4_Code les tests unitaires

Prenons une fonction qui permet de calculer la valeur absolue d'un nombre.

> **La valeur absolue d'un nombre réel est sa valeur numérique considérée sans tenir compte de son signe.**

```
def to_absolute(number):
    if number <= 0:
        return -number
    return number
```

> **Comment allons-nous coder ce test ?**

Python intègre une fonctionnalité : **les doctests**.

Écrire un doctest

Un doctest reprend la présentation d'un **terminal**. Ajoutez trois chevrons au début d'une ligne pour imiter un terminal, puis le résultat attendu juste en dessous.

```
"""
>>> to_absolute(3)
3
"""
```

Que fait-on ?

Nous exécutons la fonction `to_absolute` avec pour argument `3`, exactement comme si nous étions dans un terminal. Enfin, nous indiquons à la ligne la réponse attendue : `3`.

Nous pouvons aussi ajouter **plusieurs tests** à la suite afin de vérifier plusieurs scénarios, comme ci-dessous :

```
"""
>>> to_absolute(3)
3
>>> to_absolute(-10)
10
"""
```

Comment lancer le test ?

En utilisant la commande suivante dans votre terminal :

```
python -m doctest <nom du fichier>
```

Par exemple, si votre fonction est implémentée dans le fichier `main.py`, il faudra lancer la commande suivante :

```
python -m doctest main.py
```

Ajouter `-v` après le mot-clé `doctest` pour avoir plus de détails

```
python -m doctest -v <nom du module>
```

Les doctests sont très utiles en tant que documentation. Ils fournissent un **exemple concret** de la manière dont le programme est censé fonctionner.

C'est très bien, mais j'aimerais avoir un document dans lequel je pourrais lire uniquement mes tests. C'est pourquoi je vous propose de voir une autre façon de créer des tests dans un fichier dédié aux tests.

Créez un fichier de tests

Créez un nouveau fichier dans le dossier parent des scripts, au même niveau que le `README.md`, ou créez ce fichier dans un package `tests/` qui contiendra l'ensemble des tests de l'application. Cela nous permettra de **différencier le code source et les fichiers de test**.

> **Comment créer les tests ?**

On ne pourra pas utiliser les doctests ici, car ils sont pensés pour s'intégrer dans un script. Il nous faudra donc utiliser **une librairie de tests** qui nous donnera un **environnement complet** pour bien débuter.

Les librairies de tests

Plusieurs librairies de tests existent, dont Unittest qui est la librairie par défaut de Python. Dans ce cours, nous utiliserons principalement Pytest.

En résumé

- Les doctests se trouvent dans les docstrings et complètent la documentation d'une fonction.

- Les doctests permettent de documenter et vérifier le comportement d’une fonction.
- Il existe deux frameworks de tests, **Unittest** et **Pytest**.