

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Лабораторная работа 2.24**

Синхронизация потоков в языке программирования Python

Выполнил студент группы ИВТ-б-о-20-1

Симанский М.Ю « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_

(подпись)

**Цель работы:** приобретение навыков использования примитивов синхронизации в языке программирования Python версии 3.x.

## Задание 1

Разработать приложение, в котором выполнить решение вычислительной задачи (например, задачи из области физики, экономики, математики, статистики и т. д.) с помощью паттерна “Производитель-Потребитель”, условие которой предварительно необходимо согласовать с преподавателем. Результат выполнения отображен на рисунке 1.

```
Стреляет Миша - 2 раз
Вероятность попадания 2 выстрелов подряд - 0.28749600000000003

Стреляет Миша - 3 раз
Вероятность попадания 3 выстрелов подряд - 0.18974736000000003

Стреляет Егор - 5 раз
Вероятность попадания 5 выстрелов подряд - 0.531441

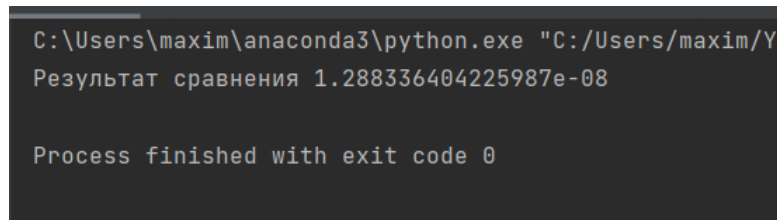
Стреляет Павел - 3 раз
Вероятность попадания 3 выстрелов подряд - 0.04100625

Стреляет Миша - 4 раз
Вероятность попадания 4 выстрелов подряд - 0.12523325760000004
```

Рисунок 1 – Результат выполнения работы

## Задание 2

Для своего индивидуального задания лабораторной работы 2.23 необходимо организовать конвейер, в котором сначала в отдельном потоке вычисляется значение первой функции, после чего результаты вычисления должны передаваться второй функции, вычисляемой в отдельном потоке. Потоки для вычисления значений двух функций должны запускаться одновременно.



```
C:\Users\maxim\anaconda3\python.exe "C:/Users/maxim/Y...
Результат сравнения 1.288336404225987e-08

Process finished with exit code 0
```

Рисунок 2 – Результат выполнения

### Ответы на контрольные вопросы

1. Lock-объект может находиться в двух состояниях: захваченное (заблокированное) и не захваченное (не заблокированное, свободное). После создания он находится в свободном состоянии. Для работы с Lock-объектом используются методы `acquire()` и `release()`. Если Lock свободен, то вызов метода `acquire()` переводит его в заблокированное состояние. Повторный вызов `acquire()` приведет к блокировке инициировавшего это действие потока до тех пор, пока Lock не будет разблокирован каким-то другим потоком с помощью метода `release()`. Вывоз метода `release()` на свободном Lock-объекте приведет к выбросу исключения `RuntimeError`.

2. В отличие от рассмотренного выше Lock-объекта `RLock` может освободить только тот поток, который его захватил. Повторный захват потоком уже захваченного `RLock`-объекта не блокирует его. `RLock`-объекты поддерживают возможность вложенного захвата, при этом освобождение происходит только после того, как был выполнен `release()` для внешнего

acquire(). Сигнатуры и назначение методов release() и acquire() RLock-объектов совпадают с приведенными для Lock, но в отличие от него у RLock нет метода locked(). RLock-объекты поддерживают протокол менеджера контекста. С помощью команды закрытия close().

3. Порядок работы с условными переменными выглядит так:

- На стороне Consumer'a: проверить доступен ли ресурс, если нет, то перейти в режим ожидания с помощью метода wait(), и ожидать оповещение от Producer'a о том, что ресурс готов и с ним можно работать. Метод wait() может быть вызван с таймаутом, по истечении которого поток выйдет из состояния блокировки и продолжит работу.

- На стороне Producer'a: произвести работы по подготовке ресурса, после того, как ресурс готов оповестить об этом ожидающие потоки с помощью методов notify() или notify\_all(). Разница между ними в том, что notify() разблокирует только один поток (если он вызван без параметров), а notify\_all() все потоки, которые находятся в режиме ожидания. Чтобы обновить данные в таблице, просто создайте соединение, затем создайте объект курсора с помощью соединения и, наконец, используйте оператор UPDATE.

4. При создании объекта Condition вы можете передать в конструктор объект Lock или RLock, с которым хотите работать. Перечислим методы объекта Condition с кратким описанием: acquire(\*args) – захват объекта-блокировки. release() – освобождение объекта-блокировки. wait(timeout=None) – блокировка выполнения потока до оповещения о снятии блокировки. Через параметр timeout можно задать время ожидания оповещения о снятии блокировки. Если вызвать wait() на Условной переменной, у которой предварительно не был вызван acquire(), то будет выброшено исключение RuntimeError.

5. Чтобы перечислить все таблицы в базе данных SQLite3, вы должны запросить данные из таблицы sqlite\_master, а затем использовать fetchall() для получения результатов из инструкции SELECT

6. При создании таблицы мы должны убедиться, что она еще не существует. Аналогично, при удалении/удалении таблицы она должна существовать. Чтобы проверить, не существует ли таблица уже, мы используем `IF NOT EXISTS` с оператором `CREATE TABLE` следующим образом.

7. Метод `executemany` можно использовать для вставки нескольких строк одновременно.

8. В базе данных Python SQLite3 мы можем легко хранить дату или время, импортируя модуль `datetime`. Следующие форматы являются наиболее часто используемыми форматами для `datetime`::