

Задание 1. Мониторинг теплицы

Ситуация: мы разрабатываем систему «умный дом» и в данный момент работаем над другой системой, контролирующей микроклимат в теплице. Для контроля условий выращивания растений установлены два датчика:

1. Датчик температуры измеряет температуру внутри теплицы.
2. Датчик влажности измеряет уровень влажности воздуха.

Задачи:

- Реализовать имитацию сбора данных с двух датчиков за заданное время. Датчики передают измерения с разным интервалом (зависит от настроек и их работы).
- Конкурентно получать данные с двух датчиков, чтобы не терять время в ожидании завершения работы одного из них.

Используем `asyncio.sleep` и библиотеку `random` для имитации задержек.

Задание 2. Асинхронная обработка данных

Ситуация: мы разрабатываем приложение для проверки доступности веб-страниц. Наша цель — асинхронно «скачать» содержимое нескольких страниц и вывести статус их обработки. Каждая загрузка выполняется с задержкой, чтобы симулировать сетевую операцию.

Задача — реализовать программу для имитации проверки доступности веб-страниц:

- дан список URL-адресов (можно использовать текстовые идентификаторы вместо реальных URL);
- каждая загрузка занимает случайное время от 1 до 5 секунд (`asyncio.sleep`, библиотека `random`);
- после завершения загрузки вывести сообщение с информацией о том, какой URL был обработан;
- задачи должны выполняться конкурентно.

Задание 3. Мониторинг теплицы с использованием `asyncio.gather`

Ситуация: мы продолжаем разработку «умного дома» из предыдущего задания. В данный момент мы работаем над системой, контролирующей микроклимат в теплице. Для контроля условий выращивания растений установлены два датчика: температуры и влажности. В этот раз наша система должна гарантировать, что время прихода данных с датчиков (температуры и влажности) не превышает 3 секунд.

Задача:

- Реализовать имитацию сбора данных с двух датчиков за заданное время (случайные значения от 1 до 3 секунд, использовать библиотеку `random`).
- Параллельно получать данные с обоих датчиков, чтобы не терять время в ожидании завершения работы одного из них.

Задание 4. Сканирование портов

Ситуация: в нашей компании случилась утечка данных, и отдел кибербезопасности предполагает, что проблема в открытом порте на одном из серверов. Нам поставили задачу написать утилиту, проверяющую наличие утечек данных. Для ускорения работы нужно выполнять проверку нескольких портов одновременно.

Задача:

Написать программу, которая будет имитировать проверку списка портов. Время проверки каждого порта составляет от 1 до 3 секунд (использовать модуль `random`). После завершения проверки каждого порта вывести сообщение об успехе или ошибке (например, порт закрыт). Все проверки должны выполняться параллельно.