IO bound операции

> run_in_executor/io.py

```
import asyncio
import concurrent.futures
import datetime

import requests


def blocking_task():
    requests.get('https://docs.python.org/3/')


async def blocking_worker():
    while True:
        blocking_task()


async def async_thread_worker():
    loop = asyncio.get_event_loop()
    with concurrent.futures.ThreadPoolExecutor(2) as pool:
        while True:
            await loop.run_in_executor(pool, blocking_task)


async def ticker():
    while True:
        print(datetime.datetime.now())
        await asyncio.sleep(1)


if __name__ == '__main__':
    loop = asyncio.get_event_loop()
    loop.create_task(ticker())
    # loop.create_task(blocking_worker())
    # loop.create_task(async_thread_worker())
    loop.run_forever()
```

Представим, что у нас есть блокирующая IO bound операция **blocking_task**, для которой нет альтернативной асинхронной библиотеки.

Есть фоновая задача **ticker**, которая просыпается раз в секунду и печатает текущее время. Она нужна для понимания, заблокирован ли основной поток исполнения. Если в логах выполнения демона мы увидим разницу между **print(datetime.datetime.now())** больше, чем в одну секунду — event loop заблокирован и не может своевременно передать управление нужной корутине.

Есть еще 2 воркера, которые выполняют «полезную работу»: **blocking_worker** и **async_thread_worker**.

blocking_worker вызывает в цикле синхронную функцию.
async_thread_worker запускает в ThreadPool выполнение синхронной функции и асинхронно ждет выполнения.

Давайте раскомментируем **loop.create_task(blocking_worker())** и попробуем запустить:

```
coder@code-server-admin-38-94c54d494-m248j:~/mercury$ python3 run_in_executor/io.py
2021-09-29 11:27:45.282881
```

Второго сообщения мы никогда не увидим, так как **blocking_worker** заблокировал event loop.

Теперь раскомментируем только **loop.create_task(async_thread_worker())** и попробуем запустить:

```
coder@code-server-admin-38-94c54d494-m248j:~/mercury$ python3 run_in_executor/io.py
2021-09-29 11:28:13.214911
2021-09-29 11:28:14.215541
2021-09-29 11:28:15.216848
2021-09-29 11:28:16.226067
2021-09-29 11:28:17.238052
```

ticker запускается по плану, все хорошо. В итоге получилось сделать синхронную операцию асинхронной.

Не злоупотребляйте этим методом. Используйте его в только в случаях, когда не смогли найти асинхронную библиотеку к подключаемому сервису.

Похожая картина получится с **ProcessPoolExecutor**: поменяйте executor и запустите **io.py**.