

Overview

Pytest is a framework that makes building simple and scalable tests easy. Tests are expressive and readable-no boilerplate code required. Get started in minutes with a small unit test or complex functional test for your application or library.

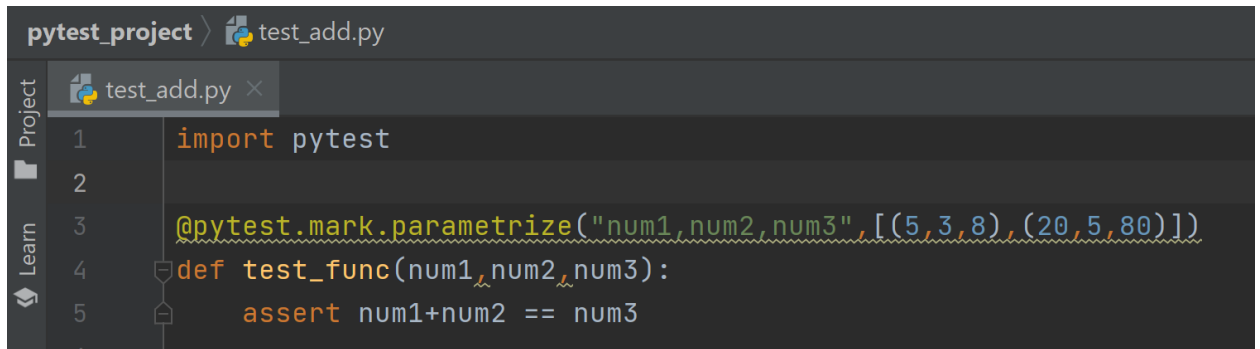
Advantages of Pytest

- Very easy to start with because of its simple and easy syntax.
- Can run multiple tests in parallel.
- Can run a specific test or a subset of tests.
- Automatically detect tests.
- Skip a subset of tests.
- Open source and free.

Usage

- Install pytest:
Run the following command- **pip install pytest**
- Once the installation is complete you can confirm it with by running the following command- **py.test -h**
- Navigate to the folder where your test file is created.
- Import pytest in the test file.
- Test name must start with test_.
- Run the test using the command-
"py.test" to run all the filenames starting with test_ and the filenames ending with _test from a folder and subfolders under that folder.
and
"py.test file_name.py" To run tests only from a specific file.

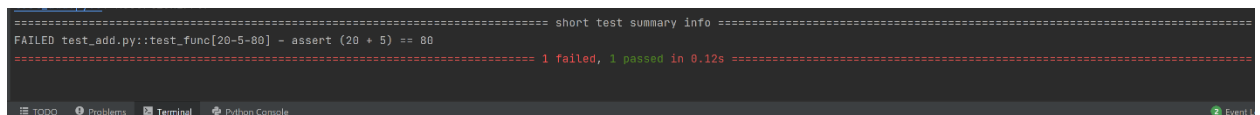
- **Testing Example**



The screenshot shows a code editor with a dark theme. The top bar indicates the project is 'pytest_project' and the file is 'test_add.py'. The left sidebar has 'Project' and 'Learn' tabs. The code in 'test_add.py' is as follows:

```
1 import pytest
2
3 @pytest.mark.parametrize("num1,num2,num3",[(5,3,8),(20,5,80)])
4 def test_func(num1,num2,num3):
5     assert num1+num2 == num3
```

- **Result**



The screenshot shows a terminal window with the following output:

```
===== short test summary info =====
FAILED test_add.py::test_func[20-5-80] - assert (20 + 5) == 80
===== 1 failed, 1 passed in 0.12s =====
```

The terminal window has a status bar at the bottom with icons for '10000', 'Problems', 'Terminal', and 'Python Console'.