Unit Testing with the pytest module

Emily Bache
TECHNICAL AGILE COACH
@emilybache coding-is-like-cooking.info

Summary

Defining test cases with `pytest`

Interpreting test failures

Test Fixtures

Organising test code

unittest is based on JUnit

JUnit was created in 1997

unittest first came out in 2001

```
class PhoneBookTest(unittest.TestCase):

    def setUp(self):
        self.phonebook = PhoneBook()

    def tearDown(self):
        pass

    def test_lookup_by_name(self):
        self.phonebook.add("Bob", "12345")
        number = self.phonebook.lookup("Bob")
        self.assertEqual("12345", number)

    def test_missing_name(self):
        with self.assertRaises(KeyError):
            self.phonebook.lookup("missing")
```

- Inherit a TestCase class
- fixtures - overridden methods
- test case
- second test case

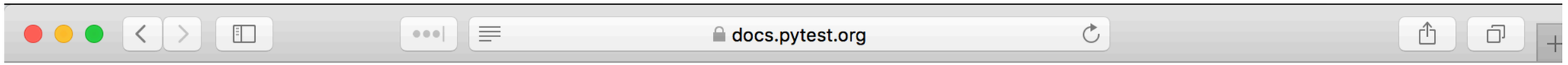
http://pytest.org

`pytest` **is a popular alternative to** `unittest`

It is not a member of the xUnit family

It's not in the standard Python distribution

Installing pytest



pytest

Table Of Contents

Installation and Getting Started

Pythons: Python 2.7, 3.4, 3.5, 3.6, 3.7, Jython, PyPy-2.3

Platforms: Unix/Posix and Windows

PyPI package name: pytest

Documentation as PDF: download latest

`pytest` is a framework that makes building simple and scalable tests easy. Tests are expressive and readable—no boilerplate code required. Get started in minutes with a small unit test or complex functional test for your application or library.

Install `pytest`

1. Run the following command in your command line:

```
pip install -U pytest
```

Phonebook Demo

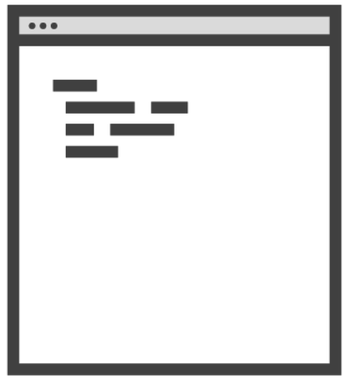
Given a list of names and phone numbers

Make a Phonebook

Determine if it is consistent:

- no number is a prefix of another
- e.g. Bob 91125426, Anna 97625992
- Emergency 911
- Bob and Emergency are inconsistent

Pytest Failure Analysis



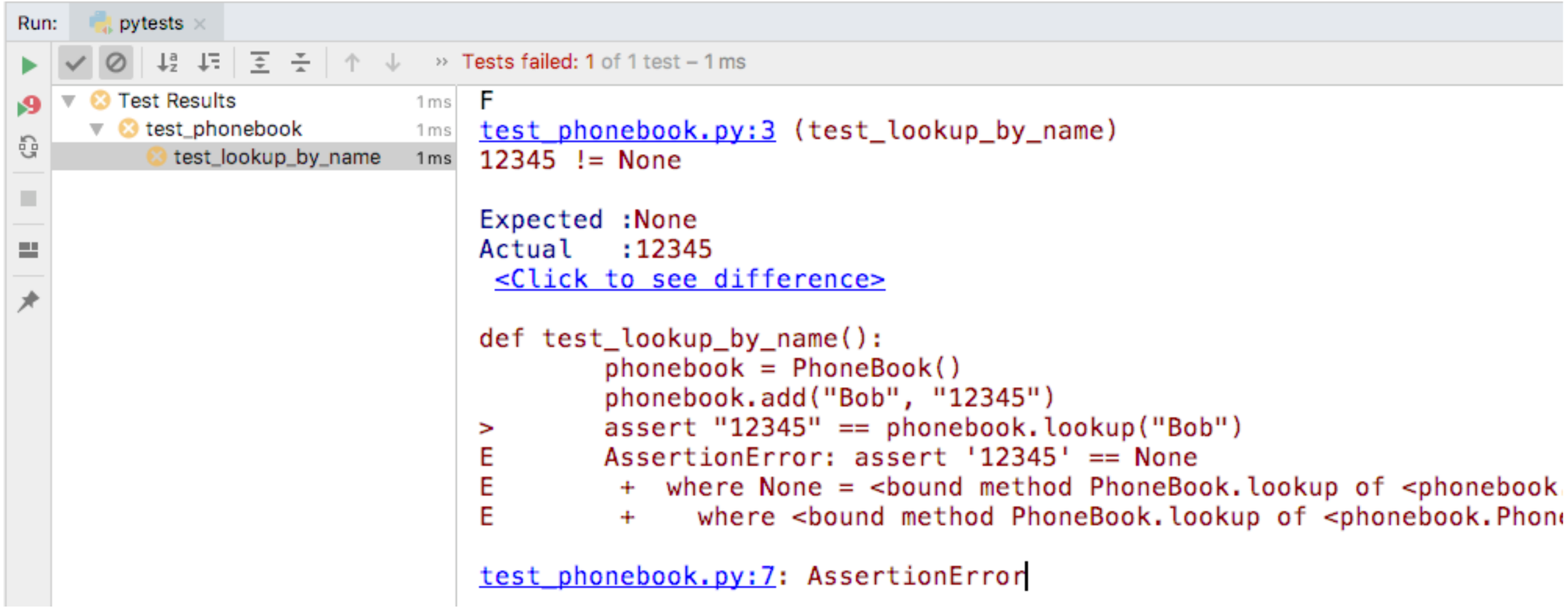
Test Case

Unit Under Test

```
[~/PycharmProjects/phonenumbers [1] > py.test
============================= test session starts ==============================
platform darwin -- Python 3.7.2, pytest-4.1.1, py-1.7.0, pluggy-0.8.1
rootdir: /Users/emily/PycharmProjects/phonenumbers, inifile:
collected 1 item

test_phonebook.py F                                                      [100%]

=================================== FAILURES ===================================
_____________________________ test_lookup_by_name _____________________________

    def test_lookup_by_name():
        phonebook = PhoneBook()
        phonebook.add("Bob", "12345")
>       assert "12345" == phonebook.lookup("Bob")
E       AssertionError: assert '12345' == None
E        +  where None = <bound method PhoneBook.lookup of <phonebook.PhoneBook o
bject at 0x1053377b8>>('Bob')
E        +    where <bound method PhoneBook.lookup of <phonebook.PhoneBook object
 at 0x1053377b8>> = <phonebook.PhoneBook object at 0x1053377b8>.lookup

test_phonebook.py:7: AssertionError
========================== 1 failed in 0.08 seconds ============================
```

Test Runner Output

PyCharm Test Runner



Test Runner in PyCharm

```
def test_lookup_by_name():
    phonebook = PhoneBook()
    phonebook.add("Bob", "12345")
    assert "12345" == phonebook.lookup("Bob")
```

◀ test name

◀ assert equal

Unit Test Vocabulary

Test Case

```
def setUp(self):
    pass

def tearDown(self):
    pass
```

Test Fixture

Unit Under Test

Test Suite

```
.......
--------------------------------
Ran 7 tests in 0.000s

OK
```

Test Runner

```
class PhoneBookTest(unittest.TestCase):

    def setUp(self):
        self.phonebook = PhoneBook()

    def tearDown(self):
        pass

    def test_lookup_by_name(self):
        self.phonebook.add("Bob", "12345")
        number = self.phonebook.lookup("Bob")
        self.assertEqual("12345", number)

    def test_missing_name(self):
        with self.assertRaises(KeyError):
            self.phonebook.lookup("missing")
```

◀ set up fixture

◀ tear down fixture

◀ test case

◀ second test case

Pytest Fixtures

Test Case


```
@pytest.fixture
def resource():
    return Resource()
```

Test Fixture

```
.......
----------------------------
Ran 7 tests in 0.000s

OK
```

Test Runner

```python
@pytest.fixture
def phonebook():
    return PhoneBook()


def test_lookup_by_name(phonebook):
    phonebook.add("Bob", "12345")
    assert "12345" == phonebook.lookup(“Bob")


def test_missing_name_raises_error(phonebook):
    with pytest.raises(KeyError):
        phonebook.lookup("Bob")
```

◀ test fixture

◀ test case using fixture

◀ second test case using fixture

Unit Test Vocabulary

Test Case

```
def setUp(self):
    pass

def tearDown(self):
    pass
```

Test Fixture

Unit Under Test

Test Suite

```
.......
--------------------------------
Ran 7 tests in 0.000s

OK
```

Test Runner

```
@pytest.fixture
def phonebook(tmpdir):
    return PhoneBook(tmpdir)


def test_lookup_by_name(phonebook):
    phonebook.add("Bob", "12345")
    assert "12345" == \
phonebook.lookup(“Bob")
```

- test fixture that uses another test fixture

- test case that uses the phonebook fixture

All available fixtures

```
~/PycharmProjects/phonenumbers [1] > pytest --fixtures
============================= test session starts ==============================
platform darwin -- Python 3.7.2, pytest-4.1.1, py-1.7.0, pluggy-0.8.1
rootdir: /Users/emily/PycharmProjects/phonenumbers, inifile:
collected 3 items
cache
    Return a cache object that can persist state between testing sessions.

    cache.get(key, default)
    cache.set(key, value)

    Keys must be a ``/`` separated value, where the first part is usually the
    name of your plugin or application to avoid clashes with other cache users.

    Values can be any object handled by the json stdlib module.
capsys
    Enable capturing of writes to ``sys.stdout`` and ``sys.stderr`` and make
    captured output available via ``capsys.readouterr()`` method calls
    which return a ``(out, err)`` namedtuple.  ``out`` and ``err`` will be ``text``
    objects.
capsysbinary
    Enable capturing of writes to ``sys.stdout`` and ``sys.stderr`` and make
    captured output available via ``capsys.readouterr()`` method calls
    which return a ``(out, err)`` tuple.  ``out`` and ``err`` will be ``bytes``
    objects.
capfd
    Enable capturing of writes to file descriptors ``1`` and ``2`` and make
    captured output available via ``capfd.readouterr()`` method calls
    which return a ``(out, err)`` tuple.  ``out`` and ``err`` will be ``text``
    objects.
capfdbinary
    Enable capturing of write to file descriptors 1 and 2 and make
    captured output available via ``capfdbinary.readouterr`` method calls
    which return a ``(out, err)`` tuple.  ``out`` and ``err`` will be
    ``bytes`` objects.
```

Pytest Fixtures

Test Case


```
@pytest.fixture
def resource():
    return Resource()
```

Test Fixture

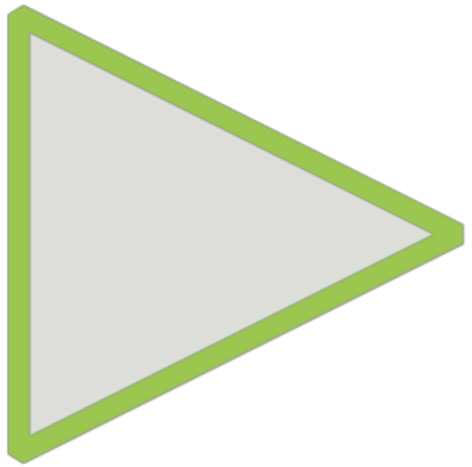
```
.......
----------------------------
Ran 7 tests in 0.000s

OK
```

Test Runner

Organising your test code

Physical structure

Where to keep test modules



Runtime structure

Controlling which tests are run

```python
@pytest.mark.slow
def test_large_file(phonebook):
    with open("test_data.txt") as f:
        csv_reader = csv.DictReader(f)
        for row in csv_reader:
            name = row["Name"]
            number = row["Phone Number"]
            phonebook.add(name, number)
    assert phonebook.is_consistent()
```

◀ test case is marked as ‘slow’

```
$> python -m pytest “not slow”
```

◀ run all tests except ‘slow’

```python
def test_lookup_by_name():
    phonebook = PhoneBook()
    phonebook.add("Bob", "12345")
    assert "12345" == phonebook.lookup(“Bob")


@pytest.mark.skip(‘WIP')
def test_phonebook_contains_names():
    phonebook = PhoneBook()
    assert 'Bob' in phonebook.names()



def test_missing_name_raises_error():
    phonebook = PhoneBook()
    with pytest.raises(KeyError):
        phonebook.lookup("Bob")
```

◀ first test case

◀ second test case (skipped)

◀ third test case

```python
def test_lookup_by_name():
    phonebook = PhoneBook()
    phonebook.add("Bob", "12345")
    assert "12345" == phonebook.lookup(“Bob")
```

◀ first test case

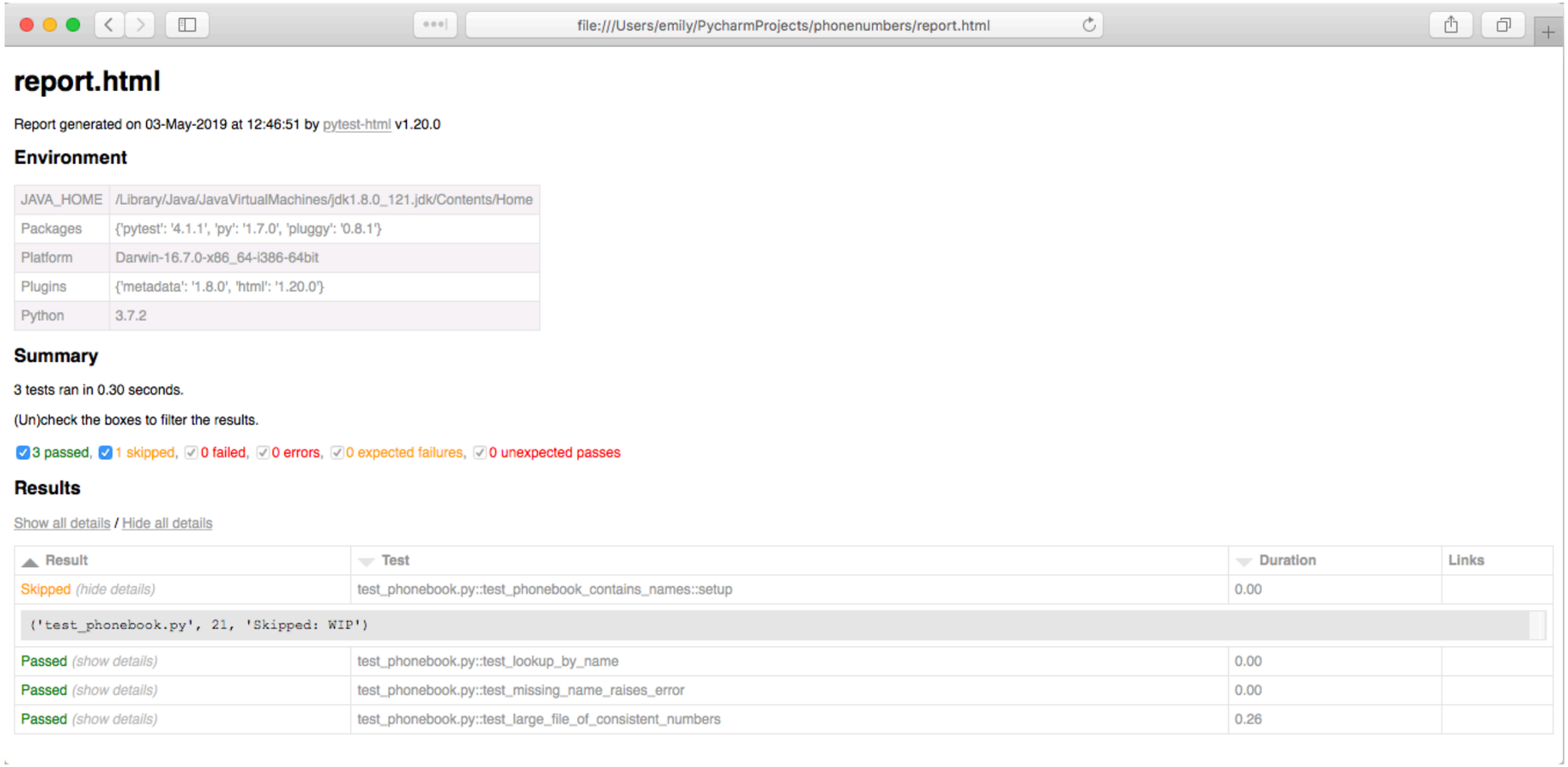
```python
@pytest.mark.skipif(sys.version_info < (3, 6),
reason="requires python3.6 or higher”)
def test_phonebook_contains_names():
    phonebook = PhoneBook()
    assert 'Bob' in phonebook.names()
```

◀ second test case (skipped if python version is too old)

```python
def test_missing_name_raises_error():
    phonebook = PhoneBook()
    with pytest.raises(KeyError):
        phonebook.lookup("Bob")
```

◀ third test case

Plugins



https://pypi.org/project/pytest-html/

Summary

Defining test cases with `pytest`

Interpreting test failures

Test Fixtures

Organising test code