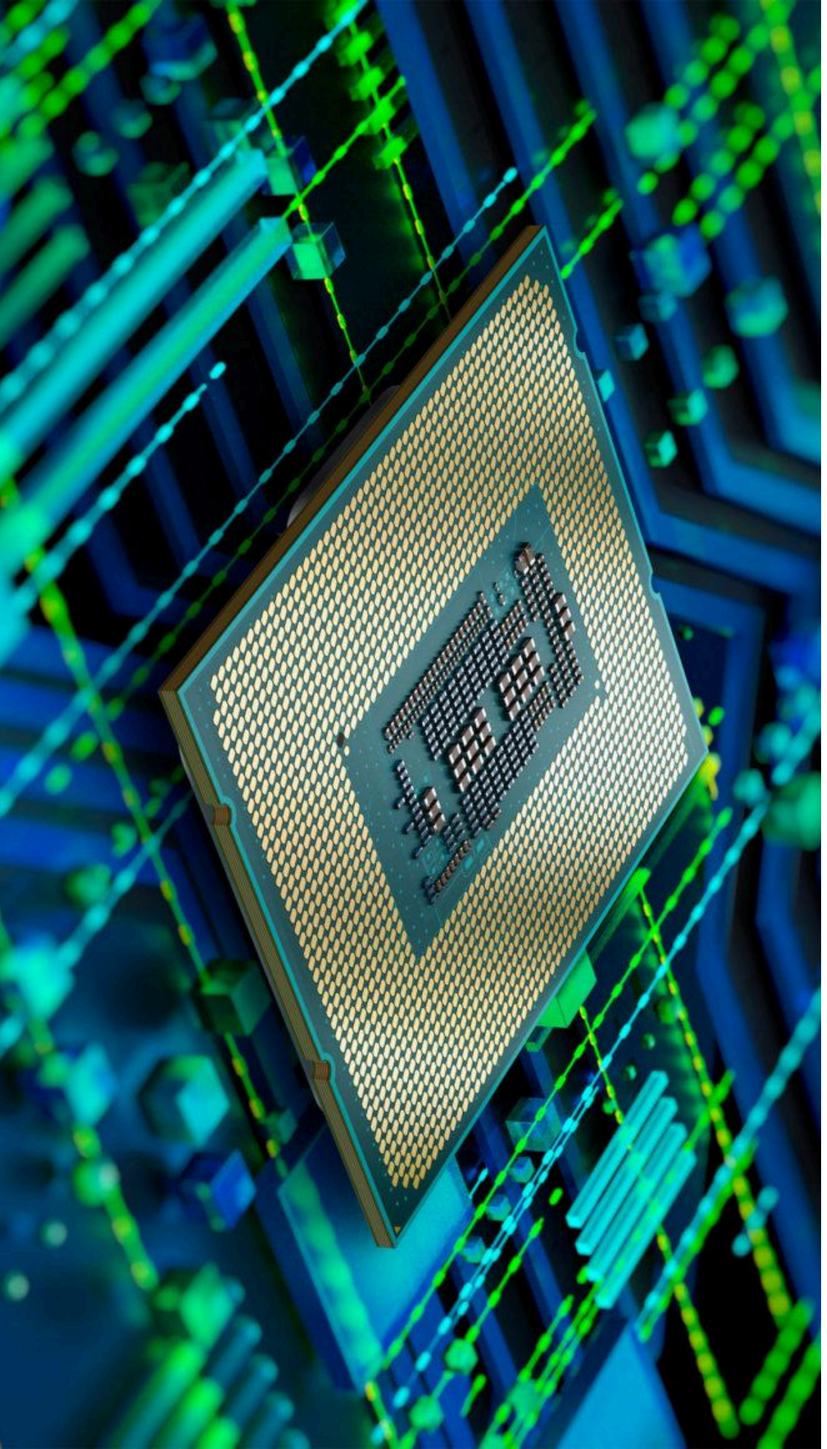
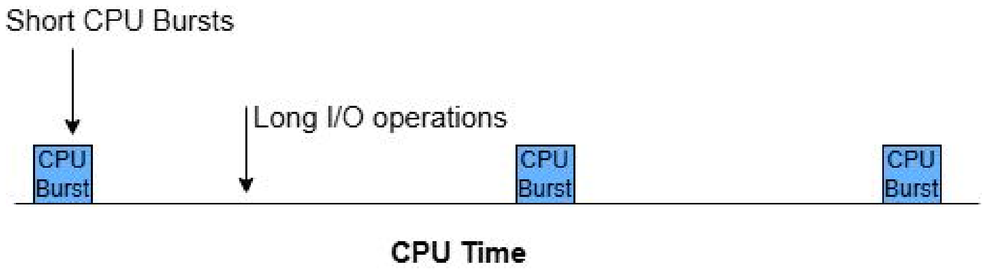


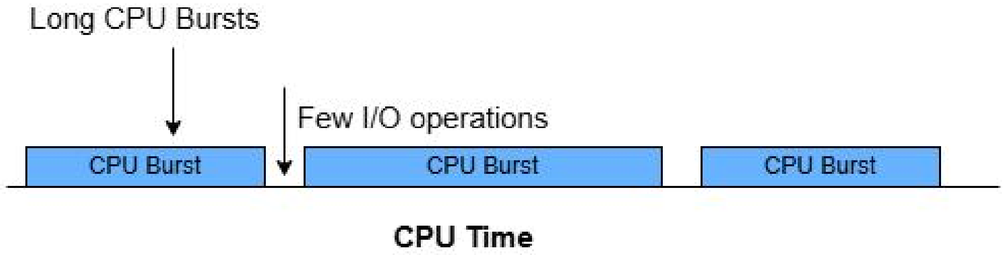
Синхронность.
Асинхронность.
Многопоточность.

“ИТ-класс”. Бэкенд-разработка на Python.
Бусыгин Дмитрий, 2025 год.

А какая вообще нагрузка бывает?



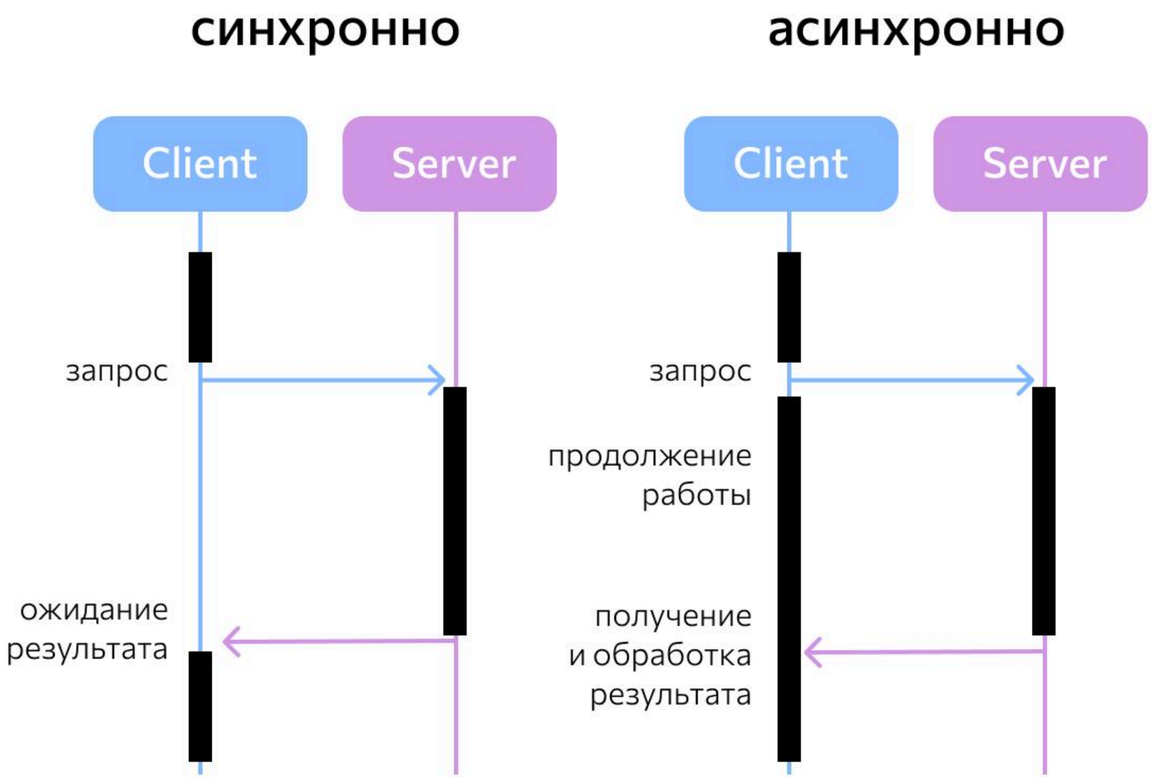
I/O-bound операции — основное время тратится на ожидание внешних систем (сеть, диск, база данных, ОС), а не на вычисления процессором.



CPU-bound операции — основное время тратится на вычисления процессора

Статейка (англ.): https://www.baeldung.com/cs/cpu-io-bound

Асинхронность — для I/O-bound задач



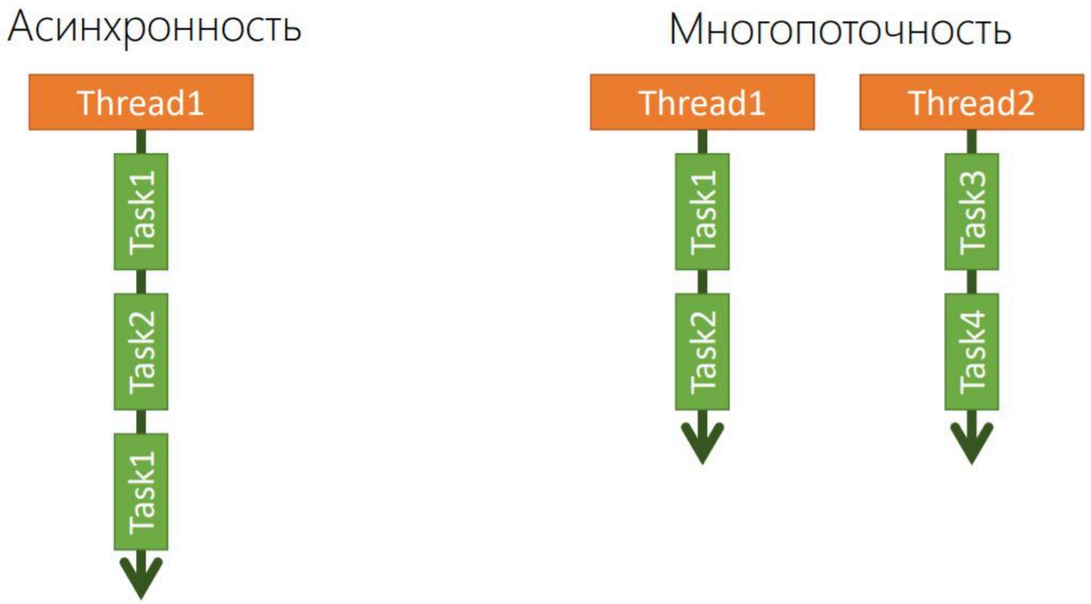
Асинхронность — это концепция выполнения задач без блокировки основного потока программы, позволяющая выполнять другие действия, пока одна из операций ожидает завершения.

Пример: пока поток №2 занят, поток №1 может работать дальше и принять ответ от №2 позже.

Многопоточность — обычно для CPU-bound задач

Современные процессоры состоят из нескольких ядер, а каждое ядро может обрабатывать несколько потоков.

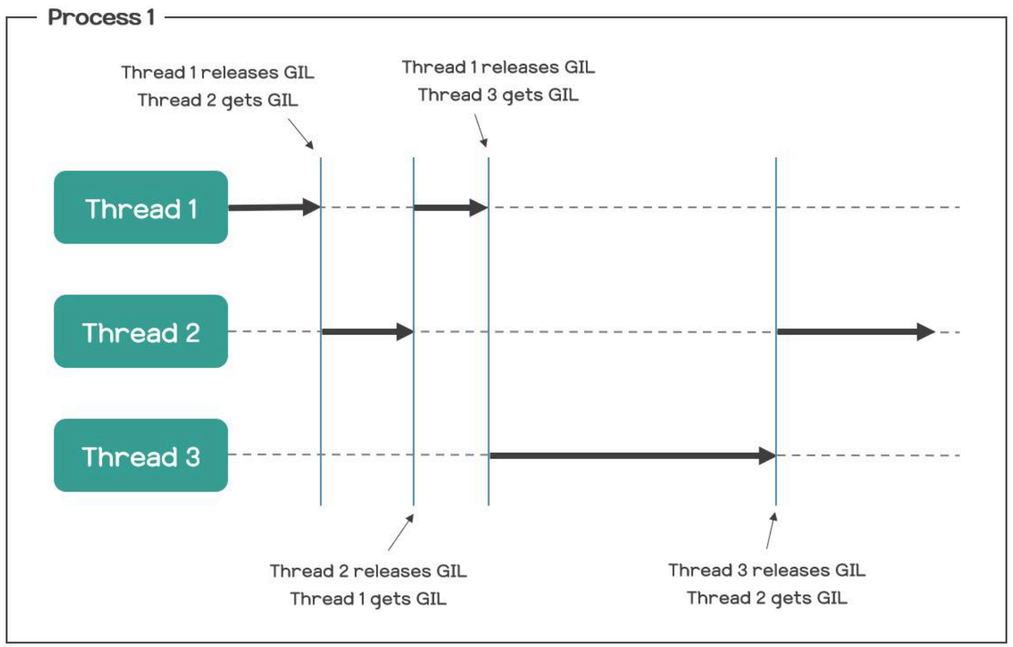
Значит, некоторые процессы можно выполнять параллельно.



Многопоточность — программа разбивается на процессы, каждый из которых выполняет свой поток.

Пример: поток 1 выполняет задачи 1 и 2, а поток 2 — задачи 3 и 4, чтобы потом объединить результаты.

Но с многопоточностью на Python всё сложнее

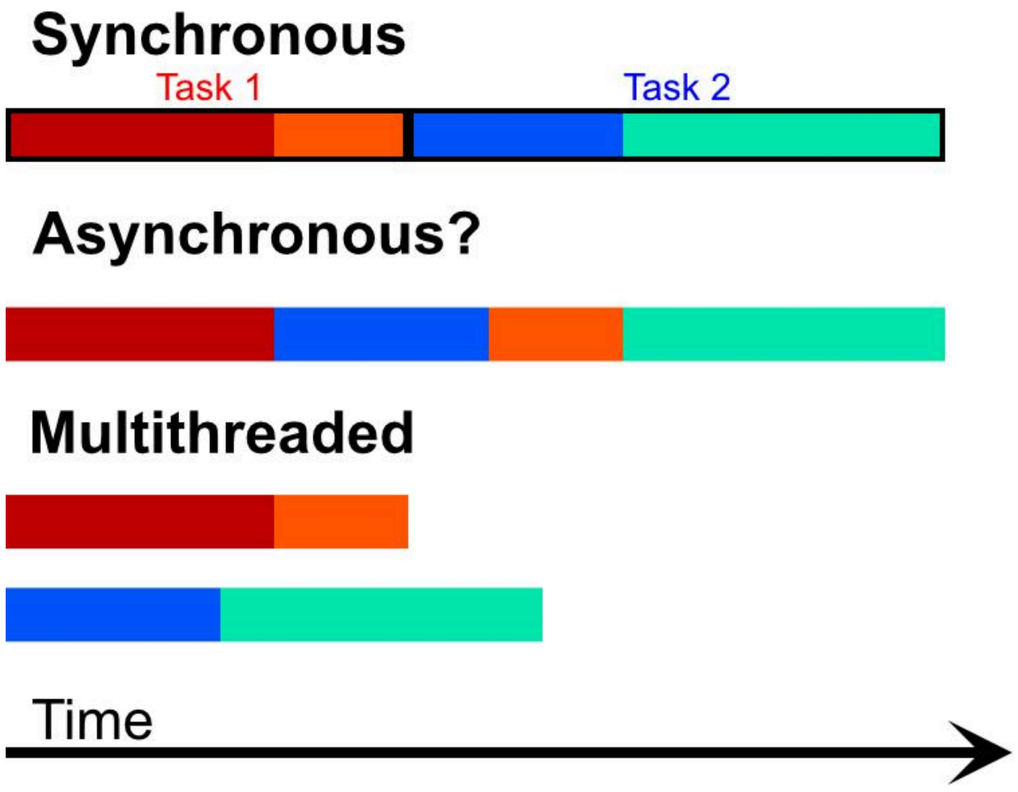


GIL (Global Interpreter Lock) — механизм Python, который органичивает использование кодом нескольких потоков.

Да, по факту, многопоточности на Python нет, поэтому для реального распараллеливания задач нужна многопроцессорность.

Поэтому библиотека threading используется для I/O-bound задач. Пока одна синхронная функция ждет ответа, программа переключается на другой поток с другой задачей.

Не путайте понятия



Синхронность — задачи выполняются строго последовательно.

Асинхронность — каждая подзадача грамотно делегируется соответствующему потоку.

Многопоточность — каждый поток берет на себя всю задачу.

Пример: сервер ждет ответ от базы данных 2 секунды. Синхронно он бы просто ждал, а асинхронно в это время обрабатывает другие запросы

Что использовать в ваших проектах?

✅ **Используйте асинхронность для I/O-bound операций:**

- Запросов к API других сервисов
- Работы с базой данных
- Одновременной обработки многих запросов, где ответ обрабатывается не процессором

⚠️ **Многопоточность тоже может понадобиться для I/O-bound, но в других случаях:**

- Синхронные библиотеки (например, requests)

По сути, threading — это костыль, позволяющий со старым синхронным кодом работать асинхронно.

⚠️ **Многопроцессорность для CPU-bound задач:**

- Параллельные вычисления (с процессами)

❌ **Не смешивайте без необходимости! И по возможности не прибегайте к threading**

А теперь практика в Google Colab