**2024/01/11 00:00|Лекция. Пропуск тестов**

## Пропуск тестов

Данный урок будет посвящён обсуждению пропуска тестов. Не всегда все тесты бывают нам нужны. Мы с вами можем показать в некоторых случаях то, что тест нам временно не нужен. Почему мы говорим временно? Потому что, когда мы говорим о каком-то тесте, считаем, что он нам не нужен и решаем его удалить либо закомментировать. На самом деле это не самый лучший способ в том плане, что тесты могут нам снова понадобиться. И конкретный тест мы можем с вами достаточно легко пропустить за счёт объяснения [«unittest.skip»](mailto:%C2%AB@unittest.skip%C2%BB) и внутри указать причину (Рис.1).

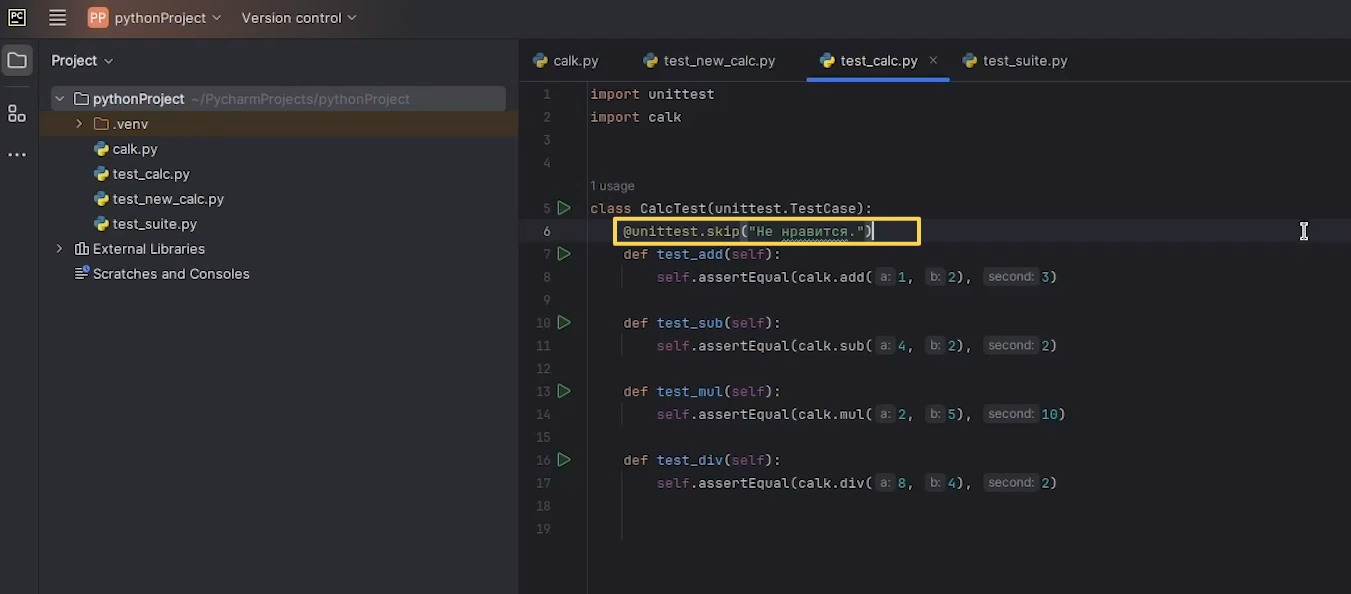


Рис.1

Запустим наш «test\_suite» (Рис.2). Почему это удобно? Потому что теперь мы не просто закомментировали тест и забыли о его существовании, а нам объясняется почему так произошло.

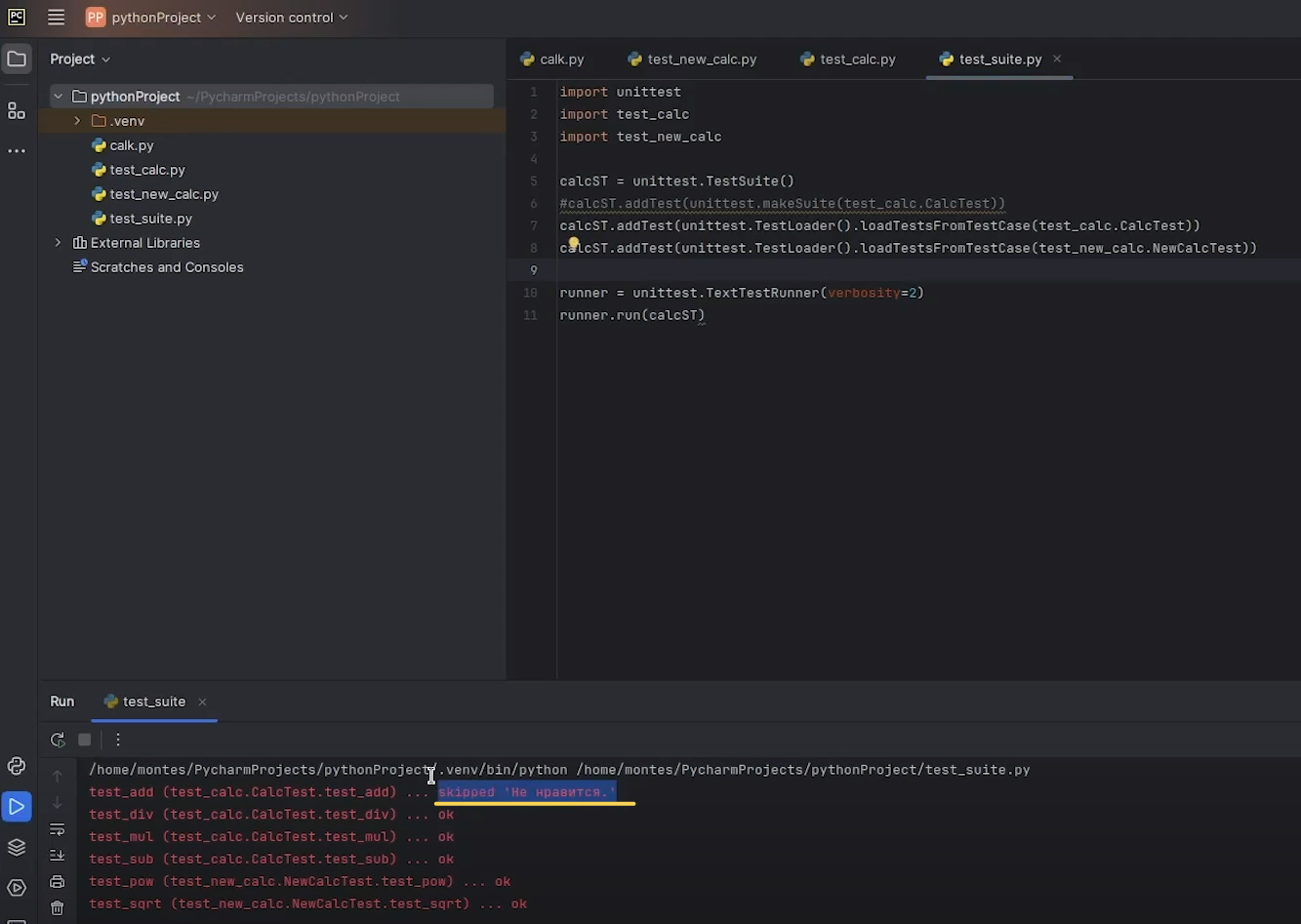


Рис.2

Помимо этого, мы можем делать проверку по условию. К примеру, [«unittest.skipIf»](mailto:%C2%AB@unittest.skip%C2%BB). Это условие, которое мы указываем как некое булево значение. Оно говорит: «если да, то пропускаем», «если нет, то не пропускаем». И есть [«unittest.skipUnless»](mailto:%C2%AB@unittest.skip%C2%BB), у которого наоборот: «если нет, то проходит», «если да, то не проходит» (Рис.3)

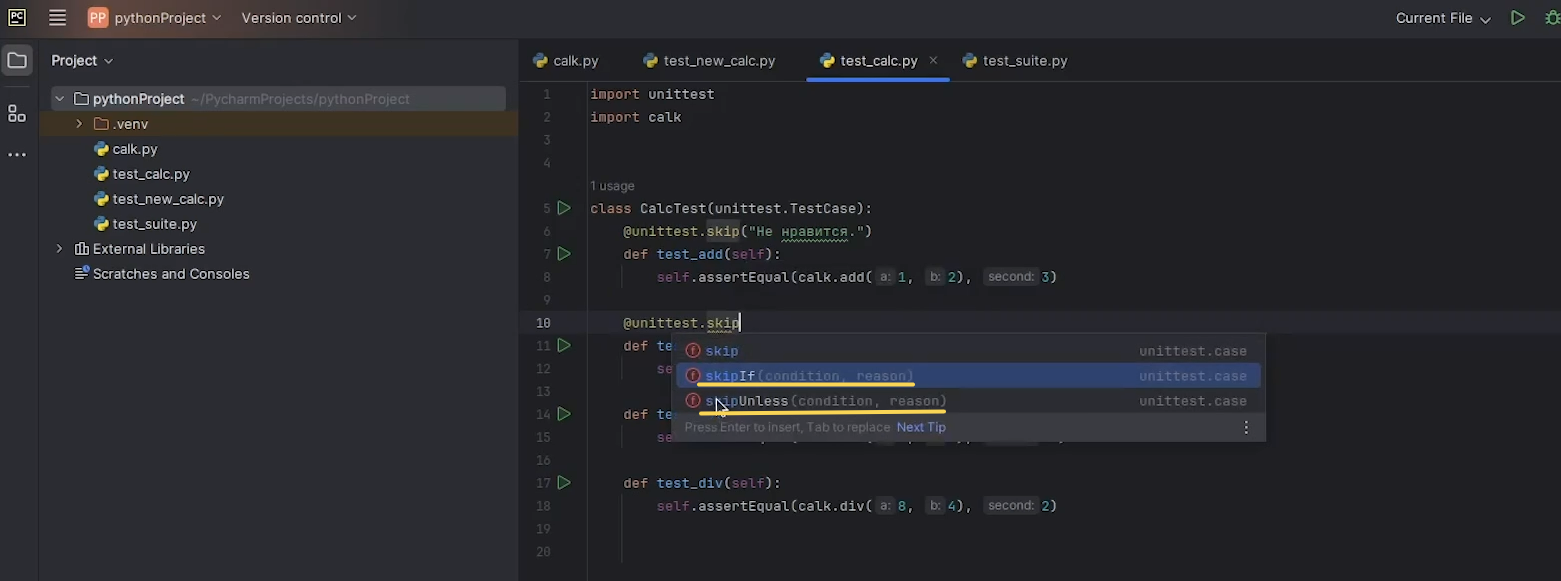


Рис.3

Сделаем «import random» (Рис.4).

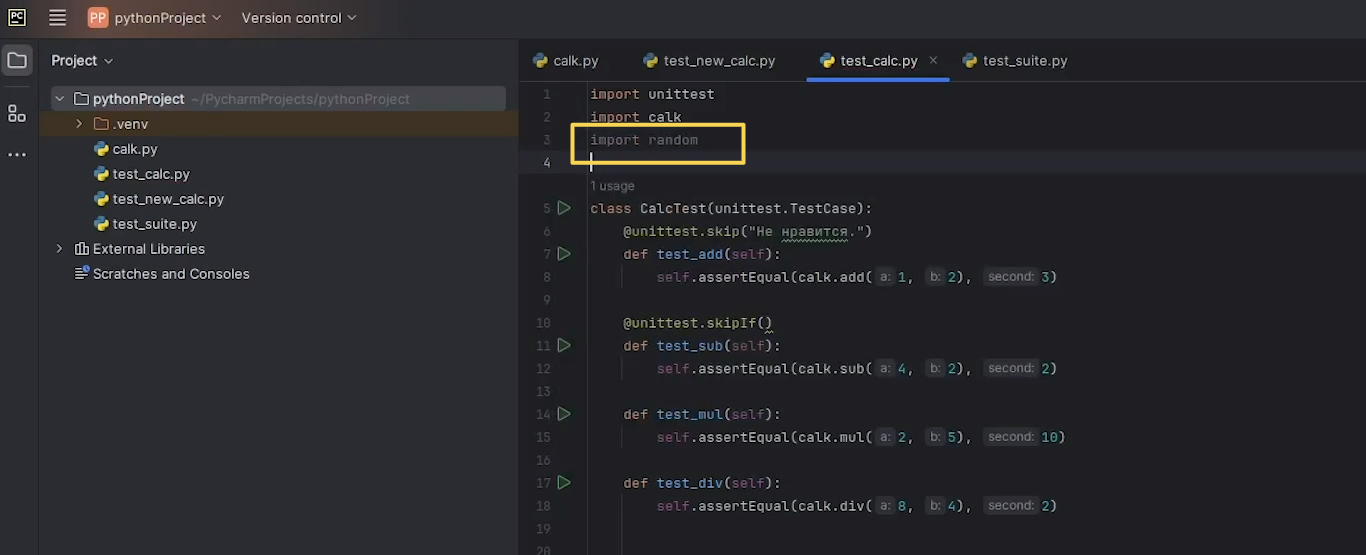


Рис.4

В [«unittest.skipIf»](mailto:%C2%AB@unittest.skip%C2%BB) допишем от «random.randint (a: 0, b: 2)». Сначала указывается значение булево, потом причина «Не повезло» (Рис.5).

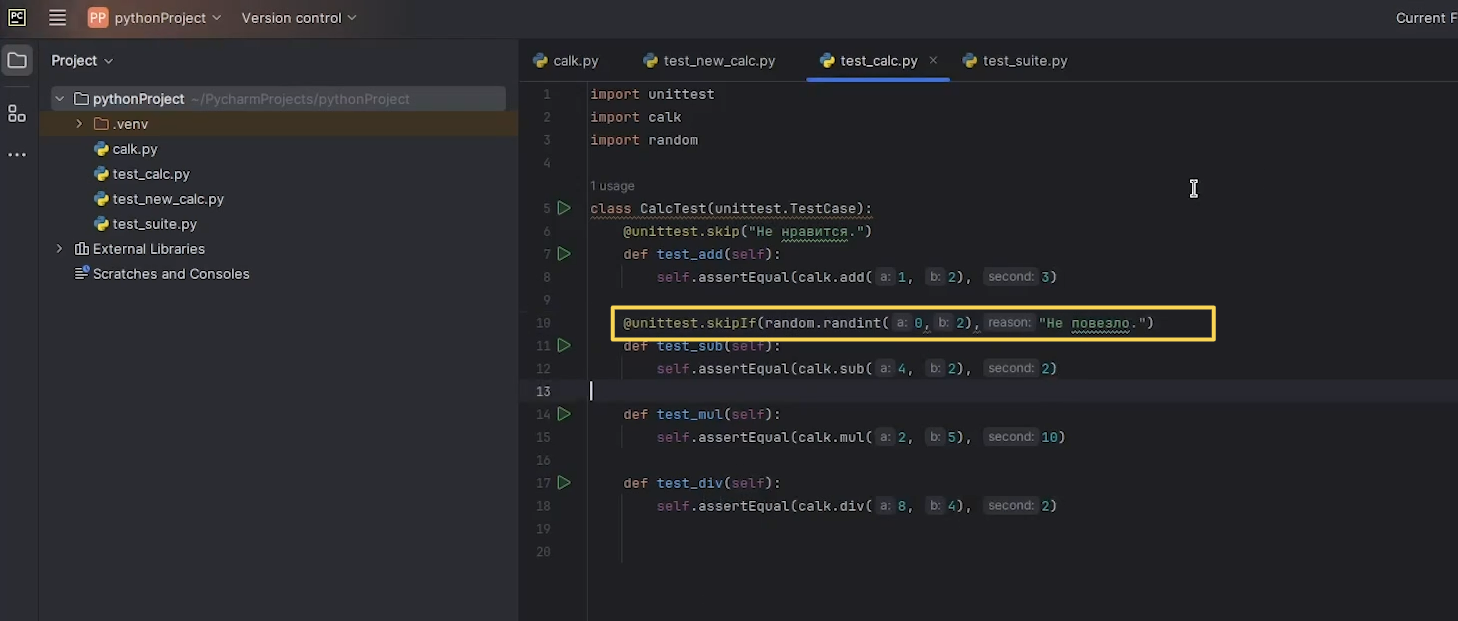


Рис.5

И давайте запустим (Рис.6). Как видите, тесту не повезло, и он не прошёл.

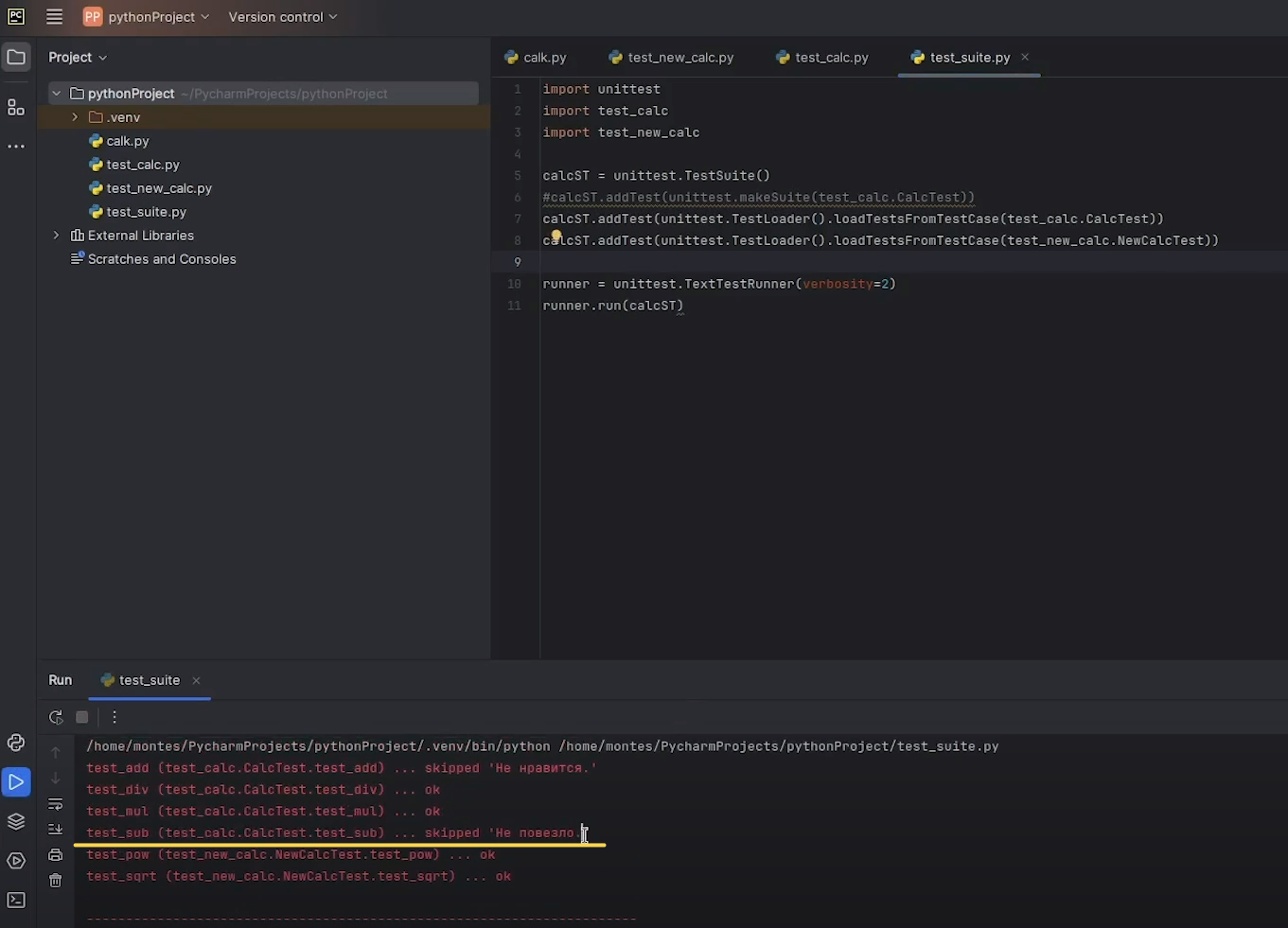


Рис.6

Вместо «random.randint (a: 0, b: 2)», укажем «False», чтобы у нас все сработало (Рис.7).

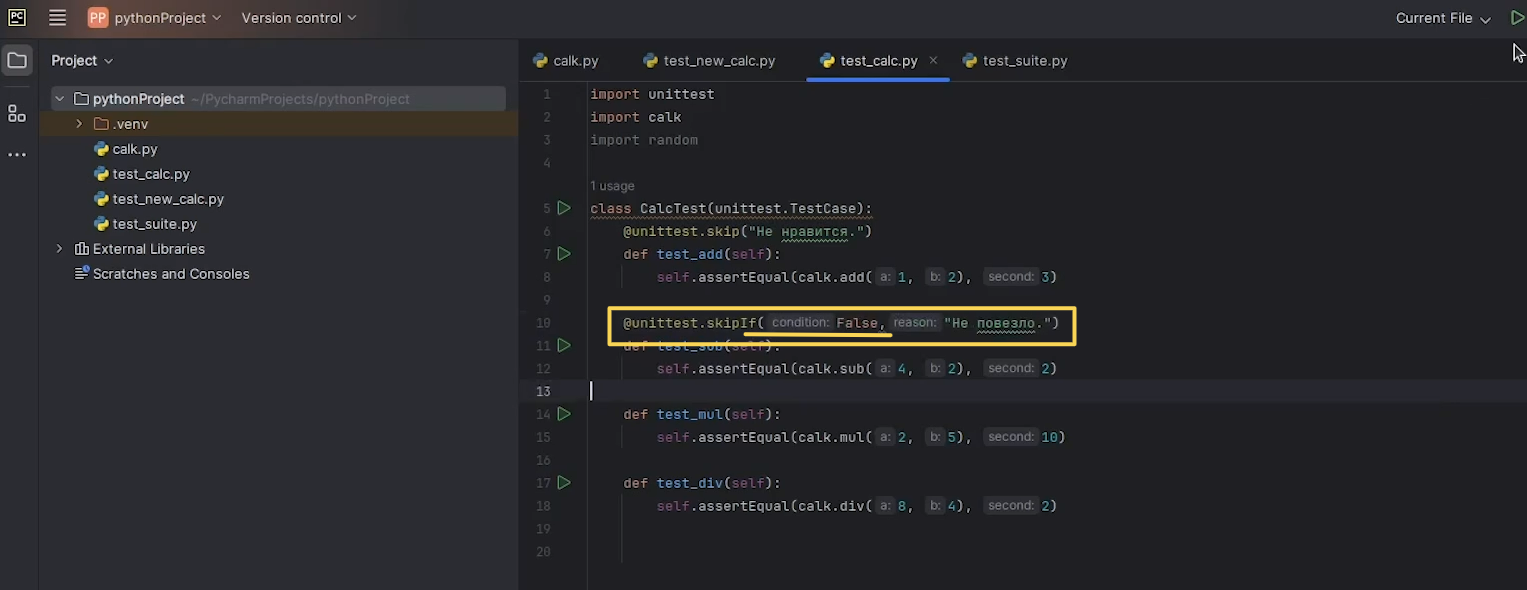


Рис.7

И когда у нас «False», тест прекрасно проходит (Рис.8).

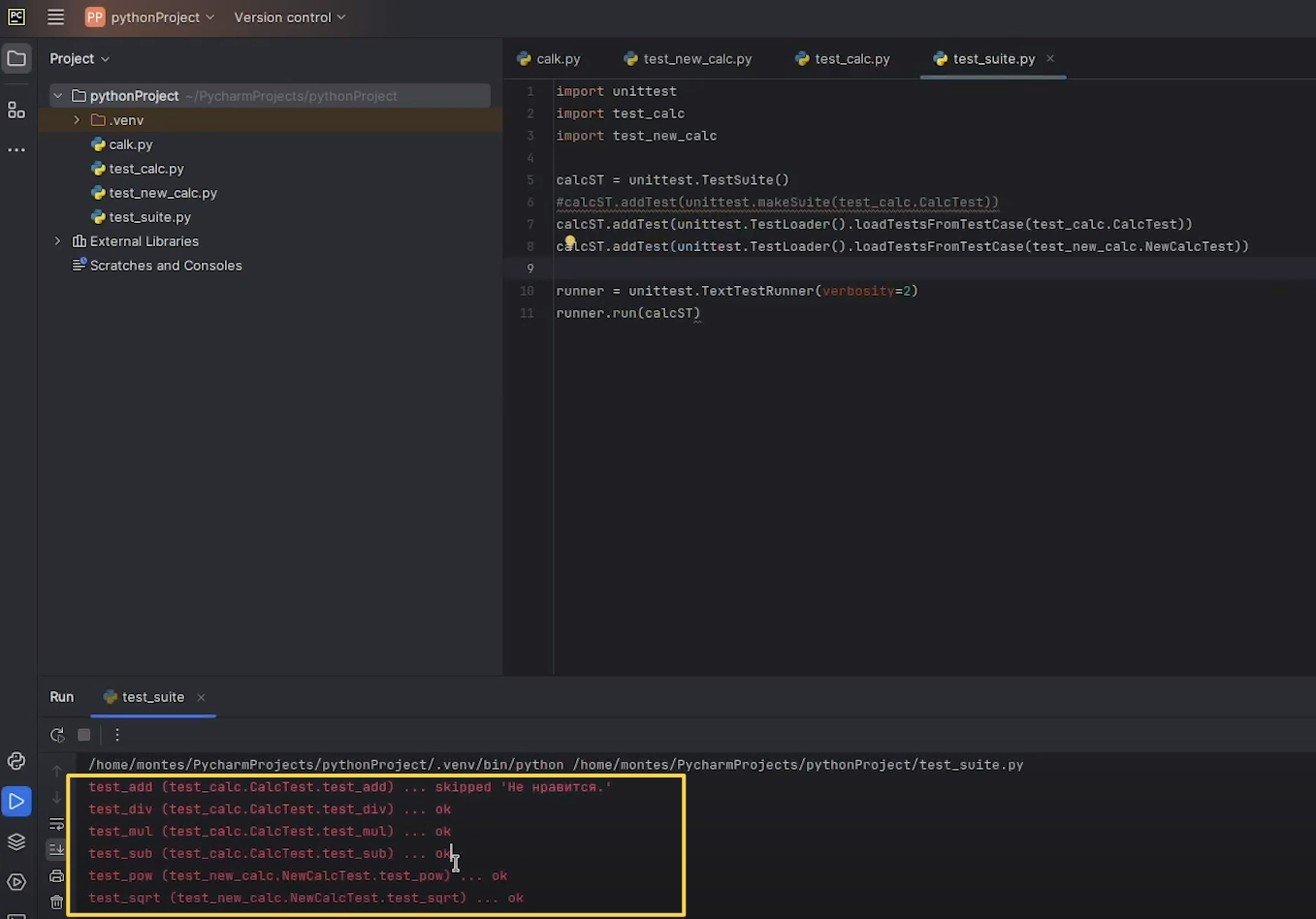


Рис.8

Соответственно, если мы указываем «True», тест не проходит (Рис.9).

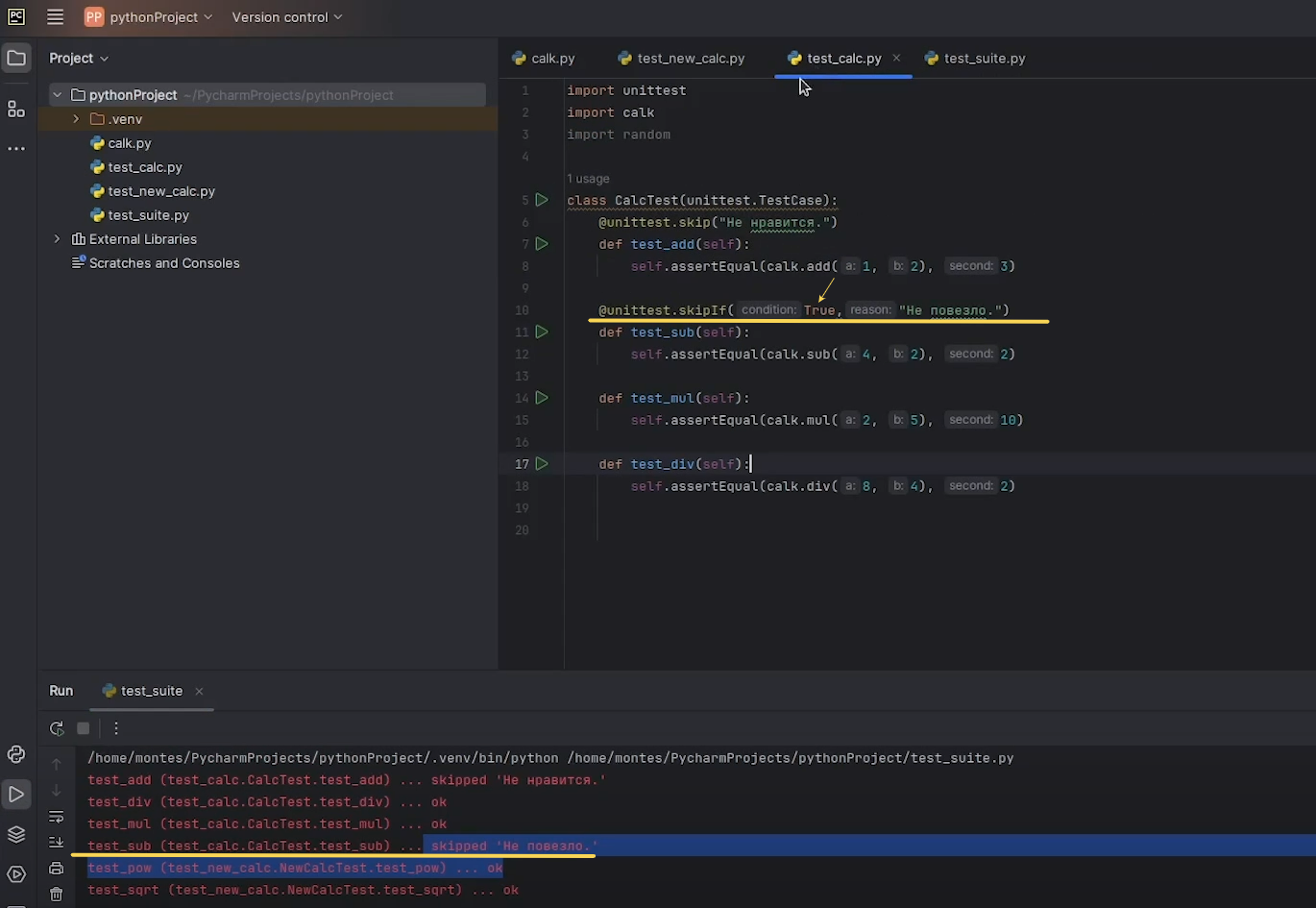


Рис.9

Для чего вообще нужен пропуск тестов? Это нужно для тех случаев, когда мы видим, что у нас есть какой-то большой блок тестов. И если у нас какой-то ключевой тест не пройдён, то зачем нам проходить ещё сотню другую тестов? Мы просто сразу их пропускаем по причине «не пройдён ключевой тест». Зачем это нужно? Сейчас у нас с вами, как видите, 6 тестиков в целом (Рис.10).

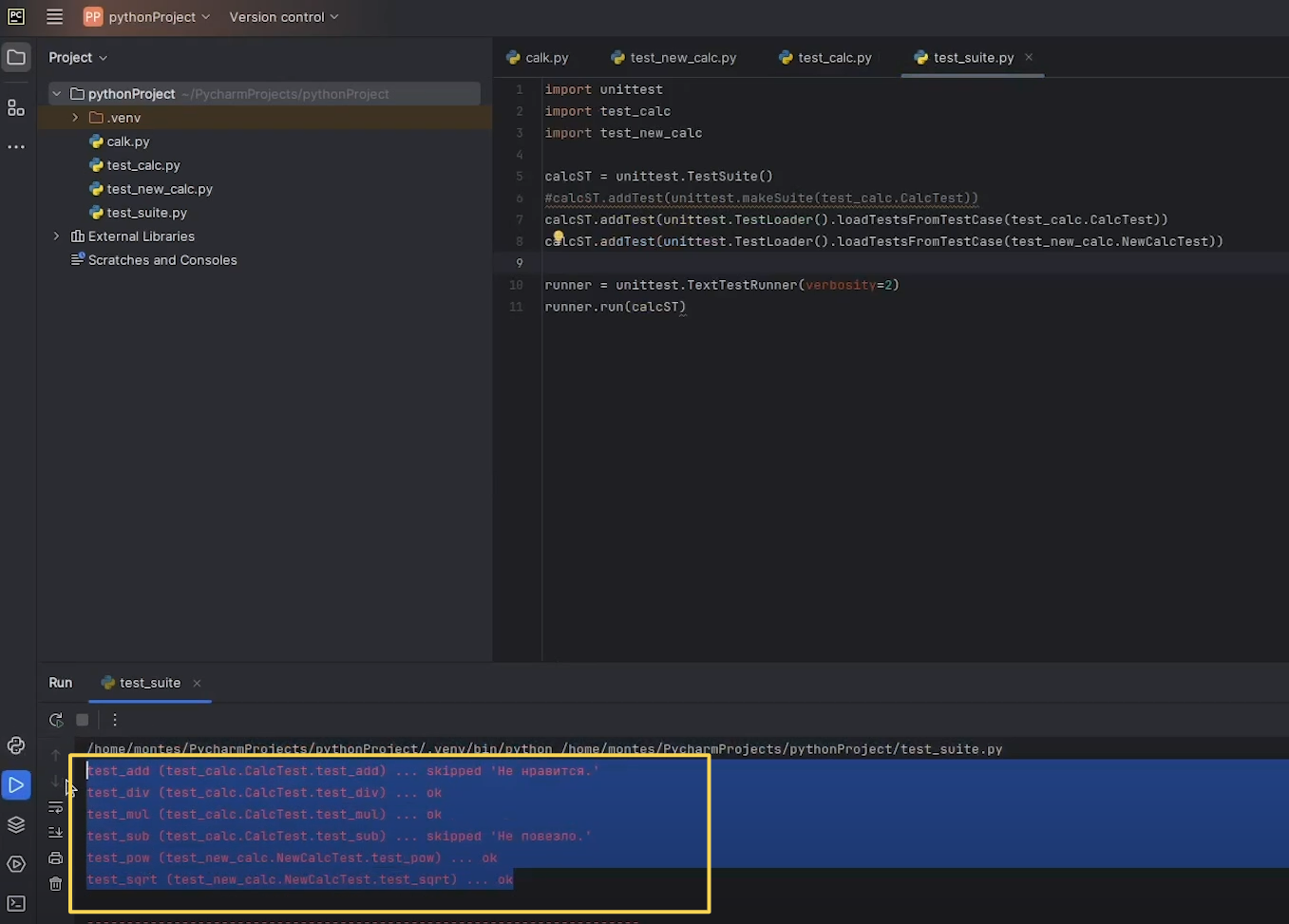


Рис.10

Ничего такого необычного. Но мы можем увеличивать это количество тестов, соизмеримо с размером проектов: уже сотни, тысячи, десятки тысяч. И там проверки могут длиться не такое время, а гораздо больше (Рис.11).



Рис.11

Для того чтобы сэкономить банально время на проверке тестов, эту связь дерева очень удобно выстраивать. То есть, когда у нас верхушка не работает, нижние ветви, мы уже не проверяем. Это, скажем так, система, когда мы делаем некую структуру, в которой у нас есть тесты, которые проверяют какую-то общую работоспособность и потом уже, скажем так, тесты-ветки, которые проверяют корректность работоспособности конкретных функций. Очень удобная система. Используется она в «TDD». Это «test-driven development»-разработка через тестирование. На самом деле «unittests» включаются в этот процесс. Это когда мы с вами разрабатываем некоторую фичу, проверяем её работоспособность, заливаем на сервер. После этого мержим это в единую нашу новую версию. И от новой версии обновляем, и добавляем новые фичи, и снова проверяем. То есть весь процесс разработки обязательно проходит процесс тестирования. Иногда один раз, иногда два раза иногда может быть и больше.

Собственно говоря, мы с вами в течение пяти занятий познакомились с «unittests». Познакомились с тем, как их запускать, как работать с самими тестами, как их компилировать в удобные ячейки-в «test\_suite», как их собирать воедино. И на самом деле этих навыков, которые вы сможете отработать на практике, будет достаточно для того, чтобы в тестах ориентироваться и местами где-то их дописать, а где-то суметь их правильно прочитать. Все-таки вы не тестировщик. В данном случае вы именно разработчик, и вашей прямой обязанностью не является написание большого количества тестов. Иногда, может быть, что-то посмотреть в тестах вам тоже обязательно придётся. Но сами тест-кейсы для вас являются очень полезным прикладным материалом, с которым вы должны быть знакомы.