test_foobar.py::test_become_a_programmer[python_package2-person1] PASSED
test_foobar.py::test_become_a_programmer[python_package2-person2] PASSED
test_foobar.py::test_become_a_programmer[python_package2-person3] PASSED
test_foobar.py::test_become_a_programmer[python_package2-person4] PASSED
test_foobar.py::test_is_open_source[python_package0] PASSED
test_foobar.py::test_is_open_source[python_package1] PASSED
test_foobar.py::test_is_open_source[python_package2] PASSED
```

```
@pytest.fixture(
    params=PACKAGES,
    ids=operator.attrgetter('name'),
)
def python_package(request):
    return request.param


@pytest.mark.parametrize('person', [
    Woman('Audrey'), Woman('Brianna'),
    Man('Daniel'), Woman('Ola'), Man('Jameson')
], ids=[
    'Audrey', 'Brianna',
    'Daniel', 'Ola', 'Jameson'
])
def test_become_a_programmer(person, python_package):
    person.learn(python_package.name)
    assert person.looks_like_a_programmer


def test_is_open_source(python_package):
    assert python_package.is_open_source
```

```
test_foobar.py::test_become_a_programmer[requests-Audrey] PASSED
test_foobar.py::test_become_a_programmer[requests-Brianna] PASSED
test_foobar.py::test_become_a_programmer[requests-Daniel] PASSED
test_foobar.py::test_become_a_programmer[requests-Ola] PASSED
test_foobar.py::test_become_a_programmer[requests-Jameson] PASSED
test_foobar.py::test_become_a_programmer[django-Audrey] PASSED
test_foobar.py::test_become_a_programmer[django-Brianna] PASSED
test_foobar.py::test_become_a_programmer[django-Daniel] PASSED
test_foobar.py::test_become_a_programmer[django-Ola] PASSED
test_foobar.py::test_become_a_programmer[django-Jameson] PASSED
test_foobar.py::test_become_a_programmer[pytest-Audrey] PASSED
test_foobar.py::test_become_a_programmer[pytest-Brianna] PASSED
test_foobar.py::test_become_a_programmer[pytest-Daniel] PASSED
test_foobar.py::test_become_a_programmer[pytest-Ola] PASSED
test_foobar.py::test_become_a_programmer[pytest-Jameson] PASSED
test_foobar.py::test_is_open_source[requests] PASSED
test_foobar.py::test_is_open_source[django] PASSED
test_foobar.py::test_is_open_source[pytest] PASSED
```

```python
from foobar import Woman, Man, Package


def pytest_make_parametrize_id(config, val):
    if isinstance(val, Woman):
        return u'👩 {}'.format(val.name)
    elif isinstance(val, Man):
        return u'👨 {}'.format(val.name)
    elif isinstance(val, Package):
        return u'📦 {}'.format(val.name)
```

```
test_foobar.py::test_become_a_programmer[📦 requests-👱 Audrey] PASSED
test_foobar.py::test_become_a_programmer[📦 requests-👱 Brianna] PASSED
test_foobar.py::test_become_a_programmer[📦 requests-👨 Daniel] PASSED
test_foobar.py::test_become_a_programmer[📦 requests-👱 Ola] PASSED
test_foobar.py::test_become_a_programmer[📦 requests-👨 Jameson] PASSED
test_foobar.py::test_become_a_programmer[📦 django-👱 Audrey] PASSED
test_foobar.py::test_become_a_programmer[📦 django-👱 Brianna] PASSED
test_foobar.py::test_become_a_programmer[📦 django-👨 Daniel] PASSED
test_foobar.py::test_become_a_programmer[📦 django-👱 Ola] PASSED
test_foobar.py::test_become_a_programmer[📦 django-👨 Jameson] PASSED
test_foobar.py::test_become_a_programmer[📦 pytest-👱 Audrey] PASSED
test_foobar.py::test_become_a_programmer[📦 pytest-👱 Brianna] PASSED
test_foobar.py::test_become_a_programmer[📦 pytest-👨 Daniel] PASSED
test_foobar.py::test_become_a_programmer[📦 pytest-👱 Ola] PASSED
test_foobar.py::test_become_a_programmer[📦 pytest-👨 Jameson] PASSED
test_foobar.py::test_is_open_source[📦 requests] PASSED
test_foobar.py::test_is_open_source[📦 django] PASSED
test_foobar.py::test_is_open_source[📦 pytest] PASSED
```

--fixtures-per-test

Live Demo

Backwards Incompatible Changes

Reinterpretation mode has now been removed. Only **plain** and **rewrite** mode are available.

As a consequence the **--assert=reinterp** option is no longer available

- **--genscript**: no longer supported
- **--no-assert**: use **--assert=plain** instead
- **--nomagic**: use **--assert=plain** instead
- **--report**: use **-r** instead

pytest warnings summary is shown by **default**.

Added a new flag **--disable-pytest-warnings** to explicitly disable the warnings summary

Deprecations

Using **pytest_funcarg__** prefix to declare fixtures is considered deprecated and will be removed in pytest 4.0

Rename **getfuncargvalue** to **getfixturevalue**.

getfuncargvalue is still present but is now considered deprecated.

Improvements

Plugins and conftest.py now benefit from **assertion rewriting**

Remove assertion reinterpretation

+45 −864 😬

#WriteTheDocs

(Beginner) user

- A beginner user is new pytest, testing or Python. It is important to give clear instructions on how to **install pytest**, how to write **basic tests** and **how to run** pytest against the tests.

- Ideally this is done in a **tutorial** like style that provides easy-to-follow steps along with an explanation on what each of the steps does.

- Documentation for a beginner users briefly explains why writing tests is important but then focuses on **explaining core concepts** of pytest, such as test discovery, assert statements and fixtures.

Advanced user

- Advanced users are familiar with core concepts of pytest and **feel comfortable writing parametrized tests** using fixtures and/or markers. He/She is interested in using **plugins**, fixture **scoping** or sharing and other features that increase efficiency or flexibility such as **hooks**.

- Guides for advanced users can be of different styles and vary from topic to topic. Comprehensive **blog-post like articles** or **cookbook style like recipes** explain how to combine several features based on real-world scenarios.

Plugin author

- An advanced user may want to share useful functionality as a **package on PyPI**. The documentation for plugin authors follows the same principles as for advanced users.

- He/She is introduced to **cookiecutter-pytest-plugin** as the most convenient way to get started writing plugins. Other section describe how to set up entry points via setuptools and might refer to use cases from existing plugins. Most importantly it is explained **how to write tests against plugin code**.

- Documentation for plugin author additionally covers how to move a plugin project to the **pytest-dev organization** on GitHub.

Contributor

- This is group is different from the other audiences in the sense that is not based on a persons experience with pytest. **Everyone is welcome to help out and contributions are greatly appreciated.**

- It is explained how the **project is organized** and how to **interact with other community members**, but also how to set up a development environment for pytest and **how to get started** with using GitHub, ideally by referring to the official guides.

- This section outlines requirements for **submitting issues or pull requests** to pytest core and covers best practices for working on the code base.

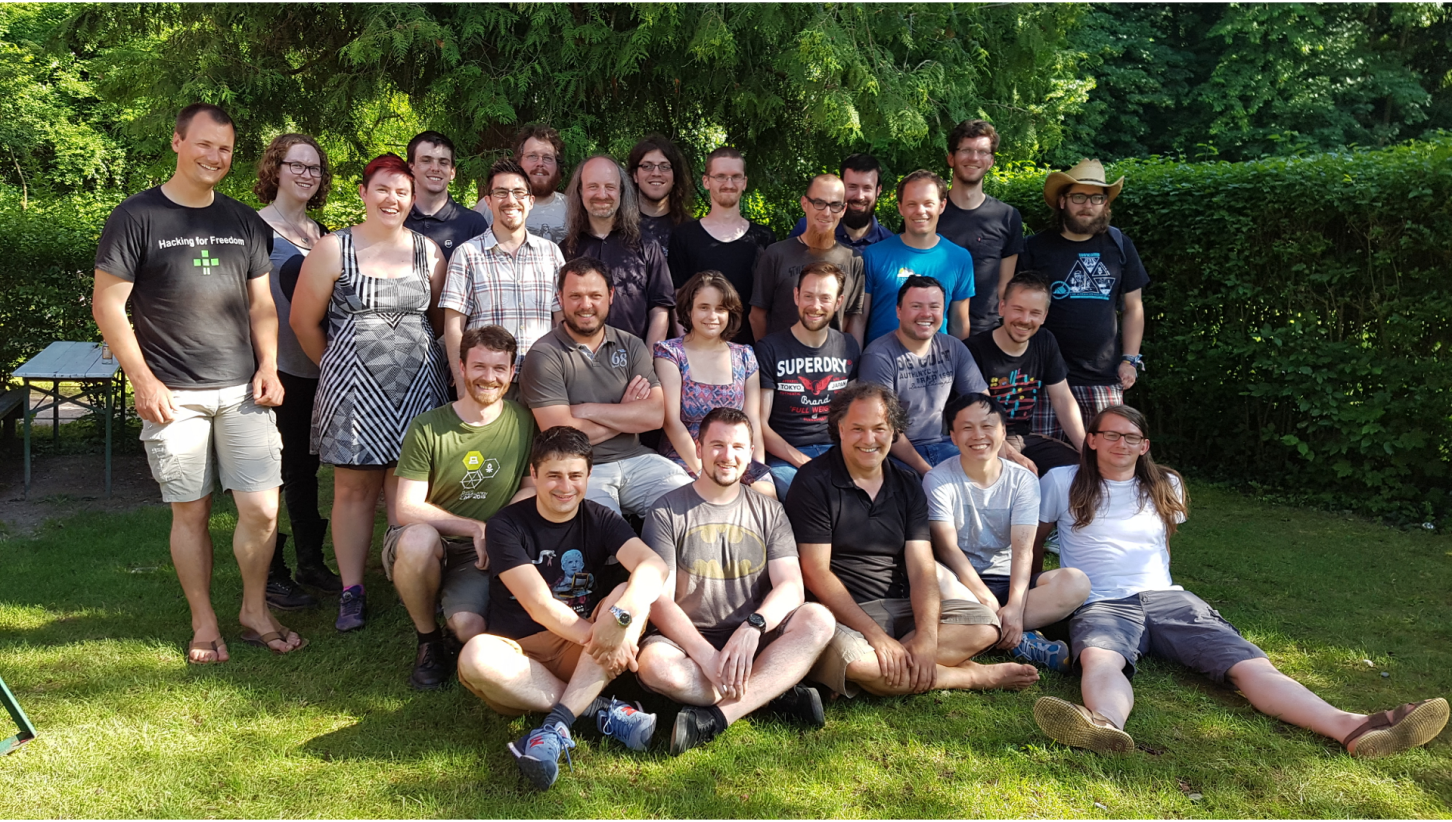

PYTEST
INSTALLATION
FOSS LICENSE
WHAT IS PYTEST?
TALKS, LINKS
PLUGIN INDEX
LINKS TO - docs - BLOG/FAQ - PYPI
CONTACT GETTING HELP
USER
ADVANCED USER
PLUGIN AUTHOR
CONTRIBUTOR
ASSERT STATEMENTS
TEST NAME CONVENTIONS
COMMAND LINE OPTIONS MOST USEFUL
CONFTEST.PY
INI CONFIG
HOOKS INTRO
SETUP.PY
MARKS 'TAGS' SELECTING
TEST DIR STRUCTURE
IDE INTEGRATION
MARKS SKIP
MARKS XFAIL
PARAMETRIZE
FIXTURES BASIC
RAISES
YIELD FIXTURES/ FINALIZERS
TYPES OF TEST RESULTS (PYTEST.EXIT)
STDOUT CAPTURING
CTYPES
MONKEY PATCH
MARKS 'TAGS' EXTRA FUNCTIONALITY
AVAILABLE PLUGINS/ RECOMMENDATIONS
XDIST RUNNING TESTS PARALLEL
PROFILING SLOW TESTS
TOX
CI JENKINS TRAVIS
COVERAGE
DEBUGGING
FLAKY TESTS
WARNINGS
LOGGING
FIXTURES SCOPE
PARAMETRIZED FIXTURES
ADVANCED PARAMETRIZE
MOCK
MIGRATE FROM UNITTEST/ NOSE
RUN NON PYTHON TESTS
FUZZY TESTING (HYPOTHESIS)
BDD
DOC TESTS
PLUGIN ECOSYSTEM
ADD COMMAND LINE OPTION
pytest-xdist writing plugins
COOKIE CUTTER
CACHE
Testing plugins
COOK BOOK
Setting up dev env
CONTRIB GIT/HG GUIDE
HISTORY
PROJECTS THAT USE PYTEST
CONTRIB SUGGESTIONS
REFERENCE
PYTEST NAMESPACE
HOOKS
COMMAND LINE OPTIONS ALL
COMMAND LINE OPTIONS
BUILT IN FIXTURES

docs.pytest.org

Funding FOSS 😕

Community

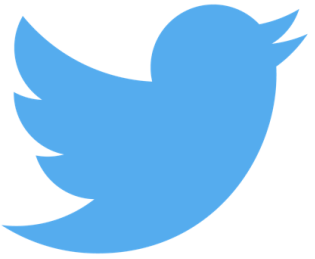

ANALMENTE


Hacking for Freedom
SUPERDRY

blog.pytest.org

speakerdeck.com/
hackebrot

Thank you #EuroPython



@hackebrot