# What are the advantages of using fixtures over setup and teardown methods?

|  |  |  |
| --- | --- | --- |
| POC | Pytest fixture | Setup/Teardown method |
| Reusability | Fixtures can be used with multiple files and multiple function so the same fixture decorator can be used with different test cases | Setup/ teardown is used within the same test class so within each test case class we will have to make multiple duplication for each test class |
| Scalability | Multiple fixtures can be used with the same class/function/file enabling modular design for complex test cases and test suites | Setup/teardown method grow into large method of complex test suit which becomes harder to scale |
| Parametrization | Fixture support parametrization which enables different set of input parameters to the test case function | Setup/teardown has no parametrization which make it more difficult to achieve the same result as fixtures |
| Failure isolation | When failure happens to certain test cases it does not affect other test cases. Also a detailed report is generated for the failed test case and cause of failure | If setup fails all test cases fails so it has more isolation issues and it’s hard to detected the root cause of the failure |
| Plugins | Lots of useful plugins as mocker, faker, html generator | Limited |
| Simplicity | More simple as less class usage and built in assert method enabling easier and clearer assertions | More complex as it requires more classes usage and has more complicated assertion method |
| Readability | Fixtures are explicitly clear in the function. Enhancing code readability | Has more implicit execution and tied to specific classes and module before and after each test case which makes it less readable |