**2024/01/16 00:00|Лекция. Асинхронность на практике.**

## Асинхронность на практике.

Основное, что дает **асинхронность** – возможность распараллеливать удобно процессы так же, как и мультипоточность, но немного по-другому.

Представим, есть функция, у которой есть 2 функции. Первая выполняется 2 секунды, вторая 4 секунды. Например, измеряем 2 состояния. Запускаем. Специально **снимаем показания датчиков времени** чтобы можно было понять сколько времени работает подобная функция. (Рис. 1)

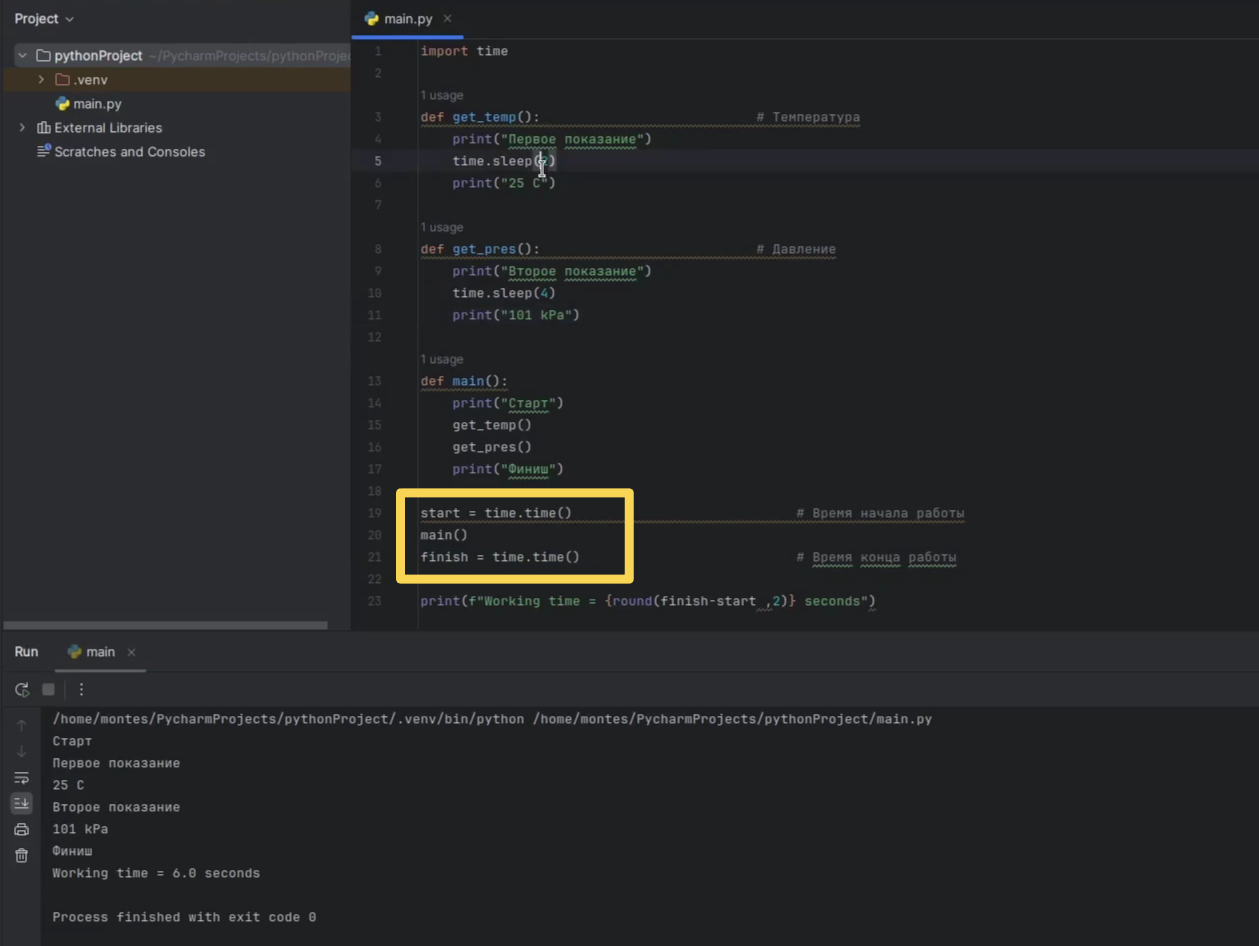


Рис. 1

Если бы не тайм-слипы, время было бы 0 секунд.

У нас есть функция, которая работает 2 секунды и 4 секунды, а время работы 6 секунд. Гипотетически можно сократить работу функции до 4 секунд. Ради этого нам и нужны асинхронности.

Делаем все, что можно асинхронным. (Рис. 2)

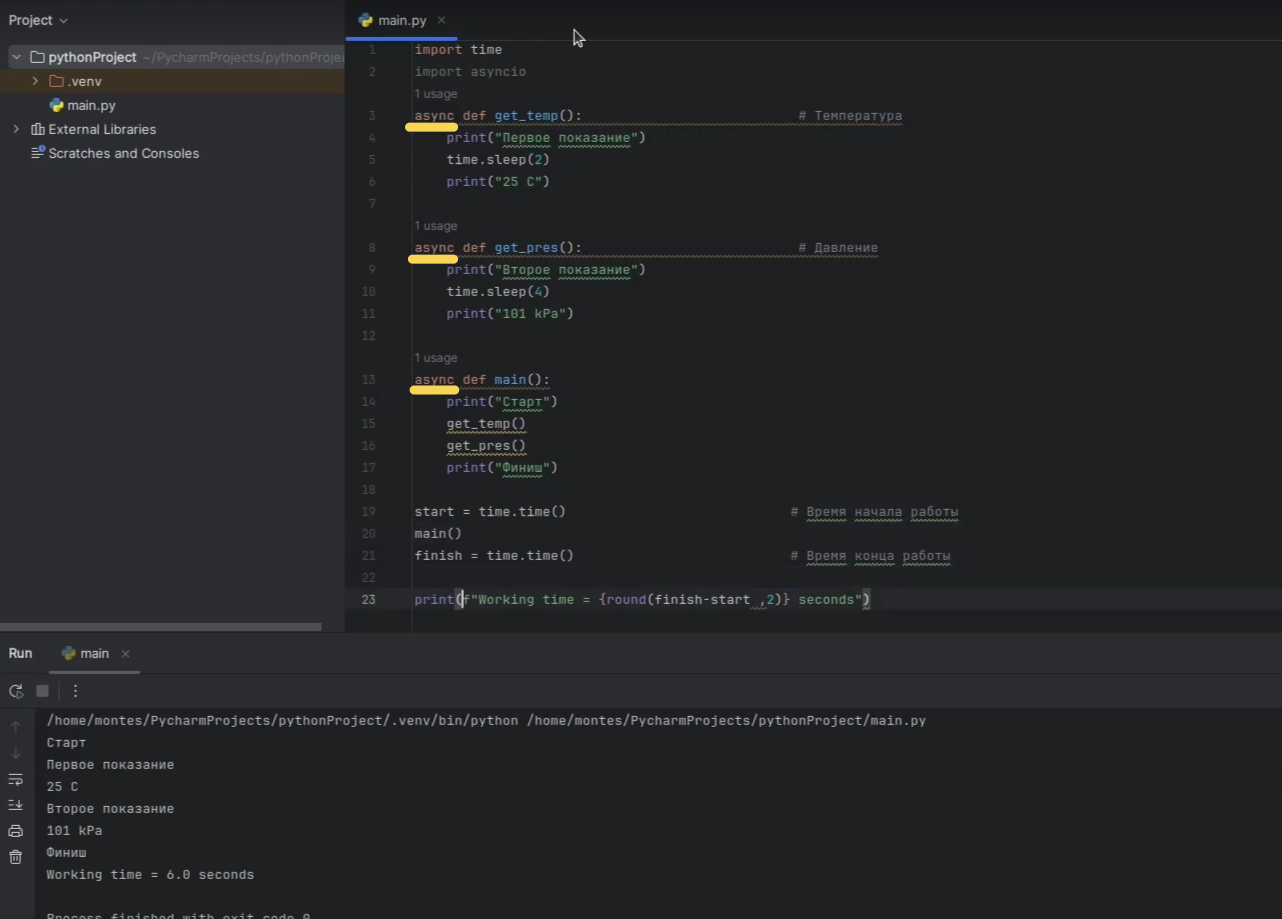


Рис. 2

В «**main**» запускаем через «**asyncio.run**». (Рис. 3)

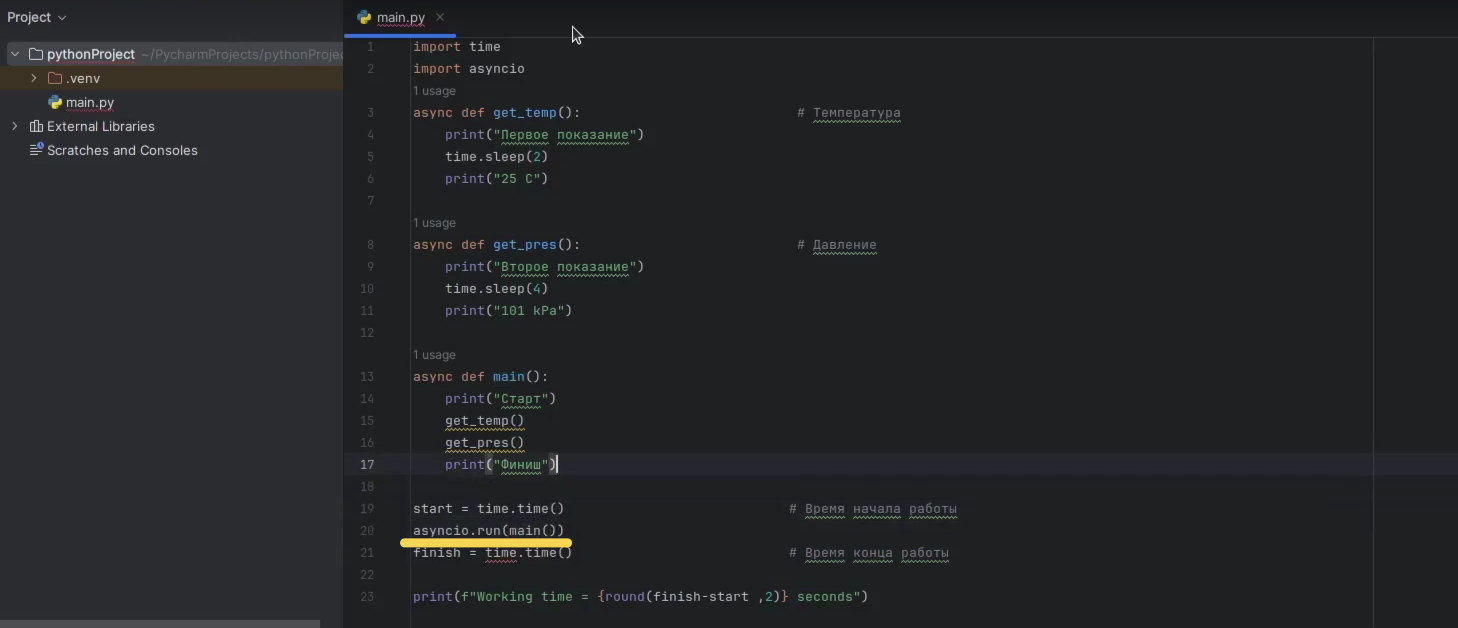


Рис. 3

Формируем «**task**», «**asyncio.create\_task**», упаковываем. Со второй делаем точно так же. (Рис. 4)

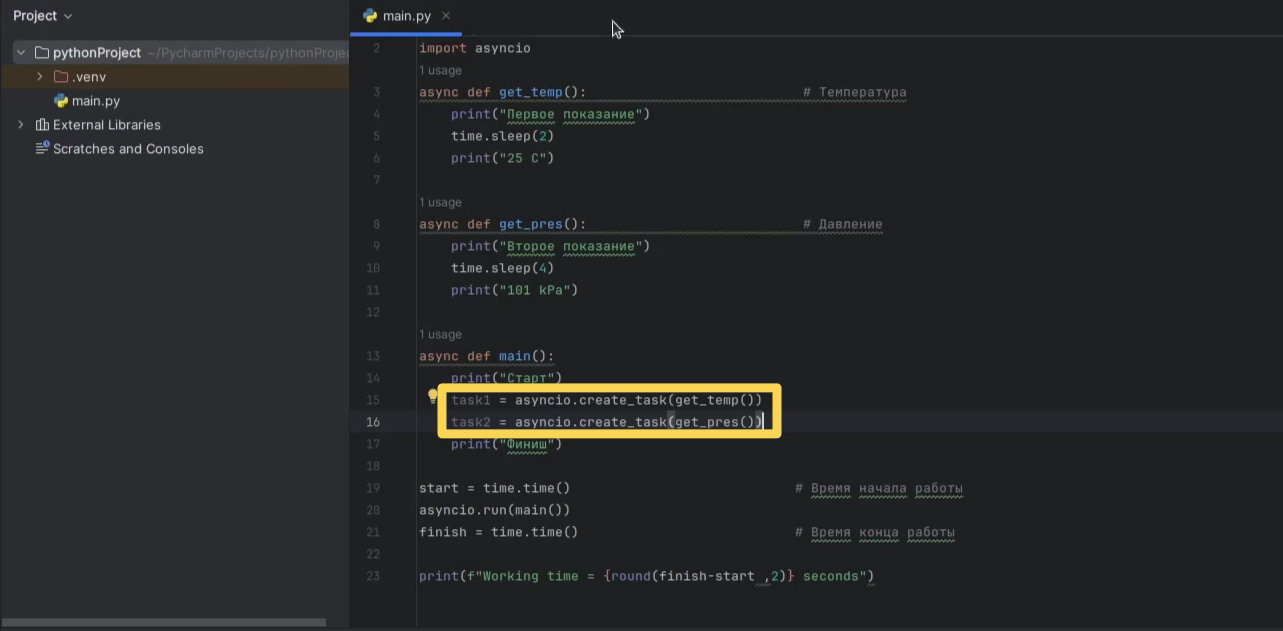


Рис. 4

Попробуем запустить. (Рис. 5)

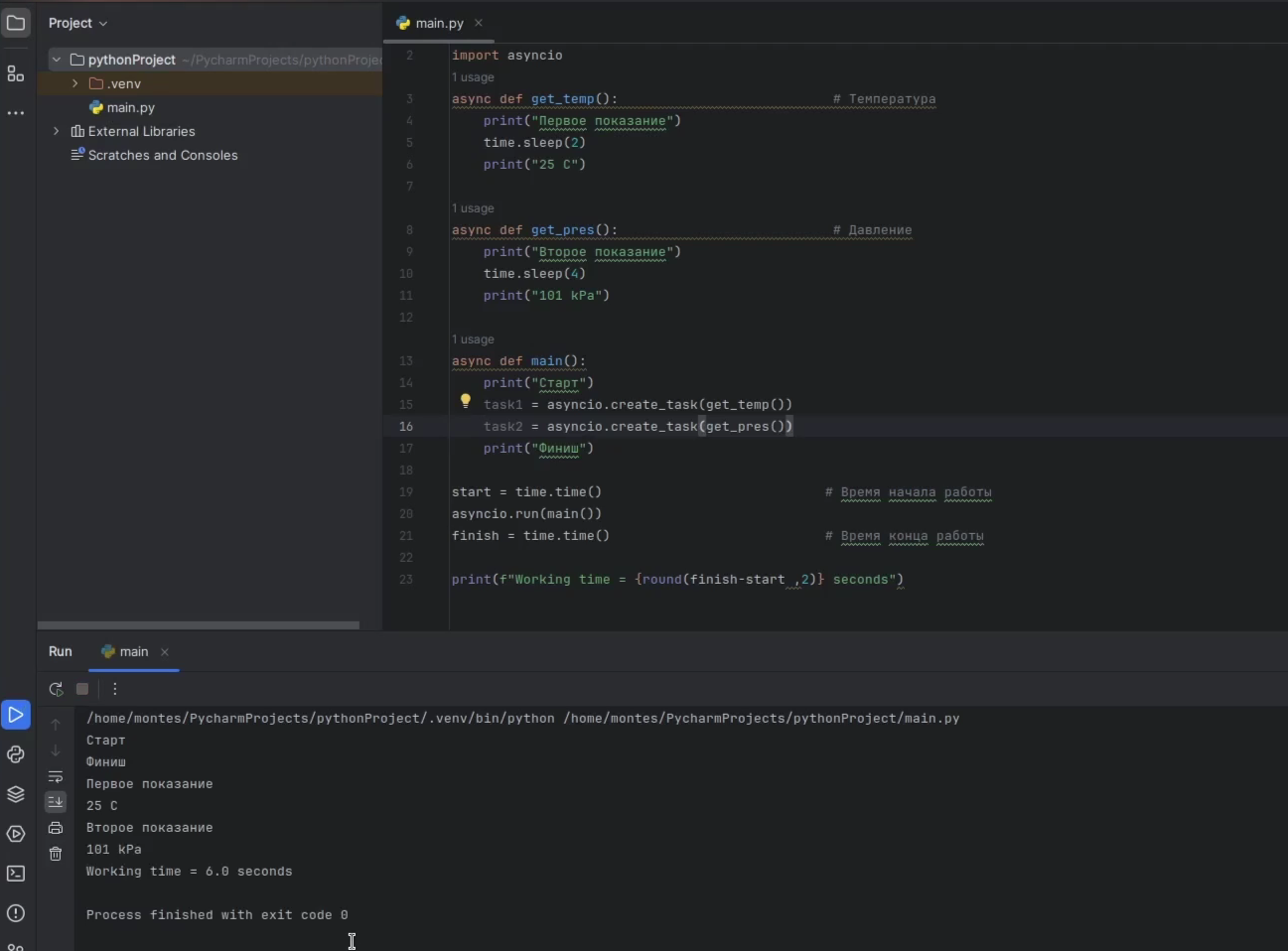


Рис. 5

**Ничего не поменялось** и дело в тайм-слипах. Если бы мы считали просто математические вычисления, сейчас бы все получилось, но тайм-слип тормозит все потоки, поэтому можем сделать «**asycio.sleep**». (Рис. 6)

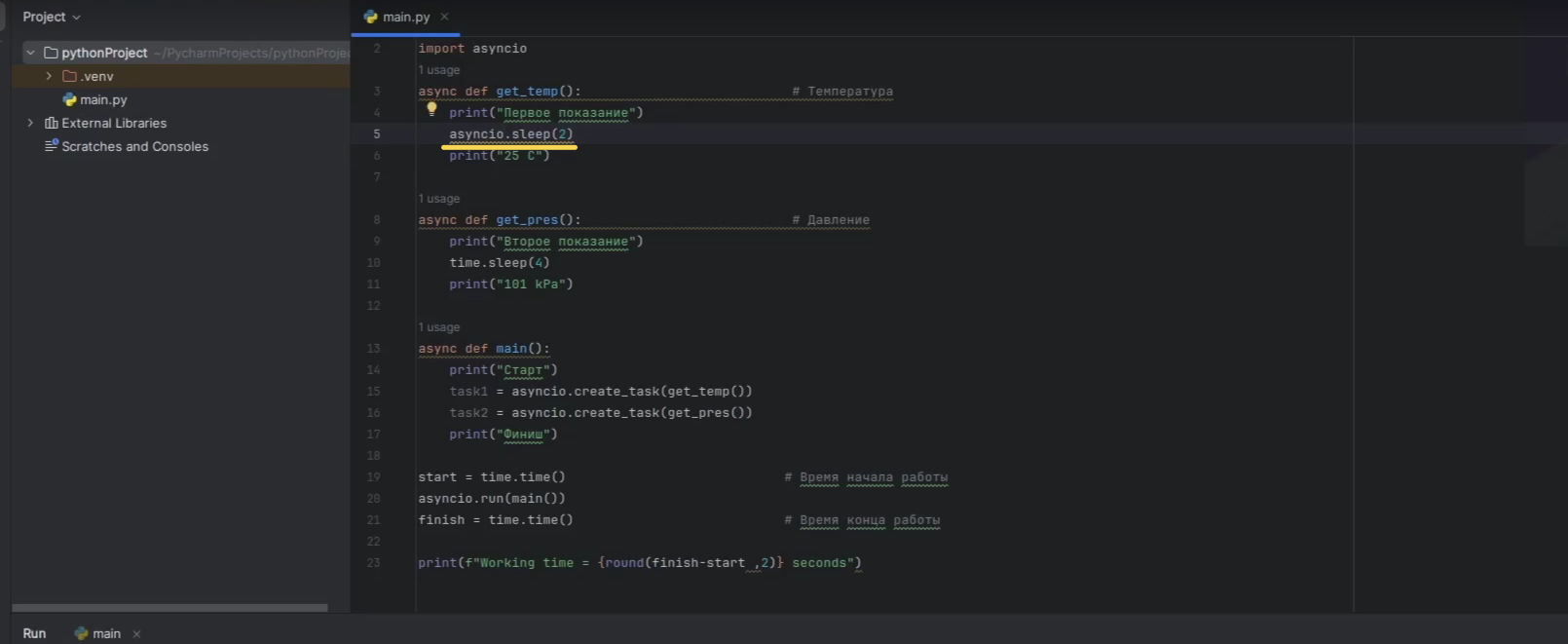


Рис. 6

Если посмотреть на эту функцию, то появится подсказка, что **«sleep» не был «awaited».** Исправим это. (Рис. 7)

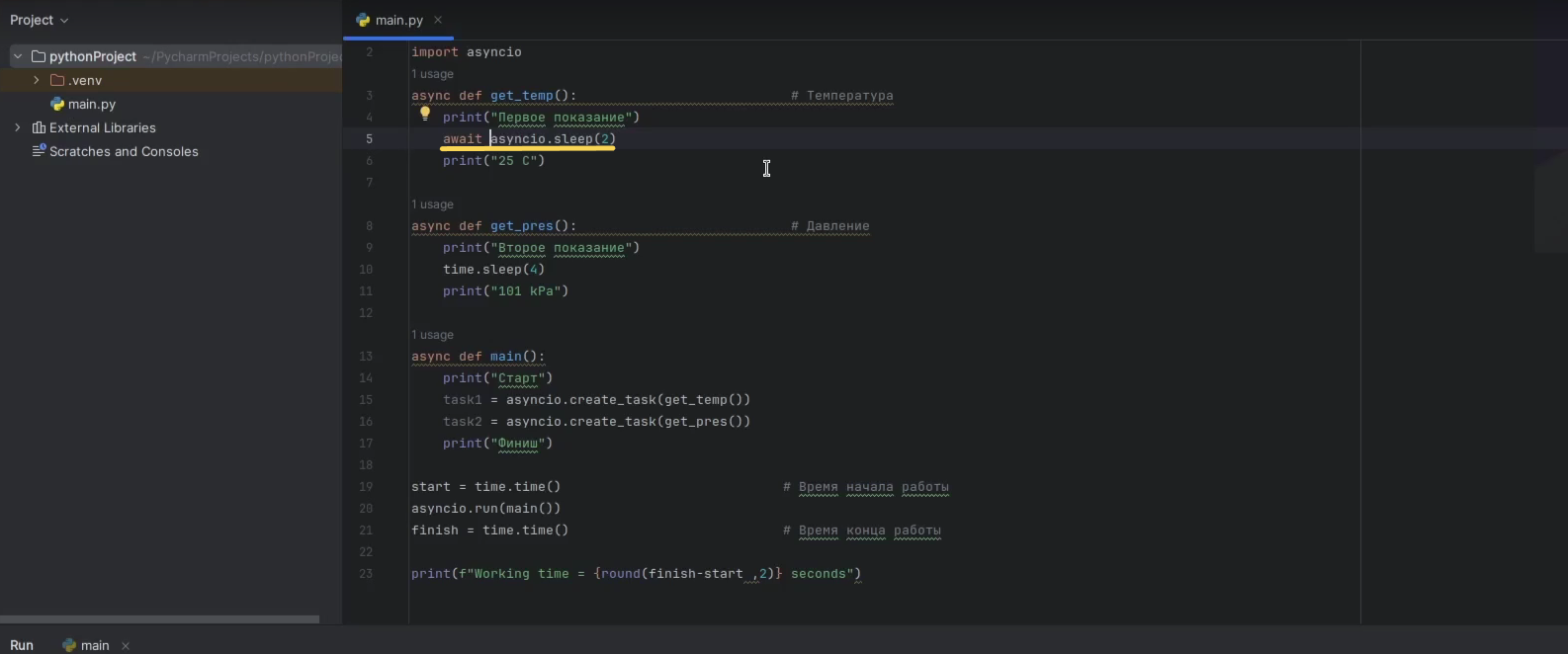


Рис. 7

### В чем разница между «awaited» и «task»?

«**task**» запускается и мы движемся дальше. В «**awaited**» пока не выполнится функция, дальше не пойдем. (Рис. 8)

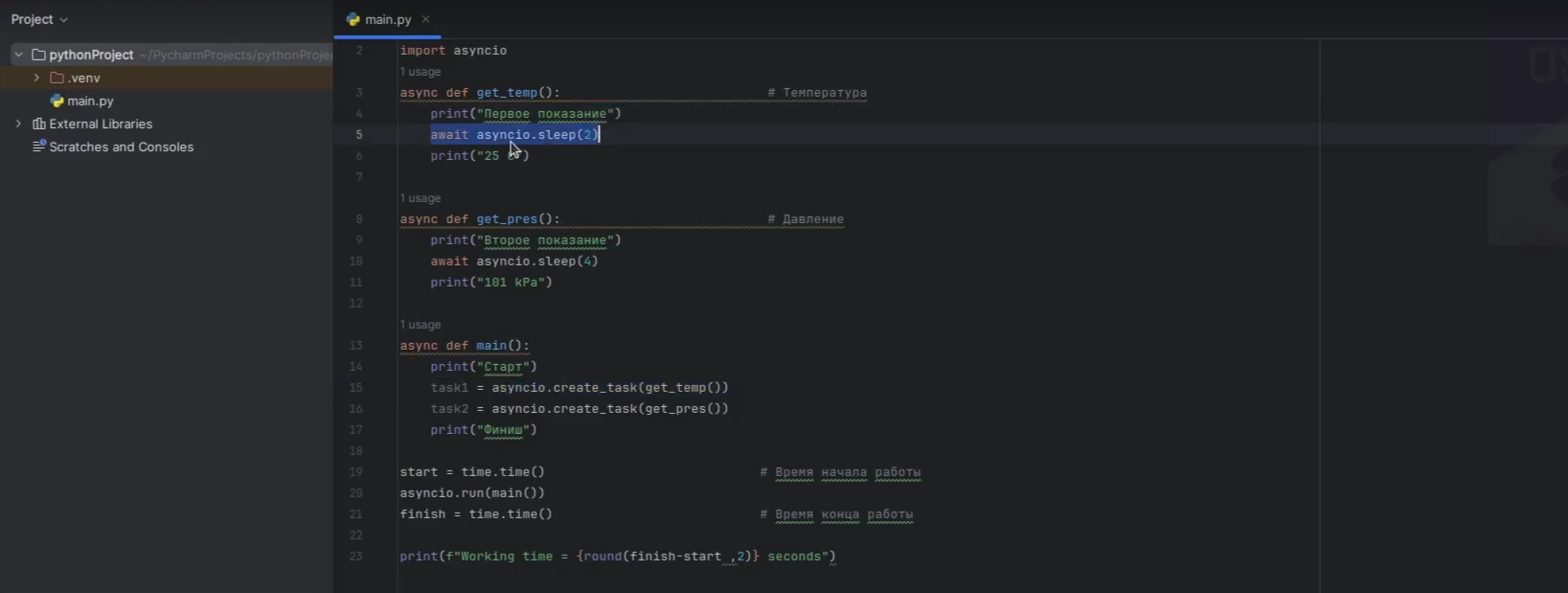


Рис. 8

Запустим. (Рис. 9)

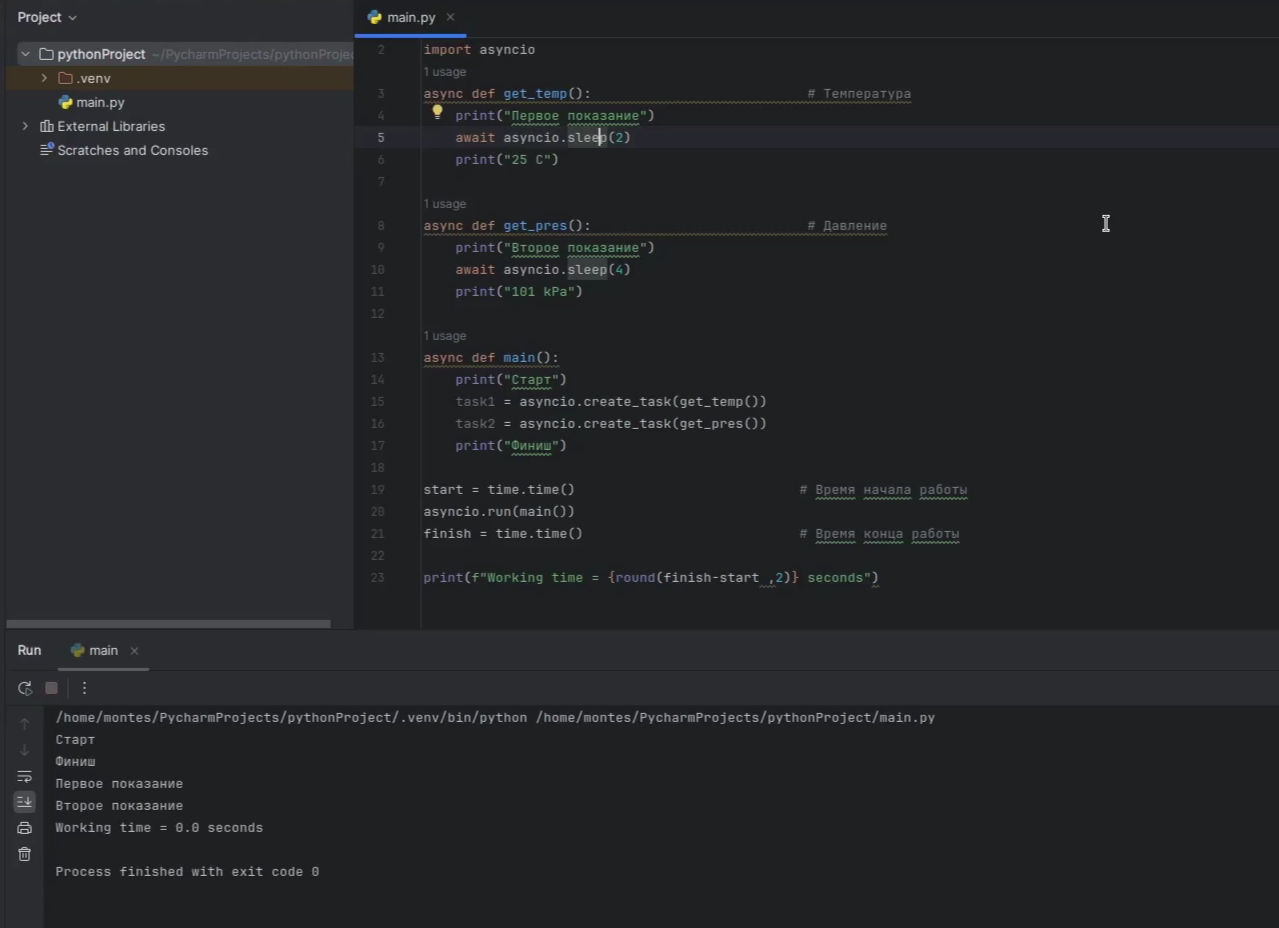


Рис. 9

Получили показание в 0 секунд, это не очень хорошо.

Нужно прописать **ожидание «task1» и «task2» в «awaited».** (Рис. 10)

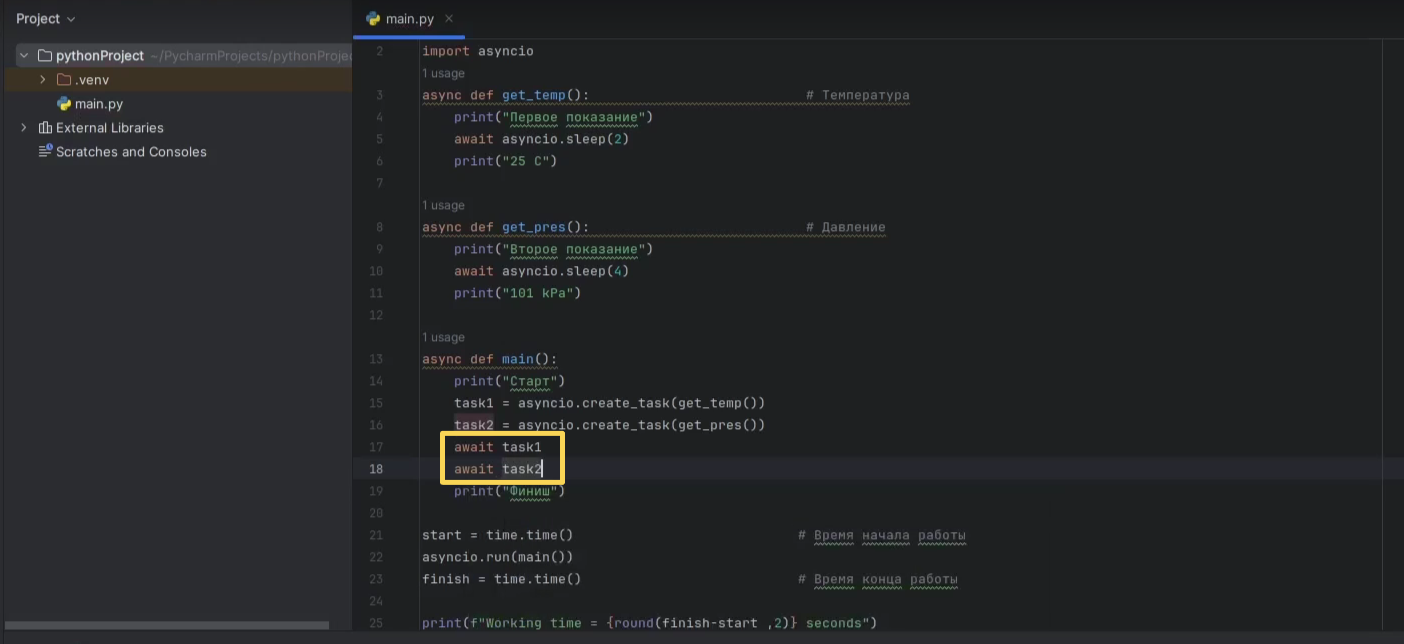


Рис. 10

Запустим. (Рис. 11)

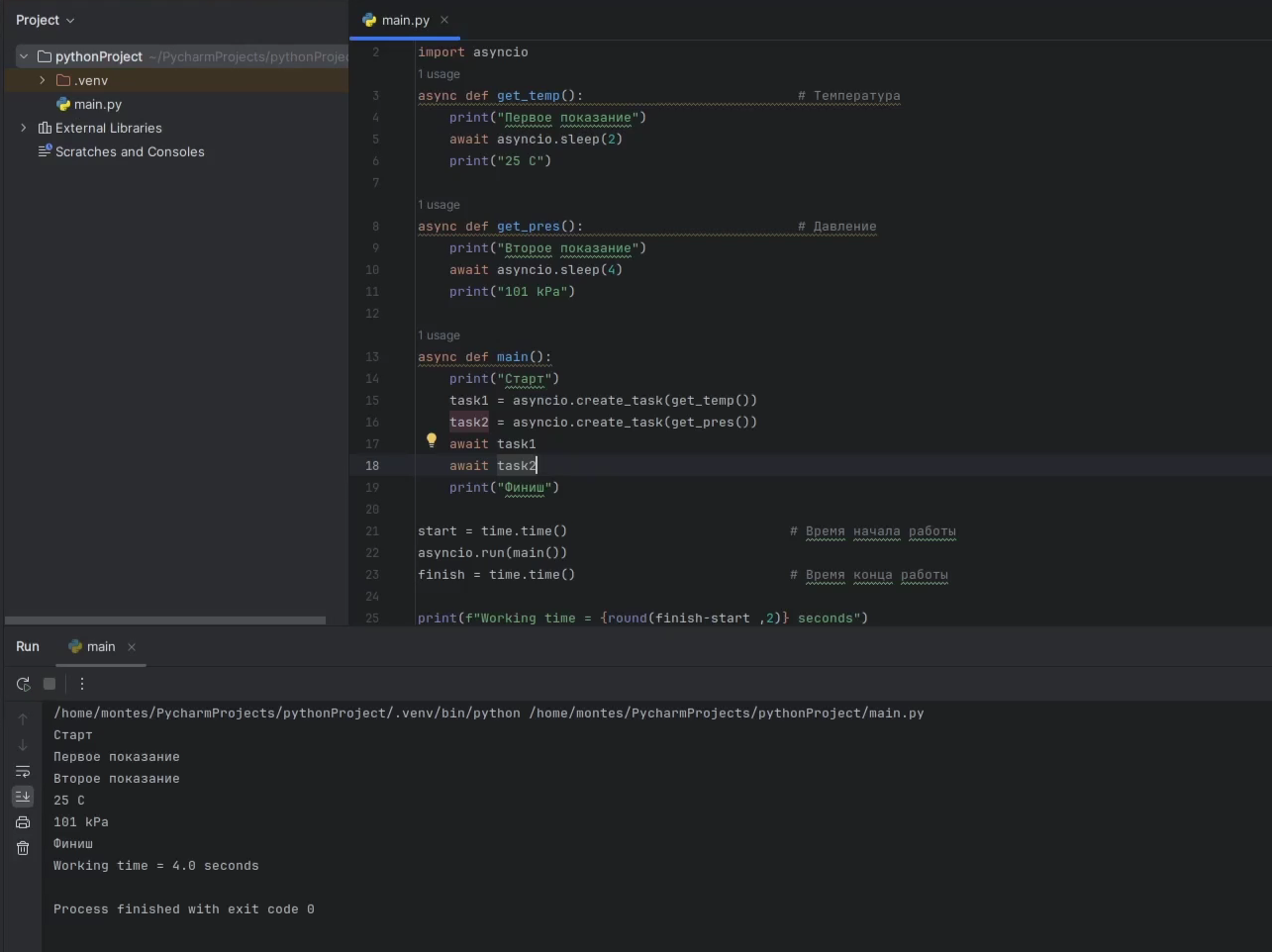


Рис. 11

### Почему понадобились «awaited» и сделали нашу программу функционирующей?

Если их убрать, то работа функции заканчивается. (Рис. 12)

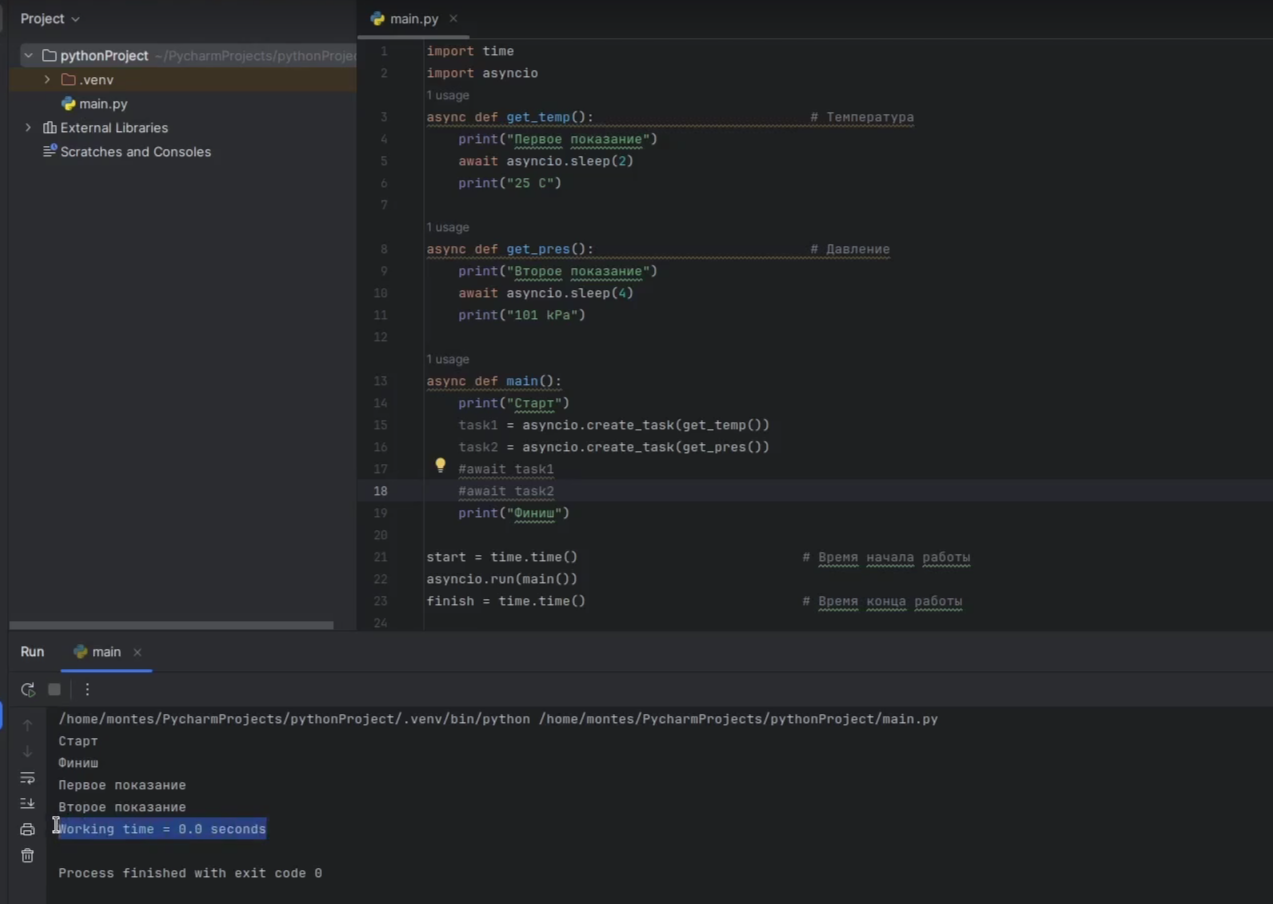


Рис. 12

Мы запустили «task», там начался отсчет, запустили второй «task», отработали print «Финиш» и программа завершилась. Мы забыли, что существуют эти 2 «таска». (Рис. 13)

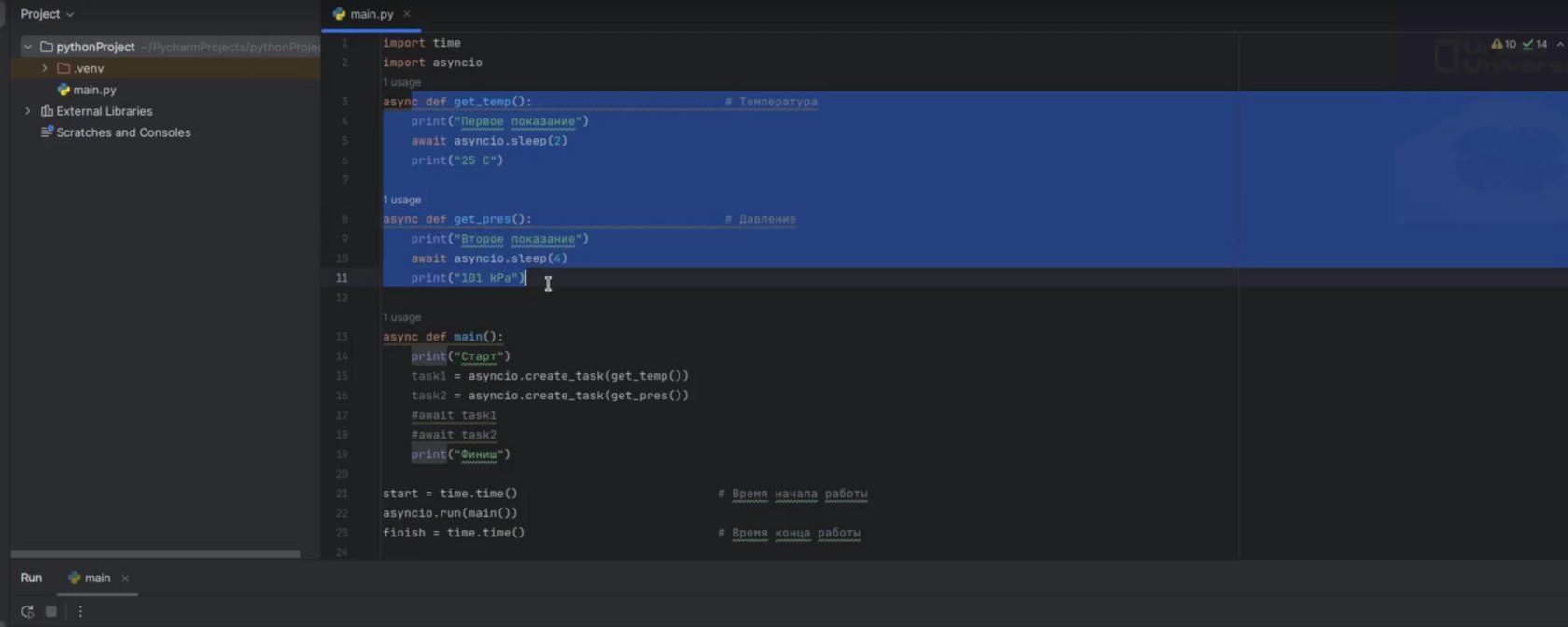


Рис. 13

Чтобы дождаться их завершения до окончания работы, мы используем «**await**», чтобы программа не завершалась, пока те «таски» не отработают, потому что они нужны нам отработанными. Именно поэтому мы **запускаем «task1», «task2» и ждем пока оба «task» отработают**. После этого пишем «Финиш» и сколько времени отработала функция. (Рис. 14)

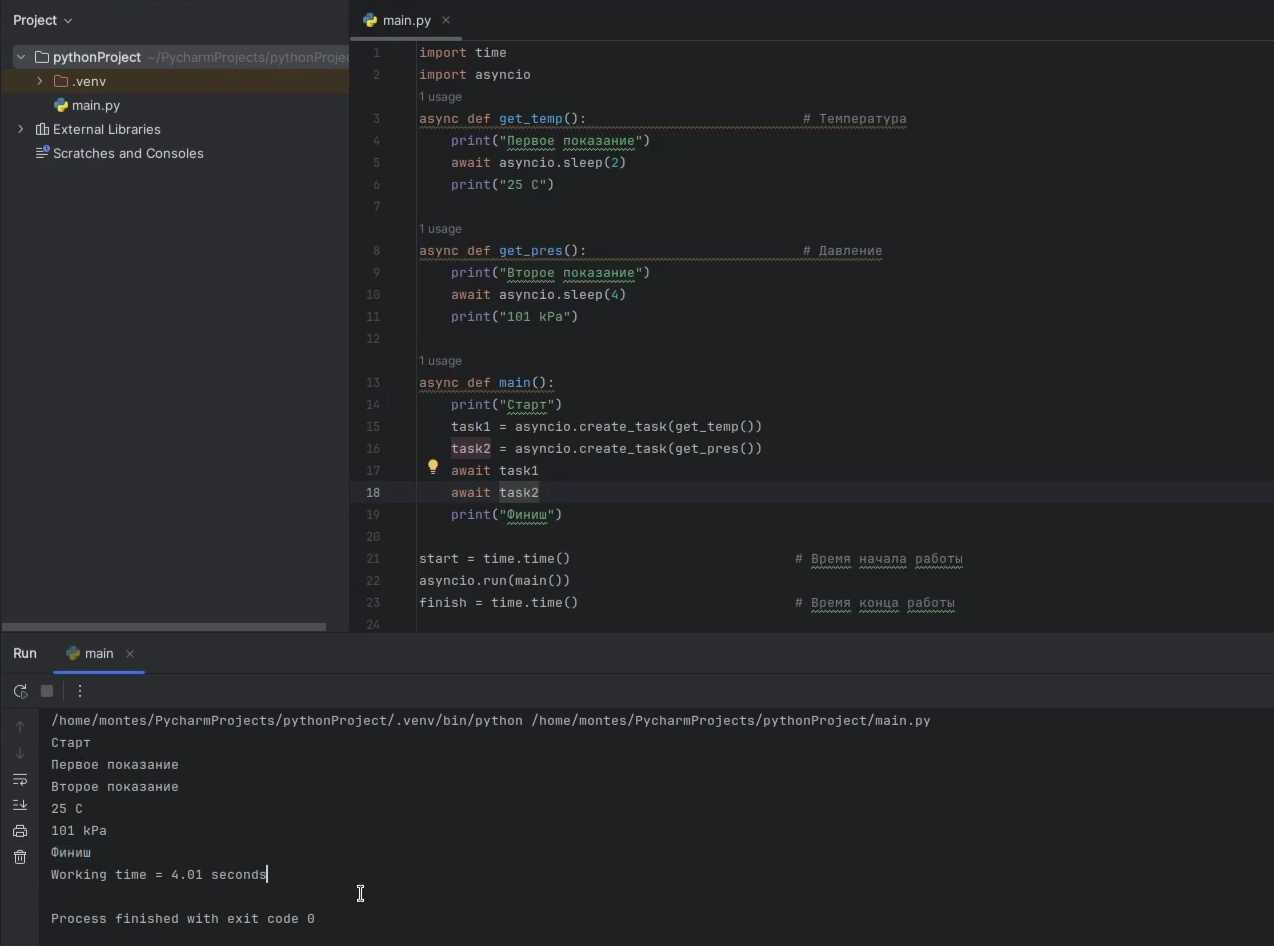


Рис. 14