06

To learn more: Markers

Part of the reason why Pytest is known as a testing framework, not a simple library, is that Pytest has a wide range of tools aimed at improving the efficiency and organization of the tests developed.

Markers are one of those incredible Pytest tools and offer not only a way to **better** organize tests with custom *tags ,* **but** also help define how certain tests will work or be executed.

skip

```
@pytest.mark.skip(reason="não quero executar isso agora")
def test_aleatorio():
    ...
```

COPY CODE

By using the *marker* `skip` we can skip a test if it is not necessary to run it at that moment.

skipif

```
import sys


@pytest.mark.skipif(sys.version_info < (3, 10), reason="Requer Python na
```


Above, the test is not executed if `sys.version_info < (3, 10)` it is true, that is, if the Python version is below version 3.10.

By using the *marker* `skipif` we can skip a test if it fits into a certain situation defined by a conditional.

xfail

```
@pytest.mark.xfail
def test_funcao():
    ...
```

COPY CODE

By using the *marker* `xfail` we specify that the test should return a failure, instead of passing.

These and many other possibilities for using **markers** to modify the mechanics of using tests can be seen in the official Pytest documentation.

How to mark test functions with attributes (https://docs.pytest.org/en/7.1.x/how-to/mark.html#mark)