Пишем свой Lock

В asyncio есть сущность Lock. Мы уже изучали его использование в «Примитивах синхронизации». Давайте повторим еще раз, зачем он нужен, чтобы выполнить связанное с ним практическое задание.

Это некоторый асинхронный mutex, который приостанавливает работу программы до того момента, как он будет разблокирован. То есть, одна корутина может захватить Lock с помощью .acquire(), держать его, пока делает какие-то свои операции, а затем отпустить с помощью команды .release().

Пример использования Lock. Есть API, которое может обслуживать одновременно только один запрос. Или функция, которая запускает трудоемкие вычисления в базе данных, и вы не хотели бы перегружать базу. Тогда вам может помочь Lock:

```
import asyncio
lock = asyncio.Lock()


...
await lock.acquire()
try:
    await db.calculate_stats()
finally:
    lock.release()
```

Lock является примитивом синхронизации. Таким же примитивом является Event — некоторое событие, которое можно дожидаться асинхронно. Например: вы запустили несколько background-задач, которые должны сходить в разные API по списку, как только какая-то другая корутина сгенерирует этот список. Вы можете передать в эти background-задачи Event и поставить их в ожидание до выполнения события.

Пример, как это может выглядеть на псевдокоде:

```
import asyncio
list_of_apis = []

async def worker(event):
  await event.wait()
  await go_to_api(list_of_apis)


async def main():
  e = asyncio.Event()
  asyncio.create_task(worker(e))
  asyncio.create_task(worker(e))
  asyncio.create_task(worker(e))
  list_of_apis = await fetch_apis_list()
  e.set()
```

В этом задании вам предстоит написать свою реализацию Lock на Event Loop, который мы делали в видеоуроке.

Нельзя: пользоваться синтаксиом async/await
Нужно: пользоваться yield, yield from, и Future, созданной нами. Все необходимые для выполнения задания сущности можно импортировать из tasks.common.

Вызвать код, где используется Lock, можно с помощью:

```
./run_lock.sh
```

Запустить тесты на Lock:

```
./test_lock.sh
```

Event уже написан. Он *очень* похож на Lock, поэтому можно действовать по аналогии. Удачи!

Задание расположено в tasks/lock, пример Event на нашем event-loop можно посмотреть в tasks/event.