**Install the plugin:**

pip install pytest-xdist – To support parallel execution

**To install pytest-rerunfailures:**

$ pip install pytest-rerunfailures – To support for running failures again

**To install pytest-html:**

$ pip install pytest-html – To generate html report

**Specifying tests / selecting tests**

Pytest supports several ways to run and select tests from the command-line.

**Run tests in a module**

pytest test\_mod.py

**Run tests in a directory**

pytest testing/

**Run tests by keyword expressions**

pytest -k "MyClass and not method"

This will run tests which contain names that match the given string expression, which can include Python operators that use filenames, class names and function names as variables. The example above will run TestMyClass.test\_something but not TestMyClass.test\_method\_simple.

**To run a specific test within a module:**

pytest test\_mod.py::test\_func

**Another example specifying a test method in the command line:**

pytest test\_mod.py::TestClass::test\_method

**Run tests by marker expressions**

pytest -m slow

Will run all tests which are decorated with the @pytest.mark.slow decorator.

**Execute the test function with “quiet” reporting mode: [Note: Does not support grouping of test cases]**

pytest -q test\_func.py

**Usage examples**

Speed up test runs by sending tests to multiple CPUs

**To send tests to multiple CPUs, type:**

pytest -n NUM

Especially for longer running tests or tests requiring a lot of I/O this can lead to considerable speed ups.

**Re-run all failures**

To re-run all test failures, use the **--reruns** command line option with the maximum number of times you’d like the tests to run:

**$ pytest --reruns 5**

Failed fixture or setup\_class will also be re-executed.

To add a delay time between re-runs use the --reruns-delay command line option with the amount of seconds that you would like wait before the next test re-run is launched:

**$ pytest --reruns 5 --reruns-delay 1**

Re-run individual failures

To mark individual tests as **flaky**, and have them automatically re-run when they fail, add the flaky mark with the maximum number of times you’d like the test to run:

**@pytest.mark.flaky(reruns=5)**

def test\_example():

import random

assert random.choice([True, False])

**Execute tests to generate html report:**

$ pytest --html=report.html