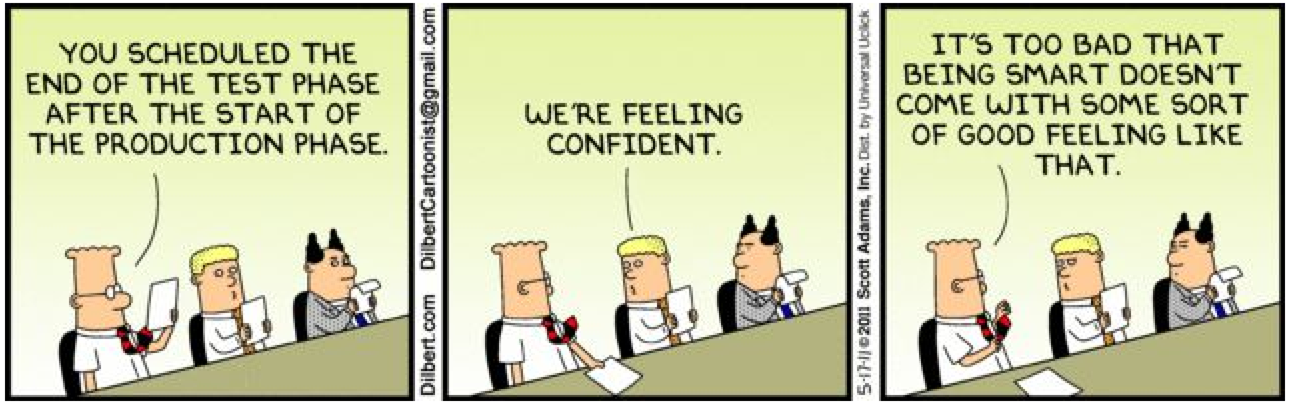
YOU SCHEDULED THE END OF THE TEST PHASE AFTER THE START OF THE PRODUCTION PHASE.
WE'RE FEELING CONFIDENT.
IT'S TOO BAD THAT BEING SMART DOESN'T COME WITH SOME SORT OF GOOD FEELING LIKE THAT.
Dilbert.com DilbertCartoonist@gmail.com
5-17-11 © 2011 Scott Adams, Inc. Dist. by Universal Uclick

Write Better Python

Unit Tests

Joanna Power
March 2016

What are they?

Programmatic verification of your code, bit by bit.

What aren't they?

Unit tests verify neither end-to-end behavior nor behavior in production.

Why should I write them?

Let me count the ways.

How do I write them?

```python
import unittest

class MyTests(unittest.TestCase):

  def test_sanity__true(self):
    self.assertTrue(True)

  def test_sanity__none(self):
    self.assertIsNone(None)


if __name__ == '__main__':
  unittest.main()
```

Seriously?

```python
class Person(object):

  def __init__(self, name=None):
    if not self.check_name(name):
      raise ValueError(
        'Invalid name')
    self.name = name

  def check_name(self, name):
    return (
      name
      and name != 'Hulk Hogan'
      and len(name.split()) > 1)
```

Are you mocking me?

```
import unittest
import unittest.mock

class PersonTests(unittest.TestCase):

  @mock.patch.object(Person, 'check_name')
  def test_init(self, mock_check_name):
    '''Verify result and calls made by __init__
    '''
    # Set up mocks and test data
    mock_check_name.return_value = True
    mock_name = mock.Mock(name='mock_name')

    # Make call
    person = Person(name=mock_name)

    # Verify result
    self.assertEqual(mock_name, person.name)

    # Verify mocks
    mock_check_name.assert_called_once_with(
      mock_name)
```

Recap

- A unit test verifies a small bit of functionality.
- Unit tests lead to better code.
- Python makes writing and running unit tests easy.
- Unit tests without mocking are not unit tests.

Recommended resources

- https://cgoldberg.github.io/python-unittest-tutorial/
- https://docs.python.org/3/library/unittest.mock-examples.html