Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЁТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №10
дисциплины «Анализ данных»
Варианты 2 и 3

Выполнил:
Репкин Александр Павлович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., канд. техн. наук,
доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Синхронизация потоков в языке программирования Python.

Цель: приобрести навыки использования примитивов синхронизации в языке программирования Python версии 3.x.

Порядок выполнения работы:

1. Выполнено индивидуальное задание – модифицировано индивидуальное задание из лабораторной работы №9 дисциплины “Анализ Данных”: “С использованием многопоточности для x = 0.7(Во 2 варианте) и x = 1.2 (В 3 варианте), находится сумма ряда S с точностью члена ряда по абсолютному значению ε = $10e^{-7}$ и производится сравнение полученной суммы с контрольным значением функции y = 1/(1-x) (Во 2 варианте) и y = 1/(2+x) (В 3 варианте) для двух бесконечных рядов”. Организован 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 запускаются одновременно.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from threading import Barrier, Lock, Thread


'''Варианты №3 и №2(По списку 28)
Для своего индивидуального задания лабораторной работы 2.23 организован
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 запускаются одновременно.

ПРИМЕЧАНИЕ: в модуле threading найден метод Barrier. Он предназначен для алгоритмов, в которых
для выполнения задачи необходимо дождаться завершения работы группы потоков.
P.S.Другой частично подходящий метод - Event, синхронизирующий операции потоков.

Индивидуальное задание из работы 2.23:
С использованием многопоточности для x = 0.7(Во 2) и x = 1.2 (В 3),
находится сумма ряда S с точностью члена ряда по абсолютному
значению ε = 10e^(-7) и производится сравнение полученной суммы с контрольным
значением функции y = 1/(1-x) (Во 2) и y = 1/(2+x) (В 3) для двух бесконечных рядов'''

# Создание блокировки для синхронизации доступа к общему ресурсу (словарю результатов results).
lock = Lock()


def second_var(results, barrier):
    # Вычисление суммы ряда для x = 0.7
    s = 0
    n = 0
    while True:
        element = 0.7**n
        if element < 1e-7:  # Проверка условия остановки (ε).
            break
        else:
            s += element
            n += 1
```

Рисунок 1. Код индивидуального задания

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_10\Программы>python Individual.py
Good day! According to our calculations:
Sum of elements in variant 2 = 3.333333083650555
At the same time, y = 3.3333333333333335 /n
For the 3 variant, sum of elements = 0.31250004145136007
y = 0.3125
```

Рисунок 2. Пример выполнения индивидуального задания

2. Ответы на вопросы.

1) Каково назначение и каковы приёмы работы с Lock-объектом?

Ответ: Lock используется для обеспечения взаимного исключения в многопоточной среде, что предотвращает одновременное выполнение критических секций кода несколькими потоками. Приёмы работы: создание (lock = threading.Lock()), захват блокировки (lock.acquire()), освобождение блокировки (lock.release()).

2) В чём отличие работы с RLock-объектом от работы с Lock-объектом?

Ответ: RLock – рекурсивная блокировка, позволяющая одному и тому же потоку захватывать блокировку несколько раз без блокировки самого себя. Полезно, когда один поток должен повторно войти в критическую секцию.

3) Как выглядит порядок работы с условными переменными?

Ответ: условные переменные позволяют потокам ждать наступления определенного состояния и уведомлять другие потоки о его наступлении. Порядок работы включает создание условной переменной, использование методов wait() для ожидания события и notify()/notify_all() для оповещения ожидающих потоков о наступлении события.

4) Какие методы доступны у объектов условных переменных?

Ответ: у объектов условных переменных доступны методы wait(), wait_for(), notify() и notify_all().

5) Каково назначение и порядок работы с примитивом синхронизации “семафор”?

Ответ: назначение семафора – ограничение доступа к ресурсу определённому числу потоков. Порядок работы включает создание семафора с начальным значением, использование метода acquire() для захвата семафора и release() для его освобождения.

6) Каково назначение и порядок работы с примитивом синхронизации “событие”?

Ответ: назначение события – управление синхронизацией потоков посредством сигналов. Порядок работы включает создание события, использование методов set() для установки флага события, clear() для сброса флага и wait() для ожидания события.

7) Каково назначение и порядок работы с примитивом синхронизации “таймер”?

Ответ: назначение таймера – выполнение функции через заданный промежуток времени. Порядок работы включает создание таймера с указанием функции и времени задержки, запуск таймера методом start(), и возможная остановка методом cancel().

8) Каково назначение и порядок работы с примитивом синхронизации “барьер”?

Ответ: назначение барьера – синхронизация определённого количества потоков, чтобы они все достигли определённой точки выполнения. Порядок работы включает создание барьера с указанным количеством потоков, использование метода wait() потоками для ожидания других потоков на барьере.

9) Общий вывод о применении тех или иных примитивов синхронизации в зависимости от решаемой задачи?

Ответ: методы синхронизации необходимо выбирать относительно поставленной задачи. Так, Lock используют для простого взаимного исключения, RLock для рекурсивного захвата, условные переменные для ожидания событий, семафоры для ограничения доступа к ресурсу, события для оповещения о состоянии, таймеры для выполнения функций через время, а барьеры для синхронизации точек выполнения нескольких потоков.

Вывод: в ходе выполнения лабораторной работы, приобретены навыки использования примитивов синхронизации в языке программирования Python версии 3.x.