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 №9
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: Управление потоками в Python.

Цель: приобрести навыки написания многопоточных приложений на языке программирования Python версии 3.x.

Порядок выполнения работы:

1. Выполнено индивидуальное задание – 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.

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

from threading import Lock, Thread


'''Варианты №3 и №2(По списку 28)
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):
    # 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
    # Блокировка доступа к общему ресурсу перед записью результата.
    with lock:
        results["second"] = s


def third_var(results):
```

Рисунок 1. Код индивидуального задания

```
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) Что такое синхронность и асинхронность?

Ответ: синхронность означает выполнение задач последовательно, одна за другой, где каждая задача должна завершиться, прежде чем начнется следующая. Асинхронность позволяет задачам выполняться независимо друг от друга, что означает, что программа может продолжать выполнение других

задач, пока одна из них ожидает завершения (например, ввода-вывода или ответа от сервера).

2) Что такое параллелизм и конкурентность?

Ответ: параллелизм – выполнение нескольких задач одновременно на разных физических ядрах процессора. Конкурентность – управление несколькими задачами, которые могут прогрессировать независимо друг от друга (Это не обязательно означает одновременное выполнение задач. Скорее чередование выполнения их частей таким образом, чтобы они казались выполняемыми одновременно).

3) Что такое GIL? Какое ограничение накладывает GIL?

Ответ: GIL (Global Interpreter Lock) – глобальная блокировка интерпретатора в языке Python, которая позволяет выполнять только один поток байт-кода Python в любой момент времени. GIL ограничивает производительность многопоточных Python программ, так как потоки не могут выполняться параллельно на нескольких ядрах процессора.

4) Каково назначение класса Thread?

Ответ: класс Thread в Python используется для создания и управления потоками. Он позволяет запускать функции в отдельных потоках, что позволяет выполнять задачи параллельно или конкурентно.

5) Как реализовать в одном потоке ожидание завершения другого потока?

Ответ: для ожидания завершения другого потока используется метод join().

6) Как проверить факт выполнения потоком некоторой работы?

Ответ: чтобы проверить, выполняется ли поток в данный момент, можно использовать метод is_alive().

7) Как реализовать приостановку выполнения потока на некоторый промежуток времени?

Ответ: для приостановки выполнения потока используется функция time.sleep(секунды).

8) Как реализовать принудительное завершение потока?

Ответ: в стандартной библиотеке Python нет безопасного и прямого способа принудительного завершения потока. Потоки следует проектировать так, чтобы они сами завершались при необходимости.

9) Что такое потоки-демоны? Как создать поток-демон?

Ответ: потоки-демоны – потоки, которые автоматически завершаются, когда завершается основной поток программы. Создать поток-демон можно, установив его свойство daemon в True перед запуском потока.

Вывод: в ходе выполнения лабораторной работы, приобретены навыки написания многопоточных приложений на языке программирования Python версии 3.x.