

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 №2.23

Дисциплина: «Анализ данных»
Тема: «Управление потоками в Python»

Выполнил:
Епифанов Алексей Александрович
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 Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

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

Порядок выполнения работы:

1. Создал новый репозиторий, клонировал его, в нем создал ветку developer и перешел на нее.

2. Выполнил индивидуальное задание: С использованием многопоточности для заданного значения x найти сумму ряда S с точностью члена ряда по абсолютному значению $\epsilon=10^{-7}$ и произвести сравнение полученной суммы с контрольным значением функции y для двух бесконечных рядов. Варианты 9 и 10

9.
$$S = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots; \quad x = 1, 4; \quad y = \sin x.$$

10.
$$S = \sum_{n=0}^{\infty} \frac{x^{2n}}{(2n)!} = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots; \quad x = \frac{1}{2}; \quad y = \frac{e^x + e^{-x}}{2}.$$

Рисунок 1. Ряды 9 и 10. вариантов

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

```
import math
from threading import Thread
```

```
# 10 V
def sum1(x, eps, s_dict):
    s = 0
    n = 0
    while True:
        k = 2 * n
        term = x**k / math.factorial(k)
        if abs(term) < eps:
            break
        else:
            s += term
            n += 1

    s_dict["s1"] = s
```

```

# 9 V
def sum2(x, eps, s_dict):
    s = 0
    n = 0
    while True:
        k = 2 * n + 1
        term = (-1) ** n * x**k / math.factorial(k)
        if abs(term) < eps:
            break
        else:
            s += term
            n += 1

    s_dict["s2"] = s

def main():
    s = {}

    eps = 10**-7
    # 10 V
    x1 = 1 / 2
    y1 = (math.e**x1 + math.e**-x1) / 2
    # 9 V
    x2 = 1.4
    y2 = math.sin(x2)

    thread1 = Thread(target=sum1, args=(x1, eps, s))
    thread2 = Thread(target=sum2, args=(x2, eps, s))

    # Запуск потоков
    thread1.start()
    thread2.start()

    # Ожидание завершения потоков
    thread1.join()
    thread2.join()

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

    print(
        f"Сумма 10 Варианта: {s1},"
        f" Ожидаемое значение y1: {y1}, Разница: {abs(s1 - y1)}"
    )

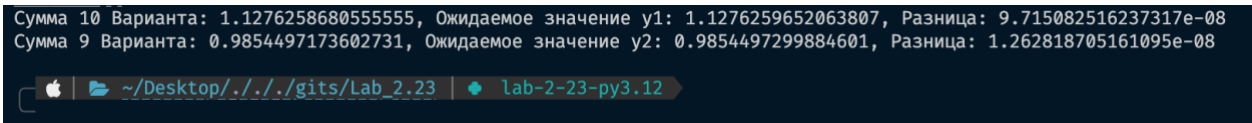
```

```

)
print(
    f"Сумма 9 Варианта: {s2},"
    f" Ожидаемое значение y2: {y2}, Разница: {abs(s2 - y2)}"
)

if __name__ == "__main__":
    main()

```



```

Сумма 10 Варианта: 1.1276258680555555, Ожидаемое значение y1: 1.1276259652063807, Разница: 9.715082516237317e-08
Сумма 9 Варианта: 0.9854497173602731, Ожидаемое значение y2: 0.9854497299884601, Разница: 1.262818705161095e-08
~/Desktop/./../.gits/Lab_2.23 lab-2-23-py3.12

```

Рисунок 1. Результат запуска всех тестов

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.

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.

4. Каково назначение класса Thread ?

За создание, управление и мониторинг потоков отвечает класс Thread из модуля threading. Поток можно создать на базе функции, либо реализовать свой класс – наследник Thread и переопределить в нем метод run().

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?

Для этого используется метод sleep() из модуля time с указанием количества мс

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.

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

Есть такая разновидность потоков, которые называются демоны (терминология взята из мира Unix-подобных систем). Python-приложение не будет закрыто до тех пор, пока в нем работает хотя бы один недемонический поток.

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 `daemon` значение `True`.

Вывод: в результате выполнения работы были получены навыки по написанию многопоточных приложений на языке программирования Python версии 3.x.