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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №9
дисциплины «Анализ данных»

Вариант №2

Выполнила:
Беседина Инга Олеговна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника», очная
форма обучения

_____
(подпись)

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

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Управление потоками в Python

Цель: 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:

$$S = \sum_{n=0}^{\infty} x^n = 1 + x + x^2 + x^3 + \ldots;\ x = 0,7;\ y = \frac{1}{1-x}.$$

$$S = \sum_{n=0}^{\infty} (-1)^n \frac{x^n}{2^{n+1}} = \frac{1}{2} - \frac{x}{4} + \frac{x^2}{8} - \ldots;\ x = 1,2;\ y = \frac{1}{2+x}.$$

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


"""
Вариант 2 и 3.
С использованием многопоточности для заданного значения x
найти сумму ряда S с точностью члена ряда по абсолютному значению e=10^-7
и произвести сравнение полученной суммы с контрольным значением функции
для двух бесконечных рядов.
"""


from threading import Thread, Lock


lock = Lock()


def calc_sum1(x, eps, s):
    sum1 = 0
    n = 0
    while True:
        el = xn
        if abs(el) < eps:
            break
        sum1 += el
        n += 1
    with lock:
```

```
        s["s1"] = sum1


def calc_sum2(x, eps, s):
    sum2 = 0
    n = 0
    while True:
        t = n + 1
        el = (-1)  n * xn / 2t
        if abs(el) < eps:
            break
        else:
            sum2 += el
            n += 1
    with lock:
        s["s2"] = sum2


def main():
    s = {}
    e = 1e-7

    x1 = 0.7
    y1 = 1/(1-x1)

    x2 = 1.2
    y2 = 1/(2+x2)

    thread1 = Thread(target=calc_sum1, args=(x1, e, s))
    thread2 = Thread(target=calc_sum2, args=(x2, e, s))

    thread1.start()
    thread2.start()

    thread1.join()
    thread2.join()

    s1 = s["s1"]
    s2 = s["s2"]

    print(f"Вариант №2. Сумма ряда S: {s1}")
    print(f"Контрольное значение функции для бесконечного ряда: {y1}")

    print(f"Вариант №3. Сумма ряда S: {s2}")
    print(f"Контрольное значение функции для бесконечного ряда: {y2}")
```

```
if __name__ == "__main__":
    main()
```

```
Вариант №2. Сумма ряда S: 3.333333083650555
Контрольное значение функции для бесконечного ряда: 3.333333333333333
Вариант №3. Сумма ряда S: 0.31250004145136007
Контрольное значение функции для бесконечного ряда: 0.3125
```

Рисунок 1. Результат выполнения программы

Контрольные вопросы:

1. Синхронность и асинхронность:

- Синхронность: Программа работает синхронно, когда задачи выполняются последовательно, одна за другой. Каждая задача должна завершиться, прежде чем начнется следующая.

- Асинхронность: Программа работает асинхронно, когда задачи могут выполняться параллельно или в разных потоках. Это позволяет продолжать выполнение программы, не ожидая завершения каждой задачи.

2. Параллелизм и конкурентность:

- Параллелизм: Это способ выполнения нескольких задач одновременно на многоядерном процессоре для увеличения производительности.

- Конкурентность: Это способ организации выполнения нескольких задач одновременно, даже если у процессора только одно ядро. Задачи могут чередоваться в выполнении.

3. GIL (Global Interpreter Lock):

- GIL — это механизм в интерпретаторе Python, который обеспечивает только один поток исполнения Python кода в каждый момент времени. Это ограничение накладывает GIL на параллельную обработку в многопоточных приложениях на Python.

4. Класс Thread:

- Класс Thread в Python используется для создания и управления потоками выполнения. Он позволяет запускать функции или методы в отдельных потоках для параллельной обработки.

5. Ожидание завершения другого потока:

- Для ожидания завершения другого потока можно использовать метод join(). Вызовите join() для потока, который нужно дождаться, чтобы продолжить выполнение основного потока.

6. Проверка факта выполнения работы потоком:

- Для проверки факта выполнения работы потоком можно использовать флаги или переменные состояния, которые будут изменяться при завершении работы потока.

7. Приостановка выполнения потока на промежуток времени:

- Для приостановки выполнения потока на определенный промежуток времени можно использовать функцию time.sleep(seconds) из модуля time.

8. Принудительное завершение потока:

- В Python нет прямого способа принудительно завершить поток из-за GIL. Однако можно использовать флаги или переменные состояния для безопасного завершения работы потока.

9. Потоки-демоны:

- Потоки-демоны (daemon threads) — это потоки, которые работают в фоновом режиме и завершаются автоматически при завершении основного потока программы. Для создания потока-демона нужно установить атрибут daemon в значение True перед запуском потока.

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