Сравнительный анализ библиотек для тестирования

Библиотека	Плюсы	Минусы
py.test	Удобный assert (стандартный из Python). Подробный отчет, в том числе выгрузка в JUnitXML (для интеграции с Jenkins). Параметризация тестов. Метки (marks), позволяющие пропустить любой тест, пометить тест, как падающий. Независимость от API (no boilerplate). Данный модуль имеет достаточно большой список дополнительных модулей, которые можно установить отдельно. Возможность запуска тестов написанных на unittest и nose, то есть полная обратная совместимость с ними. Дополнительные возможности фикстур (возвращаемое значение, финализаторы, область видимости, объект request, автоиспользование, вложенные фикстуры).	Отсутствие дополнительного уровня вложенности: для модулей, классов, методов, функций в тестах есть соответствующий уровень.
unittest	Он очень прост и его легко использовать. Есть много возможностей: проверки (assert*), декораторы, позволяющие пропустить отдельный тест (@skip, @skipIf) или обозначить сломанные тесты (@expectedFailure), при написании тестов легко прослеживается ООП стиль, что весьма удобно для тестирования процедур и классов.	Требует написания большого количества кода, стиль больше похож на Java, и потому становится менее читабельным.
nose	Девизом nose является фраза «nose extends unittest to make testing easier», что можно перевести как «nose расширяет unittest, делая тестирование проще». nose идеален, когда нужно сделать тесты «по-быстрому», без предварительного планирования и выстраивания архитектуры приложения с тестами. Функционал nose можно расширять и настраивать с помощью плагинов.	Так как представляет собой расширение над стандартными unittest, то перенимает часть его минусов.
doctest	Документация всегда соответствует коду, простота написания, можно скопировать прямо из интерактивной сессии Python.	Сложный код быстро становится нечитаемым; текстовый редактор не подсветит такой код, а статический анализатор не найдет в нём ошибок).