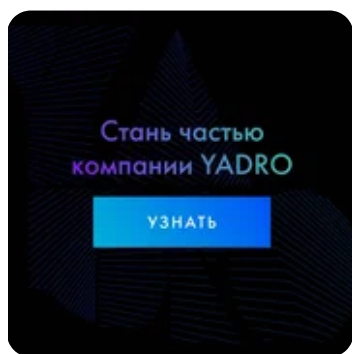


# Поддержка асинхронности async/await в Python



oneweekoffer.yadro.com

РЕКЛАМА

## Ищем ведущих программистов в команду YADRO.

Обучение и развитие • Открытое общение • Удаленно

[Узнать больше](#)[Справочник по языку Python3.](#) / Поддержка асинхронности async/await в Python

**Внимание!** Любой асинхронный код написанный на языке Python не будет работать без поддержки его распараллеливания во время выполнения.

Для написания [асинхронного кода с использованием синтаксиса async/await](#) создан [модуль asyncio](#), которая будет рассмотрена в разделе "[Стандартная библиотека Python3](#)".

Asyncio используется в качестве основы для нескольких асинхронных сред Python, которые предоставляют высокопроизводительные сетевые и веб-серверы, библиотеки подключений к базам данных, распределенные очереди задач и т. д.

Есть определенный список правил, касающийся использования команд async/await.

- Функция, которая начинается с [async def является сопрограммой](#).
- В теле сопрограммы можно использовать выражения await, [return](#) или [yield](#).
- Использование в сопрограмме ключевого слова yield создает [асинхронный генератор](#), через который можно итерироваться с помощью [async for](#).
- Использование в сопрограмме выражения async with запускает асинхронные контекстные менеджеры.
- Сопрограммы не могут использовать выражение [yield from](#). Это вызовет [исключение SyntaxError](#).
- Ключевое слово await можно использовать только в теле сопрограммы. Вызов await в другом месте спровоцирует исключение SyntaxError.
- Ключевое слово await требует наличия awaitable объекта. Такими объектами могут быть другая сопрограмма или объект у которого определен метод `__await__()`.

Ключевое слово await позволяет сопрограмме отдать контроль назад в главный цикл, который содержит порядок исполнения всех сопрограмм.

```
async def process():
    result = await func()
    return result
```

Если Python встречает ключевое слово await то это можно описать так: - "отложить исполнение кода сопрограммы process() до тех пор, пока я не получу результат выполнения func(). В это время я займусь чем-нибудь другим".

## [Сопрограммы/корутины async def в Python](#)

Внутри тела функции сопрограммы идентификаторы await и async становятся зарезервированными ключевыми словами. Выражения await, async for и async with могут использоваться только в телах функций сопрограмм.

## [Асинхронный async for в Python](#)

Асинхронный оператор `async for ... in` обеспечивает удобную итерацию по асинхронным итераторам и асинхронным генераторам. Асинхронный оператор `async for ... in` действует только в теле асинхронной функции (сопрограммы) `async def`.

## [Асинхронный контекст-менеджер async with в Python](#)

Асинхронный менеджер контекста может приостановить выполнение в своих методах `__aenter__()` и `__aexit__()`. Асинхронный оператор `with` может использоваться только в теле асинхронной функции (сопрограммы).

## [Асинхронный итератор в Python](#)

Асинхронный итератор может вызывать асинхронный код в своем методе `__anext__` и могут использоваться только в асинхронном операторе `async for`.

[Асинхронный генератор в Python](#)


Наличие выражения `yield` в функции или методе, определенном с использованием `async def`, дополнительно определяет функцию как функцию асинхронного генератора.


Содержание раздела:

- [ОБЗОРНАЯ СТРАНИЦА РАЗДЕЛА](#)
- [Сопрограммы/корутины `async def`](#)
- [Асинхронный `async for`](#)
- [Асинхронный контекст-менеджер `async with`](#)
- [Асинхронный итератор](#)
- [Асинхронный генератор](#)

ХОЧУ ПОМОЧЬ  
ПРОЕКТУ

РЕКЛАМА



 rooffalz.ru

**Производитель  
двойного фальца  
для кровли  
и фасадов.**

Классический двойной фальц  
и кликфальца. Доставка  
по России.

Завод-производитель >

Доставка по России >

Монтаж >

Узнать больше