Async/await: асинхронные возможности в Python 3.+

Asynchronous, Python 3.+, Tutorial, 3.+, Python, Async, Await

Содержание статьи

1. Цикл передач по третьим лицам
2. Потоки и исключения
3. Сетевой ввод-вывод
4. Новые слова
5. Новые грабли

Иногда у достаточно длительных операций, обращающих внимание на производительность параллельных вычислений возникает потребность в Python, встроенной поддержке асинхронных вычислений. Ранее, для решения подобных задач приходилось использовать сторонние библиотеки и надеяться, а даже для удачных и хотя бы Twisted, Tornado или gevent, разработчики приходили к выводу, что в Python 3.+ нужно иметь в виду, что асинхронные вычисления могут обеспечивать не меня и другие? Для ответа на эти и ещё несколько вопросов, а также для ознакомления с основами асинхронных вычислений в Python 3.+, сегодня, перед Вами полная экспериментальная

Цикл передач по третьим лицам

Основной принцип, на котором строится асинхронность в рамках Python 3.+ называется Event Loop (в переводе на русский язык, примерно, «цикл передач по третьим лицам»). Event Loop, или цикл бывает разных типов, зачем в основном зависимости, зависит от ОС, на которой запускается Python 3.+ и от того, что Вы выбрали, при сборке Python 3.+ (kqueue, epoll, Windows, IOCP, и т.д.). Для нас, пользователей, Event Loop, по большому счету, не интересен, поэтому, Вы можете, смело, не вникать в детали его реализации.

Суммарно — Event Loop, Вам нужен для того, чтобы в нем могли жить и выполнять свои задачи асинхронные корутины. Корутины, в свою очередь, могут приостанавливать свое выполнение, ожидая, пока, какой-то ресурс, не станет доступен, или, пока, не будет получен ответ, от другой корутины. Event Loop, в свою очередь, может, в ожидании ответа, переключиться на другую корутину, которая