**Pytest:**

Pytest is one of the unit testing frameworks for python. It allows users to write simple and as well as complex functional and integration tests for applications. It is one of the popular testing frameworks among many users due to its simple syntax, scalability and strong features. It has full documentation which provides guidance for basic and as well as advanced features.

**Strengths:**

1. **Simple to use** – Test functions can be written using simple assert statements.
2. **Detailed error reports**- Shows what went wrong without the need of additional assertion methods or frameworks.
3. **Fixtures – Fixtures help to setup test environments and manage test dependencies.**
4. **Parametrization – Support running different sets of data on the same test function by allowing wide tests coverage without having to add extra test functions.**
5. **Plugins – Wide range of plugins are available to support various functionality.**
6. **Support for Complex Systems** – pytest support not only unit tests but also functional and integration testing in complex systems.
7. **Compatibility with Other Testing Libraries** – pytest is capable of running tests written for unittest or nose without modification which allow integration into existing python projects.
8. **Community and Support** - pytest has strong community with a lot of documentation where issues get addressed quickly.

**Limitations:**

1. **Performance Issues with Large Tests -** pytest can be slower when dealing with large tests due to the need of complex setup/teardown state and detailed failure reports.
2. **Complexity of Advanced Features – Although** features like fixtures and plugins are helpful, managing them properly would be difficult for new users.
3. **Verbose Output** – Although the detailed output of failures is helpful, it can be overwhelming at the same time with large amounts of data and specially when few tests fail simultaneously.
4. **Overhead of Plugins** - Relying heavily on plugins despite its substantial functionality can cause potential conflicts between plugins.
5. **Integration with IDEs** - Integration in some IDEs may not be as seamless based on the requirements.

**UnitTest**

Unittest is a testing framework which supports test automation. It supports sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework.

**Strengths:**

1. **Standard Library Integration** – unittest is part of Python's standard library and no additional installation is required. This makes it readily available and easily accessible.
2. **Structured Framework -** unittest follows an object-oriented approach to create test cases.
3. **Test Discovery** – It supports automatic discovery of test cases in a project, which can significantly simplify testing in large projects.
4. **Rich Assertions – It** provides a comprehensive set of assertion methods.
5. **Test Fixtures -** Support for setup and teardown methods help tests to run in a controlled environment.
6. **Suite Support – unittest** allows grouping of tests into suites which is helpful in large projects where tests need to be organized into categories.

**Limitations:**

1. **Verbosity -** Compared to modern frameworks, unittest can be less concise, which may slow down test development.
2. **Less Flexible Fixture Management – Although it** provides setup and teardown methods, it lacks more flexible fixture management systems.
3. **Lack of Advanced Features – Although it** supports a good range of basic testing needs, it lacks features such as parameterized testing and powerful plugins which are useful in complex scenarios.
4. **Difficult for Non-OOP Users** – Users who are not familiar with OOP concepts may find it difficult.
5. **Limited Support for modern features – unittest is** slower to adopt newer Python features compared to other community-driven testing frameworks.