pytest

Sommaire

1/ Qu’est-ce que Pytest et quelle est son utilitée ?

2/ Pourquoi l’utiliser ?

3/ Installation et configuration requises

4/ Écrire un test avec Pytest

5/ Les fixtures et les markers

6/ Organisation des tests

7/ Les différents plugins

8/ Architecture Pytest vs Unittest

1/ Qu’est-ce que Pytest et quelle est son utilitée ?

-- Pytest est un framework de test Python issu du projet PyPy(implémentation alternative de Python)

-- Il est utilisé pour écrire divers types de tests logiciels comme:

° des tests unitaires

° des tests d'intégration

° des tests fonctionnels

2/ Pourquoi l’utiliser ?

-- Pytest est très facile à démarrer grâce à sa **syntaxe simple et facile**

-- Il peut exécuter **plusieurs tests en même temps** grâce au plugin pytest-xdist

-- Il peut exécuter **un test spécifique ou un sous-ensemble de tests**

-- Pytest est **Open source**

3/ Installation et configuration requises

```
pip install pytest
```

ini

```
[pytest]
adopts = -v --maxfail=1
testpaths = tests
```

Il est possible de personnaliser pytest via un fichier de configuration comme pytest.ini
Ce fichier permet de définir des options par défaut (mode verbeux, arrêt après le premier échec, répertoire de tests)

4/ Écrire un test

Editeur de code -->

```
def fonction_a_tester(x):
    return x + 1
def test_fonction_a_tester():
    assert fonction_a_tester(3) == 4
```

Terminal -->

```
pytest
```

5/ Les fixtures et les marqueurs (markers)

Les fixtures sont des fonctions qui permettent de **préparer l'environnement de test** et de les **partager entre plusieurs tests**. Elles facilitent le **réemploi** du code et la gestion des ressources avant et après les tests.

python

```
import pytest

@pytest.fixture
def sample_data():
    return[1, 2, 3, 4]

def test_sample_data(sample_data):
    assert sum(sample_data) == 10
```

Les marqueurs permettent de **catégoriser** les tests et de les **exécuter** sélectivement.Les marqueurs sont également utilisés pour **signaler des tests attendus en échec ou des tests ignorés.**

```python
import pytest


@pytest.mark.lent
def test_processus_complexe():
    pass


@pytest.mark.rapide
def test_calcul_simple():
    assert 1 + 1== 2
```

Pour exécuter uniquement les tests rapides, on peut lancer

```python
pytest -m rapide
```

6/ Organisation des tests

-- Nommage des fichiers : Par convention, les fichiers de test commencent souvent par test_ ou se terminent par _test.py.

-- Structure des dossiers : Il est nécessaire d’organiser les tests dans un dossier dédié nommé “tests” qui aide à maintenir une structure claire dans le projet

7/ Les différents plugins

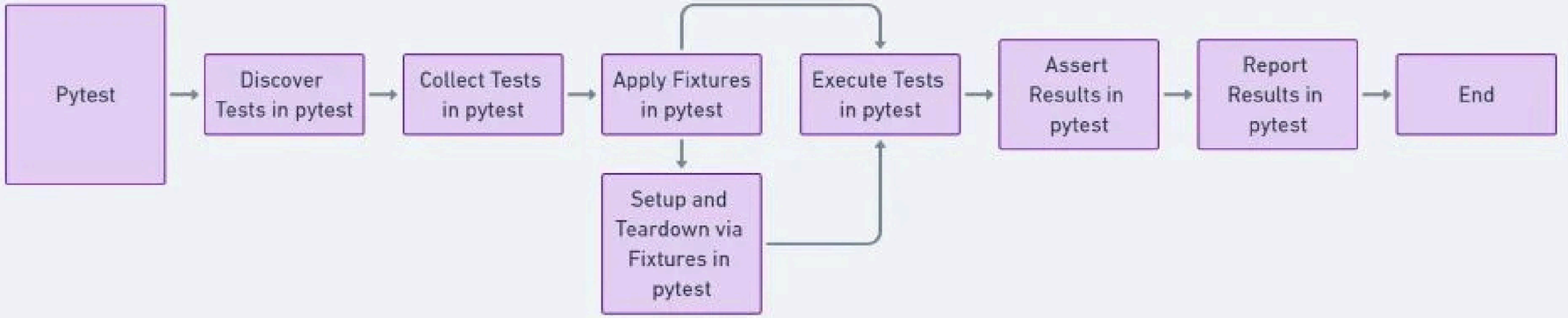
--> **pytest-cov** : Génère des rapports de couverture de code (métrique qui permet de comprendre la part testée de la source)

--> **pytest-mock** : Facilite la création et la gestion des mocks dans les tests (mock = objet virtuel)

--> **pytest-xdist** : Exécute les tests en parallèle et accélérer les suites de tests.

--> **pytest-django / pytest-flask** : Intégrer pytest dans des environnements web spécifiques

Pytest Architecture



Unittest Architechture