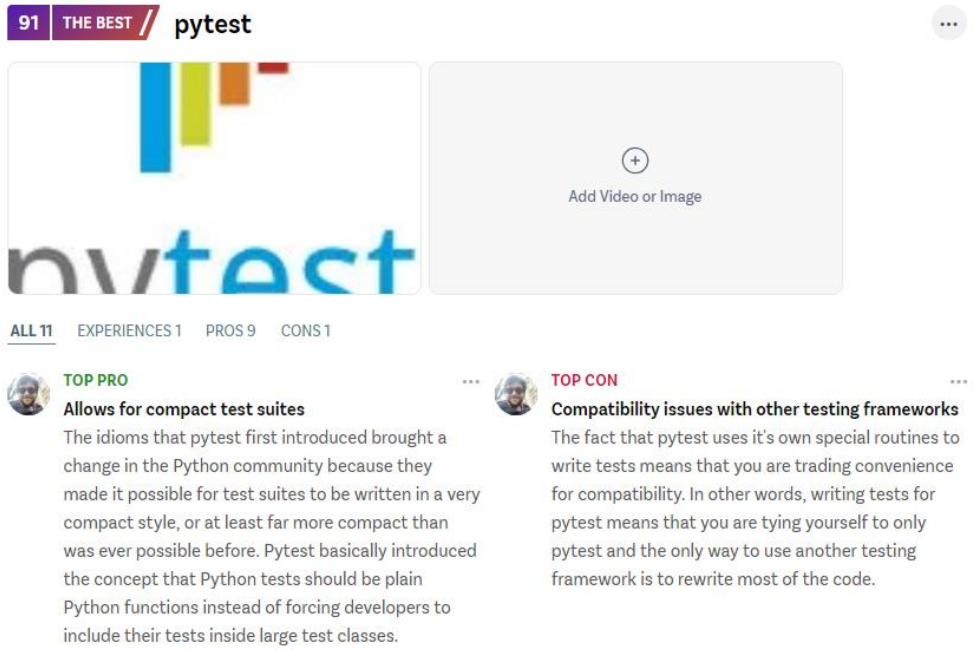
Pytest Magic

- An overview of what pytest offers
- Where to learn more.

What is Pytest?

- A python testing framework
- Compared to Unittest, it:
  - Reduces boilerplate
  - Nicer syntax
  - Provides additional features
- Can run Unittest tests
  - And Tensorflow tests (b/c subclasses Unittest)
- Is the best python testing framework
  - according to slant



Pytest Basics

Pytest Basics

fcns.py

```
import sys


def fib(n):
    if n < 0 or not
   isinstance(n, int):
        raise ValueError
    elif n < 2:
        return n
    return fib(n-1) + \
           fib(n-2)


def fib_argv():
    n=int(sys.argv[1])
    return fib(n)
```

test_fcns.py

```
import sys
from fcns import \
    fib, fib_argv

# It's important that
# this file's and
# function's names
# both start with "test_"
# This is how pytest
# knows it's a test.
def test_fib_base():
    assert fib(0) == 0
    assert fib(1) == 1
def test_fib_ind():
    assert fib(3) == 2
    assert fib(6) == 9
def test_fib_raise():
    fib(-1)
    fib(3.6)

def test_fib_argv():
    sa = sys.argv
    sys.argv = ["", "6"]
    assert fib_argv()==13
    sys.argv = sa
```

Pytest Basics - *note the assert introspection!*

fcns.py

```python
import sys


def fib(n):
    if n < 0 or not
    isinstance(n, int):
        raise ValueError
    elif n < 2:
        return n
    return fib(n-1) + \
           fib(n-2)


def fib_argv():
    n=int(sys.argv[1])
    return fib(n)
```

test_fcns.py

```python
import sys
from fcns import \
    fib, fib_argv

# It's important that
# this file's and
# function's names
# both start with "test_"
# This is how pytest
# knows it's a test.
def test_fib_base():
    assert fib(0) == 0
    assert fib(1) == 1
def test_fib_ind():
    assert fib(3) == 2
    assert fib(6) == 9
def test_fib_raise():
    fib(-1)
    fib(3.6)

def test_fib_argv():
    sa = sys.argv
    sys.argv = ["", "6"]
    assert fib_argv()==13
    sys.argv = sa
```

```
>pytest -v
============================= test session starts
 ============================
platform win32 -- Python 3.6.4, pytest-3.3.2, py-
1.5.2, pluggy-0.6.0 -- C:\Users\Acer\Anaconda3\py
thon.exe
cachedir: .cache
hypothesis profile 'default' -> database=Director
yBasedExampleDatabase('C:\\Users\\Acer\\Desktop\\
git_repos\\learning\\pytest_presentation\\Basics\
\.hypothesis\\examples')
rootdir: C:\Users\Acer\Desktop\git_repos\learning
\pytest_presentation\Basics, inifile:
plugins: hypothesis-3.82.6
collecting 0 items
collecting 4 items
collected 4 items


test_fcns.py::test_fib_base PASSED
                          [ 25%]
test_fcns.py::test_fib_ind FAILED
                          [ 50%]
test_fcns.py::test_fib_raise FAILED
                          [ 75%]
test_fcns.py::test_fib_argv FAILED
                          [100%]

================================= FAILURES =====
==============================
_____________________________ test_fib_ind ___
_____________________________

    def test_fib_ind():
        assert fib(3) == 2
>       assert fib(6) == 9
E       assert 8 == 9
E        +  where 8 = fib(6)

test_fcns.py:16: AssertionError
```

Pytest Basics - *note the assert introspection!*

fcns.py

```python
import sys


def fib(n):
    if n < 0 or not
    isinstance(n, int):
        raise ValueError
    elif n < 2:
        return n
    return fib(n-1) + \
           fib(n-2)


def fib_argv():
    n=int(sys.argv[1])
    return fib(n)
```

test_fcns.py

```python
import sys
from fcns import \
    fib, fib_argv

# It's important that
# this file's and
# function's names
# both start with "test_"
# This is how pytest
# knows it's a test.
def test_fib_base():
    assert fib(0) == 0
    assert fib(1) == 1
def test_fib_ind():
    assert fib(3) == 2
    assert fib(6) == 9
def test_fib_raise():
    fib(-1)
    fib(3.6)

def test_fib_argv():
    sa = sys.argv
    sys.argv = ["", "6"]
    assert fib_argv()==13
    sys.argv = sa
```

```
_____________________________ test_fib_raise ___
_____________________________

    def test_fib_raise():
>       fib(-1)

test_fcns.py:18:
- - - - - - - - - - - - - - - - - - - - - - - - -
 - - - - - - - - - - - - - - - -

n = -1

    def fib(n):
      if n < 0 or not isinstance(n, int):
>       raise ValueError
E       ValueError

fcns.py:6: ValueError
_____________________________ test_fib_argv ___
_____________________________

    def test_fib_argv():
        sa = sys.argv
        sys.argv = ["", "6"]
>       assert fib_argv()==13
E       assert 8 == 13
E        +  where 8 = fib_argv()

test_fcns.py:24: AssertionError
==================== 3 failed, 1 passed in 0.18
seconds ====================
```

Marks, Fixtures

Skipping tests marked “slow”

Before

```
import sys
from fcns import \
    fib, fib_argv

# It's important that
# this file's and
# function's names
# both start with "test_"
# This is how pytest
# knows it's a test.
def test_fib_base():
    assert fib(0) == 0
    assert fib(1) == 1
def test_fib_ind():
```

After

```
import sys
import pytest
from fcns import \
    fib, fib_argv

# It's important that
# this file's and
# function's names
# both start with "test_"
# This is how pytest
# knows it's a test.
@pytest.mark.slow
def test_fib_base():
    assert fib(0) == 0
    assert fib(1) == 1
def test_fib_ind():
```

```
13:31:24.38  C:\Users\Acer\Desktop\git_repos\learnin
ation\fixtures_parametrize
   >pytest -m "not slow"
============================= test session starts ==
==========
platform win32 -- Python 3.6.4, pytest-3.3.2, py-1.5

hypothesis profile 'default' -> database=DirectoryBa
se('C:\\Users\\Acer\\Desktop\\git_repos\\learning\\p
on\\fixtures_parametrize\\.hypothesis\\examples')
rootdir: C:\Users\Acer\Desktop\git_repos\learning\py
n\fixtures_parametrize, inifile:
plugins: hypothesis-3.82.6
collecting 0 items
collecting 8 items
collected 8 items


test_fcns.py .FFFFFF
     [100%]

================================== FAILURES =======
==========
_____________________________ test_fib_raise[(-1,)] _
```

```
============================= 1 tests deselected ==================
==========
============== 6 failed, 1 passed, 1 deselected in 0.29 seconds ====
==========
```

Running parametrized tests

Before (2 asserts in 1 test)

```
def test_fib_ind():
    assert fib(3) == 2
    assert fib(6) == 9
```

After (4 tests from parametrizing a single method)

```
# We run the same test, multiple times
# with different values
@pytest.mark.parametrize("arg, val", [
    (3, 2),
    (4, 3),
    (5, 6),
    (6, 8),
], ids=repr)
def test_fib_ind(arg, val):
    assert fib(arg) == val
```

```
   >pytest -v test_fcns.py::test_fib_ind
============================= test session starts =============================
platform win32 -- Python 3.6.4, pytest-3.3.2, py-1.5.2, pluggy-0.6.0 -- C:\Users
cachedir: .cache
hypothesis profile 'default' -> database=DirectoryBasedExampleDatabase('C:\\User
OS\\pytest_presentation\\fixtures_parametrize\\.hypothesis\\examples')
rootdir: C:\Users\Acer\Desktop\git_repos\ONE_OFF_REPOS\pytest_presentation\fixtu
plugins: hypothesis-3.82.6
collected 4 items

test_fcns.py::test_fib_ind[3-2] PASSED                                   [ 25%]
test_fcns.py::test_fib_ind[4-3] PASSED                                   [ 50%]
test_fcns.py::test_fib_ind[5-6] FAILED                                   [ 75%]
test_fcns.py::test_fib_ind[6-8] PASSED                                   [100%]

================================== FAILURES ===================================
_____________________________ test_fib_ind[5-6] ______________________________

arg = 5, val = 6

    @pytest.mark.parametrize("arg, val", [
        (3, 2),
        (4, 3),
        (5, 6),
        (6, 8),
    ], ids=repr)
    def test_fib_ind(arg, val):
>       assert fib(arg) == val
E       assert 5 == 6
E        +  where 5 = fib(5)

test_fcns.py:31: AssertionError
==================== 1 failed, 3 passed in 0.13 seconds =====================
```

Testing that errors are raised

```python
@pytest.mark.parametrize("n", [
    -1, -2, -10,
    3.6, 1.1, 3
],ids=repr)
def test_fib_raise(n):
    # Will fail the test UNLESS
    # a ValueError is raised
    with pytest.raises(ValueError):
        fib(n)
```

```
>pytest -v test_fcns.py::test_fib_raise
============================= test session starts =============================
platform win32 -- Python 3.6.4, pytest-3.3.2, py-1.5.2, pluggy-0.6.0 -- C:\Users
cachedir: .cache
hypothesis profile 'default' -> database=DirectoryBasedExampleDatabase('C:\\User
OS\\pytest_presentation\\fixtures_parametrize\\.hypothesis\\examples')
rootdir: C:\Users\Acer\Desktop\git_repos\ONE_OFF_REPOS\pytest_presentation\fixtu
plugins: hypothesis-3.82.6
collected 6 items

test_fcns.py::test_fib_raise[-1] PASSED                                  [ 16%]
test_fcns.py::test_fib_raise[-2] PASSED                                  [ 33%]
test_fcns.py::test_fib_raise[-10] PASSED                                 [ 50%]
test_fcns.py::test_fib_raise[3.6] PASSED                                 [ 66%]
test_fcns.py::test_fib_raise[1.1] PASSED                                 [ 83%]
test_fcns.py::test_fib_raise[3] FAILED                                   [100%]

================================== FAILURES ===================================
_____________________________ test_fib_raise[3] ______________________________

n = 3

    @pytest.mark.parametrize("n", [
        -1, -2, -10,
        3.6, 1.1, 3
    ],ids=repr)
    def test_fib_raise(n):
        # Will fail the test UNLESS
        # a ValueError is raised
        with pytest.raises(ValueError):
>           fib(n)
E           Failed: DID NOT RAISE <class 'ValueError'>

test_fcns.py:43: Failed
==================== 1 failed, 5 passed in 0.14 seconds =====================
```

Fixtures (Setup and Teardown)

```python
# We can do targetted
# setup/teardown
# with fixtures
@pytest.fixture()
def argv6():
    sa = sys.argv
    try:
        sys.argv = ["", "6"]
        yield sys.argv
    finally:
        sys.argv = sa
def test_fib_argv(argv6):
    print(argv6)
    assert fib_argv()==13
```

```
   >pytest -v test_fcns.py::test_fib_argv
============================= test session starts =============================
platform win32 -- Python 3.6.4, pytest-3.3.2, py-1.5.2, pluggy-0.6.0 -- C:\Users\Acer
\Anaconda3\python.exe
cachedir: .cache
hypothesis profile 'default' -> database=DirectoryBasedExampleDatabase('C:\\Users\\Ac
er\\Desktop\\git_repos\\learning\\pytest_presentation\\fixtures_parametrize\\.hypothe
sis\\examples')
rootdir: C:\Users\Acer\Desktop\git_repos\learning\pytest_presentation\fixtures_parame
trize, inifile:
plugins: hypothesis-3.82.6
collected 1 item

test_fcns.py::test_fib_argv FAILED                                       [100%]

================================== FAILURES ===================================
_________________________________ test_fib_argv _________________________________

argv6 = ['', '6']

    def test_fib_argv(argv6):
        print(argv6)
>       assert fib_argv()==13
E       assert 8 == 13
E        +  where 8 = fib_argv()

test_fcns.py:45: AssertionError
---------------------------- Captured stdout call -----------------------------
['', '6']
========================== 1 failed in 0.11 seconds ===========================
```

Built-in Fixtures

Built-in Fixtures

```
>pytest --fixtures
============================= test session starts =============================
platform win32 -- Python 3.6.4, pytest-3.3.2, py-1.5.2, pluggy-0.6.0
hypothesis profile 'default' -> database=DirectoryBasedExampleDatabase('C:\\User
er\\Desktop\\git_repos\\learning\\pytest_presentation\\fixtures_parametrize\\.hy
sis\\examples')
rootdir: C:\Users\Acer\Desktop\git_repos\learning\pytest_presentation\fixtures_p
trize, inifile:
plugins: hypothesis-3.82.6
collected 8 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.
```

```
monkeypatch
    The returned ``monkeypatch`` fixture provides these
    helper methods to modify objects, dictionaries or os.environ::

        monkeypatch.setattr(obj, name, value, raising=True)
        monkeypatch.delattr(obj, name, raising=True)
        monkeypatch.setitem(mapping, name, value)
        monkeypatch.delitem(obj, name, raising=True)
        monkeypatch.setenv(name, value, prepend=False)
        monkeypatch.delenv(name, value, raising=True)
        monkeypatch.syspath_prepend(path)
        monkeypatch.chdir(path)

    All modifications will be undone after the requesting
    test function or fixture has finished. The ``raising``
    parameter determines if a KeyError or AttributeError
    will be raised if the set/deletion operation has no target.
recwarn
    Return a WarningsRecorder instance that provides these methods:

    * ``pop(category=None)``: return last warning matching the category.
    * ``clear()``: clear list of warnings

    See http://docs.python.org/library/warnings.html for information
    on warning categories.
tmpdir_factory
    Return a TempdirFactory instance for the test session.
tmpdir
    Return a temporary directory path object
    which is unique to each test function invocation,
    created as a sub directory of the base temporary
    directory.  The returned object is a `py.path.local`_
    path object.

---------------------- fixtures defined from test_fcns ----------------------
argv6
    test_fcns.py:36: no docstring available
```

The Built-in monkeypatch Fixture

- This saves a lot of boilerplate!

Before

```
# We can do targetted
# setup/teardown
# with fixtures
@pytest.fixture()
def argv6():
    sa = sys.argv
    try:
        sys.argv = ["", "6"]
        yield sys.argv
    finally:
        sys.argv = sa
def test_fib_argv(argv6):
    print(argv6)
    assert fib_argv()==13
```

After

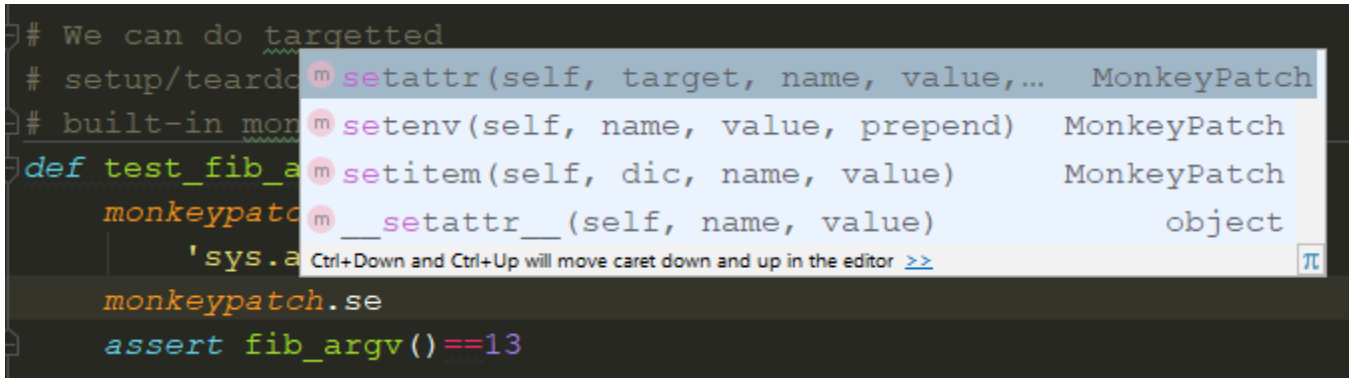
```
# We can do targetted
# setup/teardown with
# built-in monkeypatch fixture
def test_fib_argv(monkeypatch):
    monkeypatch.setattr(
        'sys.argv', ["", "6"])
    assert fib_argv()==13
```

PyCharm and Pytest

- The Pytest testing framework is popular
- Tools are being programmed to understand its magic.
- I use PyCharm

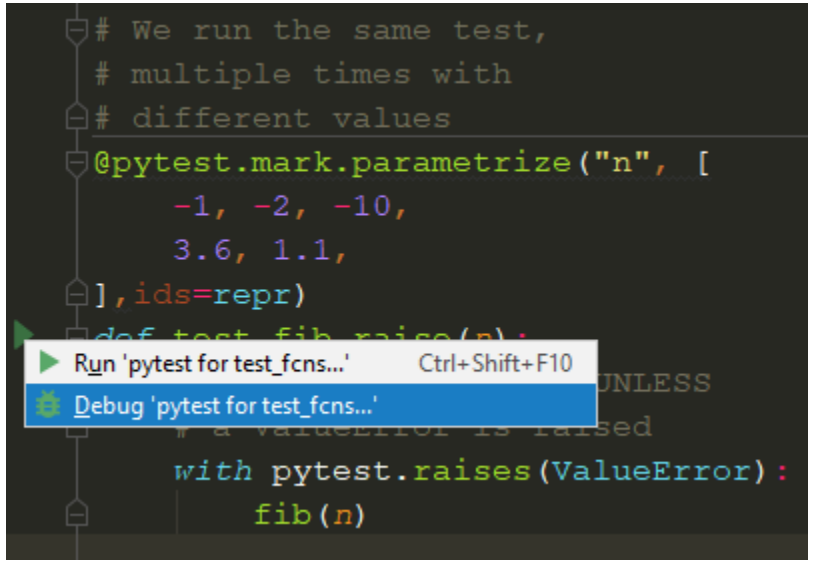
The Built-in monkeypatch Fixture

- PyCharm understands pytest (and maybe other editors do too?)

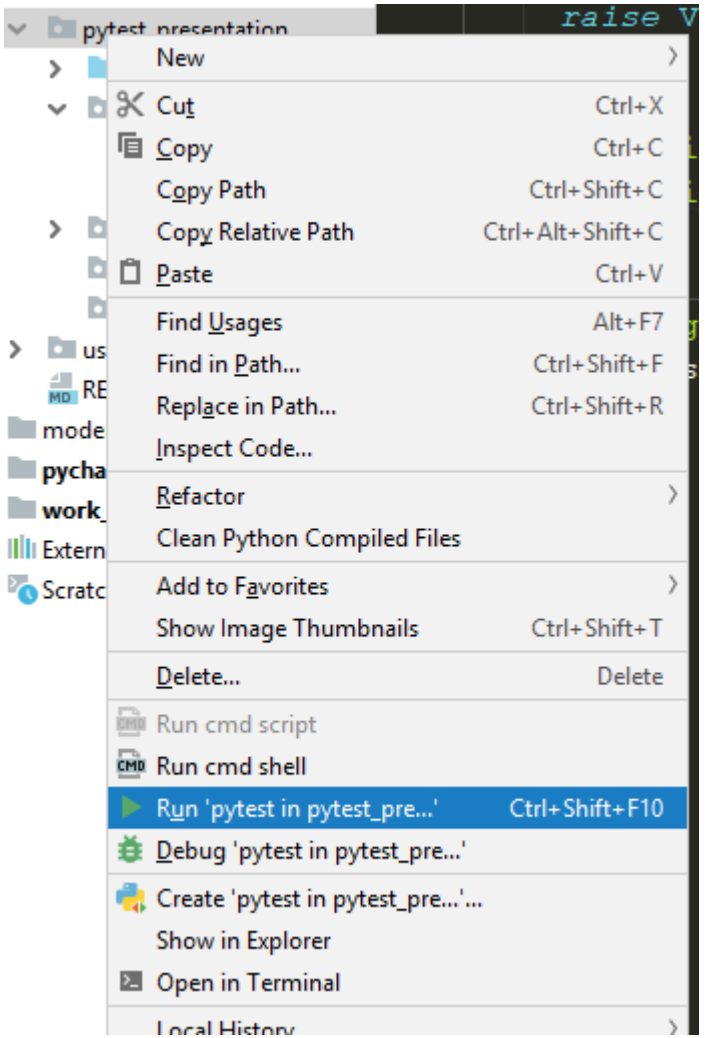


Easily run/debug tests

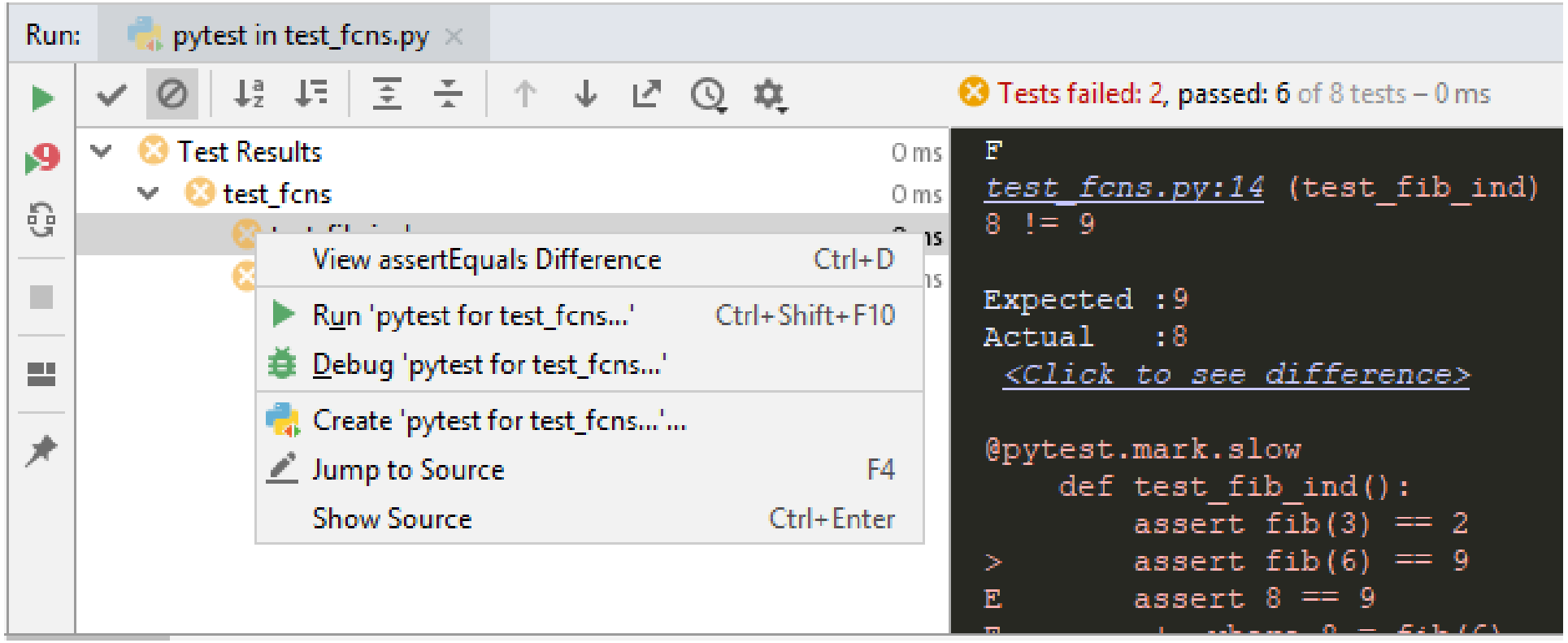
Run/debug individual test



Run/debug all tests in a file/directory



Easily re-run/debug failed tests



Github Repo of Code for this presentation

- https://github.com/wbkdef/pytest_magic_presentation

Resources I used for Learning PyTest

- *I found learning just from pytest.org difficult. I used this material from* Brian Okken
- **BOOK: Python Testing with Pytest: Simple, Rapid, Effective, and Scalable**
  - by Brian Okken, Oct 2017
  - https://www.oreilly.com/library/view/python-testing-with/9781680502848/ , 4.23 on Goodreads

- **VID: Visual Testing with PyCharm and pytest** => https://www.youtube.com/watch?v=FjojZxDZscQ
- **VID: Productive pytest with PyCharm** => https://www.youtube.com/watch?v=ixqeebhUa-w

- **VID: How Testing Strategy can Increase Developer Efficiency and Effectiveness** https://www.youtube.com/watch?v=o9D2ZxMO3Nw

- **Google**