Effective Programming Practices for Economists

Software engineering

What does pytest do?

Janoś 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

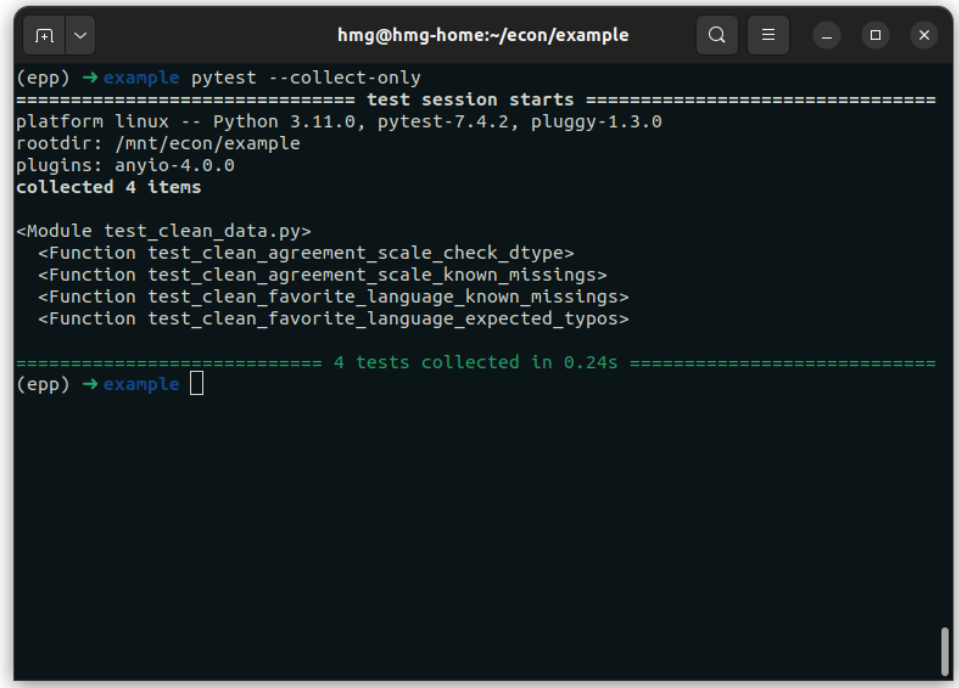
```
def _clean_agreement_scale(sr):
    sr = sr.replace({"-77": pd.NA, "-99": pd.NA})
    categories = ["strongly disagree", "disagree", "neutral", "agree", "strongly agree"]
    dtype = pd.CategoricalDtype(categories=categories, ordered=True)
    return sr.astype(dtype)


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

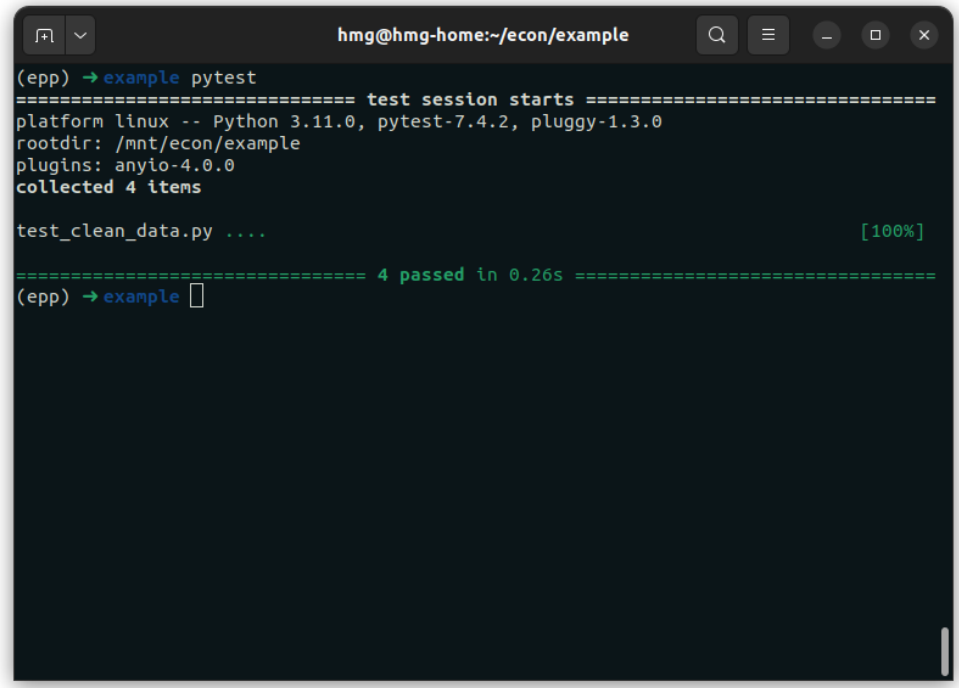
Step 1: Collection

- 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)



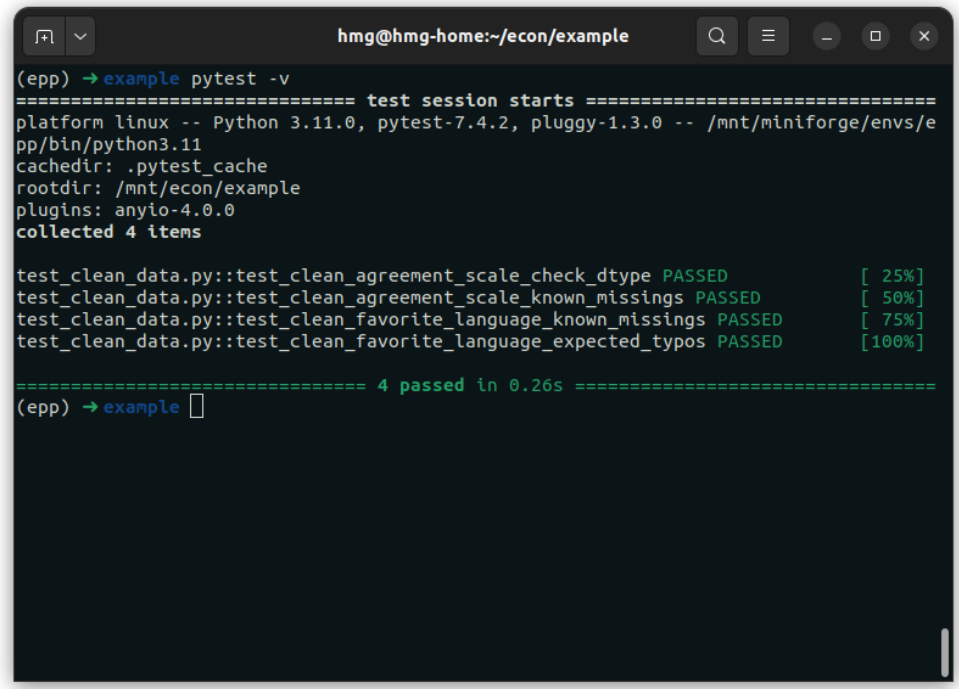
Step 2: Execute the tests

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

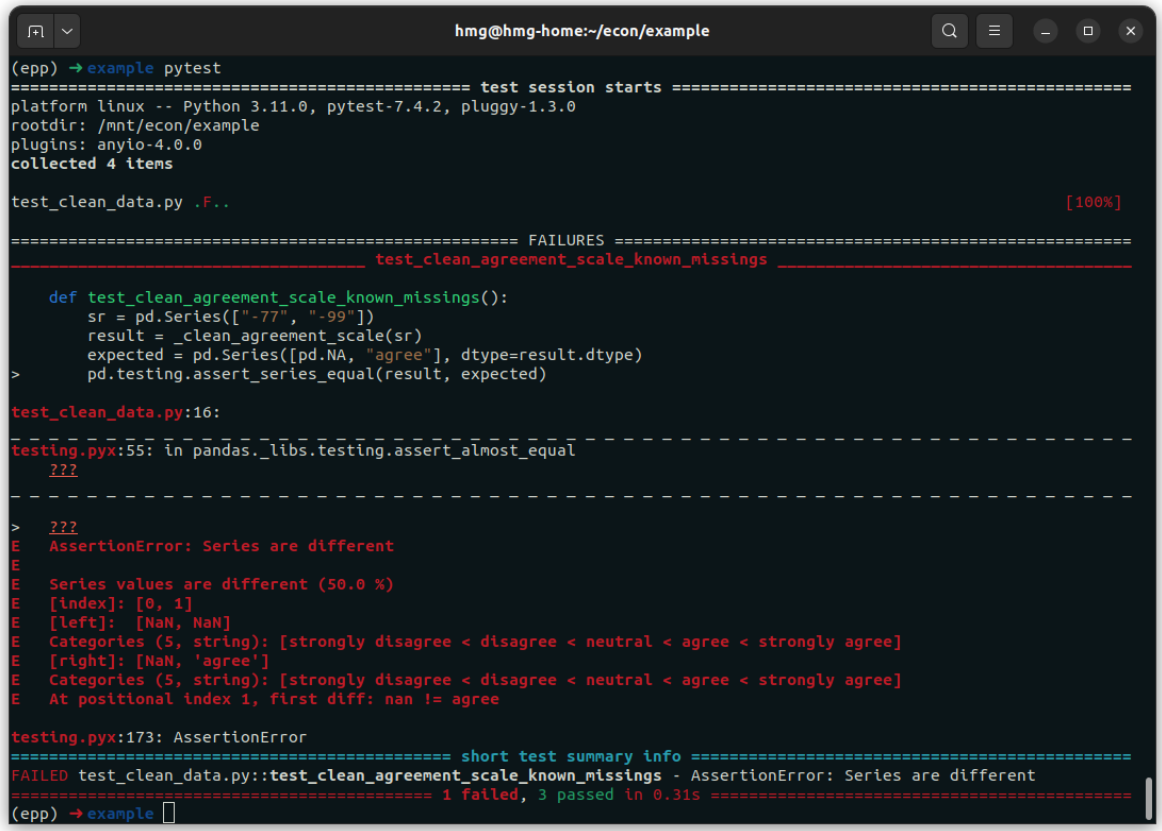
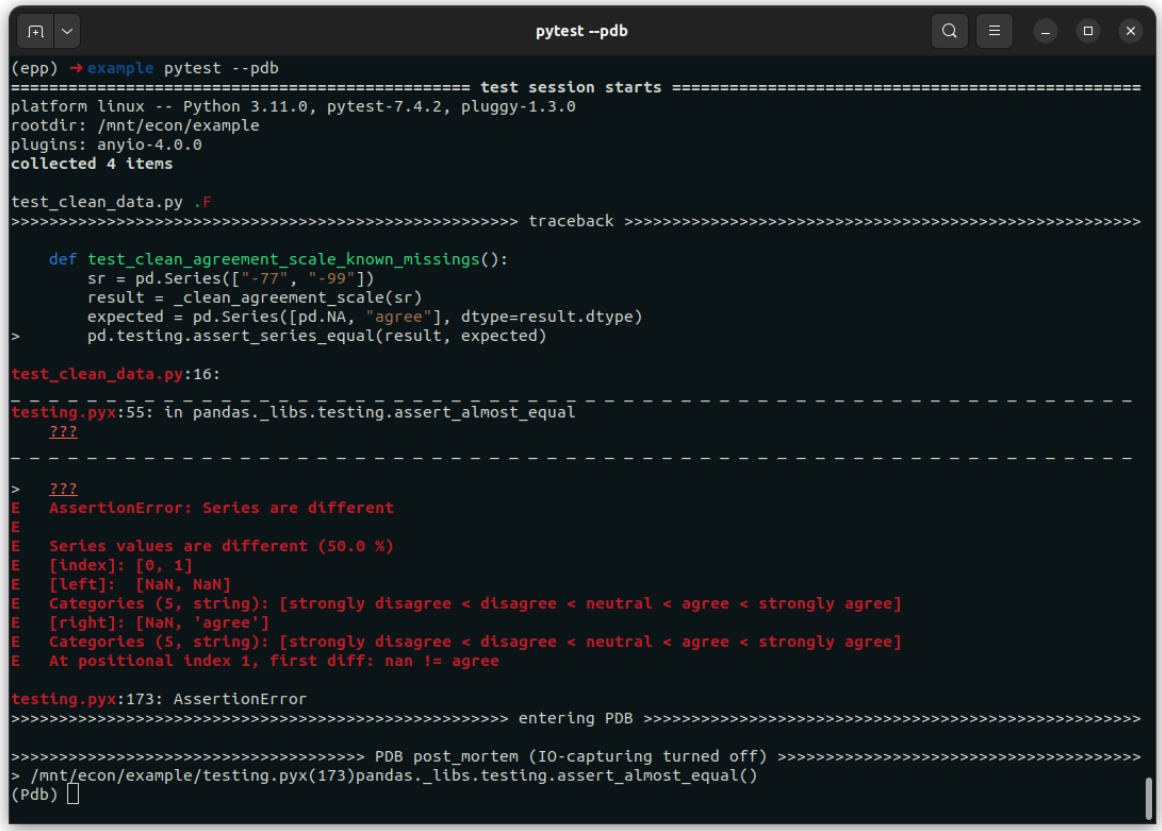


Step 2: Execute the tests

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



Step 3: Inspect failures



Step 3: Inspect failures with pdb