pytest-phmdoctest

Release 1.0.0

Mark Taylor

Apr 15, 2022

CONTENTS:

1 pytest-phmdoctest 1.0.0 **1**
- 1.1 Introduction . . . 1
- 1.2 Description . . . 1
- 1.3 Installation . . . 2
- 1.4 Usage . . . 2
- 1.5 Testing Python interactive sessions . . . 3
- 1.6 Generate test files . . . 4
- 1.7 Generate and collect test files . . . 5
- 1.8 Help . . . 6
- 1.9 Configure collection . . . 6
- 1.10 Hints . . . 9
- 1.11 Related projects . . . 9

2 Recent changes **11**

3 Contributing **13**

4 Search **15**

CHAPTER

ONE

PYTEST-PHMDOCTEST 1.0.0

1.1 Introduction

Python syntax highlighted Markdown doctest pytest plugin.

A pytest plugin based on the phmdoctest command line tool.

If you have Python syntax highlighted examples in Markdown like this Python code...

```
print("Hello World!")
```

plus expected output.

```
Hello World!
```

This pytest plugin will test them, as is, without edits. On this file try the command ...

```
pytest -v --phmdoctest README.md
```

pytest console output ...

```
plugins: phmdoctest-1.0.0
collected 1 item

::README.py::test_code_12_output_16 PASSED
```

The plugin also tests Python interactive sessions described by doctest. See *testing examples* below.

The 12 in the function name `test_code_12_output_16` is the line number of the first line of python code. 16 shows the line number of the expected terminal output.

1.2 Description

The plugin is based on the Python tool phmdoctest version >= 1.4.0 It generates a pytest test file from a Markdown file.

- Reads these from Markdown fenced code blocks:
  - Python source code and expected terminal output.
  - Python interactive sessions described by doctest.
- Simple use case is possible with no Markdown edits.

- More features selected by adding HTML comment directives to the Markdown. See Directives in phmdoctest Directives.
- Run on pytest >= 5.4.3.

1.2.1 main branch status

Website | Docs | Repos | pytest | Codecov | License

Introduction | *Description* | *Installation* | *Usage* | *Testing Python interactive sessions* | *Generate test files* | *Generate and collect test files* | *Help* | *Configure collection* | *Hints* | *Related projects*

Changes | *Contributions*

1.3 Installation

It is advisable to install in a virtual environment.

```
python -m pip install pytest-phmdoctest
```

1.4 Usage

Consider a project with the following files. Not all files are shown.

```
README.md
doc/directive2.md
doc/nocode.md
tests/test_example.py
```

```
pytest -v
```

```
plugins: phmdoctest-1.0.0
collected 1 item

tests/test_example.py::test_example PASSED
```

Use `--phmdoctest` to also collect Markdown files.

```
pytest -v --phmdoctest
```

```
plugins: phmdoctest-1.0.0
collected 6 items

::README.py::test_code_10_output_17 PASSED
::doc__directive2.py::test_code_25_output_32 PASSED
::doc__directive2.py::test_code_42_output_47 PASSED
::doc__directive2.py::test_code_52_output_56 PASSED
```

(continues on next page)

(continued from previous page)

```
::doc__project.py::test_code_12_output_19 PASSED
tests/test_example.py::test_example PASSED
```

- The sample project above can be viewed on GitHub at `tests/sample`.
- The doc__ indicates the Markdown file was collected from the doc folder.
- Markdown ".md" files get discovered by pytest. pytest finds them in the same way it finds Python test files. For each Markdown file discovered, the plugin generates a pytest test file.
- A Markdown file that does not have any Python fenced code block examples is not tested.
- To avoid collecting .md files use pytest `--ignore` and `--ignore-glob` on the command line or in the addopts part of the pytest ini file. These commands work on .md files and use Unix shell-style wildcards.

Use `--phmdoctest` in a pytest ini file instead of on the command line as shown below:

```
# pytest.ini and tox.ini
[pytest]
addopts = --phmdoctest
```

```
# pyproject.toml
[tool.pytest.ini_options]
addopts = "--phmdoctest"
```

```
# setup.cfg  Note: Use is discouraged by pytest docs.
[tool:pytest]
addopts = --phmdoctest
```

1.5 Testing Python interactive sessions

The plugin also tests Python interactive sessions described by doctest like this one:

```
>>> import math
>>> math.floor(9.1)
9
```

Use the `--phmdoctest-docmod` option to collect both Python code/expected output and Python interactive sessions.

```
pytest -v --phmdoctest-docmod
```

```
plugins: phmdoctest-1.0.0
collected 10 items

::README.py::README.session_00001_line_24 PASSED
::README.py::test_code_10_output_17 PASSED
::doc__directive2.py::test_code_25_output_32 PASSED
::doc__directive2.py::test_code_42_output_47 PASSED
::doc__directive2.py::test_code_52_output_56 PASSED
::doc__project.py::doc__project.session_00001_line_31 PASSED
::doc__project.py::doc__project.session_00002_line_46 PASSED
::doc__project.py::doc__project.session_00003_line_55 PASSED
```

(continues on next page)

(continued from previous page)

```
::doc__project.py::test_code_12_output_19 PASSED
tests/test_example.py::test_example PASSED
```

- The `--phmdoctest-docmod` option uses a non-public pytest class DoctestModule. There is a slight chance a pytest future major release changes the DoctestModule API.
- DoctestModule works ok on pytest major releases 5, 6, and 7 as verified by tests in .github/workflows/ci.yml.
- If `--phmdoctest-docmod` doctest collection breaks, the rest of the plugin `--phmdoctest` and `--phmdoctest-generate` options still work. The test suite simulates such breaking changes. See the test_readme and test_docmod.py test functions that take the monkeypatch fixture.
- The `line_24` in the function name `session_00001_line_24` is the line number in tests/sample/README.md of the first line of the interactive session. `00001` is a sequence number to order the doctests.

Here is simulated output captured when `--phmdoctest-docmod` doctest collection breaks due to incompatible DoctestModule API.

```
pytest -v --phmdoctest-docmod
```

```
plugins: phmdoctest-1.0.0
collected 8 items

::README.py::test_code_10_output_17 PASSED
::README.py::test_unable_to_collect_doctests FAILED
::doc__directive2.py::test_code_25_output_32 PASSED
::doc__directive2.py::test_code_42_output_47 PASSED
::doc__directive2.py::test_code_52_output_56 PASSED
::doc__project.py::test_code_12_output_19 PASSED
::doc__project.py::test_unable_to_collect_doctests FAILED
tests/test_example.py::test_example PASSED
```

- There is one FAILED test called for each Markdown file with Python interactive sessions.
- The Python code/expected output examples still run successfully.
- The pytest test case in tests/test_example.py succeeds.

1.6 Generate test files

Save generated test files to the file system. Do not collect them. The plugin does not use the non-public DoctestModule when doing this.

```
pytest -v --phmdoctest-generate .gendir
```

```
plugins: phmdoctest-1.0.0
collected 1 item

tests/test_example.py::test_example PASSED
```

- Note that only test_example.py was collected.
- With `--phmdoctest-generate` the test files generated from Markdown do not get collected.
- Run pytest again on the generated test files.

- The generated test files become stale with time.
- Test files should be regenerated after modifying the Markdown.
- See below to generate and collect test files with a single pytest command.
- .gendir is cleaned of all *.py files before writing new test files. Pre-existing *.py files in the output directory get renamed. If output_directory inadvertently gets pointed at a Python source directory, the renamed files can be recovered by renaming them.
  - The FILENAME.py files found in the output directory are renamed to noFILENAME.sav.
  - If a noFILENAME.sav already exists it is not modified.
  - Files in target_dir with other extensions are not modified.
  - A FILENAME.py pre-existing in target_dir is only renamed and not deleted.
- .gendir is cleaned of all *.md files as well. Pre-existing FILENAME.md files in the output directory get renamed to FILENAME_md.sav.
- If .gendir was empty, it will now have these *.py files:

```
test_doc__directive2.py
test_doc__project.py
test_README.py
```

1.7 Generate and collect test files

A single pytest command will generate test files and collect them. The plugin does not use the non-public DoctestModule when doing this.

```
pytest -v --phmdoctest-generate .gendir . .gendir --doctest-modules --ignore src
```

```
plugins: phmdoctest-1.0.0
collected 10 items

tests/test_example.py::test_example PASSED
.gendir/test_README.py::test_README.session_00001_line_24 PASSED
.gendir/test_README.py::test_code_10_output_17 PASSED
.gendir/test_doc__directive2.py::test_code_25_output_32 PASSED
.gendir/test_doc__directive2.py::test_code_42_output_47 PASSED
.gendir/test_doc__directive2.py::test_code_52_output_56 PASSED
.gendir/test_doc__project.py::test_doc__project.session_00001_line_31 PASSED
.gendir/test_doc__project.py::test_doc__project.session_00002_line_46 PASSED
.gendir/test_doc__project.py::test_doc__project.session_00003_line_55 PASSED
.gendir/test_doc__project.py::test_code_12_output_19 PASSED
```

How it works:

- The plugin writes the generated test files during the pytest collection phase.
- This happens while pytest is collecting from `"."`
- pytest does not collect from `.gendir` until after discovering and collecting files in `"."`.
- The leading `"."` in `.gendir` prevents pytest from searching there for test files while searching `"."`. See norecursedirs default values in Pytest Documentation | API reference | Configuration Options | norecursedirs.

- The `--doctest-modules` option tells pytest to look for doctests in docstrings of *.py files.
- When doing `--doctest-modules`, the `--ignore src` option tells pytest not to collect modules from the src folder. We only want to collect doctests from .gendir. This prevents pytest from importing modules there to look for doctests.

These are the ini file settings:

```
# pytest.ini and tox.ini
[pytest]
addopts = --phmdoctest-generate=.gendir --doctest-modules --ignore src
```

```
# pyproject.toml
[tool.pytest.ini_options]
addopts = "--phmdoctest-generate=.gendir --doctest-modules --ignore src"
```

```
# setup.cfg  Note: Use is discouraged by pytest docs.
[tool:pytest]
addopts = --phmdoctest-generate=.gendir --doctest-modules --ignore src
```

Here is a demo that runs on a checked out copy of the repository.

```
# With a terminal in the tests/sample directory
# The first line collects 5 items.
# The second line collects 3 items.

pytest -v --phmdoctest-generate=.gendir "." .gendir --ignore README.md --ignore doc/
↪directive2.md --doctest-modules --ignore src
pytest -v --phmdoctest-generate=.gendir "." .gendir  --ignore-glob */*.md --doctest-
↪modules --ignore src
```

1.8 Help

pytest `--help` contains a **phmdoctest:** group in the middle and an ini-option near the bottom. The help contains:

- `--phmdoctest`
- `--phmdoctest-generate`
- `--phmdoctest-docmod`
- `phmdoctest-collect`

1.9 Configure collection

An optional `phmdoctest-collect =` section can be placed in the pytest ini file. It is a list of lines of the format

```
glob [options]
```

Consider using the section to pass collection options described below. The options have the same names and behave like the options accepted by phmdoctest usage.

- The Markdown file must match one of the globs.

- The glob is processed by Path.glob() from the Python standard library pathlib. Path.glob() offers a "**" recursive pattern that means "this directory and all subdirectories recursively."
- The globs are checked from top to bottom. The first glob to match the Markdown file determines the phmdoctest command line options.
- If there is no match the file will **not** be collected.
- A line can have just the glob and no options. The glob is required.
- The options should look like and have the same spacing as the command line options passed to the tool phmdoctest usage.
  - Use double quotes as needed in TEXT.
  - The plugin does not support \" escaped double quote.
  - Look for list of options in the next section.
- If a line that does not parse is needed, the plugin collects a special test file that contains a failing test case with an embedded error message.

Example

```
# pytest.ini
[pytest]
addopts = --phmdoctest-docmod
phmdoctest-collect =
    doc/project.md --skip greeting --skip enjoyment
    **/*.md
```

Then run this pytest command on the project files from the Usage section ...

```
pytest -v --ignore tests/test_example.py
```

output

```
plugins: phmdoctest-1.0.0
collected 7 items

::README.py::README.session_00001_line_24 PASSED
::README.py::test_code_10_output_17 PASSED
::doc__directive2.py::test_code_25_output_32 PASSED
::doc__directive2.py::test_code_42_output_47 PASSED
::doc__directive2.py::test_code_52_output_56 PASSED
::doc__project.py::doc__project.session_00001_line_31 PASSED
::doc__project.py::doc__project.session_00002_line_46 PASSED
```

- The example passes the options `--skip greeting` and `--skip enjoyment` when testing doc/project.md. The `--skip greeting` and `--skip enjoyment` options cause 2 examples to be skipped. These are the first and last examples in the file.
- The example tests the two Python interactive sessions in the midlle of the file as the last two items above.
- The "**/*.md" recursively matches other Markdown files. There are no otions.
- The ini file globs above only apply to .md files. We ignore the Python pytest test file tests/test_example.py by adding the `--ignore` option.
- Look for more phmdoctest-collect examples on GitHub in tests/test_collect_section.py.

1.9.1 phmdoctest-collect options

`-s, --skip TEXT`

Do not test blocks with substring TEXT. Allowed multiple times.

`--fail-nocode`

Markdown file with no code blocks left after applying skips generates a failing test.

`-u, --setup TEXT`

Run block with substring TEXT at test module setup time.

`-d, --teardown TEXT`

Run block with substring TEXT at test module teardown time.

`--setup-doctest`

Make globals created by the `--setup` Python code block or setup directive visible to Python interactive session `>>>` blocks. **Caution:** The globals are set at pytest Session scope. The globals are visible to all doctests in the test suite. This includes doctests collected by the plugin and doctests collected from other files due to `--doctest-modules`.

1.9.2 Notes

- Fenced code blocks are searched for the substring TEXT.
- `--skip TEXT` can apply to more than one block.
- Exactly one block can match `--setup` and `--teardown`.

1.9.3 Equivalent directives

The HTML directive comments below placed in the Markdown file can be used instead of specifying options in the phmdoctest-collect section. Directives select a single fenced code block. There are 10 directives in phmdoctest Directives. Here are the directives equivalent to the collect section options.

```
  collect-section          HTML Directive
  option                   comment
-------------------------------------------------
-s, --skip TEXT         <!--phmdoctest-skip-->
-u, --setup TEXT        <!--phmdoctest-setup-->
-d, --teardown TEXT     <!--phmdoctest-teardown-->
```

1.10 Hints

- When invoking pytest, cwd must be in the subpath of the files to be collected to avoid this error from pathlib.py in relative_to(): ValueError: <file to be collected> is not in the subpath of <working directory>
- Note the plugin does not accept single quoted phmdoctest args in the phmdoctest-collect section. A failing test will be collected.
- Use underscore in conftest.py for pytest_plugins: `pytest_plugins = ["pytest_phmdoctest"]`
- An ImportPathMismatchError indicates two test files have the same name.
- If using `--phmdoctest-generate` add `.gendir` to .gitignore.
- pytest -vv output shows the path to the plugin temporary directory.
- The `--report` option of the phmdoctest command lists all fenced code blocks in the Markdown file.
- phmdoctest can generate test files for multiple Markdown files with one call by specifying a configuration file.
- phmdoctest offers pytest pytester fixtures (suitable for plugin development) to generate and run tests for a single Markdown file.

1.11 Related projects

- phmdoctest
- rundoc
- byexample
- sphinx.ext.doctest
- sybil
- doxec
- egtest
- pytest-codeblocks

CHAPTER

TWO

RECENT CHANGES

1.0.0 - 2022-04-15

- `--phmdoctest` option only does Python code/expected output.
- Add `--phmdoctest-docmod` option to do both:
  - Python code/expected output.
  - Python interactive sessions.
- Rename `--phmdoctest-save` option to `--phmdoctest-generate`.
- Collect Markdown only if it has Python examples.
- Runs on pytest 7, as well as 5.4.3 and 6.

0.0.3 - 2021-11-10

- Initial upload to Python Package Index.

CHAPTER

THREE

CONTRIBUTING

- Create an issue or submit a pull request forked from the develop branch.
- For pull requests please refer to steps 1-6 at the top of Contributing to Simple Icons

Preconditions for pull request merge:

- For bug fixes a test that fails.
- Documentation and test updates for features.

CHAPTER

FOUR

SEARCH

- search