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:

Пустяков Андрей Сергеевич

2 курс, группа ИВТ-б-о-22-1,

09.03.01 «Информатика и вычислительная техника (профиль) «Программное обеспечение средств вычислительной
техники и автоматизированных систем», очная форма обучения

_____

(подпись)

Руководитель практики:

Воронкин Р. А. кандидат технических наук, доцент, доцент кафедры инфокоммуникаций

_____

(подпись)

Ставрополь, 2024 г.

Тема: Управление потоками в Python.

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

Ход работы:

Создание общедоступного репозитория на «GitHub», клонирование репозитория, редактирование файла «.gitignore», организация репозитория согласно модели ветвления «git-flow» (рис. 1).

```
C:\Users\Andrey\Desktop\Анализ_данных\9_лаба\Analysis_data_laboratory_work_9>git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
```

Рисунок 1 – Организация модели ветвления «git-flow»

Выполнение индивидуальных заданий:

Задание 1.

Необходимо с использованием многопоточности для заданного значения x найти сумму ряда S с точностью члена ряда по абсолютному значению и произвести сравнение полученной суммы с контрольным значением функции y(x) для двух бесконечных рядов.

Сумма ряда (Вариант 26 (1)):

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

Сумма ряда (Вариант 26 (2)):

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

Код программы данной задачи:

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

import math
from threading import Lock, Thread
```

```
import sympy as sp


"""
Необходимо с использованием многопоточности для заданного значения x найти
сумму ряда S
с точностью члена ряда по абсолютному значению и произвести сравнение
полученной суммы с
контрольным значением функции y(x) для двух бесконечных рядов (Вариант 26 (1
и 2)).
"""

E = 1e-7  # Точность
# Создание блокировки для синхронизации доступа к общему ресурсу.
lock = Lock()


def series1(x, eps, results):
    s = 0
    n = 0
    while True:
        term = x**n * sp.log(3)**n / math.factorial(n)
        if abs(term) < eps:
            break
        s += term
        n += 1
    with lock:
        results["series1"] = s


def series2(x, eps, results):
    s = 0
    n = 0
    while True:
        term = x**n
        if abs(term) < eps:
            break
        s += term
        n += 1
    with lock:
        results["series2"] = s


def main():
    results = {"series1": 0, "series2": 0}

    # Определение символа n
    n = sp.symbols('n')

    # Первое выражение: сумма от n=0 до бесконечности x^n (ln^n 3)/n! ; x=1
    x1 = 1
    control1 = sp.Sum(x1**n * sp.log(3)**n / sp.factorial(n), (n, 0,
sp.oo)).evalf()

    # Второе выражение: сумма от n=0 до бесконечности x^n ; x=0.7
    x2 = 0.7
    control2 = sp.Sum(x2**n, (n, 0, sp.oo)).evalf()

    thread1 = Thread(target=series1, args=(x1, E, results))
    thread2 = Thread(target=series2, args=(x2, E, results))

    thread1.start()
    thread2.start()
```

```
    thread1.join()
    thread2.join()

    sum1 = results["series1"]
    sum2 = results["series2"]

    print(f"x1 = {x1}")
    print(f"Sum of series 1: {sum1:.7f}")
    print(f"Control value 1: {control1:.7f}")
    print(f"Match 1: {round(sum1, 7) == round(control1, 7)}")

    print(f"x2 = {x2}")
    print(f"Sum of series 2: {sum2:.7f}")
    print(f"Control value 2: {control2:.7f}")
    print(f"Match 2: {round(sum2, 7) == round(control2, 7)}")


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

Результаты работы данной программы при заданных значениях x (рис. 2).

```
C:\Users\Andrey\anaconda3\envs\lab_9\python.exe C:\Users\Andrey\Des
x1 = 1
Sum of series 1: 2.9999999
Control value 1: 2.7182818
Match 1: False
x2 = 0.7
Sum of series 2: 3.3333331
Control value 2: 1.0000000
Match 2: False

Process finished with exit code 0
```

Рисунок 2 – Результаты работы программы с многопоточностью

Ответы на контрольные вопросы:

1. Что такое синхронность и асинхронность?

Синхронность: в синхронном выполнении задачи каждый шаг ожидает завершения предыдущего. То есть код выполняется последовательно, шаг за шагом. Асинхронность: в асинхронном выполнении задачи код может продолжать выполнение, не дожидаясь завершения предыдущего шага. Это

позволяет эффективнее использовать ресурсы и обрабатывать множество задач одновременно.

2. Что такое параллелизм и конкурентность?

Параллелизм: это выполнение нескольких задач одновременно, фактически в один и тот же момент времени. Каждая задача выполняется независимо от других.

Конкурентность: это координация выполнения нескольких задач. Задачи могут выполняться в разное время, но между ними существует взаимодействие.

3. Что такое GIL? Какое ограничение накладывает GIL?

GIL — это механизм, используемый в некоторых интерпретаторах, например, в CPython (стандартная реализация Python). Он предназначен для обеспечения безопасности в многопоточной среде, но ограничивает возможность использования нескольких ядер процессора для параллельного выполнения Python-кода.

4. Каково назначение класса Thread?

Класс Thread в языке программирования Python предоставляет средства для создания и управления потоками выполнения. Потоки представляют собой легковесные процессы, которые выполняются независимо друг от друга.

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

Можно использовать метод join() для ожидания завершения другого потока.

6. Как проверить факт выполнения потоком некоторой работы?

Это может зависеть от конкретной реализации, но обычно можно использовать флаги или другие механизмы для сигнализации о выполнении работы.

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

В Python можно использовать time.sleep(seconds) для приостановки выполнения потока на определенное количество секунд.

8. Как реализовать принудительное завершение потока?

В Python принудительное завершение потока не всегда рекомендуется, но можно использовать флаги или исключения для безопасного завершения.

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

Это потоки, которые выполняются в фоновом режиме и завершаются, когда основной поток завершается. В Python можно создать демонический поток, установив атрибут daemon объекта Thread в True.

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