Основы работы с потоками на Python

Основы работы с потоками и объектами Concurrency.

Лабораторная работа №2. Задачи.

Комплект 1: Потоки в Python.

1.1: Напишите программу, которая создаёт несколько потоков, а затем выводит их имена изнутри каждого потока.

1.2: Напишите программу для одновременной загрузки нескольких файлов (например картинок) с использованием потоков. Используйте для скачивания одну из библиотек `urllib`, `requests` или `wget`.

1.3: Напишите программу на Python, которая выполняет одновременные HTTP-запросы с использованием потоков. Используйте библиотеку `requests`.

1.4: Напишите программу для вычисления факториала числа с использованием нескольких потоков.

1.5: Напишите программу для реализации многопоточного алгоритма быстрой сортировки.

Комплект 2: Concurrency и Futures в Python.

2.1: Дополните решение из лабораторной работы № 1 с кодом функции integrate следующим кодом, расположенном по этой ссылке (`https://replit.com/@zhukov/sem7-task3#main.py`) или по этой ссылке (`https://replit.com/@zhukov/prog7-t1-lr2#main.py`) и также проведите замеры времени вычисления для аналогичных параметров модуля timeit для кратного числа потоков и процессов (2, 4, 6). Замеры вычислений — только для количества итераций `n_iter = 10**6`:

```
integrate(math.atan, 0, math.pi / 2, n_iter=10**6)
```

Дополните этой информацией отчёт. Количество повторений / repeat — 100, единицы измерения — msec. Ссылку на борд с кодом с комментариями на Python в repl.it приведите как ответ на это задание.

2.2: Напишите программу, которая будет симулировать банк с использованием потоков и объектов типа `Lock`. Реализуйте методы `deposit` и `withdraw`, которые будут добавлять и снимать деньги со счета клиента соответственно. Гарантируйте, что доступ к счету будет синхронизирован с помощью объекта типа `Lock`.

2.3: Напишите программу, используя объекты типа `Future`, чтобы асинхронно скачать несколько изображений с Интернета. Каждое изображение должно быть загружено в отдельный поток и сохранено на диск. Используйте семафор, чтобы ограничить количество одновременно выполняющихся потоков загрузки, чтобы избежать блокировки по IP от сервера или серверов.

2.4: Создайте два потока, один из которых будет записывать данные в файл, а другой - считывать данные с файла. Используйте объекты типа `Future`, чтобы синхронизировать потоки и избежать гонки потоков.

2.5: Напишите программу, которая создает 3 потока. Первый поток должен устанавливать состояние объекта типа `Event` каждую секунду. Второй поток должен ждать наступления события и выводить сообщение `"Event␣occurred"`. Третий поток должен выводить сообщение `"Event␣did␣not␣occur"` каждую секунду, до тех пор, пока не наступит событие. Как только событие происходит, третий поток должен остановиться.

2.6: Создайте программу, которая имеет свой класс "очередь" (Queue), у которого есть методы для добавления элементов в очередь и удаления элементов из очереди. Используйте рекурсивный блокировщик `RLock` для обеспечения безопасности доступа к очереди из нескольких потоков.

2.7: Сделайте два потока, представляющие сервер и клиент, где клиентский поток будет ждать готовности сервера, прежде чем отправлять ему какой-либо запрос. Используйте `threading.Barrier`

2.8: Написать программу для параллельного поиска файла в директории. Каждый поток должен обрабатывать свой фрагмент файлов директории и синхронизироваться с другими потоками, чтобы убедиться, что файл найден только один раз. Как только первый файл по его шаблону найден - все потоки поиска завершаются.