Scientific Computing Using Python

Testing

14th of June, 2023

Thomas Arildsen

tari@its.aau.dk

CLAAUDIA

Aalborg University

Testing

Testing
Overview

Why test – and how?

- It is a quality assurance to validate and verify that the software works as it should (meets specifications).
- Typically this demands many different specific tests (catching bad input, check for correct output, no unforeseen artefacts etc.).

Testing levels:

- Unit tests $\rightarrow$ test of individual functions etc.
- Integration tests $\rightarrow$ test of interfaces between modules or software components.
- System tests $\rightarrow$ complete end-2-end test (could typically be a selection of well chosen examples of the use of the software).
- Acceptance tests $\rightarrow$ test performed by the end-user to ensure that the software performs as required.

Testing
Overview

Testing types:

- Sanity testing $\rightarrow$ an initial test to decide if further testing makes sense – could be to check that outputs are of the correct types etc.
- Installation testing $\rightarrow$ test that the software is correctly installed and aligned with e.g. the operating system.
- Regression testing $\rightarrow$ test that modifications have not caused regression of errors.
- Functional test $\rightarrow$ black box approach; feed input, test the output against oracle or known specifications.
- . . . more types exist . . . performance test, interface testing, . . .

Testing
Overview

Some aspects:

- At Google [WAC12]: “Testing must not create friction that slows down innovation and development.”.
- Once a bug is found, write a test to catch it. Then correct the code and confirm it passes the test. [WMV03].
- Functional tests . . . [must be] easier to write: If writing them is a bottleneck to writing production code, they’ll be considered optional and quickly become incomplete and obsolete. [And07]
- Quote [Edd09].: “. . . with Matlab users, people often understand the need for software testing, but might not know how to go about it.”
- Quote [EMT05]: “The key aspect of TDD [Test Driven Design] is that programmers write low-level functional tests before production code.”

In computational science:

- The objective is to do research . . . not to write software which is “just” a tool. Therefore, testing is not generally a priority.
- Pragmatically: all the testing you do is better than no testing. But be careful not to take passed tests as a guarantee for safe software. False positives are “dangerous”.

Testing
The easy win

We believe that most computational science can have an easy win by adopting

- Unit and integration ...
- for functional and regression testing ..
- using automatic and easy to handle/maintain tools

Doctest

- Part of the standard library.
- Uses “examples” in the docstring

```python
def function(x):
    """
    This function ....

    example
    >>> function(40)
    [10, 20, 30, 40]

    """
```

- A test script, e.g. test.py

```python
import mymodule
import doctest
doctest.testmod(mymodule, verbose=True)
```

- Can be executed as python test.py

Doctest
Mini-exercise

Implement doctest for an exercise from this course.

- If the program uses floating-point numbers, then compare with `numpy.allclose(a, b)`

Pytest

```python
def primes(x):
    ...
```

```python
from find_primes import primes


class TestPrimes:

    def test_boundary(self):
        assert primes(0) == []
        assert primes(1) == []
        assert primes(2) == [2]

    def test_case(self):
        assert primes(30) == [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

- Simply uses Python's `assert` to check if expectations are met.
- Pytest automatically discovers tests in the current folder and subfolders from where you run it: all files called `test_*.py` or `*_test.py`, all classes therein prefixed with `Test` and functions prefixed with `test_`.

Pytest
Mini-exercise

Implement Pytest tests for an exercise from this course.

- If the program uses floating-point numbers, then compare with `numpy.allclose(a, b)`

Testing I
Literature

[And07] Jennitta Andrea. "Envisioning the Next Generation of Functional Testing Tools". In: *IEEE Software* 24.3 (2007), pp. 58–66. ISSN: 0740-7459. DOI: `http://doi.ieeecomputersociety.org/10.1109/MS.2007.73`.

[Edd09] Steven L Eddins. "Automated software testing for matlab". In: *Computing in science & engineering* 11.6 (2009), pp. 48–55.

[EMT05] H. Erdogmus, Maurizio Morisio, and Marco Torchiano. "On the effectiveness of the test-first approach to programming". In: *Software Engineering, IEEE Transactions on* 31.3 (Mar. 2005), pp. 226–237. ISSN: 0098-5589. DOI: `10.1109/TSE.2005.37`.

Testing II
Literature

[WAC12] James A Whittaker, Jason Arbon, and Jeff Carollo. *How Google Tests Software*. Addison-Wesley Professional, 2012.

[WMV03] Laurie Williams, E Michael Maximilien, and Mladen Vouk. "Test-driven development as a defect-reduction practice". In: *Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on*. IEEE. 2003, pp. 34–45.