**2024/01/09 00:00|Лекция. Методы Юнит-тестирования**

## Методы Юнит-тестирования

Общая логика того, что есть в блоке тестирования и какой функционал доступен.

Несколько понятий:

1.**Фикстура**– конкретный тест, т.е. некоторый элемент, который позволяет произвести подготовку для выполнения тестов, так же саму отработку теста и мероприятие по его завершению.

2.**«TestCase»**- набор тестов, который мы имеем в виду. Мы ее воспринимаем как элементарную единицу тестирования, в рамках которой проходят различные фикстуры.

3.**«Test Suite»** - набор тестов, тест-кейсов, который позволяет выполнять системы и их отдельно проверять. Количество таких тест-кейсов не ограничено.

4.**«Test Runner»**- компонент, который отвечает за запуск проверки тестирования и того, как со всем этим взаимодействовать.

Кроме вышеперечисленного доступно еще 3 типа вещей того, что можно сделать с нашим тест-кейсом.

1.Методы, запускаемые при запуске теста

2.Методы, запускаемые после тестов

3.Методы, которые запускаются при написании, проверке теста

Так же есть метод, который позволяет собирать информацию.

Метод «**def setUp**»

Это метод, который производится перед каждым тестированием. Он позволяет создать некую переменную, которая позволяет что-либо изменять. (Рис. 1)

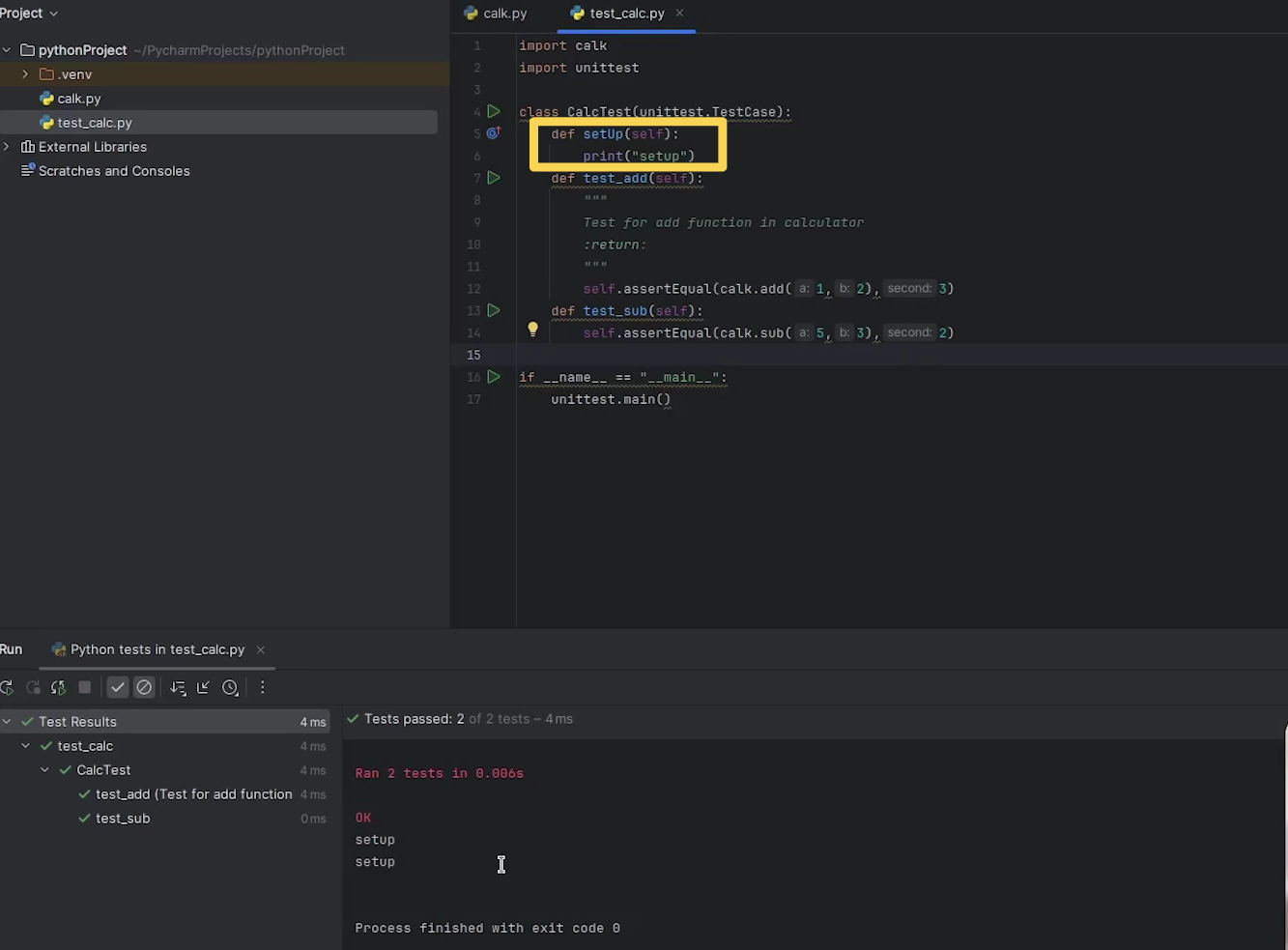


Рис. 1

При каждой отработке теста выводится «**setUp**». **ВАЖНО!**Перед «**Test\_abb**», «t**est\_sub**» производится функция «setUp», в ней можно производить какие-то действия заранее.

### Функция «setUpClass»

Для нее нужно указать декоратор через @, она называется «setUpClass». Напишем «**MegaSetup**». «MegaSetup» происходит в самом начале и 1 раз.

В чем идея?

Есть некий тест-кейс, которому нужно производить некоторые действия, например, в этом методе можно производить процесс подключения к базе данных, т.е. перед тестированием подключаемся к базе данных и перед каждым тестом производим возвращение к стандартному состоянию БД. (Рис. 2)

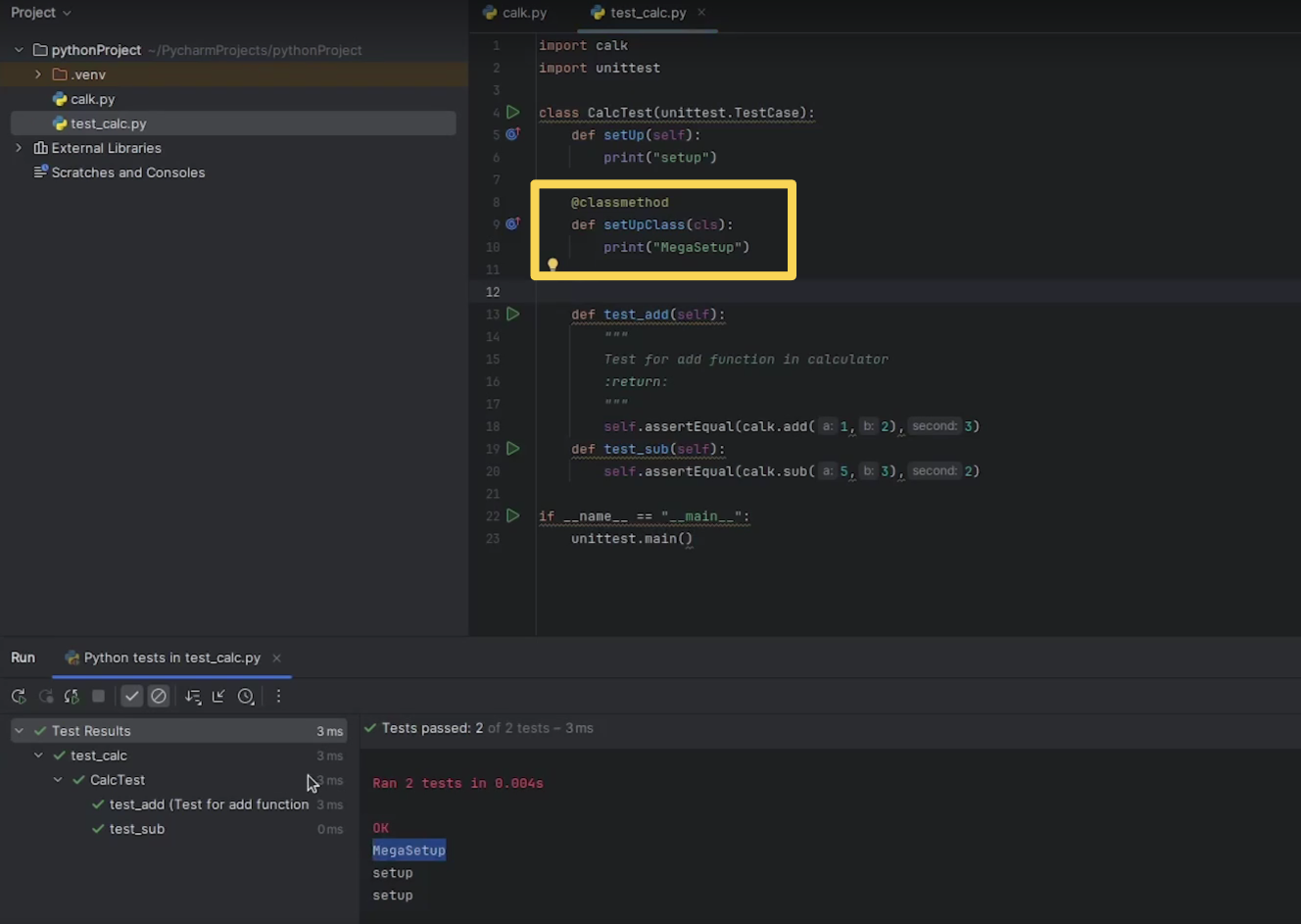


Рис. 2

### «def tearDown»

Аналогичная функция. Запускается после любого тест-кейса.

«**classmethod**» «**tearDownClass**»

Происходит после завершения отработки всех тестов.

Это служебные методы, которые могут в некоторых случаях облегчить какую-либо ситуацию, чтобы перед каждым тест-кейсом не подключались к БД или не обновляли ее каждый раз в каждом тест-кейсе.

Набор других вариантов того, как можно сравнивать.

Лучше заглянуть в документацию.

«**assertEqual**» - достаточно стандартный вариант, который используется для сравнения. (Рис. 3)

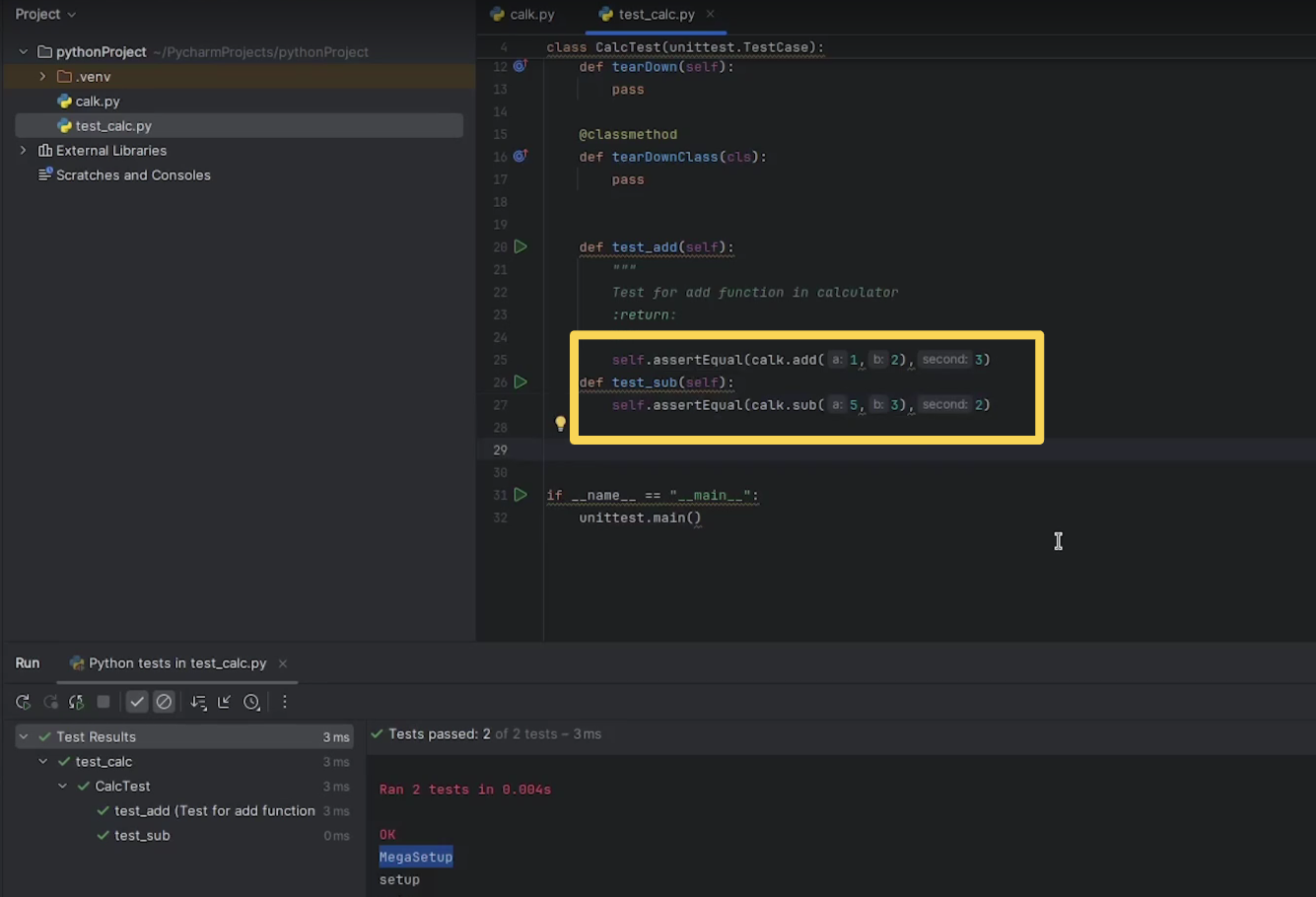


Рис. 3

Есть большое количество методов (assert). (Рис. 4)

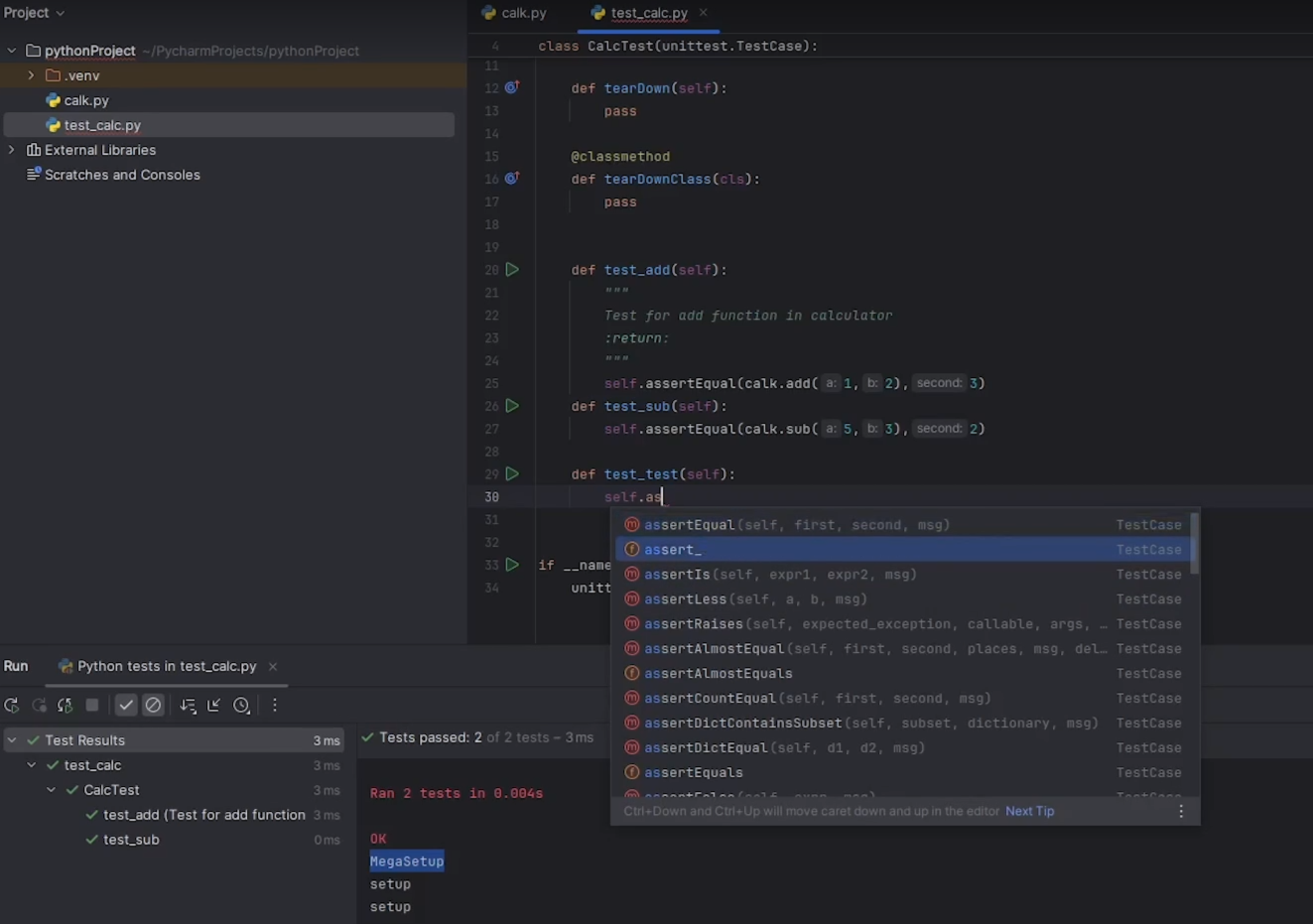


Рис. 4

«**assertTrue**» позволяет проверить равна ли передаваемая переменная. Можем передать некую логическую переменную. (Рис. 5)

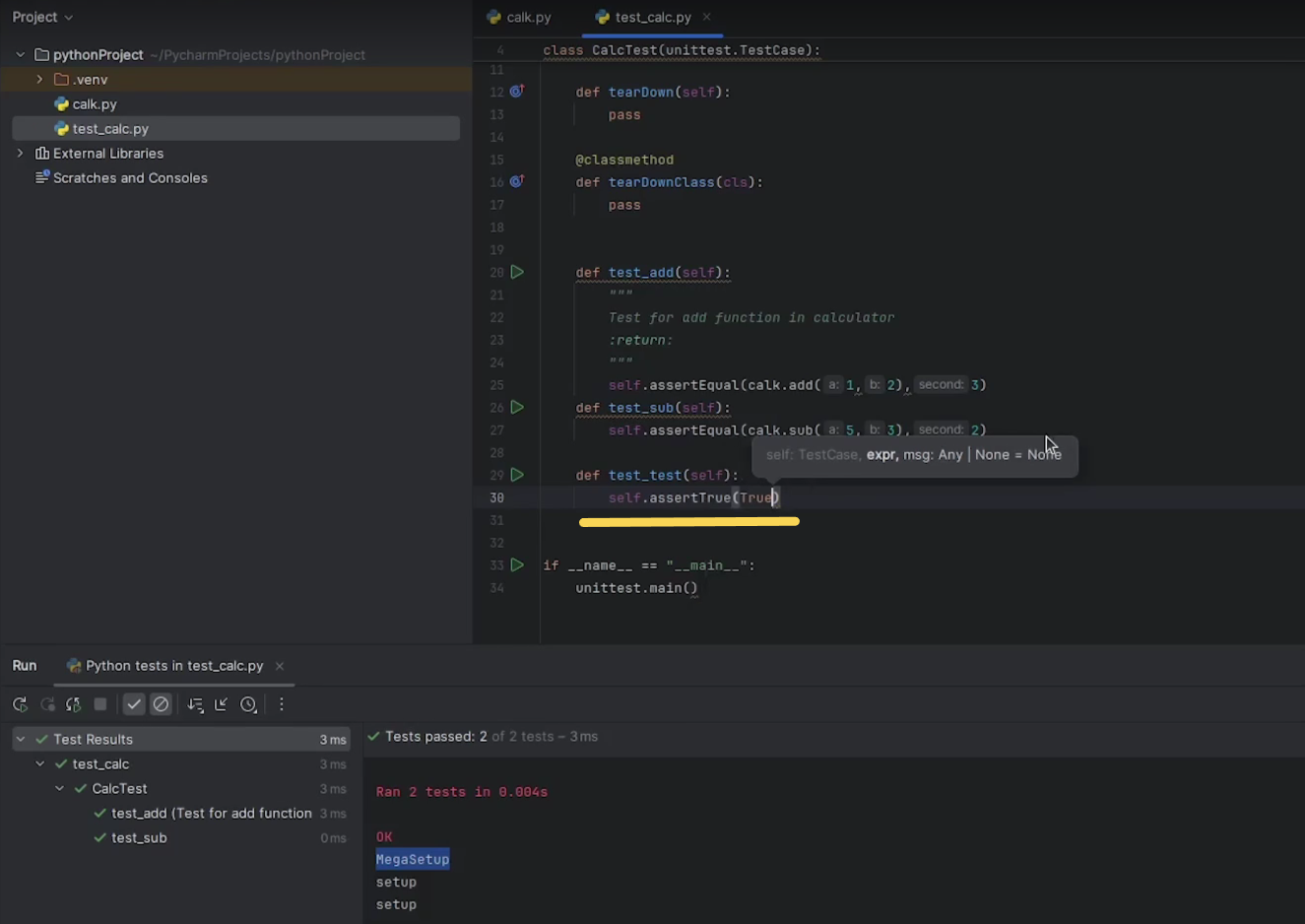


Рис. 5

1.«**assertFalse**» проверяет не равно ли значение «False».

2.«**assertIs**» позволяет проверять через оператор «Is» является ли «a, b» «a Is b».

3.«**assertIsNone**» проверяет не равна ли переменная пустому значению. Если она пустая, то вернется «True», если не пустая, то «notNone» (False).

4.«**assertIn**» - проверка наличия, подходит для строк. Проверяем есть ли строка «a» в строке «b». Если вы что-то забыли и неудобно открывать документацию, наведите мышку, посмотрите что это делает. (Рис. 6)

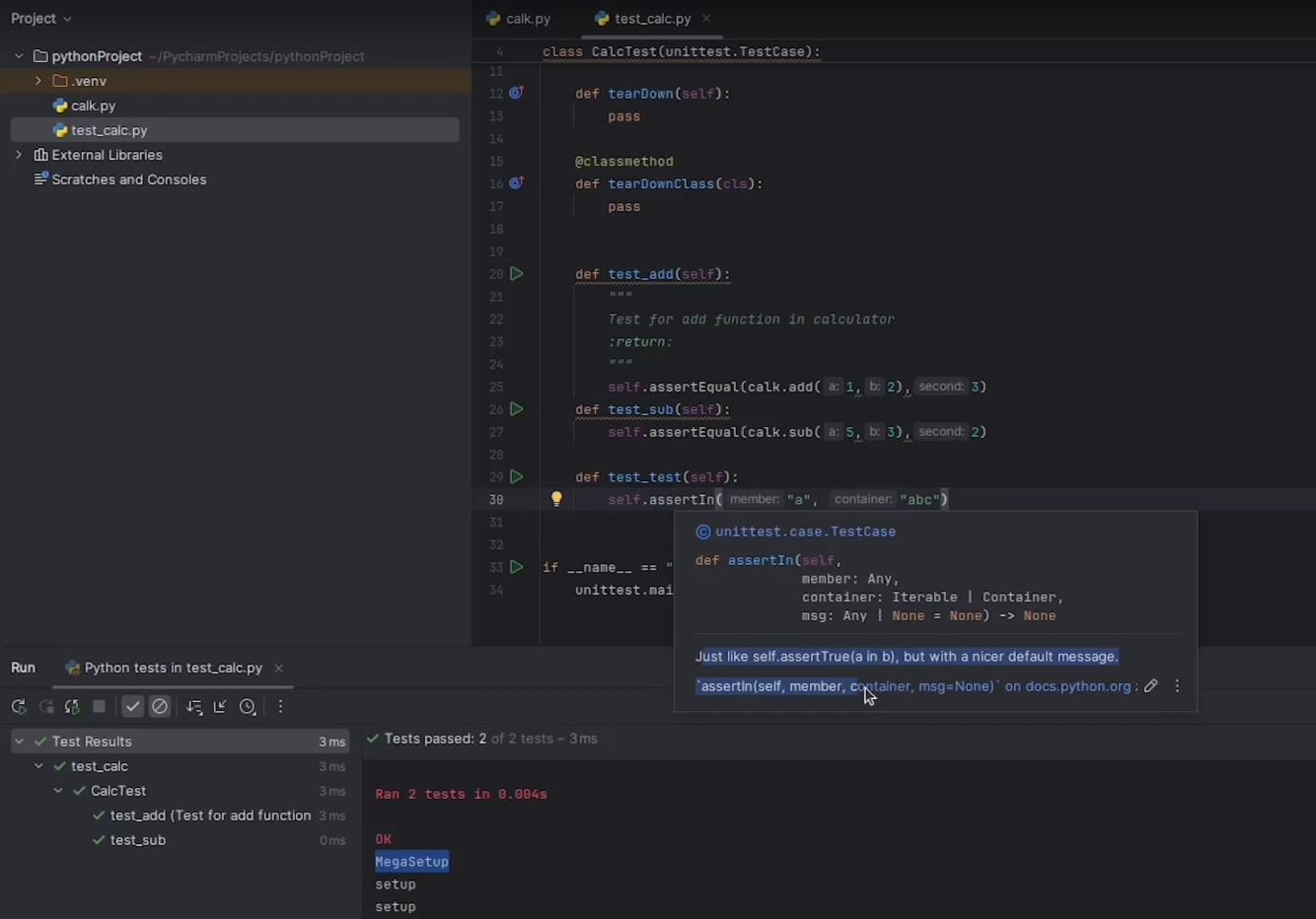


Рис. 6

**«assertRaises»**позволяет проверять была ли вызвана какая-либо ошибка, например, деление на 0.

**«assertWarnings»**(ворнинги) можно их получать и проверять не появились ли они. Для них есть доп. функция через регулярные выражения.

**«assertAlmostEquals»** - функция, которая позволяет сравнить 2 числа с точностью до 7 знака после запятой. Это нужно для сравнения вещественных чисел. Виду специфики хранения вещественных чисел, они могут отличаться, например, на 25 знаке, в зависимости от того, как они хранятся. (Есть так же **«assertAlmostNotEquals»**)

**«assertGreater», «assertNotGreater», «assertGreaterEqual», «assertLess», «assertLessEqual», т.е. больше, больше либо равно, меньше, меньше либо равно.**