Библиотеки для тестирования

Библиотека	Преимущества	Недостатки
unittest	он почти такой же в Java (Junit), C# (Nunit). его тесты запускает nose и pytest уже стоит (входит в стандартную поставку). nittest стар и надежен. Дает базовый, универсальный тест-класс.	Он немного неповоротлив – он навязывает наследование классов вместо того, чтобы разрешать функции в качестве тест-кейсов. Стиль ООП не очень "пайтоновский", по ощущениям. Тесты также нельзя параметризировать. Использовать unittest нужно, если необходим базовый фреймворк для юнит-тестов без дополнительных зависимостей. В прочих случаях есть фреймворки получше – например, pytest.
2. pytest	1) Независимость от API. 2) Подробный отчет. В том числе выгрузка в JUnitXML (для интеграции с Jenkins). 3) Удобный assert (стандартный из Python). 4) Динамические фикстуры всех уровней, которые могут вызываться как автоматически, так и для конкретных тестов. 5) Параметризация тестов, то есть запуск одного и того же теста с разными наборами параметров. 6) Метки (marks), позволяющие пропустить любой тест, пометить тест, как падающий (и это его ожидаемое поведение, что полезно при разработке) или просто именовать набор тестов, чтобы можно было запускать только его по имени.	1) Отсутствие дополнительного уровня вложенности: Для модулей, классов, методов, функций в тестах есть соответствующий уровень. 2) Необходимость отдельной установки модуля. 3) Для использования PyTest требуется немного больше знаний Python, чем для того же unittest.
3. mock	Макеты в Mock можно подставлять всюду и как угодно. Ваш код не придётся подстраивать под тестирование, что значит быстрее разработка, и, возможно, быстрее прототипирование. Можно выбросить из тестов	

	всё лишнее, всё занимающее время и ресурсы, оставив работать только тот код, который нужно проверить. Настройки макетов где нужно жёсткие (spec), где нужно гибкие, и патчи не оставляют за собой следов.	