Using pytest

Emily Bache

Technical Coach

@emilybache | coding-is-like-cooking.info

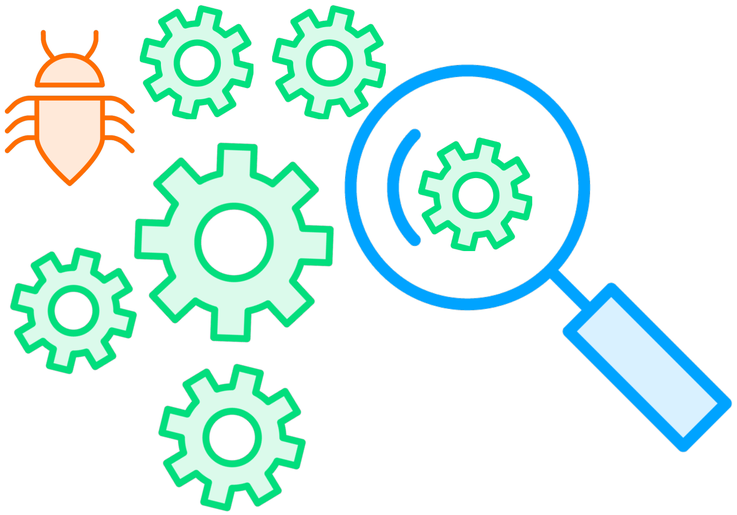
Overview

Defining test cases with `pytest`

Interpreting test failures

Avoiding duplication in test code

Organising tests in a larger project



unittest is based on JUnit

JUnit was created in 1997

unittest first came out in 2001

About pytest

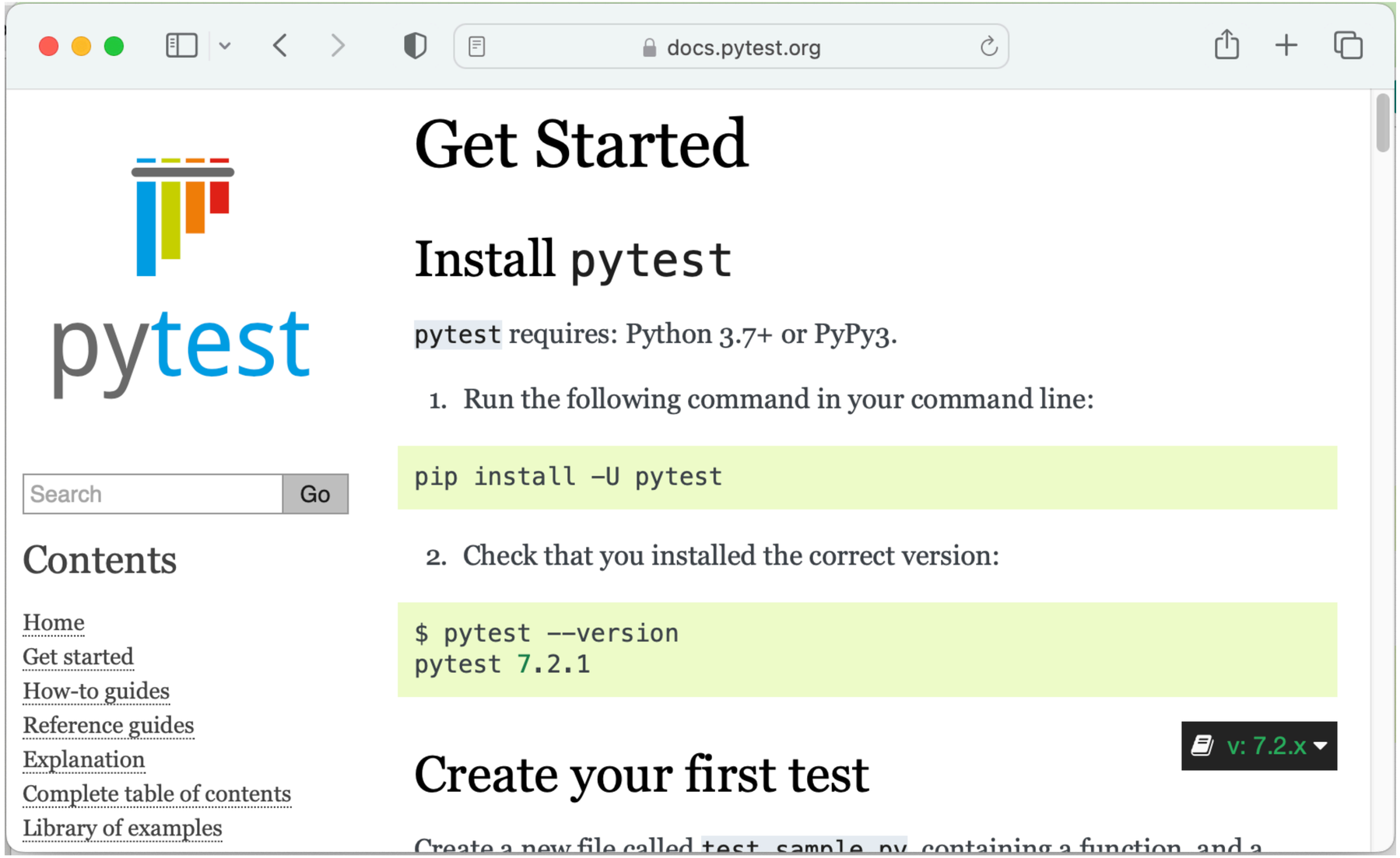
http://pytest.org

`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



Phonebook

Demo

Given a list of names and phone numbers

Make a Phonebook

Determine if it is consistent, i.e. no number is a prefix of another

Phone Numbers Exercise

Using pytest

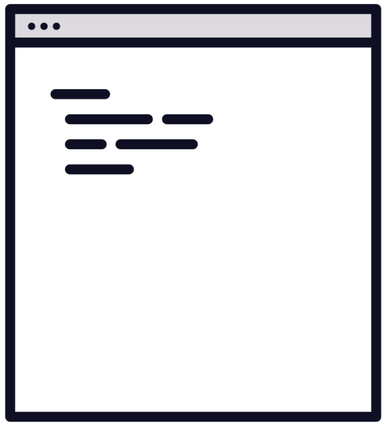
```
class PhoneBookTest(unittest.TestCase):


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

- Inherit a TestCase class
- Test case naming convention
- Inherited ‘assertEqual’ method

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

- Test case naming convention
- Built-in ‘assert’ function

Failure Analysis in pytest



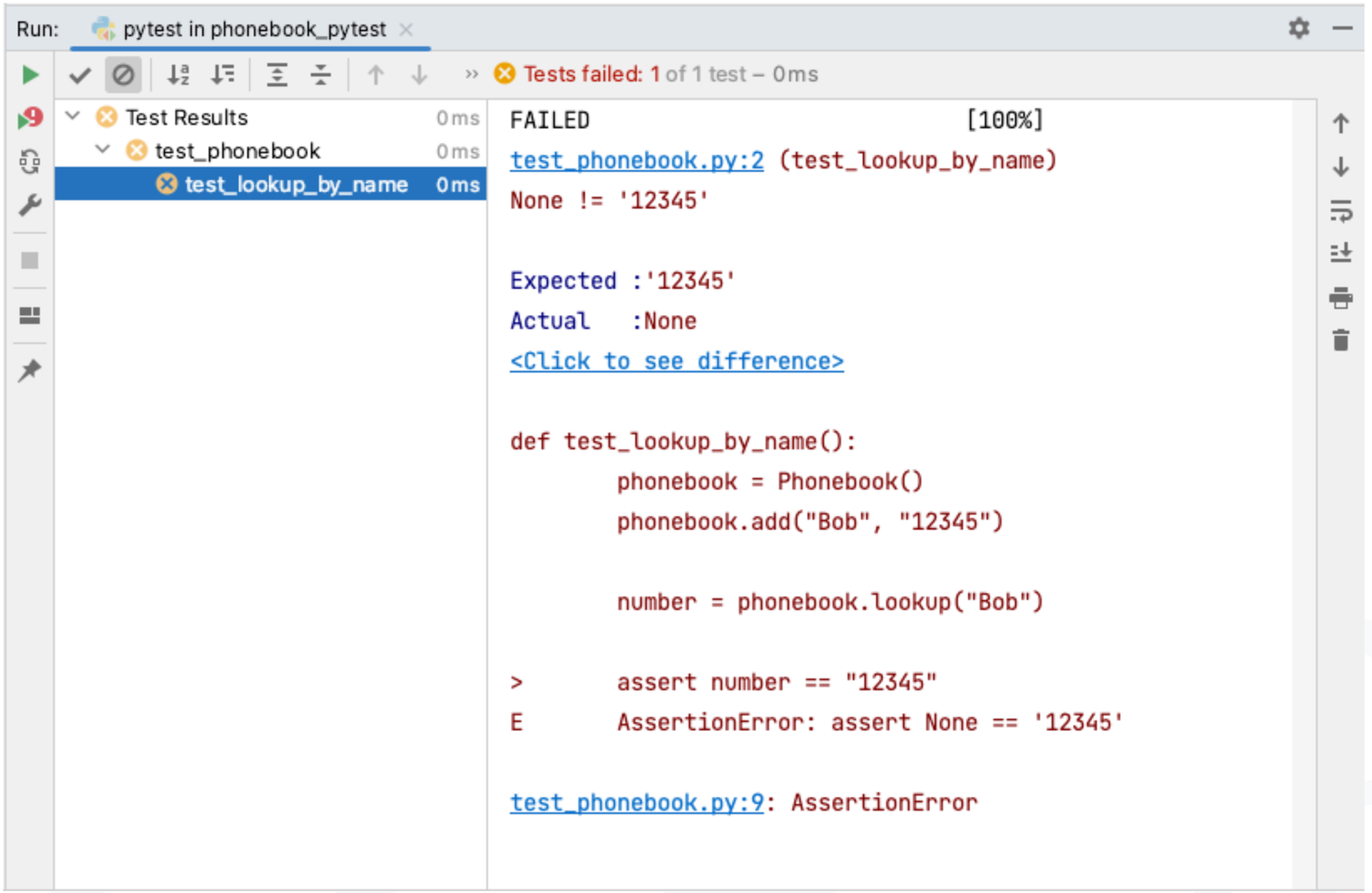
Test Case

Unit Under Test

```
(phonebook_pytest) emily:phonebook_pytest/ $ python -m pytest                    [10:57:47]
========================================= test session starts ==========================================
platform darwin -- Python 3.11.2, pytest-7.2.1, pluggy-1.0.0
rootdir: /Users/emily/workspace/training/phonebook_pytest
collected 1 item

test_phonebook.py F                                                                  [100%]

============================================== FAILURES ==============================================
_____________________________________ test_lookup_by_name _____________________________________

    def test_lookup_by_name():
        phonebook = Phonebook()
        phonebook.add("Bob", "12345")

        number = phonebook.lookup("Bob")

>       assert number == "12345"
E       AssertionError: assert None == '12345'

test_phonebook.py:9: AssertionError
======================================= short test summary info ========================================
FAILED test_phonebook.py::test_lookup_by_name - AssertionError: assert None == '12345'
========================================== 1 failed in 0.29s ===========================================
```

Test Runner Output

Test Runners



Test Runner in PyCharm

Phone Numbers Exercise

Make the test pass

Phonebook

Demo

More kinds of assertions

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

- Test name
- Assert equal

Phone Numbers Exercise

More assertions

Phonebook

Demo

Test Fixtures

Unit Test Vocabulary

Test Case

```
def setUp (self):
    pass

def tearDown (self):
    pass
```

Test Fixture

Units 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

Fixtures in pytest

Test Case

pytest

Fixtures in pytest

Test Case

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

Test Fixture

Fixtures in pytest

Test Case

pytest

```
@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

Units Under Test

Test Suite

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

OK
```

Test Runner

```python
@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

```
(phonebook_pytest) emily:phonebook_pytest/ $ python -m pytest --fixtures                    [10:57:50]
========================================= test session starts ==========================================
platform darwin -- Python 3.11.2, pytest-7.2.1, pluggy-1.0.0
rootdir: /Users/emily/workspace/training/phonebook_pytest
collected 1 item
cache -- lib/python3.11/site-packages/_pytest/cacheprovider.py:510
    Return a cache object that can persist state between testing sessions.

capsys -- lib/python3.11/site-packages/_pytest/capture.py:905
    Enable text capturing of writes to ``sys.stdout`` and ``sys.stderr``.

capsysbinary -- lib/python3.11/site-packages/_pytest/capture.py:933
    Enable bytes capturing of writes to ``sys.stdout`` and ``sys.stderr``.

capfd -- lib/python3.11/site-packages/_pytest/capture.py:961
```

```
------------------------------- fixtures defined from test_phonebook --------------------------------
phonebook -- test_phonebook.py:7
    no docstring available
```

Fixtures in pytest

Test Case

pytest

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

Test Fixture

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

OK
```

Test Runner

Phonebook

Demo

Parameterized tests

Parameterized Testing

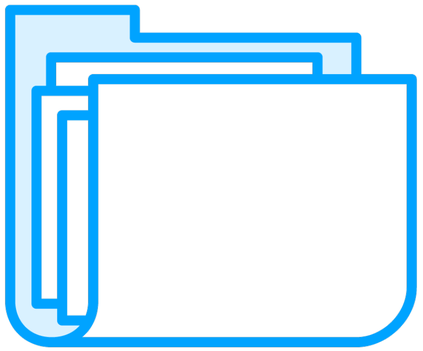
To reduce duplication in test code:

- The ‘Act’ step is the same in several tests

- Vary the data via arguments to the test

Organising Your Test Code

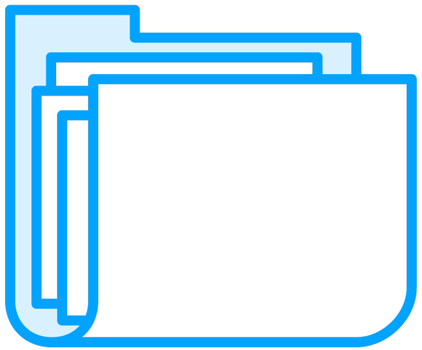
Organising Your Test Code



Physical structure

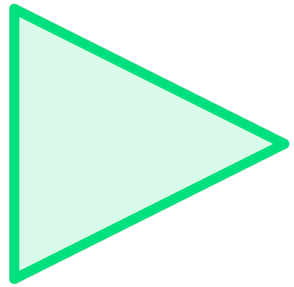
Where to keep test modules

Organising Your Test Code



Physical structure

Where to keep test modules



Runtime structure

Controlling which tests are run

Phonebook

Demo

Organising test code in a large project

```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")
```

◀ First test case

```python
@pytest.mark.skip('Work In Process')
def test_phonebook_new_feature():
    assert False
```

◀ Second test case (skipped)

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

◀ 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

Summary

Defining test cases with `pytest`

Interpreting test failures

Avoiding duplication in test code

Organising tests in a larger project