Синхронный код в асинхронных функциях

Еще раз отметим, что асинхронный код дает свое преимущество только при использовании его с неблокирующими операциями ввода/вывода.

Например, вызов сихронной функции **time.sleep()** превратит нашу программу в синхронную.

```
import asyncio
import time

async def sync_sleep():
    time.sleep(1)

SUB_PROGS_COUNT = 10
async def main():
    begin = time.time()
    await asyncio.gather(
        *[asyncio.sleep(1) for _ in range(SUB_PROGS_COUNT)]
    )
    print(f"async sleep execution: {time.time() - begin}")

    begin = time.time()
    await asyncio.gather(
        *[sync_sleep() for _ in range(SUB_PROGS_COUNT)]
    )
    print(f"common sleep execution: {time.time() - begin}")

asyncio.run(main())
```

```
async sleep execution: 1.0053341388702393
common sleep execution: 10.028470039367676
```

Функция **time.sleep(),** в отличие от **asyncio.sleep(),** блокирующая.

Именно поэтому то же число короутин отрабатывает во столько раз больше, сколько короутин синхронно запущено.