pytest - exemple complet

👍 Voici un **exemple plus complet** pour bien montrer l'usage de `pytest` avec plusieurs fonctions et ce qui se passe si un test échoue.

5TTR 6TTR

Exemple complet avec `pytest`

Fichier : `test_algo.py`

```python
# Fonctions à tester
def addition(a, b):
    return a + b

def moyenne(liste):
    return sum(liste) / len(liste)

def est_palindrome(mot):
    mot = mot.lower()
    return mot == mot[::-1]


# Tests
def test_addition():
    assert addition(2, 3) == 5
    assert addition(-2, 5) == 3
    assert addition(0, 7) == 7

def test_moyenne():
    assert moyenne([10, 20, 30]) == 20
    assert moyenne([5, 15]) == 10
    assert moyenne([42]) == 42

def test_palindrome():
    assert est_palindrome("radar") == True
    assert est_palindrome("kayak") == True
    assert est_palindrome("python") == False
```

Lancer les tests

Dans ton terminal :

```
pytest test_algo.py
```

Cas 1 — Tous les tests passent

Résultat attendu :

```
==================== test session starts ====================
collected 3 items

test_algo.py ...                                    [100%]

===================== 3 passed in 0.01s =====================
```

Le `...` signifie que les 3 fonctions de test ont réussi.

Cas 2 — Un test échoue

Supposons que tu aies fait une erreur volontaire dans `moyenne` :

```
def moyenne(liste):
    return sum(liste)   # ❌ oublié de diviser par len(liste)
```

Quand tu relances :

```
==================== test session starts ====================
collected 3 items

test_algo.py ..F                                    [100%]

================================= FAILURES =================================
_____________________________ test_moyenne _____________________________

    def test_moyenne():
>       assert moyenne([10, 20, 30]) == 20
E       assert 60 == 20
E        +  where 60 = moyenne([10, 20, 30])

test_algo.py:16: AssertionError
================== 1 failed, 2 passed in 0.02s ==========================
```

👉 `pytest` te montre directement :

- Quelle fonction a échoué (`test_moyenne`).
- Quelle ligne (`assert moyenne([10, 20, 30]) == 20`).
- Ce qu'il a obtenu (`60`) vs ce qu'il attendait (`20`).

À retenir

- `pytest` lance tous les fichiers `test_*.py` automatiquement.
- Le retour est **très lisible** : un point = un test réussi, `F` = échec.
- Les messages d'erreur sont clairs et montrent la différence entre attendu et obtenu.
