Unit Testing with Python

Emily Bache
http://coding-is-like-cooking.info
emily@bacheconsulting.com

pluralsight
hardcore developer training

Course Summary

- Unit testing vocabulary & basic example using **`unittest`**
- Unit testing - Why and When
- An alternative to unittest: **`pytest`**
- Testable documentation using **`doctest`**
- Test Doubles
- Assessing Test Coverage
- Maintainable Unit Tests

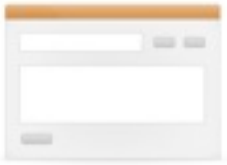
"Goat in Tree" - Cuno de Boer (flickr)

Module Summary

- Unit Testing vocabulary & "Phonebook" example
- Test Case design

"Goat in Tree" - Cuno de Boer (flickr)

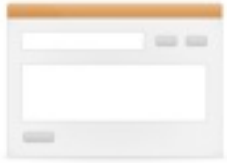
Review of Fundamentals



System Under Test

- a Unit Test checks the behaviour of an **element of code**
  - a method or function
  - a module or class
- an automated test
  - is designed by a human
  - runs without intervention
  - reports results unambiguously as “pass” or “fail”

Strictly speaking



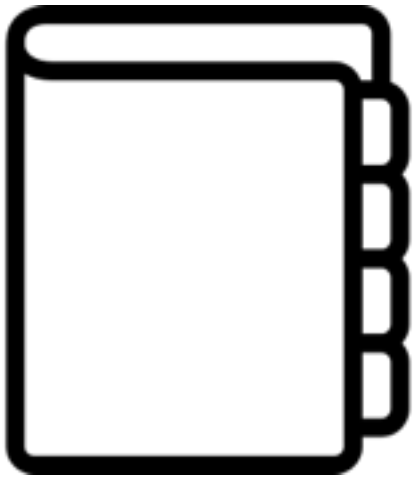
System Under Test

It's not a unit test if it uses...

- the file system
- a database
- the network

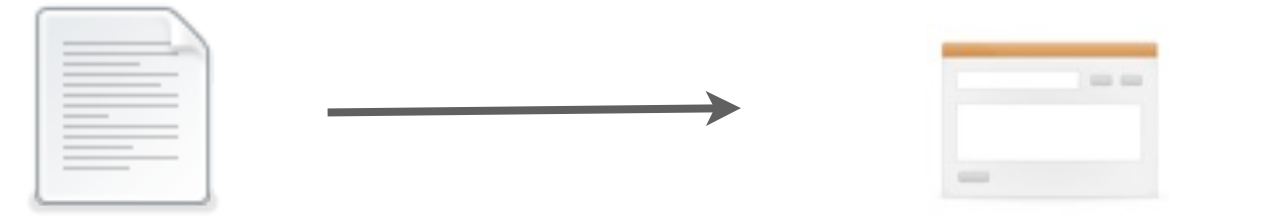
(but it might still be a useful test)

Exercise - Phone Numbers

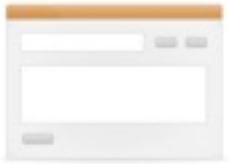


- Given a list of names and phone numbers, make a Phonebook that allows you to look up numbers by name.
- Determine if a given Phonebook is consistent.
  - In a consistent phone list no number is a prefix of another.
  - For Example:
    - Bob 91125426
    - Alice 97 625 992
    - Emergency 911
  - Bob and Emergency are inconsistent

Unit test vocabulary: TestCase



Unit Test Vocabulary



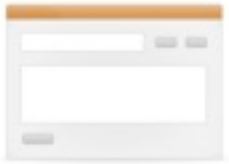
Test Case

System Under Test

```
.
----------------------
Ran 1 test in 0.001s

OK
```

Test Runner

Unit Test Vocabulary

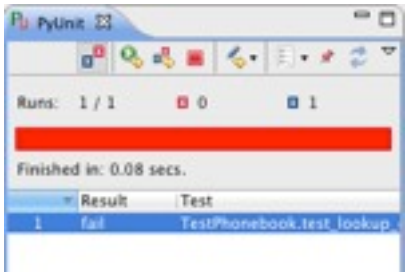


Test Case

System Under Test

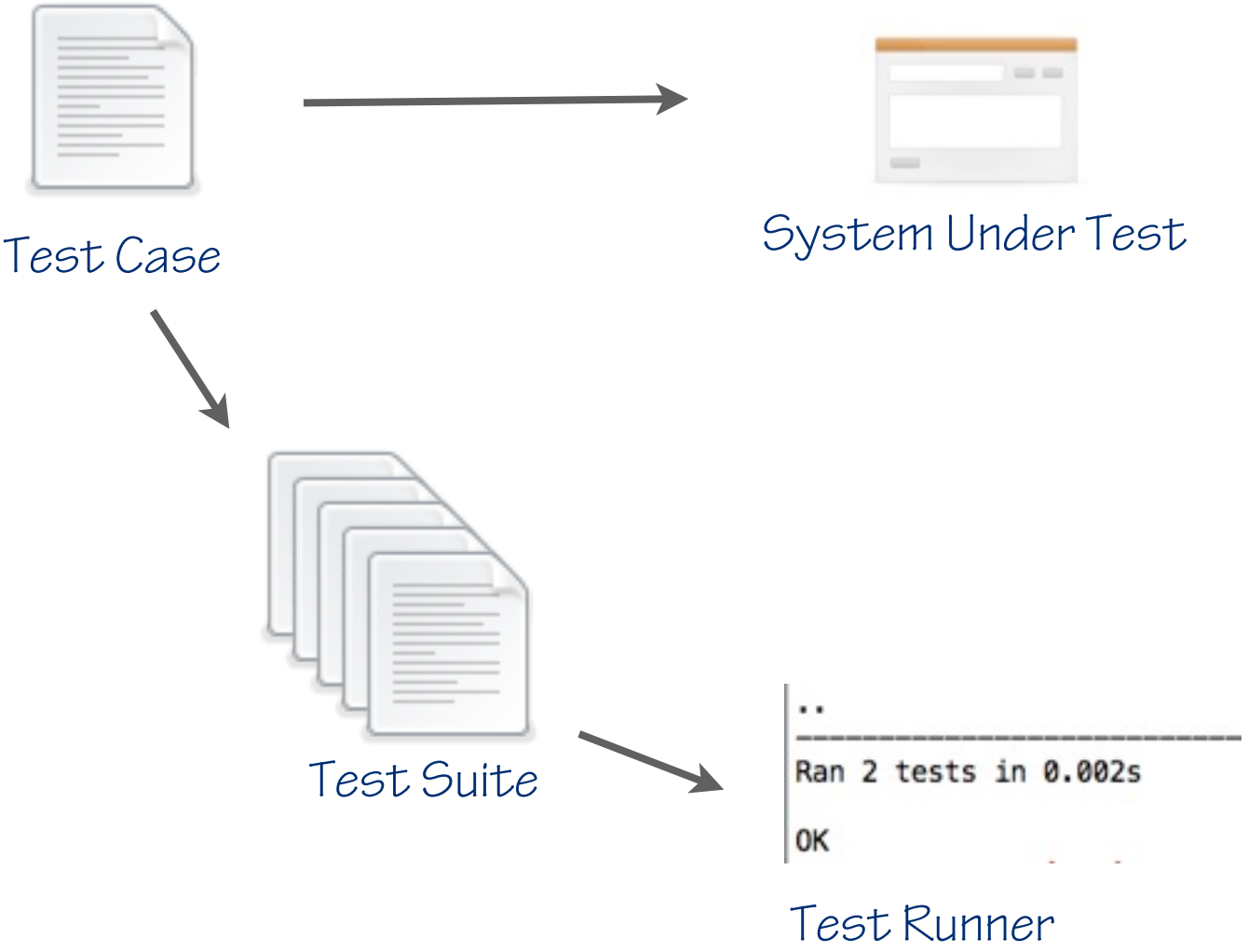
```
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
```

or



Test Runner

Unit Test Vocabulary

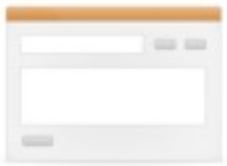


Unit Test Vocabulary

Test Case

```
def setUp(self):
    pass

def tearDown(self):
    pass
```

Test Fixture



System Under Test

Test Suite

```
..
----------------------------------------
Ran 2 tests in 0.002s

OK
```

Test Runner

Test Fixture

```
def setUp(self):
    self.phonebook = Phonebook()


def test_lookup_entry_by_name(self):
    self.phonebook.add("Bob", "12345")
    self.assertEqual("12345", self.phonebook.lookup("Bob"))


def tearDown(self):
    pass
```

Test Fixture - Test Fails

```
def setUp(self):
    self.phonebook = Phonebook()


def test_lookup_entry_by_name(self):
    self.phonebook.add("Bob", "12345")
    self.assertEqual("123456", self.phonebook.lookup("Bob"))


def tearDown(self):
    pass
```

Test Fixture - Setup Fails

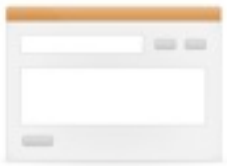
```
def setUp(self):
    self.phonebook = Phonebook("extraneous argument")

def test_lookup_entry_by_name(self):
    self.phonebook.add("Bob", "12345")
    self.assertEqual("123456", self.phonebook.lookup("Bob"))

def tearDown(self):
    pass
```

Unit Test Vocabulary

Test Case

```
def setUp(self):
    pass

def tearDown(self):
    pass
```

Test Fixture



System Under Test

Test Suite

```
..
----------------------------------------
Ran 2 tests in 0.002s

OK
```

Test Runner

Test Case Design

Test Case

- Test Case Name
  - Arrange
  - Act
  - Assert

The Three Parts of a Test

Test Case

Arrange: Set up the object to be tested & collaborators.

Act: Exercise functionality on the object.

Assert: Make claims about the object & its collaborators

The Three Parts of a Test

Four

Test Case

Arrange: Set up the object to be tested & collaborators.

Act: Exercise functionality on the object.

Assert: Make claims about the object & its collaborators

Cleanup: Release resources, restore to original state

Method	Checks that	New in
`assertEqual(a, b)`	`a == b`	
`assertNotEqual(a, b)`	`a != b`	
`assertTrue(x)`	`bool(x) is True`	
`assertFalse(x)`	`bool(x) is False`	
`assertIs(a, b)`	`a is b`	3.1
`assertIsNot(a, b)`	`a is not b`	3.1
`assertIsNone(x)`	`x is None`	3.1
`assertIsNotNone(x)`	`x is not None`	3.1
`assertIn(a, b)`	`a in b`	3.1
`assertNotIn(a, b)`	`a not in b`	3.1
`assertIsInstance(a, b)`	`isinstance(a, b)`	3.2
`assertNotIsInstance(a, b)`	`not isinstance(a, b)`	3.2

Module Review

- Unit Testing vocabulary
  - Test Case
  - Test Runner
  - Test Suite
  - Test Fixture
- Test Case design
  - Test name
  - Arrange - Act - Assert

"Goat in Tree" - Cuno de Boer (flickr)