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»

Выполнил:
Середа Кирилл Витальевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович

_____
(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

Ход работы:

Индивидуальное задание (Вариант 11, 12): С использованием многопоточности для заданного значения найти сумму ряда с точностью члена ряда по абсолютному значению и произвести сравнение полученной суммы с контрольным значением функции для двух бесконечных рядов.

Создал файл (ind.py) в котором было выполнено индивидуальное задание: созданы две функции расчета рядов (рис.1), в которых происходит также расчет контрольного значения

```
# 11 Вариант
def sum_1(target, x):

    a = sin(x)
    S, k = a, 2
    # Найти сумму членов ряда.
    while fabs(a) > EPS:
        coef = 2 * k - 1
        a = sin(coef * x)/ coef
        S += a
        k += 1

    target[0] = S
    print(pi/4)


# 12 Вариант
def sum_2(target, x):

    a = cos(x)
    S, k = a, 2
    # Найти сумму членов ряда.
    while fabs(a) > EPS:
        a = cos(k * x)/ k
        S += a
        k += 1

    target[1] = S
    print(-log(2 * sin(x / 2)))
```

Рисунок 1 – Функции расчета рядов

Вывод: в ходе выполнения практической работы были приобретены навыки написания многопоточных приложений на языке программирования Python

Контрольные вопросы:

1. **Синхронность и асинхронность**

Синхронность относится к выполнению задач в определенном порядке, где каждая задача ожидает завершения предыдущей перед началом своего выполнения. **Асинхронность** позволяет выполнению задач в любом порядке, без необходимости ожидания завершения предыдущих задач.

2. **Параллелизм и конкурентность**

Параллелизм относится к выполнению нескольких задач одновременно, фактически в одно и то же время.

Конкурентность позволяет задачам продвигаться вперед, даже если они выполняются не одновременно, но в пересекающиеся промежутки времени.

3. **GIL (Global Interpreter Lock)**

GIL – это механизм в Python, который ограничивает выполнение только одного потока Python-кода за раз из-за особенностей управления памятью в интерпретаторе CPython.

Это ограничение накладывает ограничение на параллельное выполнение потоков Python-кода.

4. **Назначение класса Thread**

Класс **Thread** в Python предназначен для создания и управления потоками выполнения.

5. **Ожидание завершения другого потока**

Для ожидания завершения другого потока в Python можно использовать метод **join()** для ожидания завершения работы другого потока.

6. **Проверка факта выполнения работы потоком**

Для проверки факта выполнения работы потоком в Python можно использовать метод **is_alive()**, который возвращает True, если поток выполняется, и False, если завершил свою работу.

7. **Приостановка выполнения потока**

Для приостановки выполнения потока на некоторый промежуток времени в Python можно использовать метод **sleep()** из модуля **time**.

8. **Принудительное завершение потока**

В Python нет прямого способа принудительно завершить поток. Однако, можно использовать флаги или другие механизмы для безопасной остановки потока.

9. **Потоки-демоны и создание потока-демона**

Потоки-демоны – это потоки, которые работают в фоновом режиме и завершаются, когда основной поток завершает свою работу.

Для создания потока-демона в Python, установите атрибут **daemon** потока в значение **True** перед его запуском.