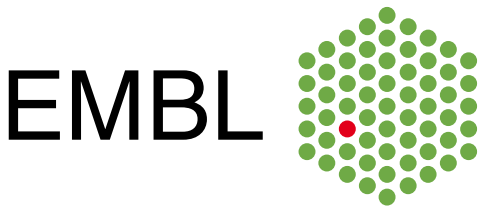
Unit testing using Python nose

Luis Pedro Coelho
§
On the web: http://luispedro.org
On twitter: @luispedrocoelho

European Molecular Biology Laboratory

June 11, 2014



Motivation

Scientific code must not just produce nice looking output, but actually be correct.
http://bit.ly/testing-science

Motivation (II)

Some recent code-related scientific catastrophes:

- Geoffrey Chang
- Abortion reduces crime? maybe not so much once you fix the bug
- Rogoff's "Growth in a Time of Debt" paper is a famous example (even if the bug itself is only a small part of the counter-argument)
- East Anglia "Climategate"

Why do things go wrong?

1. Your code is correct, but input files are wrong/missing/, the network goes down ...
2. Your code is buggy.

Never fail silently!

- The worst thing is to fail silently.
- Fail loudly and clearly

(This is partially why Unix tradition is to produce no output when things go well)

Defensive Programming

Defensive programming means writing code that will catch bugs early.

Assertions

```
def stddev(values):
    '''
    S = stddev(values)

    Compute standard deviation
    '''
    assert len(values) > 0, 'stddev: got empty list.'
    ...
```

Assertions

```
def stddev(values):
    '''
    S = stddev(values)

    Compute standard deviation
    '''
    if len(values) <= 0:
        raise AssertionError(
            'stddev: got empty list.')
    ...
```

Preconditions

In computer programming, a precondition is a condition or predicate that must always be true just prior to the execution of some section of code.

(Wikipedia)

Preconditions

Other Languages

- C/C++ #include <assert.h>
- Java assert pre-condition
- Matlab assert() (in newer versions)
- ... ...

Assertions Are Not Error Handling!

- Error handling protects against outside events; assertions protect against programmer mistakes.
- Assertions should never be false.

Programming by Contract

1. pre-conditions.
2. post-conditions.
3. invariants.

Pre-condition

What must be true before calling a function.

Post-condition

What is true after calling a function.

Testing

Do you test your code?

Unit Testing

```
def test_stddev_const():
    assert stddev([1]*100) < 1e-3

def test_stddev_positive():
    assert stddev(range(20)) > 0.
```

Nosetest

Nose software testing framework:

- Tests are named test_something.
- Conditions are asserted.

Software Testing Philosophies

1. Test everything. Test it twice.
2. Write tests first.
3. Regression testing.

Regression Testing

Make sure bugs only appear once!

Practical Session: some preliminaries

statistics.py

```
def stddev(xs):
    ...
```

test_statistics.py

```
def test_stddev_const():
    assert stddev([1]*100) < 1e-3

def test_stddev_positive():
    assert stddev(range(20)) > 0.
```

Practical Session: some preliminaries

statistics.py

```
def stddev(xs):
    ...
```

test_statistics.py

```
import statistics
def test_stddev_const():
    assert statistics.stddev([1]*100) < 1e-3

def test_stddev_positive():
    assert statistics.stddev(range(20)) > 0.
```

Practical: Python III & Unit testing

1. You can either start from scratch or check the files I give you (or any combination of both).
2. Goal is to write code to do a simple task & test it.

Types of tests

- Smoke test: just check it runs
- Corner/edge cases: check “complex” cases.
- Case testing: test a “known case”
- Regression testing: create a test when you find a bug.
- Integration test: test that different parts work together.

Goals

1. Download files from https://github.com/luispedro/2014-06-10-cyi-support/testing
2. There is a data file (data.txt)
3. See the code in main.py, which loads it.
4. Write a function average in a file called robust.py, which computes the average of a sequence of numbers, whilst ignoring the maximum and minimum.
5. Write tests for robust.average.
6. (If you have the time, you can look at plots.py)