**2024/01/07 00:00|Лекция. Идея Юнит-тестирования**

## Идея Юнит-тестирования

Представим, что есть калькулятор или любое другое более сложное приложение, нужно понять работает ли приложение. Проверить это достаточно легко. Можно вызвать любую из функций, проверить, что она работает корректно.

Проблема, с которой мы сегодня столкнемся, чуть более обширная. Представим, что загружаем приложение и через некоторое время вы или сокомандник обновляет приложение и тем самым затрагивает логику функции. Проблема в том, что обновление может вызвать какое-то изменение работы или ее сломать, поэтому нужно проводить постоянную проверку работы функций.

**Как проводить проверку?**

Самое простое все проверять, но это достаточно долго. Рассмотрим несколько вариантов как автоматизировать этот процесс.

Создадим новый файл, который будет привязан к калькулятору и импортируем туда калькулятор. Это позволит подключить функцию и можно его использовать. (Рис. 1, 2)

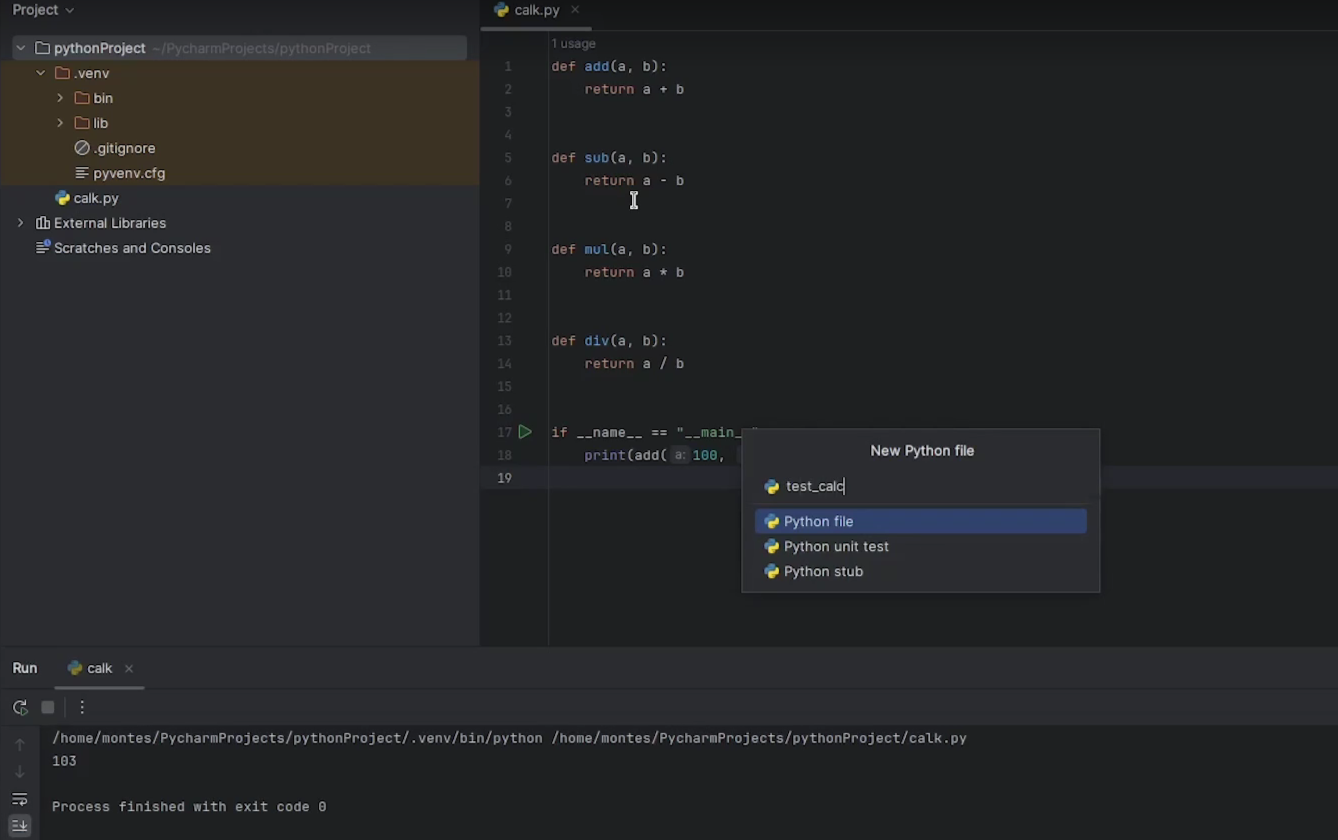


Рис. 1

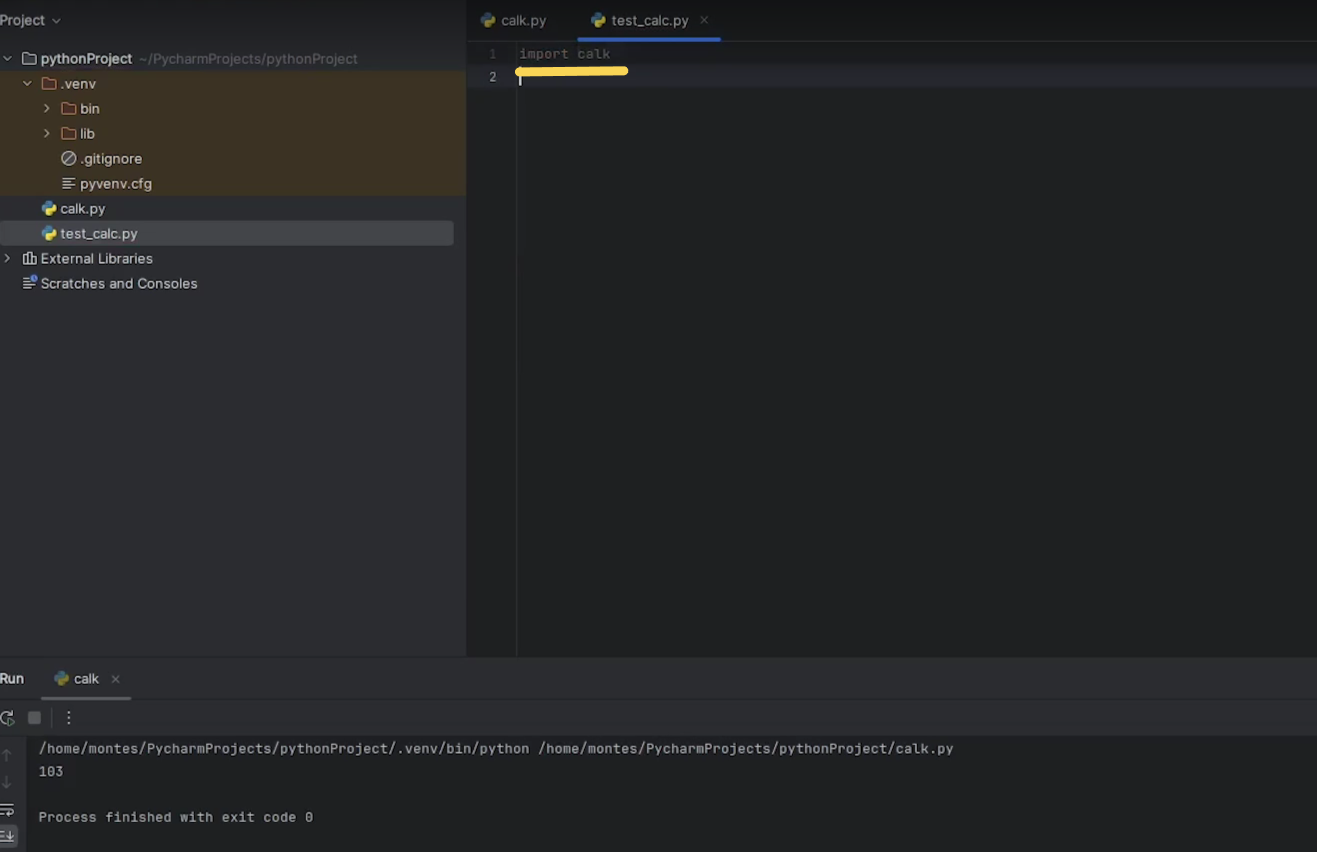


Рис. 2

Создадим специальную функцию, которая будет проверять работу некой функции, например «**test\_add**» будет проверять функцию сложения. Добавляем проверку. Вызовем «**calk.add**», даем (a: 1, b: 2) и проверяем равно ли значение этой функции. Если сумма равна 3, то можно напечатать, что «Test add (a,b) is OK». (Рис. 3)

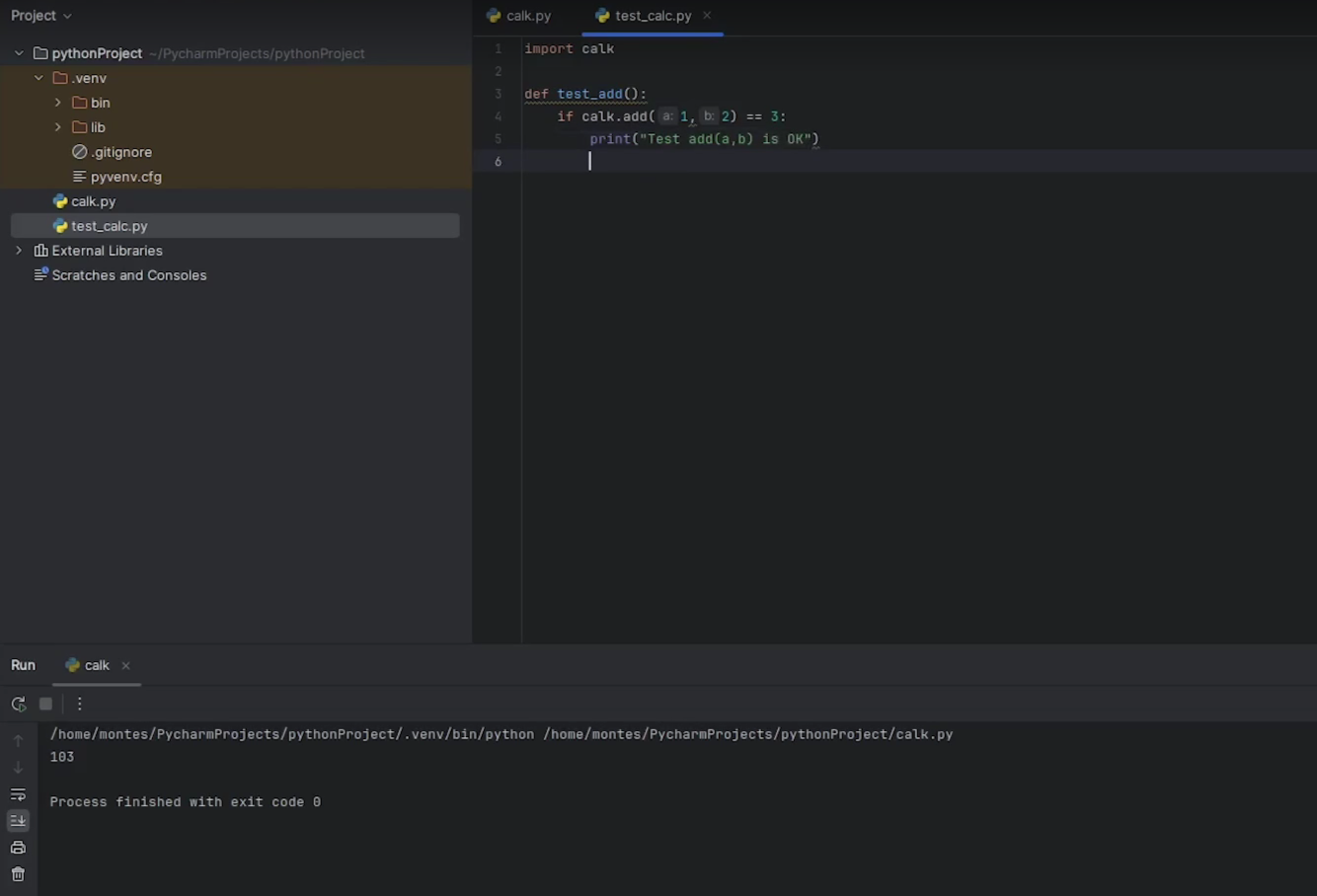


Рис. 3

Если результат не равен 3, то печатаем, что тест не пройден. (Рис. 4)

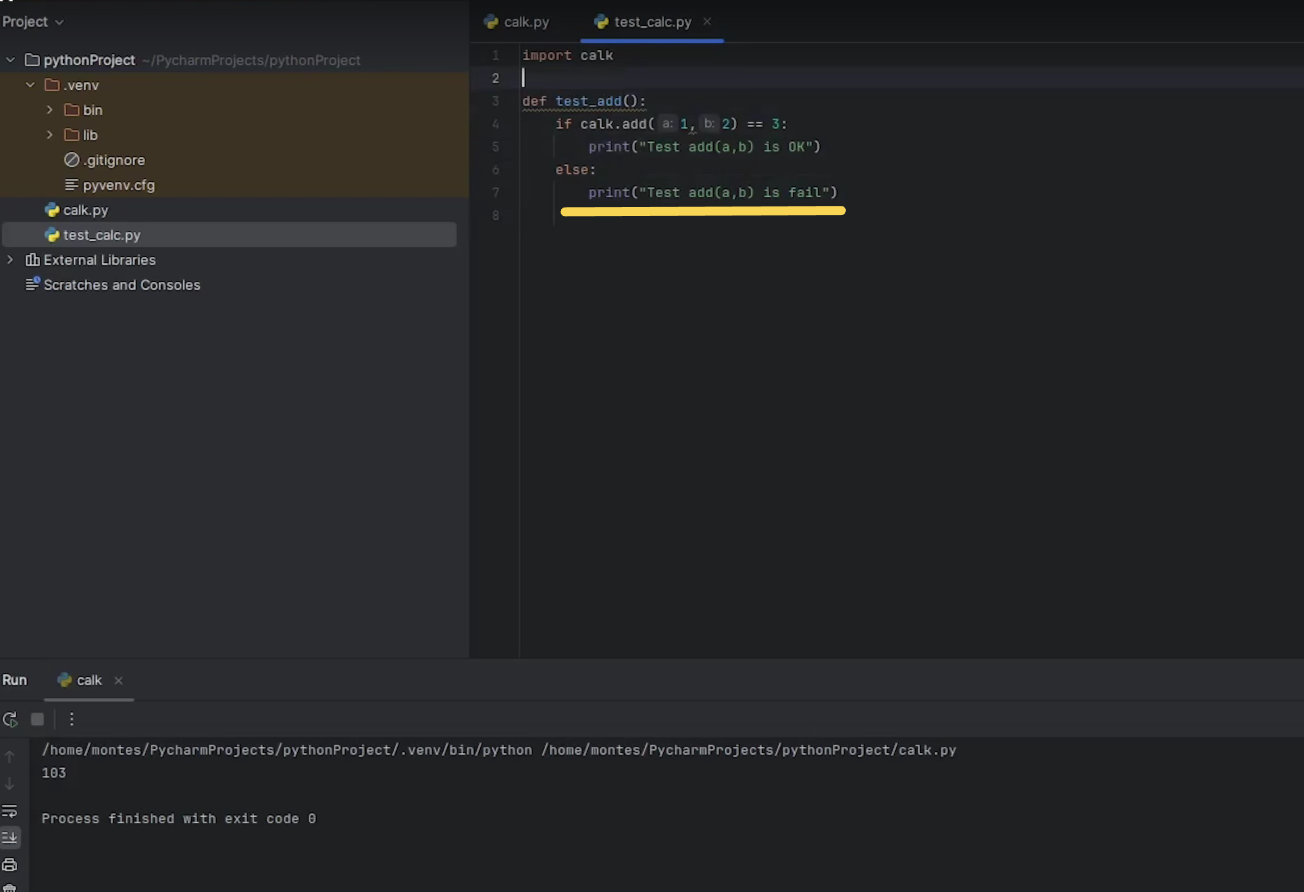


Рис. 4

Запустим и увидим, что функция работает. (Рис. 5)

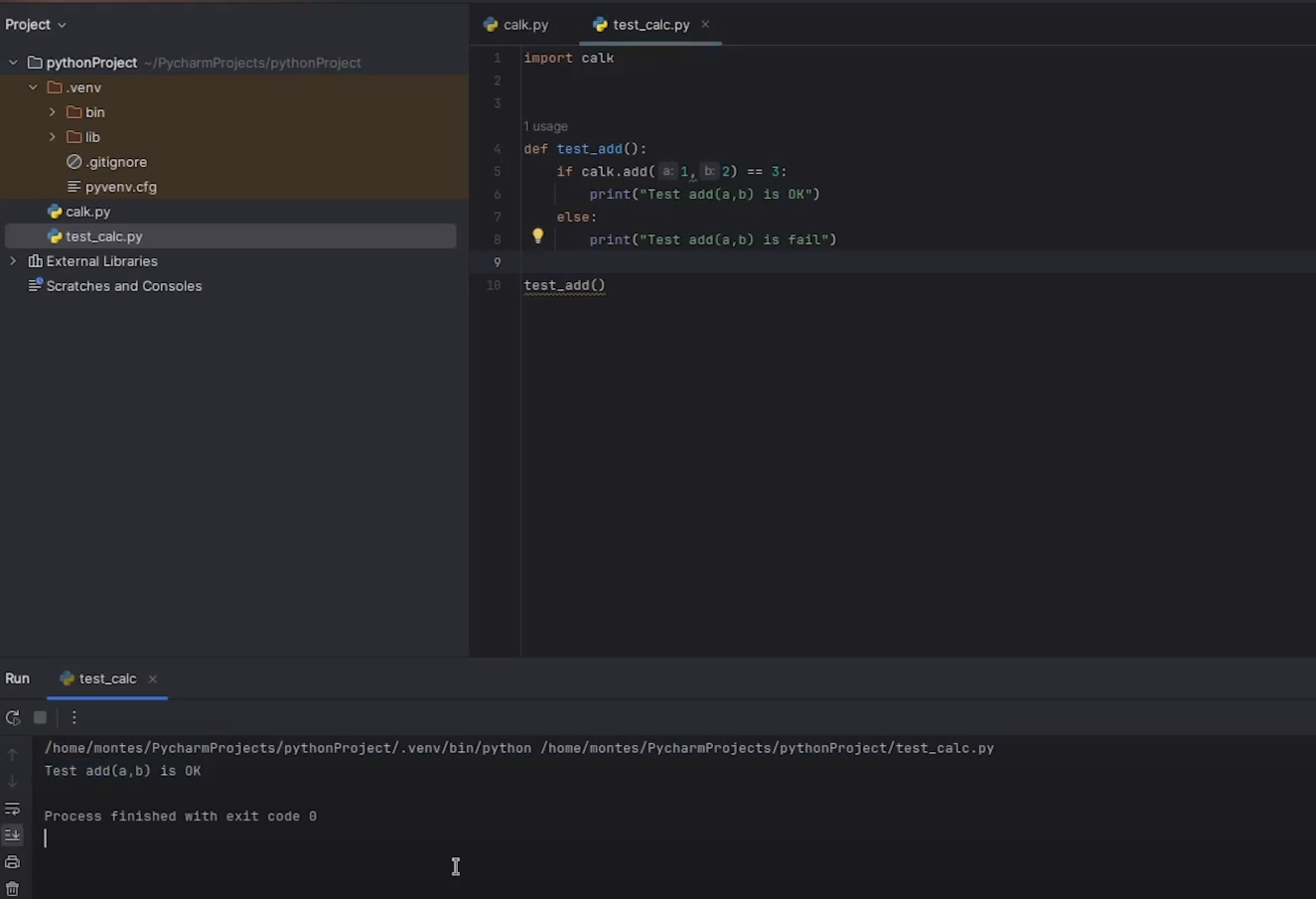


Рис. 5

**Для чего нужно?**

На тот случай если потом кто-то или вы напишете новую функцию и решите, что она должна, например, не складывать, а складывать квадрат. Получим, что функция будет возвращать сумму квадратов и результат сложения 1 и 2 не будет равен 3. (Рис. 6, 7)

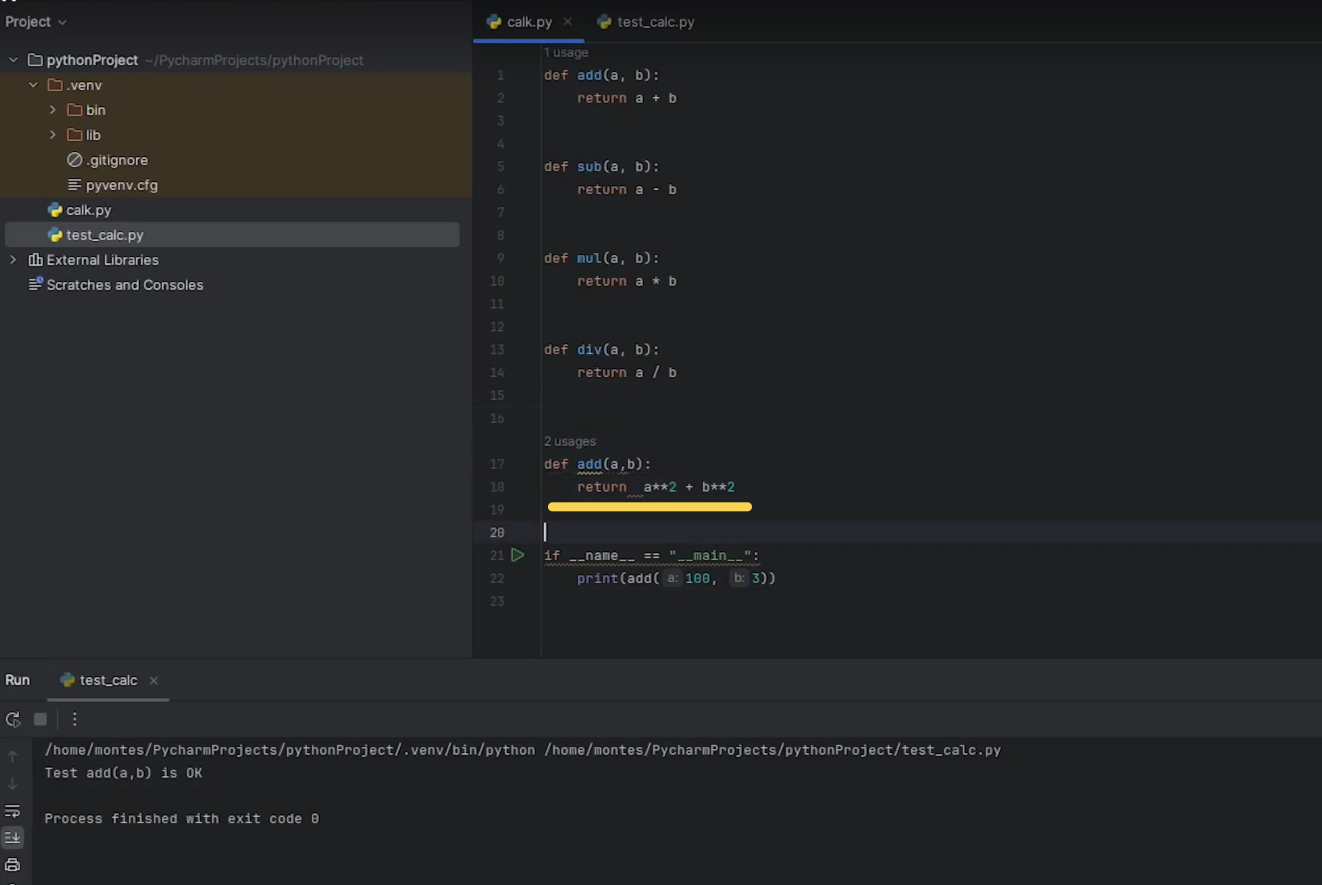


Рис. 6

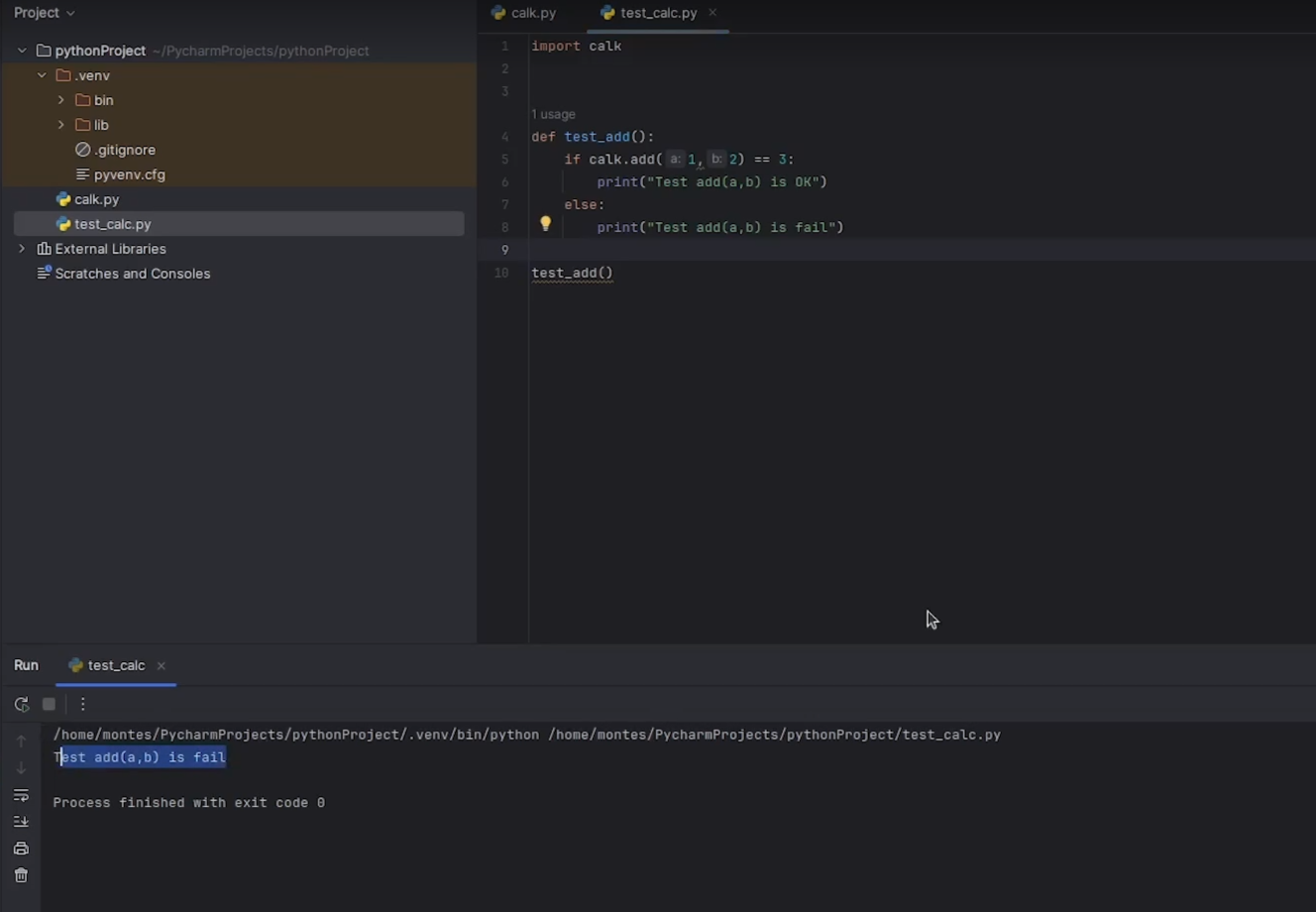


Рис. 7

Этот набор тестов можно заключить в специальную небольшую программу, которая будет запускаться при проверке.