# Instalar

pip install pytest

pip install python-dotenv

pytest -h

# Ubicación

Poner las pruebas en CP/tests/

Busca las pruebas en los archivos **test\_\*.py** o **\*\_test.py**

Y en específico las funciones que comienzan con test

# assert

En algún lado dentro de la función se evalúa con assert expresión booleana

import pytest

def test\_greater():

num = 100

assert num > 100

# probarlos

## todos

Para probar los test

CP> pytest -v

El busca automáticamente la carpeta

## filename

Para probar un solo archivo

CP/test> pytest <filename> -v

## subnombre

Si se tiene def test\_subnombreprueba() se puede ejecutar esta prueba especifica con

CP/test> pytest -k subnombreprueba -v

## marcas

Se puede configurar para solo se ejecuten un conjunto de pruebas específicas, marcando las pruebas con decoradores en la forma

@pytest.mark.nombremarca

def test\_subnombreprueba()

y luego

CP/test> pytest -m nombremarca -v

## maxfail

Para detener la ejecución de las pruebas cuando se han detectado un determinado numero de fallos es agregado al comando de ejecución el parámetro --maxfail = NUMERO MAXIMO

CP> pytest -v --maxfail=5

# fixture

Se pueden declarar funciones atributos, que serán pasados a las funciones pruebas, marcando estas funciones con decoradores del tipo

@pytest.fixture

def input\_value():

input = 39

return input

def test\_divisible\_by\_3(input\_value):

assert input\_value % 3 == 0

def test\_divisible\_by\_6(input\_value):

assert input\_value % 6 == 0

# conftest

se pueden declarar funciones que van a estar accesibles para todos los test, generalmente los fixture, estas hay que ubicarlas dentro del archivo CP/tests/conftest.py, las funciones declaradas dentro de este archivo no hay que importarlas en los demás

# pares de parámetros

para mandar a ejecutar una misma función con distintos pares de parámetros

@pytest.mark.parametrize("num, output",[(1,11),(2,22),(3,35),(4,44)])

def test\_multiplication\_11(num, output):

assert 11\*num == output

# xfail

El decorador @pytest.mark.xfail hace que la función de prueba se ejecute pero que no se consideren sus resultados, ni se imprimen en consola

# skip

El decorador @pytest.mark.skip hace que la función de prueba no se ejecute