Pytest

December 12, 2020

1 pytest

pytest es un framework que nos va a permitir escribir pequeños tests para porciones pequeñas de código y además nos permitirá escalar para realizar tests más complejos y de aplicaciones más complejas

[1]:
```
import unittest

def suma_1(x):
    return x + 1

# Unittest:
class TestEjemplo(unittest.TestCase):
    def test_suma_1(self):
        self.assertEqual(11, suma_1(10))

#pytest:
def test_suma_1(x):
    assert suma_1(10) == 11

# cuando no se cumpla una condicion se lanza -> AssertionError
```

Que es lo que nos da pytest: * No hace falta usar los self.assert. Información detallada de los errores. * Auto discovery de tests. * Se pueden escribir fixtures modulares. * Se puede seguir escribiendo tests usando `unittest` * Una gran cantidad de plugins (+315)

Documentación: https://docs.pytest.org/

1.1 Instalación

```
pip3 install -U pytest
```

Luego ver si se instaló correctamente

```
pytest --version
```

1.2 Assert errores

Se puede usar el método raises para captar todas las cosas que se “raiseen”

[4]:
```python
import pytest

def f():
    raise ValueError("error")

def test_f():
    with pytest.raises(ValueError):
        f()
```

1.3 Agrupar tests

Podemos agrupar los tests en clases. Esto puede ser beneficioso por: * Organización de tests. * Compartir Fixtures solo en una clase en particular * Aplicar *marks* a nivel de clases.

[6]:
```python
class TestEjemplo:
    def test_f(self):
        with pytest.raises(ValueError):
            f()
    def test_suma_1(self):
        assert 11 == test_suma_1(10)
```

1.4 Fixtures

Hay muchas fixtures que ya viene creadas dentro del framework (built-in). La lista de builtin la podemos ver acá: https://docs.pytest.org/en/stable/builtin.html

[8]:
```python
def test_tmpdir_fixture(tmpdir):
    print(tmpdir)
    assert 0
```

Los `test fixtures` hacen un setup del sistema e inicializa los tests (https://en.wikipedia.org/wiki/Test_fixture#Software). Esto se hace para lograr que los tests sean confiables, repetibles, sencillos de leer, sencillo de escribir, etc.

En pytest se utiliza el decorador: `@pytest.fixture`

[9]:
```python
# Ejemplo (usar smtp)
```

1.5 Marcas

Las marcas son utilizadas para agregar metadata a los tests. Existen marcas creadas dentro del pytest que las pueden ver en https://docs.pytest.org/en/stable/reference.html#marks-ref

Los más utilizados son: * usefixture * skip * skipif * xfail * parametrize

[12]:
```python
# Ejemplo
```