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

Институт цифрового развития
Кафедра инфокоммуникаций

«Лабораторная работа 2.23 Управление потоками в Python»

ОТЧЕТ
по лабораторной работе №26
дисциплины
«Основы программной инженерии»

Выполнил:
Луценко Дмитрий Андреевич
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил:
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Лабораторная работа 2.23 Управление потоками в Python

Цель работы: приобретение навыков написания многопоточных приложений на языке программирования Python версии 3.x.

Ход работы:

Индивидуальное задание

С использованием многопоточности для заданного значения x найти сумму ряда S с точностью члена ряда по абсолютному значению $\varepsilon = 10^{-7}$ и произвести сравнение полученной суммы с контрольным значением функции y для двух бесконечных рядов. Номера вариантов необходимо уточнить у преподавателя:

18. $$S = \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1} = x + \frac{x^3}{3} + \frac{x^5}{5} + \ldots;\ x = 0,35;\ y = \ln\sqrt{\frac{1+x}{1-x}}.$$

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


from threading import Thread
from math import log, sqrt


EPS = .0000001


def sum_func(x):
    summ = x
    prev = 0
    i = 1
    while abs(summ - prev) > EPS:
        prev = summ
        summ += x ** (i * 2 + 1) / (i * 2 + 1)
        i += 1

    print(f"Sum is {summ}")


def check_func(x):
    res = log(sqrt((1 + x) / (1 - x)))
    print(f"Check: {res}")


if __name__ == '__main__':
    x = 0.35
    th1 = Thread(target=sum_func(x))
    th2 = Thread(target=check_func(x))
    th1.start()
    th2.start()
```

Листинг 1 – Индивидуальное задание

```
Sum is 0.3654437434397286
Check: 0.3654437542713962
```

Рисунок 1 – Результат работы программы

Вывод: приобретены навыки написания многопоточных приложений на языке программирования Python версии 3.x.

Ответы на контрольные вопросы:

1. Что такое синхронность и асинхронность? Синхронное выполнение программы подразумевает последовательное выполнение операций. Асинхронное – предполагает возможность независимого выполнения задач.

2. Что такое параллелизм и конкурентность? Конкурентность предполагает выполнение нескольких задач одним исполнителем. Параллельность предполагает параллельное выполнение задач разными исполнителями.

3. Что такое GIL? Какое ограничение накладывает GIL? GIL — это аббревиатура от Global Interpreter Lock – глобальная блокировка интерпретатора. Он является элементом эталонной реализации языка Python, которая носит название CPython. Суть GIL заключается в том, что выполнять байт код может только один поток. Это нужно для того, чтобы упростить работу с памятью (на уровне интерпретатора) и сделать комфортной разработку модулей на языке C. Это приводит к некоторым особенностям, о которых необходимо помнить. Условно, все задачи можно разделить на две большие группы: в первую входят те, что преимущественно используют процессор для своего выполнения, например, математические, их ещё называют CPU-bound, во вторую – задачи работающие с вводом выводом (диск, сеть и т.п.), такие задачи называют IO-bound. Если вы запустили в одном интерпретаторе несколько потоков, которые в основном используют процессор, то скорее всего получите общее замедление работы, а не прирост производительности. Пока выполняется одна задача, остальные простаивают (из-за GIL), переключение происходит через определенные промежутки времени. Таким образом, в каждый конкретный момент времени, будет выполняться только один поток, несмотря на то, что у вас может быть многоядерный процессор (или многопроцессорный сервер), плюс ко всему,

будет тратиться время на переключение между задачами. Если код в потоках в основном выполняет операции ввода-вывода, то в этом случае ситуация будет в вашу пользу. В CPython все стандартные библиотечные функций, которые выполняют блокирующий ввод-вывод, освобождают GIL, это дает возможность поработать другим потокам, пока ожидается ответ от ОС.

4. Каково назначение класса Thread? За создание, управление и мониторинг потоков отвечает класс Thread из модуля threading.

5. Как реализовать в одном потоке ожидание завершения другого потока? Если необходимо дождаться завершения работы потока(ов) перед тем как начать выполнять какую-то другую работу, то воспользуйтесь методом join()

6. Как проверить факт выполнения потоком некоторой работы? Для того чтобы определить выполняет ли поток какую-то работу или завершился используется метод is_alive().

7. Как реализовать приостановку выполнения потока на некоторый промежуток времени? С помощью метода sleep() из модуля time.

8. Как реализовать принудительное завершение потока? В Python у объектов класса Thread нет методов для принудительного завершения работы потока. Один из вариантов решения этой задачи – это создать специальный флаг, через который потоку будет передаваться сигнал остановки. Доступ к такому флагу должен управляться объектом синхронизации.

```
lock.acquire()
if stop_thread is True:
    break lock.release()
```

9. Что такое потоки-демоны? Как создать поток-демон? Для того, чтобы потоки не мешали остановке приложения (т.е. чтобы они останавливались вместе с завершением работы программы) необходимо при создании объекта Thread аргументу daemon присвоить значение True, либо после создания потока, перед его запуском присвоить свойству deamon значение True.

```
th = Thread(target=func, deamon=True)
```