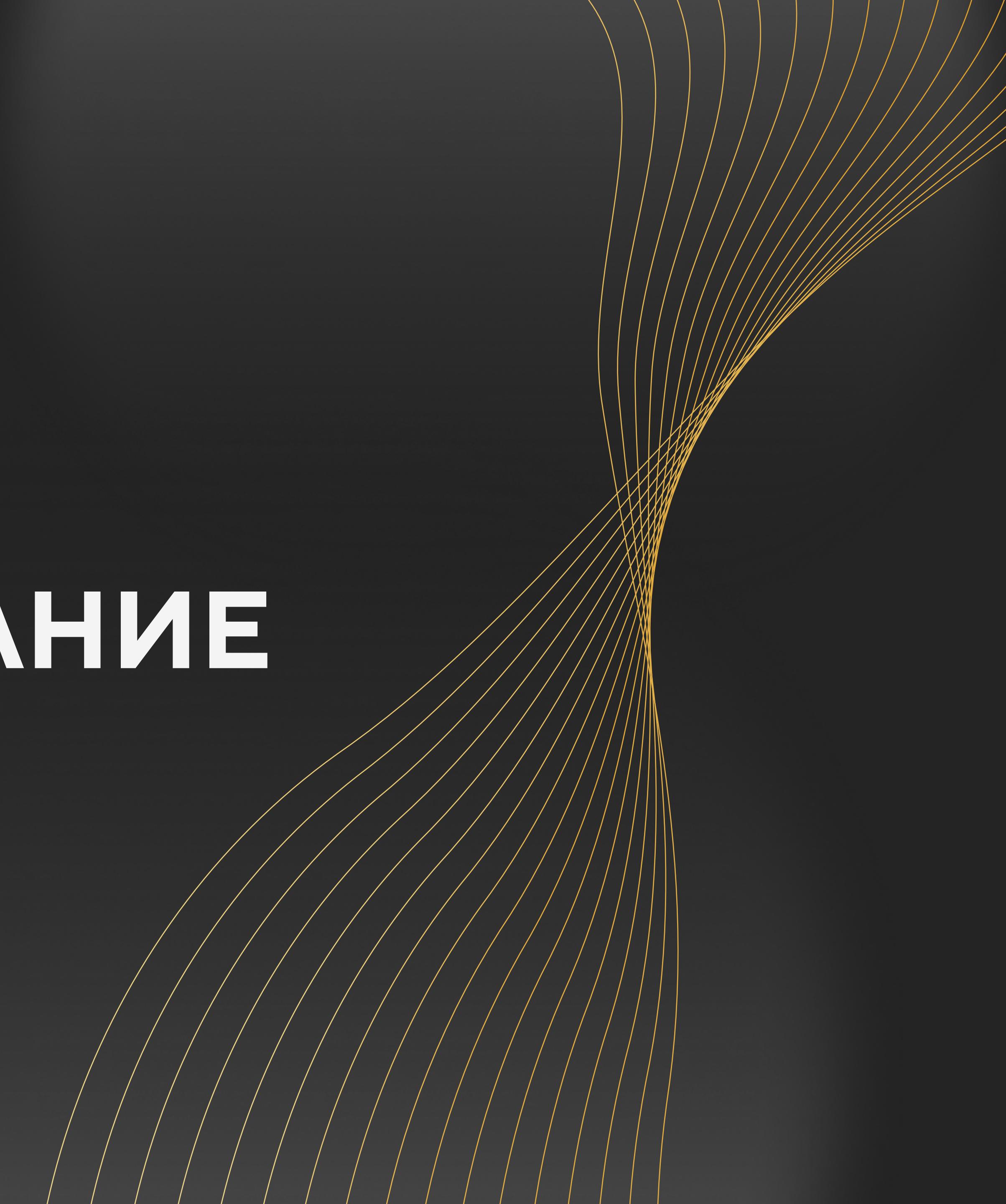


АСИНХРОННОЕ ПРОГРАММИРОВАНИЕ

Продвинутый Python

Андрей Вильмов



СОДЕРЖАНИЕ ТЕМЫ

Продвинутый Python

01 Библиотека asyncio

02 Различие асинхронности и многопоточности

03 Вызов функций по расписанию

АСИНХРОННОСТЬ И ПОТОКИ

Продвинутый Python

Параллелизм & Асинхронность

Асинхронность

Асинхронность представляет собой модель выполнения, при которой **программа не ждет завершения операций ввода-вывода или других долгих операций**, а продолжает свою работу, так как вместо блокировки ожидания результатов можно продолжать выполнение других задач

Параллелизм

Параллельное программирование предполагает выполнение задач **последовательно или параллельно на нескольких процессорах или ядрах одного процессора**. Оно основано на идее разделения задач на более мелкие подзадачи, которые могут быть выполнены одновременно, чтобы увеличить общую производительность

ASYNCIO

это модуль стандартной библиотеки Python,
который предоставляет возможности для
асинхронного программирования

Главная концепция asyncio – использование

Корутины также известны как асинхронные функции

Позволяют выполнять операции ввода-вывода асинхронно без блокировки основного потока выполнения

Сопрограмм

(асинхронные генераторы) для описания асинхронных операций

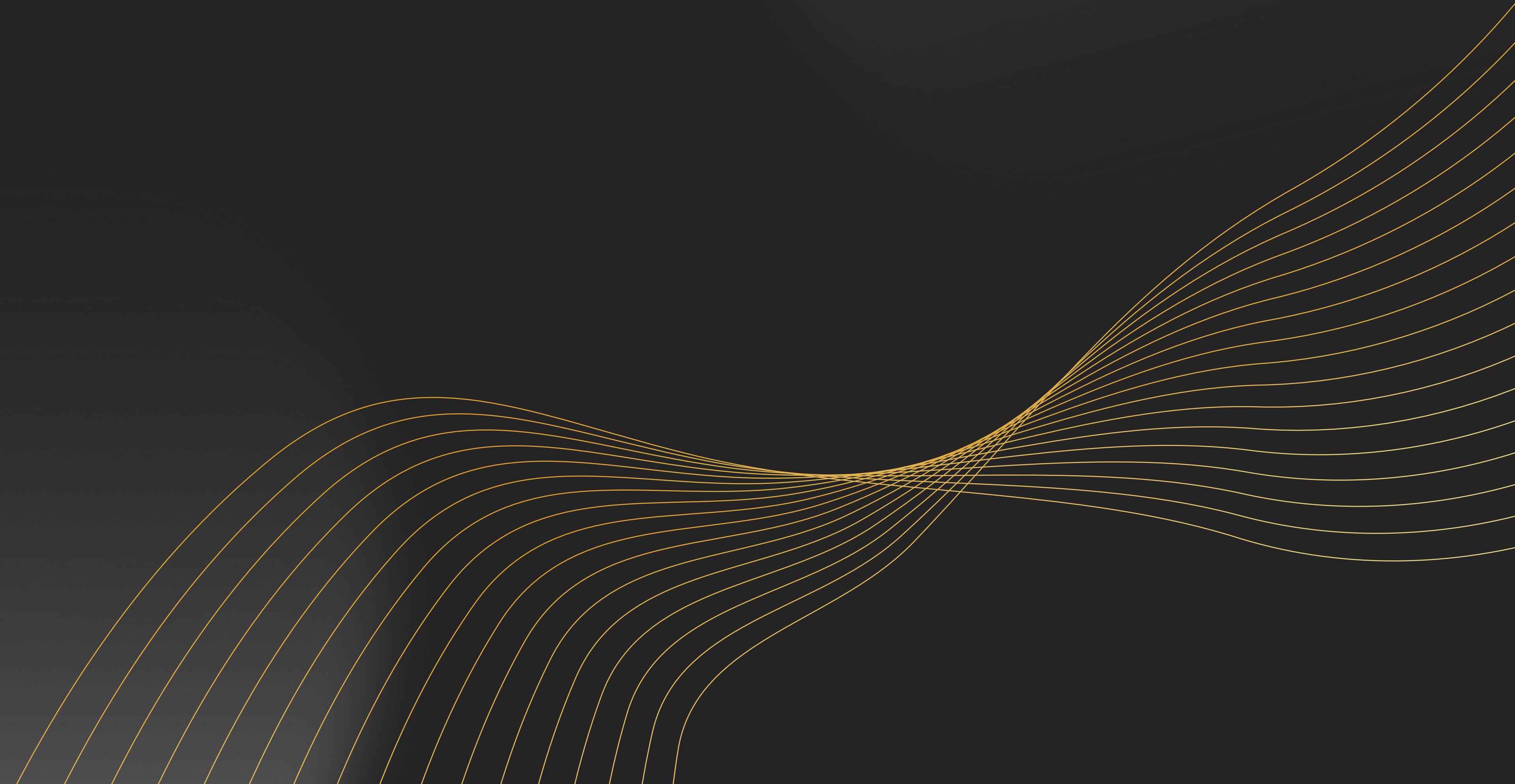
Asyncio также предоставляет механизмы для координации и синхронизации различных асинхронных задач. С помощью специальных объектов, таких как Event, Semaphore или Queue, можно организовывать взаимодействие между корутинаами

ASYNCIO

Продвинутый Python

Для работы с **asyncio** эту библиотеку необходимо импортировать

```
...  
import asyncio
```



RUN_UNTIL_COMPLETE

Продвинутый Python

`run_until_complete()`

Метод является одним из ключевых методов в модуле asyncio. Он используется для запуска асинхронной операции и блокирует выполнение программы до ее завершения

`run_until_complete()`

Принимает на вход одну асинхронную операцию в качестве аргумента. Эта операция может быть представлена в виде корутины, асинхронной функции или Future объекта. Она представляет собой задачу или операцию, которую нужно выполнить асинхронно

`run_until_complete()`

После завершения асинхронной операции `run_until_complete()` возвращает результат выполнения операции. Если операция выбрасывает исключение, метод также выбрасывает это исключение

RUN_UNTIL_COMPLETE

Продвинутый Python

Для работы с **asyncio** эту библиотеку необходимо импортировать

```
...
event_loop = asyncio.get_event_loop()

event_loop.run_until_complete(asyncio.gather(*parsers))
event_loop.close()
```



asyncio.gather()

это функция в модуле asyncio, которая позволяет выполнить несколько асинхронных операций параллельно и дождаться их завершения

Она принимает в качестве аргументов несколько корутин (асинхронных функций), и возвращает корутину, которая объединяет результаты выполнения всех переданных корутин. Это позволяет выполнять несколько асинхронных операций "одновременно" и ожидать их завершения вместе

asyncio.gather() возвращает список с результатами выполнения переданных корутин в том порядке, в котором они были переданы в функцию. Если корутины выбрасывают исключения, **gather()** возвращает эти исключения вместо результатов

ПОДВЕДЁМ ИТОГИ

Продвинутый Python

- ✓ Асинхронный код выполняется в одном потоке
- ✓ Библиотека `asyncio` позволяет реализовать асинхронный запуск
- ✓ Асинхронный код помогает увеличить скорость выполнения программы