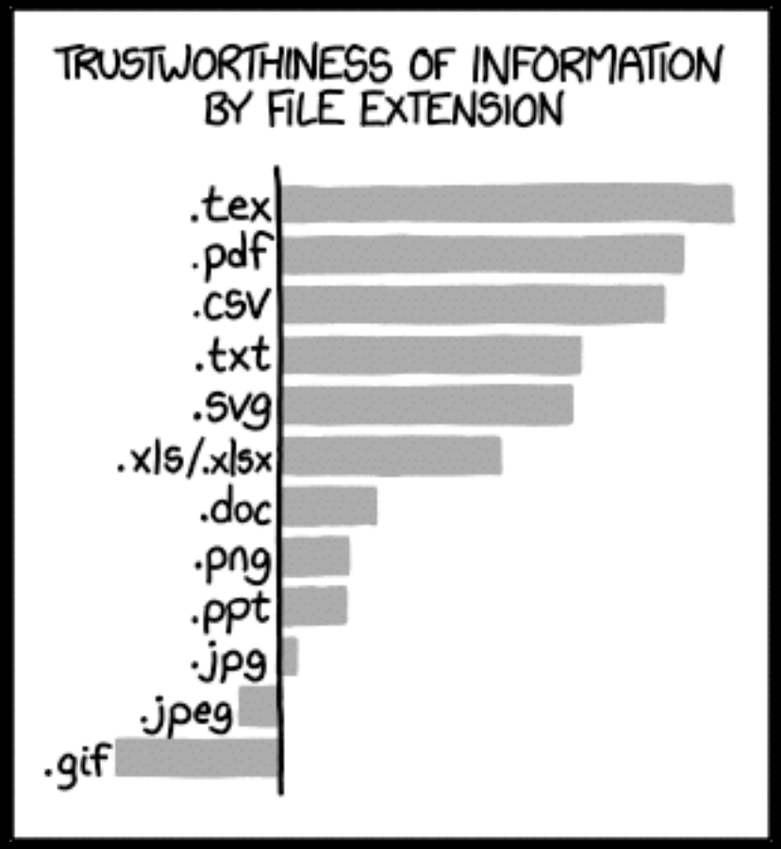
Unit Testing

Sebastian Raschka
January 8, 2014



http://xkcd.com/1301/

What is unit testing?

testing of a "unit" of code:
module, class, function, data file

in isolation!

What is unit testing?

a "unit":

1) fixture (e.g., function)
2) action (e.g., invoking function with particular input)
3) expected result (e.g., return value of a function)
4) actual result (e.g., actual return value of a function
5) report (e.g., success or failure)

Why unit testing?

testing functionality (incl. edge cases)

Why unit testing?

alter your code and make
sure that you didn't break anything

Why unit testing?

debugging

Why unit testing?

trust your code

Why unit testing?

saving time in the long run

Why unit testing?

trust code from collaborators

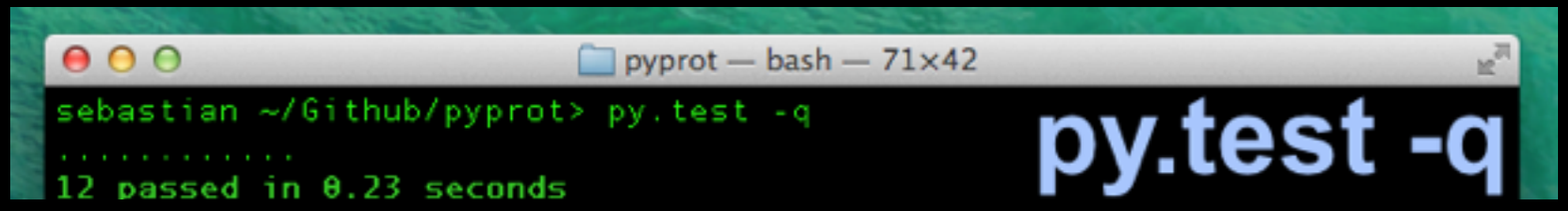
Why unit testing?

credibility for publication

unit testing frameworks in python

- unittest
- nose
- py.test

Running unit tests



pyprot — bash — 71×42

```
sebastian ~/Github/pyprot> py.test -q
............
12 passed in 0.23 seconds
sebastian ~/Github/pyprot> py.test
======================== test session starts ========================
platform darwin -- Python 2.7.5 -- pytest-2.5.0
collected 12 items

test/test_bfact.py ...
test/test_center_of_mass.py .
test/test_filefunc.py .
test/test_filter_cont.py ..
test/test_pdbobj.py ...
test/test_rmsd.py .
test/test_mol2/test_mol2_in.py .

===================== 12 passed in 0.23 seconds =====================
```

py.test -q

py.test

Here, each "dot" represents a single test

pyprot — bash — 71×42

```
sebastian ~/Github/pyprot> py.test -q
............
12 passed in 0.23 seconds
```

py.test -q

```
sebastian ~/Github/pyprot> py.test
======================== test session starts ========================
platform darwin -- Python 2.7.5 -- pytest-2.5.0
collected 12 items

test/test_bfact.py ...
test/test_center_of_mass.py .
test/test_filefunc.py .
test/test_filter_cont.py ..
test/test_pdbobj.py ...
test/test_rmsd.py .
test/test_mol2/test_mol2_in.py .

===================== 12 passed in 0.23 seconds =====================
```

Here, each "dot" represents a single test

py.test

```
sebastian ~/Github/pyprot> nosetests
............
----------------------------------------------------------------------
Ran 12 tests in 0.109s

OK
```

nosetests

pyprot — bash — 71×42

```
sebastian ~/Github/pyprot> py.test -q
............
12 passed in 0.23 seconds
```

py.test -q

```
sebastian ~/Github/pyprot> py.test
======================= test session starts =======================
platform darwin -- Python 2.7.5 -- pytest-2.5.0
collected 12 items

test/test_bfact.py ...
test/test_center_of_mass.py .
test/test_filefunc.py .
test/test_filter_cont.py ..
test/test_pdbobj.py ...
test/test_rmsd.py .
test/test_mol2/test_mol2_in.py .

==================== 12 passed in 0.23 seconds ====================
```

Here, each "dot" represents a single test

py.test

```
sebastian ~/Github/pyprot> nosetests
............
----------------------------------------------------------------------
Ran 12 tests in 0.109s

OK
```

nosetests

```
sebastian ~/Github/pyprot> nosetests -v
test_bfact.test_get_bfactor ... ok
test_bfact.test_median_bfactor ... ok
test_bfact.test_mean_bfactor ... ok
test_center_of_mass.test_center_of_mass ... ok
test_filefunc.test_open_file ... ok
test_filter_cont.test_filter_column_match ... ok
test_filter_cont.test_filter_col_match_exclude ... ok
test_mol2_in.test_multi_mol2list ... ok
test_pdbobj.test_constructor ... ok
test_pdbobj.test_calpha ... ok
test_pdbobj.test_main_chain ... ok
test_rmsd.test_rmsd ... ok

----------------------------------------------------------------------
Ran 12 tests in 0.095s

OK
sebastian ~/Github/pyprot>
```

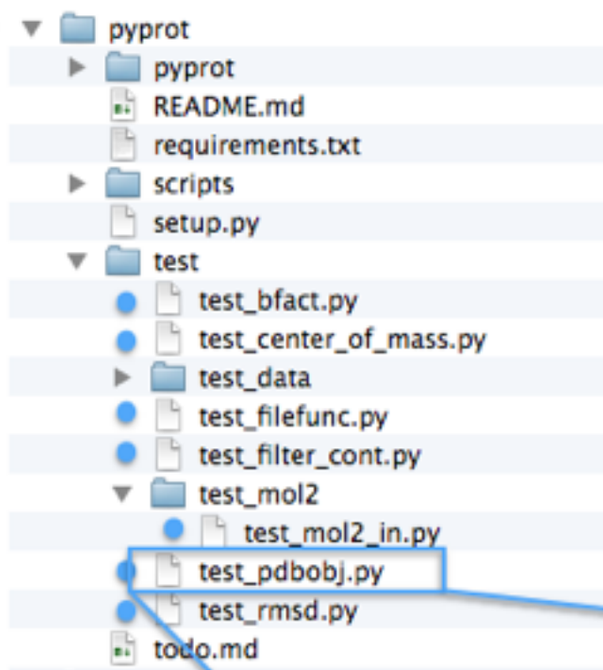
nosetests -v

PSA lab: python3 module

add it to your `.bashrc` (or `.personal`) file:

```
module load python/3.3.3
```

nose/py.test →

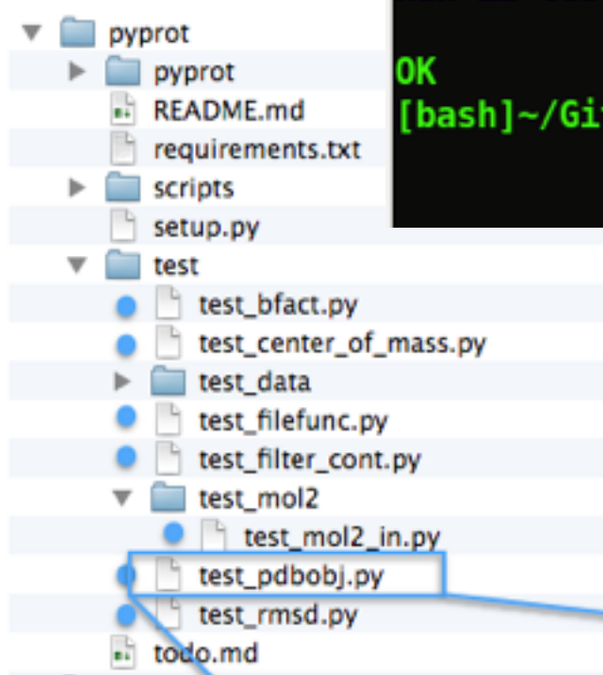


nose descending the directory tree looking for prefix "`test`"

test_pdbobj.py (~/Github/pyprot/test) – VIM

```
import pyprot.pdb as ppdb

pdb1 = ppdb.PdbObj("./test/test_data/3EIY.pdb")
pdb2 = ppdb.PdbObj("./test/test_data/small_3EIY.pdb")


def test_constructor():
    assert len(pdb1.cont) == 2147
    assert len(pdb1.atom) == 1330
    assert len(pdb1.hetatm) == 151
    assert len(pdb1.conect) == 54

def test_calpha():
    atom = [
    'ATOM      2  CA  SER A   2       3.259  54.783  -0.368  1.00 52.54          C',
    'ATOM      8  CA  PHE A   3       4.261  51.413   1.102  1.00 48.73          C',
    'ATOM     19  CA  SER A   4       4.593  49.745  -2.323  1.00 47.00          C',
    'ATOM     25  CA  ASN A   5       7.351  52.145  -3.480  1.00 41.42          C',
    'ATOM     33  CA  VAL A   6       9.636  51.959  -0.457  1.00 32.41          C'
    ]
    assert atom == pdb2.calpha()
```

nose/py.test →



Terminal

File Edit View Search Terminal Help

```
[bash]~/Gits/Github/pyprot >nosetests
.............
----------------------------------------------------------------------
Ran 12 tests in 0.143s

OK
[bash]~/Gits/Github/pyprot >
```

test_pdbobj.py (~/Github/pyprot/test) – VIM

```
import pyprot.pdb as ppdb

pdb1 = ppdb.PdbObj("./test/test_data/3EIY.pdb")
pdb2 = ppdb.PdbObj("./test/test_data/small_3EIY.pdb")


def test_constructor():
    assert len(pdb1.cont) == 2147
    assert len(pdb1.atom) == 1330
    assert len(pdb1.hetatm) == 151
    assert len(pdb1.conect) == 54

def test_calpha():
    atom = [
    'ATOM      2  CA  SER A   2       3.259  54.783  -0.368  1.00 52.54          C',
    'ATOM      8  CA  PHE A   3       4.261  51.413   1.102  1.00 48.73          C',
    'ATOM     19  CA  SER A   4       4.593  49.745  -2.323  1.00 47.00          C',
    'ATOM     25  CA  ASN A   5       7.351  52.145  -3.480  1.00 41.42          C',
    'ATOM     33  CA  VAL A   6       9.636  51.959  -0.457  1.00 32.41          C'
    ]
    assert atom == pdb2.calpha()
```

nose/py.test →

- pyprot
  - pyprot
  - README.md
  - requirements.txt
  - scripts
  - setup.py
  - test
    - test_bfact.py
    - test_center_of_mass
    - test_data
    - test_filefunc.py
    - test_filter_cont.py
    - test_mol2
      - test_mol2_in.py
    - test_pdbobj.py
    - test_rmsd.py
  - todo.md

Terminal

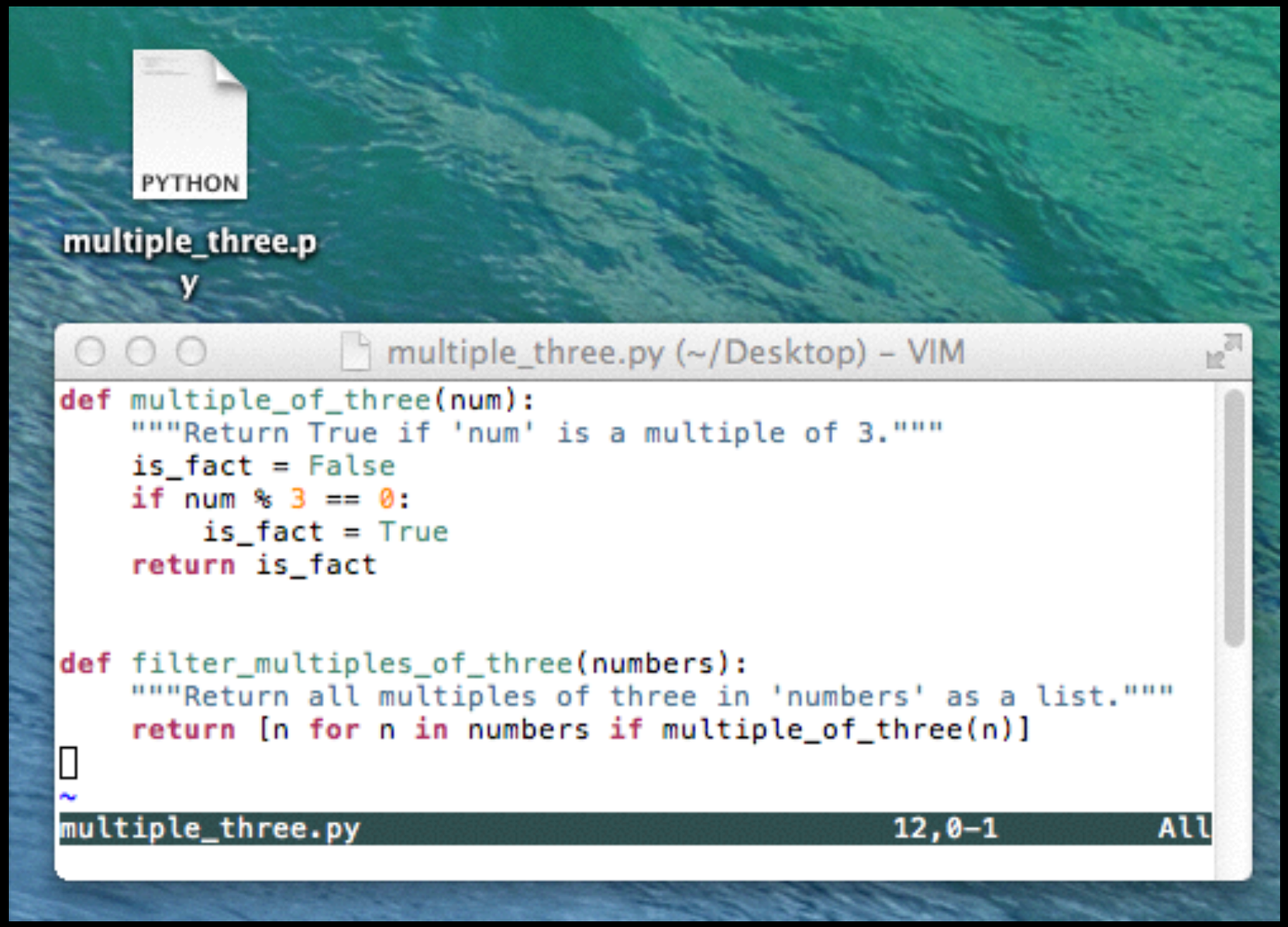
File Edit View Search Terminal Help

```
[bash]~/Gits/Github/pyprot >nosetests -v
test_bfact.test_get_bfactor ... ok
test_bfact.test_median_bfactor ... ok
test_bfact.test_mean_bfactor ... ok
test_center_of_mass.test_center_of_mass ... ok
test_filefunc.test_open_file ... ok
test_filter_cont.test_filter_column_match ... ok
test_filter_cont.test_filter_col_match_exclude ... ok
test_mol2_in.test_multi_mol2list ... ok
test_pdbobj.test_constructor ... ok
test_pdbobj.test_calpha ... ok
test_pdbobj.test_main_chain ... ok
test_rmsd.test_rmsd ... ok

----------------------------------------------------------------------
Ran 12 tests in 0.128s

OK
[bash]~/Gits/Github/pyprot >
```

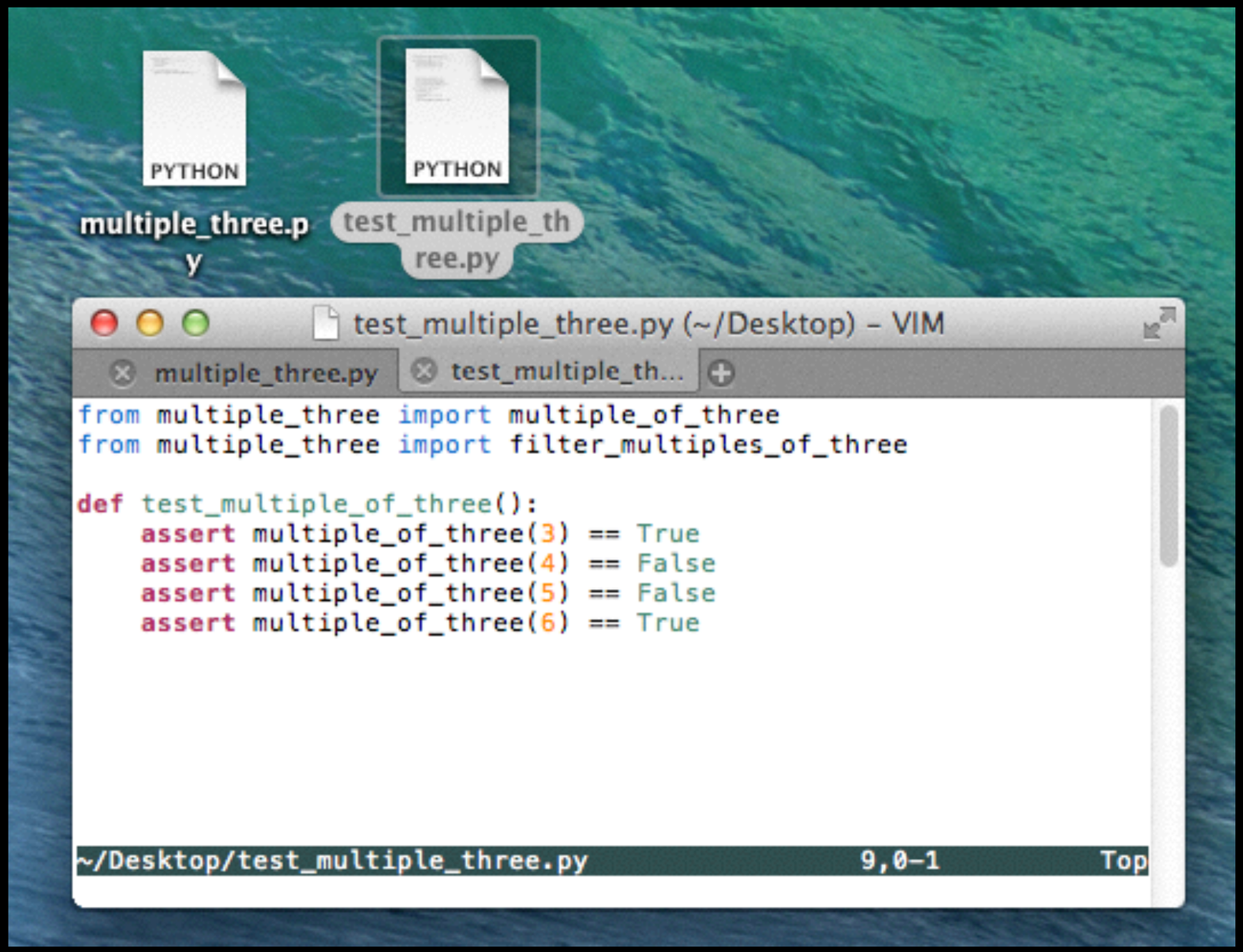
test_pdbobj.py (~/Github/pyprot/test) – VIM

```
import pyprot.pdb as ppdb

pdb1 = ppdb.PdbObj("./test/test_data/3EIY.pdb")
pdb2 = ppdb.PdbObj("./test/test_data/small_3EIY.pdb")


def test_constructor():
    assert len(pdb1.cont) == 2147
    assert len(pdb1.atom) == 1330
    assert len(pdb1.hetatm) == 151
    assert len(pdb1.conect) == 54

def test_calpha():
    atom = [
    'ATOM      2  CA  SER A   2       3.259  54.783  -0.368  1.00 52.54           C',
    'ATOM      8  CA  PHE A   3       4.261  51.413   1.102  1.00 48.73           C',
    'ATOM     19  CA  SER A   4       4.593  49.745  -2.323  1.00 47.00           C',
    'ATOM     25  CA  ASN A   5       7.351  52.145  -3.480  1.00 41.42           C',
    'ATOM     33  CA  VAL A   6       9.636  51.959  -0.457  1.00 32.41           C'
    ]
    assert atom == pdb2.calpha()
```

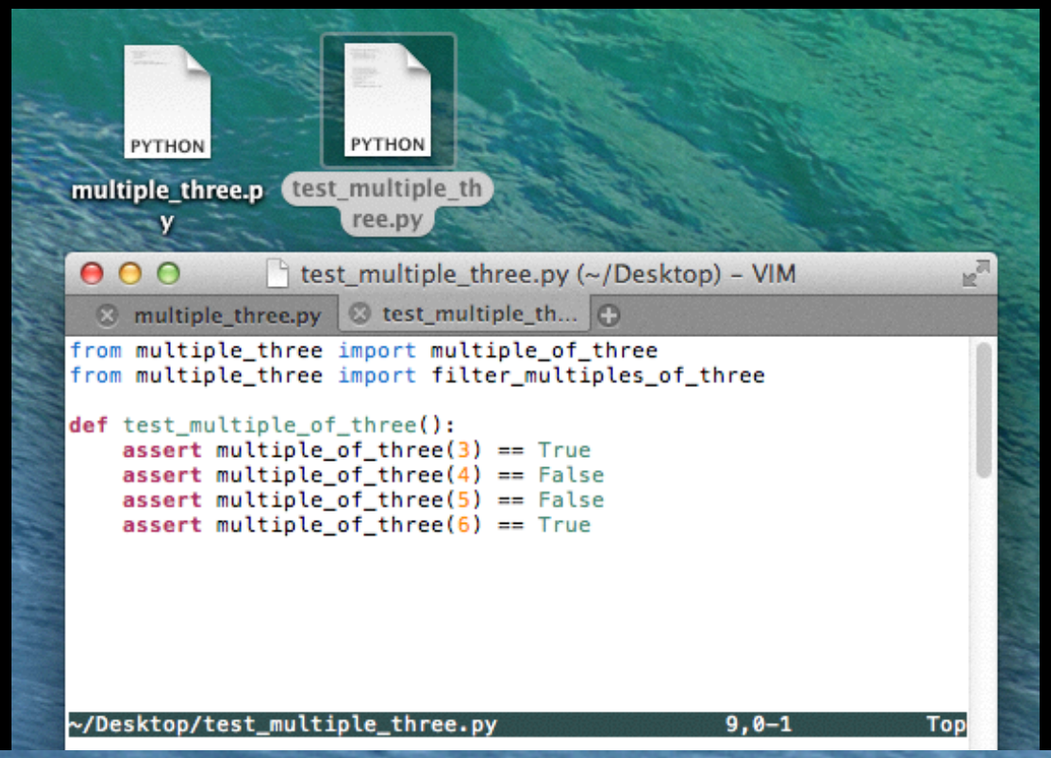
Simple functions to test

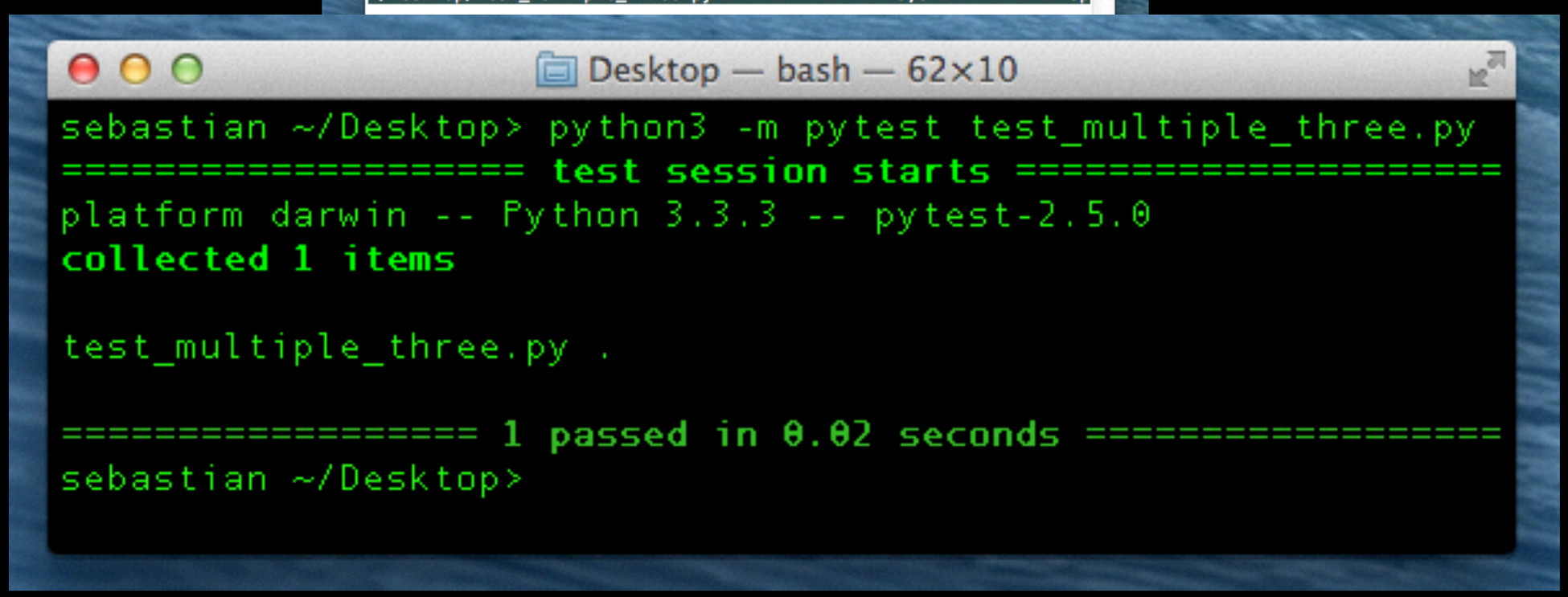


Creating a unit test file



Running the unit test





Adding edge cases

test_multiple_three.py (~/Desktop) – VIM

multiple_three.py | test_multiple_th...

```
from multiple_three import multiple_of_three
from multiple_three import filter_multiples_of_three

def test_multiple_of_three():
    assert multiple_of_three(3) == True
    assert multiple_of_three(4) == False
    assert multiple_of_three(5) == False
    assert multiple_of_three(6) == True
    assert multiple_of_three(0) == False
```

~/Desktop/test_multiple_three.py 10,0-1 Top

"~/Desktop/test_multiple_three.py" 30L, 810C written

test_multiple_three.py (~/Desktop) – VIM

multiple_three.py | test_multiple_th...

```python
from multiple_three import multiple_of_three
from multiple_three import filter_multiples_of_three

def test_multiple_of_three():
    assert multiple_of_three(3) == True
    assert multiple_of_three(4) == False
    assert multiple_of_three(5) == False
    assert multiple_of_three(6) == True
    assert multiple_of_three(0) == False
```

Desktop — bash — 63×22

```
sebastian ~/Desktop> python3 -m pytest test_multiple_three.py
====================== test session starts ======================
platform darwin -- Python 3.3.3 -- pytest-2.5.0
collected 1 items

test_multiple_three.py F

========================== FAILURES ==========================
_________________ test_multiple_of_three _________________

    def test_multiple_of_three():
        assert multiple_of_three(3) == True
        assert multiple_of_three(4) == False
        assert multiple_of_three(5) == False
        assert multiple_of_three(6) == True
>       assert multiple_of_three(0) == False
E       assert True == False
E        +  where True = multiple_of_three(0)

test_multiple_three.py:9: AssertionError
================== 1 failed in 0.04 seconds ==================
sebastian ~/Desktop>
```

Fixing the code

multiple_three.py (~/Desktop) – VIM

multiple_three.py | test_multiple_th...

```
def multiple_of_three(num):
    """Return True if 'num' is a multiple of 3."""
    is_fact = False
    if num != 0 and num % 3 == 0:
        is_fact = True
    return is_fact


def filter_multiples_of_three(numbers):
    """Return all multiples of three in 'numbers' as a list."""
    return [n for n in numbers if multiple_of_three(n)]

~
~
~
~
~
multiple_three.py                    12,0-1        All
"multiple_three.py" 12L, 338C written
```

More edge cases

test_multiple_three.py (~/Desktop) – VIM

multiple_three.py | test_multiple_th...

```
from multiple_three import multiple_of_three
from multiple_three import filter_multiples_of_three

def test_multiple_of_three():
    assert multiple_of_three(3) == True
    assert multiple_of_three(4) == False
    assert multiple_of_three(5) == False
    assert multiple_of_three(6) == True
    assert multiple_of_three(0) == False
    assert multiple_of_three(3.0) == True
    assert multiple_of_three(6.6) == False
    assert multiple_of_three(3.00000001) == False
    assert multiple_of_three(6*1000) == True
    assert multiple_of_three(-9) == False
~
~
~/Desktop/test_multiple_three.py                15,0-1          All
```

test_multiple_three.py (~/Desktop) – VIM

multiple_three.py | test_multiple_th...

```python
from multiple_three import multiple_of_three
from multiple_three import filter_multiples_of_three

def test_multiple_of_three():
    assert multiple_of_three(3) == True
    assert multiple_of_three(4) == False
    assert multiple_of_three(5) == False
    assert multiple_of_three(6) == True
    assert multiple_of_three(0) == False
    assert multiple_of_three(3.0) == True
    assert multiple_of_three(6.6) == False
    assert multiple_of_three(3.00000001) == False
    assert multiple_of_three(6*1000) == True
    assert multiple_of_three(-9) == False
```

~/Desktop/test_multiple_three.py 15,0-

Desktop — bash — 63×27

```
sebastian ~/Desktop> python3 -m pytest test_multiple_three.py
==================== test session starts ====================
platform darwin -- Python 3.3.3 -- pytest-2.5.0
collected 1 items

test_multiple_three.py F

========================= FAILURES =========================
_________________ test_multiple_of_three __________________

    def test_multiple_of_three():
        assert multiple_of_three(3) == True
        assert multiple_of_three(4) == False
        assert multiple_of_three(5) == False
        assert multiple_of_three(6) == True
        assert multiple_of_three(0) == False
        assert multiple_of_three(3.0) == True
        assert multiple_of_three(6.6) == False
        assert multiple_of_three(3.00000001) == False
        assert multiple_of_three(6*1000) == True
>       assert multiple_of_three(-9) == False
E       assert True == False
E        +  where True = multiple_of_three(-9)

test_multiple_three.py:14: AssertionError
================= 1 failed in 0.02 seconds =================
sebastian ~/Desktop>
```

Fixing the code (again)

multiple_three.py (~/Desktop) – VIM

multiple_three.py | test_multiple_th...

```
def multiple_of_three(num):
    """Return True if 'num' is a multiple of 3."""
    is_fact = False
    if num != 0 and num % 3 == 0:
        is_fact = True
    return is_fact


def filter_multiples_of_three(numbers):
    """Return all multiples of three in 'numbers' as a list."""
    return [n for n in numbers if multiple_of_three(n)]

~
~
~
~
~
multiple_three.py                                  12,0-1          All
"multiple_three.py" 12L, 338C written
```

Fixing the code (again)

multiple_three.py (~/Desktop) – VIM

multiple_three.py | test_multiple_th...

```
def multiple_of_three(num):
    """Return True if 'num' is a multiple of 3."""
    is_fact = False
    if num > 0 and num % 3 == 0:
        is_fact = True
    return is_fact


def filter_multiples_of_three(numbers):
    """Return all multiples of three in 'numbers' as a list."""
    return [n for n in numbers if multiple_of_three(n)]
~
~
multiple_three.py                                   12,0-1        All
"multiple_three.py" 12L, 337C written
```

Testing the next function

test_multiple_three.py (~/Desktop) – VIM

multiple_three.py | test_multiple_th...

```
from multiple_three import multiple_of_three
from multiple_three import filter_multiples_of_three

def test_multiple_of_three():
    assert multiple_of_three(3) == True
    assert multiple_of_three(4) == False
    assert multiple_of_three(5) == False
    assert multiple_of_three(6) == True
    assert multiple_of_three(0) == False
    assert multiple_of_three(3.0) == True
    assert multiple_of_three(6.6) == False
    assert multiple_of_three(3.00000001) == False
    assert multiple_of_three(6*1000) == True
    assert multiple_of_three(-9) == False

def test_filter_multiples_of_three():
    in1 = [1,2,3,4,5,6,7,8,9]
    out1 = [3,6,9]
    in2 = [-2,-1,33,99]
    out2 = [33,99]
    assert (filter_multiples_of_three(in1)) == out1
    assert (filter_multiples_of_three(in2)) == out2

~
~
~/Desktop/test_multiple_three.py                    24,0-1          All
"~/Desktop/test_multiple_three.py" 24L, 791C written
```

Unit tests okay (for now)

Desktop — bash — 63×11

```
sebastian ~/Desktop> python3 -m pytest test_multiple_three.py
====================== test session starts ======================
platform darwin -- Python 3.3.3 -- pytest-2.5.0
collected 2 items

test_multiple_three.py ..

================== 2 passed in 0.02 seconds ==================
sebastian ~/Desktop>
```

More Resources

- Titus' tutorial: An Extended Introduction to the nose Unit Testing Framework
  http://ivory.idyll.org/articles/nose-intro.html

- Software Carpentry Video Tutorial: Unit Testing (nose)
  http://software-carpentry.org/v4/test/unit.html

- Jeff Knupp's Tutorial (unittest)
  http://www.jeffknupp.com/blog/2013/12/09/improve-your-python-understanding-unit-testing/