

Effective Programming Practices for Economists

Software engineering

What does pytest do?

Janoš Gabler and Hans-Martin von Gaudecker

Example

consider this hypothetical survey about a programming course

```
>>> raw = pd.read_csv("survey.csv")
>>> raw
```

Q001	Q002	Q003
0 strongly disagree	agree	python
1 strongly agree	strongly agree	Python
2 -77	disagree	R
3 agree	-77	Python
4 -99	-99	Python
5 nan	strongly agree	Python
6 neutral	strongly agree	Python
7 disagree	agree	python
8 strongly disagree	-99	PYTHON
9 -77	-99	Ypthon

From the metadata you know

- Q001: I am a coding genius
- Q001: I learned a lot
- Q003: What is your favourite language
- -77 not readable
- -99 no reply

Two functions in clean_data.py

```
def _clean_agreement_scale(sr):  
    sr = sr.replace({"-77": pd.NA, "-99": pd.NA})  
    categories = ["strongly disagree", "disagree", "neutral", "agree", "strongly agree"]  
    dtype = pd.CategoricalDtype(categories=categories, ordered=True)  
    return sr.astype(dtype)
```

```
def _clean_favorite_language(sr):  
    sr = sr.replace({"-77": pd.NA, "-99": pd.NA})  
    sr = sr.str.lower().str.strip()  
    sr = sr.replace("ypthon", "python")  
    return sr.astype(pd.CategoricalDtype())
```

New module: test_clean_data.py

- 4 assertions whether actual results match our expectation
- Will look at syntax in subsequent screencast

Step 1: Collection



- Go through all folders in working directory
- Collect all files with name ``test_XXX.py``
- Go through those files and collect all functions that start with ``test_``
- All these test functions will be executed (fine-grained control possible)

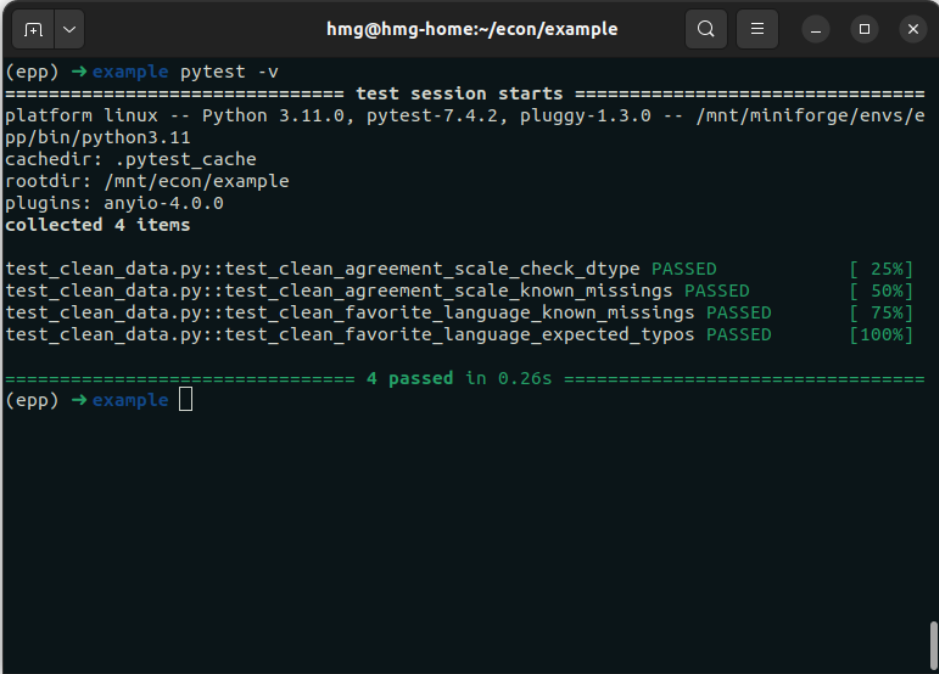
Step 2: Execute the tests

- All test functions are executed
- A report is printed to the screen



Step 2: Execute the tests

- `pytest -v` gives more detailed progress reports
- Can be very helpful for long-running tests

A terminal window with a dark background and light text. The title bar shows the user 'hmg' at 'hmg-home' in the directory '~/econ/example'. The prompt is '(epp) → example'. The command 'pytest -v' has been executed. The output shows a detailed test session start, including platform, Python version, pytest version, and plugins. It lists 4 collected items and shows the results of 4 tests, all of which passed. The progress is shown as a percentage in brackets on the right: [25%], [50%], [75%], and [100%]. The final summary line indicates '4 passed in 0.26s'. The prompt is now '(epp) → example' followed by a cursor.

```
hmg@hmg-home:~/econ/example
(epp) → example pytest -v
===== test session starts =====
platform linux -- Python 3.11.0, pytest-7.4.2, pluggy-1.3.0 -- /mnt/miniforge/envs/e
pp/bin/python3.11
cachedir: .pytest_cache
rootdir: /mnt/econ/example
plugins: anyio-4.0.0
collected 4 items

test_clean_data.py::test_clean_agreement_scale_check_dtype PASSED [ 25%]
test_clean_data.py::test_clean_agreement_scale_known_missings PASSED [ 50%]
test_clean_data.py::test_clean_favorite_language_known_missings PASSED [ 75%]
test_clean_data.py::test_clean_favorite_language_expected_typos PASSED [100%]

===== 4 passed in 0.26s =====
(epp) → example
```

Step 3: Inspect failures

```

hmg@hmg-home:~/econ/example
(epp) → example pytest
===== test session starts =====
platform linux -- Python 3.11.0, pytest-7.4.2, pluggy-1.3.0
rootdir: /mnt/econ/example
plugins: anyio-4.0.0
collected 4 items

test_clean_data.py .F.. [100%]

===== FAILURES =====
_____ test_clean_agreement_scale_known_missings _____

    def test_clean_agreement_scale_known_missings():
        sr = pd.Series(["-77", "-99"])
        result = _clean_agreement_scale(sr)
        expected = pd.Series([pd.NA, "agree"], dtype=result.dtype)
>         pd.testing.assert_series_equal(result, expected)

test_clean_data.py:16:
testing.pyx:55: in pandas._libs.testing.assert_almost_equal
    ???

-----
> ???
E   AssertionError: Series are different
E
E   Series values are different (50.0 %)
E   [index]: [0, 1]
E   [left]: [NaN, NaN]
E   Categories (5, string): [strongly disagree < disagree < neutral < agree < strongly agree]
E   [right]: [NaN, 'agree']
E   Categories (5, string): [strongly disagree < disagree < neutral < agree < strongly agree]
E   At positional index 1, first diff: nan != agree

testing.pyx:173: AssertionError
===== short test summary info =====
FAILED test_clean_data.py::test_clean_agreement_scale_known_missings - AssertionError: Series are different
===== 1 failed, 3 passed in 0.31s =====
(epp) → example

```


Step 3: Inspect failures with pdb

