Async and await

№ урока: 10 **Курс:** Flask

Средства обучения: Python3 и любимая среда разработки, например, PyCharm.

Обзор, цель и назначение урока

Познакомимся с асинхронным программированием в Python, рассмотрим асинхронный фреймворк Sanic и обсудим, когда лучше применять асинхронный подход, а когда синхронный. Кроме того, изучим концепты асинхронного программирования и какие у него преимущества, и недостатки перед синхронным подходом.

Изучив материал данного занятия, учащийся сможет:

- Использовать асинхронный функционал в Python.
- Правильно выбирать подход - асинхронный или синхронный для разработки конкретной задачи.
- Понимать основные концепты асинхронного программирования.

Содержание урока

1. Конкурентность или параллелизм
2. Асинхронное программирование
3. Основные концепты асинхронного программирования
4. Sanic

Резюме

- **Конкурентность** — это свойство систем (программы, сети, компьютера и т.д.), допускающее одновременное выполнение нескольких вычислительных процессов, которые могут взаимодействовать друг с другом. Вычисления запускаются, проходят и завершаются в пересекающихся промежутках времени; они также могут происходить абсолютно одновременно (параллелизм), но это не обязательно.
- Конкурентность реализована в логике программирования таким образом, что она явно устанавливает отдельные точки исполнения вычислений или процессов, называемые управляющими потоками. Они позволяют этим вычислениям избежать ожидания завершения всех остальных вычислений — как это происходит в случае последовательного программирования. Хотя и считается, что конкурентные вычисления включают в себя параллельные, у них есть существенные отличия.
- Параллельные вычисления используют более одного вычислительного ядра, поскольку все управляющие потоки работают одновременно и занимают весь рабочий цикл ядра на время исполнения — именно поэтому параллельное вычисление невозможно на одноядерном компьютере. В этом они отличаются от конкурентных вычислений, которые фокусируются на пересечениях жизненных циклов вычислений. Например, этапы выполнения процесса могут быть разбиты на временные промежутки, и, если процесс не заканчивает своё существование до конца промежутка, он приостанавливается, предоставляя другому процессу возможность работать.
- Главным преимуществом этого подхода является максимально возможное использование ресурсов системы. В начале 2000-ых стало популярным использование многоядерных процессоров, пришедших на смену одноядерным, пускай и с очень

CyberBionic systematics

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

мощным на то время ядром. Это в первую очередь позволяет оптимизировать время выполнения программы путём разделения нагрузки на ядра.

- В синхронных операциях задачи выполняются друг за другом. В асинхронных - задачи могут запускаться и завершаться независимо друг от друга. Одна асинхронная задача может запускаться и продолжать выполняться, пока выполнение переходит к новой задаче. Асинхронные задачи не блокируют (не заставляют ждать завершения выполнения задачи) операции и обычно выполняются в фоновом режиме.
    - Синхронность: блокирует операции (блокирующие)
    - Асинхронность: не блокирует операции (неблокирующие)
    - Конкурентность: совместный прогресс (совместные)
    - Параллелизм: параллельный прогресс (параллельные)

Базовые определения основных понятий asyncio:

- Coroutine (сопрограмма) – это следующий этап развития генераторов. Синтаксически генератор и сопрограммы похожи. Это функции, в которых присутствует ключевое слово yield. Task – это объект, который оборачивает coroutine, предоставляя методы для контроля ее выполнения и запроса ее статуса. Task может быть создан с помощью asyncio.create_task() или asyncio.gather(). Мы не можем вручную запустить Task или корутину. Для запуска корутины или Task нужно передать в event_loop с помощью create_task() или await.
- Event Loop – это бесконечный цикл, который берёт события из очереди и как-то их обрабатывает, а в некоторых промежутках — смотрит, не произошло ли каких-нибудь IO-событий и не просрочились ли какие-либо таймеры — тогда добавляет в очередь событие об этом, чтобы потом обработать.
- Идея очень проста. Есть цикл обработки событий и у нас есть функции, которые выполняют асинхронные операции ввода-вывода. Мы передаем свои функции циклу событий и просим его запустить их для нас. Цикл событий возвращает нам объект Future, словно обещание, что в будущем мы что-то получим.
- Future – объекты, в которых хранится текущий результат выполнения какой-либо задачи. Это может быть информация о том, что задача ещё не обработана или уже полученный результат; а может быть вообще исключение.

- Нужно понимать, что asyncio и threading максимизируют нагрузку 1 ядра компьютера. Если компьютер имеет 12 ядер, то 11 будут простаивать. Для задействования всех ядер нужно использовать multiprocessing.
- Главным минусом асинхронного программирования является то, что вам понадобится неблокирующая версия всего, что вы делаете (сокет, коннект к базе данных, подпроцесс, многопоточность, многопроцессорность, time.sleep). Соответственно, в асинхронном мире есть огромная экосистема инструментов поддержки. Это увеличивает время обучения.
- Асинхронное программирование отличается от многопоточного тем, что мы используем кооперативную многозадачность, и мы сами говорим, когда отдать контроль управления с помощью yield и await. Из этого следует, что вы контролируете, когда происходит переключение задач, поэтому блокировки и другая синхронизация больше не нужны.

Закрепление материала

- Что такое конкурентность?
- В чем разница между конкурентностью и параллелизмом?
- Что такое Event Loop?
- Что такое Task?
- Что такое Future?

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Что такое Coroutine?
- В чем разница между асинхронным подходом и многопоточным?
- В чем минусы асинхронного подхода?
- Сколько ядер компьютера будет задействовано при использовании asyncio / threading?

Дополнительное задание

Задание

Реализуйте поиск фильмов с помощью любого публичного API фильмов, как из предыдущего задания, но теперь используйте для этого Sanic, а не потоки во Flask.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные понятия, рассмотренные на уроке.

Задание 2
Изучите материалы из рекомендуемых ресурсов.

Задание 3
Перепишите методы, которые синхронно общаются с базой данных, используя SQLAlchemy, на асинхронные, которые в свою очередь используют asyncpg или aiopg. Для этого замените все контекстные менеджеры сессий SQLAlchemy на async with pool.acquire() as conn.

Рекомендуемые ресурсы

Miguel Grinberg Asynchronous Python for the Complete Beginner PyCon 2017:
https://www.youtube.com/watch?v=iG6fr81xHKA

How the heck does async/await work in Python 3.5?
https://snarky.ca/how-the-heck-does-async-await-work-in-python-3-5/

Async IO in Python: A Complete Walkthrough:
https://realpython.com/async-io-python/

Waiting in asyncio:
https://hynek.me/articles/waiting-in-asyncio/

Sanic documentation:
https://sanic.readthedocs.io/en/latest/

Конкурентность: Параллелизм:
https://habr.com/ru/post/318374/

Курс одного из разработчика asyncio: import asyncio: Learn Python's AsyncIO:
https://www.youtube.com/playlist?list=PLhNSoGM2ik6SIkVGXWBwerucXjgP1rHmB

CyberBionic systematics

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com