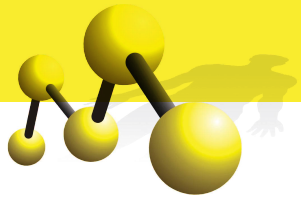
Testing

Luis Pedro Coelho

Programming for Scientists

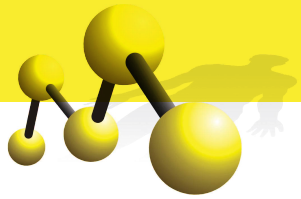
September 24, 2012

Defensive Programming



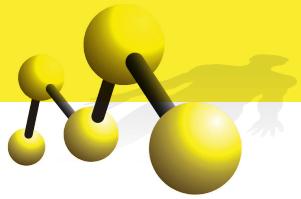
Defensive programming means writing code that will catch bugs early.

Remember the Homework?



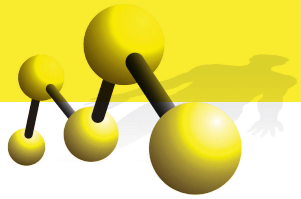
```
def trim(qs, thresh):
    ...
    assert len(qs) > 0, 'trim: got empty list'
```

Assert



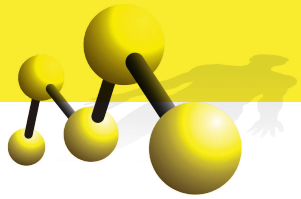
```
assert <condition>, <error message>
```

Testing



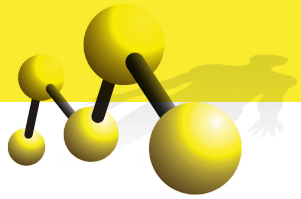
Do you test your code?

Testing trim



```
import numpy as np
from trimfq import trim

qs = np.array([])
trim(qs, 20)

qs = np.array([20,20])
trim(qs, 20)
```

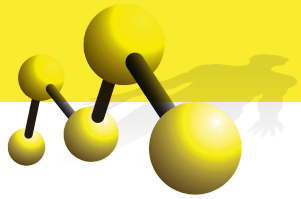
Testing trim



```
import numpy as np
from trimfq import trim

qs = np.array([])
trim(qs, 20)

qs = np.array([20,20])
trim(qs, 20)
```

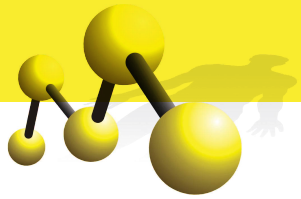
These simple sort of tests are called smoke tests.

Testing trim II



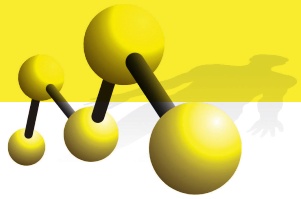
```
qs = np.array([10,10,10,20,20,20,20,10])
s,e = trim(qs, 15)
assert np.all(qs[s:e] >= 15)
assert s < e
```

Testing trim III



Where are errors likely to lurk?

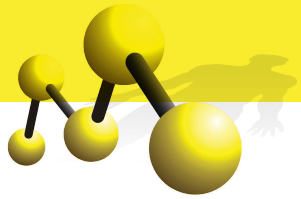
Testing trim III



Where are errors likely to lurk?

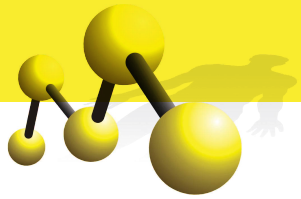
- At the edges?
- What if the whole string is above threshold?
- What if the whole string is below threshold?

Testing trim IV



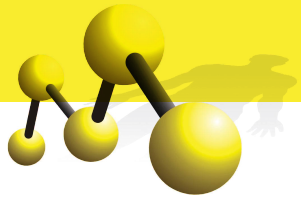
```
s,e = trim(np.array([10,10,10,10]), 5)
assert s == 0
assert e == 4

s,e = trim(np.array([10,10,10,10]), 15)
assert s == e # Note that we
              # DO NOT care about
              # actual values
```

Testing trim V



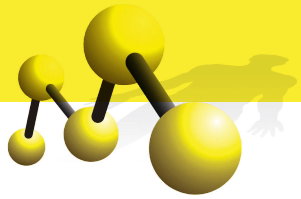
```
s,e = trim(np.array([10,10,20,20]), 15)
assert s == 2
assert e == 4

s,e = trim(np.array([20,20,10,10]), 15)
assert s == 0
assert e == 2
```

Fencepost Errors



If you build a straight fence 100 meters long with posts 10 meters apart, how many posts do you need?

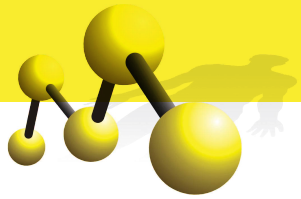
Fencepost Errors



If you build a straight fence 100 meters long with posts 10 meters apart, how many posts do you need?

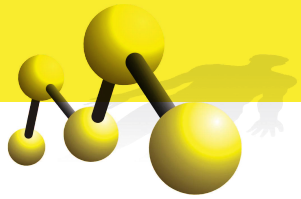
Eleven, but we often think 10.

What is the use of testing?



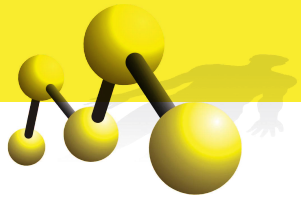
- Ok, I tested it
- It seems to work
- Now, I am happy

What is the use of testing?



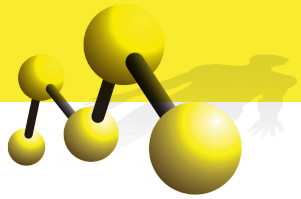
- Ok, I tested it
- It seems to work
- Now, I am happy
- But save those tests!

When your code changes



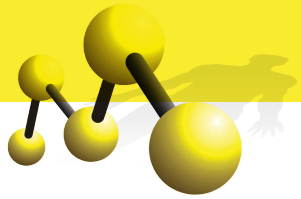
- When your code changes...

When your code changes



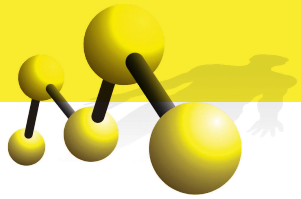
- When your code changes...
- ...you rerun your tests.
- Over time, you will accumulate a collection of tests.

Software Testing Philosophies



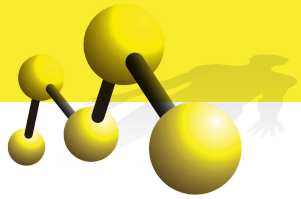
1. Test everything. Test it twice.
2. Write tests first.
3. Regression testing.

Regression Testing



Make sure bugs only appear once!

Nose testing



- Many utilities already exist to help manage test suites (A test suite is a fancy name for "a bunch of tests).
- In Python, nose is the most popular one.

http://nose.readthedocs.org/en/latest/