pytest

Dan Swain, Roc Python user group, 2/20/18
dan.t.swain@gmail.com github.com/dantswain @dantswain

Who the @#$Q are you?

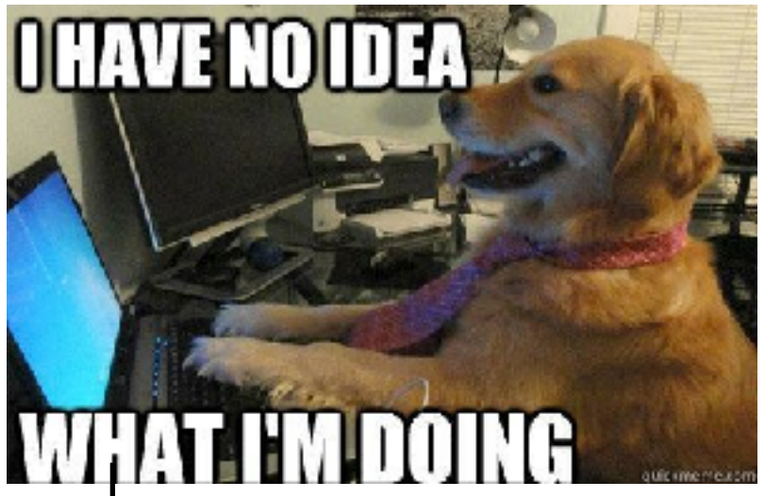
- I went to school for a while
- Fellow at simpli.fi
  - Adtech, Realtime bidding, “big data”
  - 3.5M QPS, 50 TB/day to hdfs

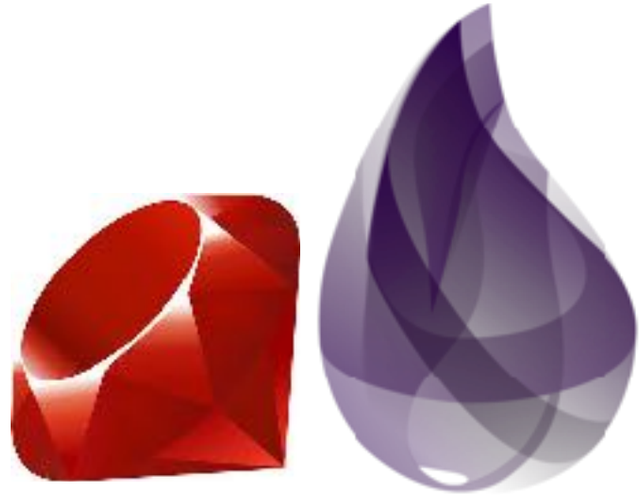
Disclaimer

penfieldrobotics.com
https://www.firstinspires.org/robotics/frc

Also Disclaimer

On a scale of...





Get on with it

Testing

unittest

- unittest is the baked-in python unit testing library

- https://docs.python.org/3/library/unittest.html

unittest

```
import unittest

class TestStringMethods(unittest.TestCase):

    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        # check that s.split fails when the separator is not a string
        with self.assertRaises(TypeError):
            s.split(2)

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

Uses inheritance

Variety of assertion methods

Manual invocation

pytest

- https://docs.pytest.org/en/latest/
- pytest is a unit testing framework for python
- https://github.com/dantswain/pytest_examples
- https://travis-ci.org/dantswain/pytest_examples

pytest

```
import pytest

def test_upper():
    assert 'foo'.upper() == 'FOO'

def test_isupper():
    assert 'FOO'.isupper()
    assert not 'Foo'.isupper()
```

No inheritance

Regular asserts

```
def test_split():
    s = 'hello world'
    assert s.split() == ['hello', 'world']
    with pytest.raises(TypeError):
        s.split(2)
```

Automatic detection, invocation via ‘pytest’ command

unittest vs pytest

```
import unittest

class TestStringMethods(unittest.TestCase):

    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        with self.assertRaises(TypeError):
            s.split(2)

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

```
import pytest

def test_upper():
    assert 'foo'.upper() == 'FOO'

def test_isupper():
    assert 'FOO'.isupper()
    assert not 'Foo'.isupper()

def test_split():
    s = 'hello world'
    assert s.split() == ['hello', 'world']
    with pytest.raises(TypeError):
        s.split(2)
```

Fixtures

- https://docs.pytest.org/en/latest/fixture.html#fixtures

- “pytest fixtures: explicit, modular, scalable”

- “Fixtures: a prime example of dependency injection”

Fixtures

```
import pytest

class Message(object):
    def __init__(self, content):
        self.content = content

    def reverse(self):
        return self.content[::-1]

    def __add__(self, other_msg):
        return Message(' '.join([self.content, other_msg.content]))

@pytest.fixture
def hello():
    return Message("Hello")

@pytest.fixture
def world():
    return Message("World")

def test_hello(hello):
    assert hello.reverse() == "olleH"

def test_hello_world(hello, world):
    assert (hello + world).content == "Hello World"
```

Fixtures

```
import os
import shutil
import pytest

@pytest.fixture(scope='module')
def output_dir():
    return 'test-output'

@pytest.fixture(scope='module', autouse=True)
def clean_output(output_dir):
    shutil.rmtree(output_dir, ignore_errors=True)
    os.mkdir(output_dir)
    yield
    shutil.rmtree(output_dir)

def test_write_file(output_dir):
    with open(os.path.join(output_dir, 'test.txt'), 'w') as f:
        f.write('Hello!')
    assert os.path.exists(os.path.join(output_dir, 'test.txt'))

def test_read_file(output_dir):
    with open(os.path.join(output_dir, 'test.txt'), 'r') as f:
        assert f.read() == 'Hello!'
```

Fixtures

- Shared fixtures automatically loaded from ‘conftest.py’
- Can be overridden at local scopes
- Built-in fixtures
  - monkeypatch
  - tmpdir
  - capsys
  - …

monkeypatch

```
'''
A stupid client for devnull-as-a-service.com
'''

import http.client, urllib.parse

class DevNull(object):
    def _do_query_post(self, params):
        conn = http.client.HTTPSConnection("devnull-as-a-service.com")
        conn.request("POST","/dev/null", params)
        return conn.getresponse()

def post(self, data):
    params = urllib.parse.urlencode(data)
    response = self._do_query_post(params)
    return response.status
```

```
from collections import namedtuple
import http.client

import pytest

from dev_null_client import DevNull

DummyResponse = namedtuple('DummyResponse', ['status'])

class DummyConnection(object):
    def __init__(self):
        self.last_method = None
        self.last_path = None
        self.last_params = None
        self.host = None

    def request(self, method, path, params):
        self.last_method = method
        self.last_path = path
        self.last_params = params

    def getresponse(self):
        return DummyResponse(status=200)

@pytest.fixture
def dummy_conn(monkeypatch):
    dummy = DummyConnection()
    def get_dummy(host):
        dummy.host = host
        return dummy
    monkeypatch.setattr(http.client, 'HTTPSConnection', get_dummy)
    return dummy

def test_post(dummy_conn):
    dev_null = DevNull()
    assert dev_null.post({'a': 1}) == 200
    assert dummy_conn.last_method == "POST"
    assert dummy_conn.last_path == "/dev/null"
    assert dummy_conn.last_params == 'a=1'
    assert dummy_conn.host == 'devnull-as-a-service.com'
```

Marks

- Can run tests by arbitrary marks - e.g., integration vs unit
- @pytest.mark.xfail
- @pytest.mark.skip
- @pytest.mark.parametrize

Marks

```
'''
Demonstrating pytest marks
'''

import pytest

@pytest.mark.skip
def test_never_runs():
    assert False

@pytest.mark.xfail
def test_would_fail():
    assert False

@pytest.mark.parametrize('x, double_x',
                         [(1, 2),
                          (-1, -2),
                          ('a', 'aa')],
                         ids=['integer', 'negative integer', 'string'])
def test_parametrized(x, double_x):
    assert x * 2 == double_x
```

Misc features

- pytest.ini
    - Configure pytest
    - Set env vars
- conftest.py
    - Define shared fixtures

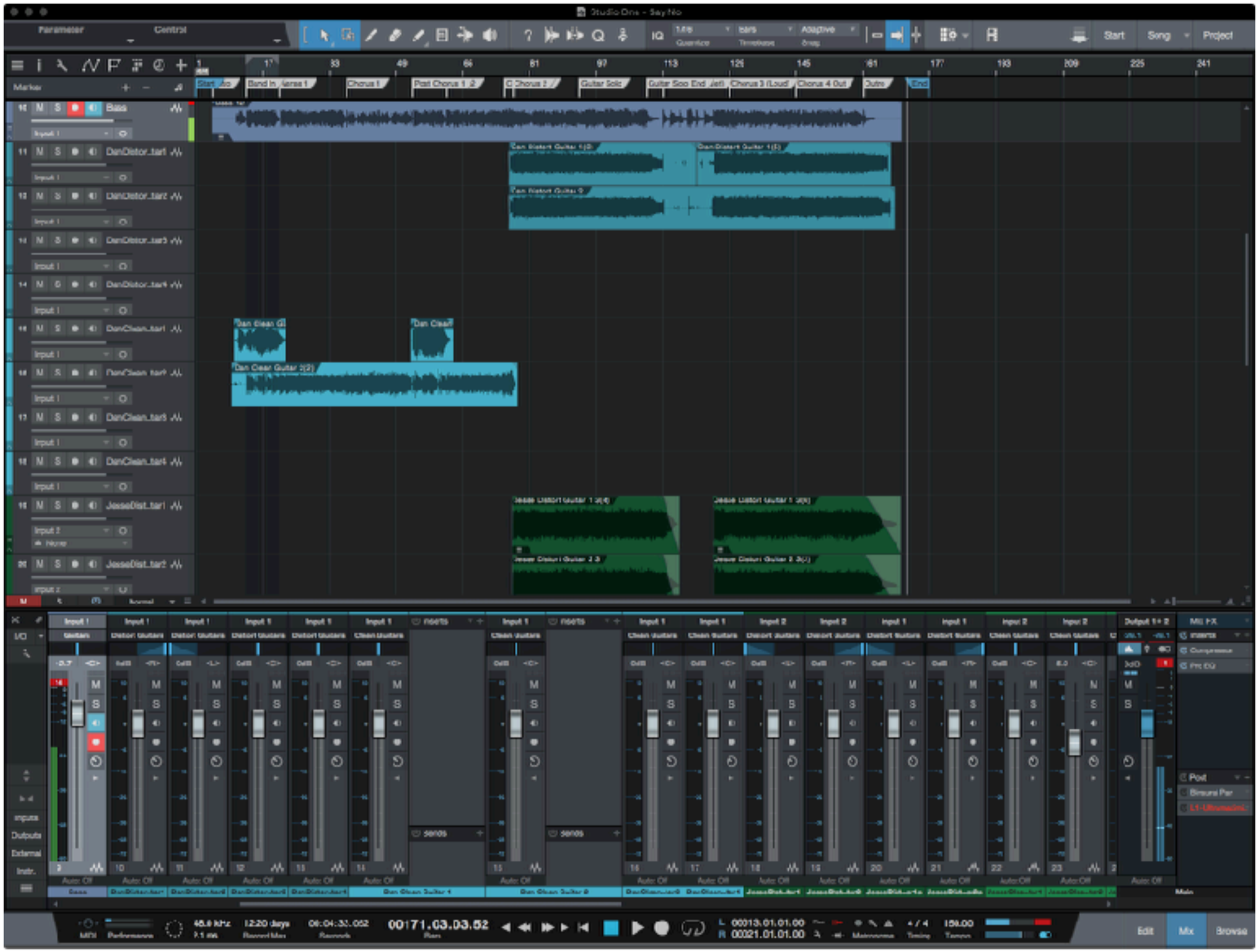
pytest with Django

- https://pytest-django.readthedocs.io/en/latest/
- Can be configured to run with manage.py or just ‘pytest’
- No database access by default (this is a good thing)
- Built-in fixtures
  - client
  - rf
  - mailoutbox
  - …

But first

My Happy Place

Recording


Parameter
Control
Start
Song
Project
Marker
Bass
DanDistor.tar1
DanDistor.tar2
DanDistor.tar3
DanDistor.tar4
DanClean.tar1
DanClean.tar3
DanClean.tar4
JesseDist.tar1
JesseDist.tar2
Dan Clean Guitar 2(2)
Edit
Mix
Browse

But…

- 9 Pedals - ~25 knobs, 5 switches
- 4 Amp Channels - 11 knobs, 6 switches
- 2 Guitars - 7 knobs, 13 switch positions
- Mic Placement - $#^!@#

Heavy Meta

- Track metadata management
- Object Model
    - Song has many tracks
    - Track has many takes
    - Take has many comments
- https://github.com/dantswain/heavy_meta
    - This is just a dummy project and doesn't really actually do anything!