

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
№2.23
дисциплины «Анализ данных»**

Выполнил:
Говоров Егор Юрьевич
2 курс, группа ИВТ-б-о-22-1,
18.05.2024 «Информатика
и вычислительная
техника», направленность
(профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

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

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема. Лабораторная работа 2.23 Управление потоками в Python

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

Ход работы

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python. Выполнил клонирование созданного репозитория.
2. Дополнил файл .gitignore необходимыми правилами.
3. Организовал созданный репозиторий в соответствии с необходимыми требованиями.
4. Добавил в файл README.md информацию о группе и ФИО студента, выполняющего лабораторную работу.
5. Выполнил индивидуальное задание. Привел в отчете скриншоты работы программы решения индивидуального задания.

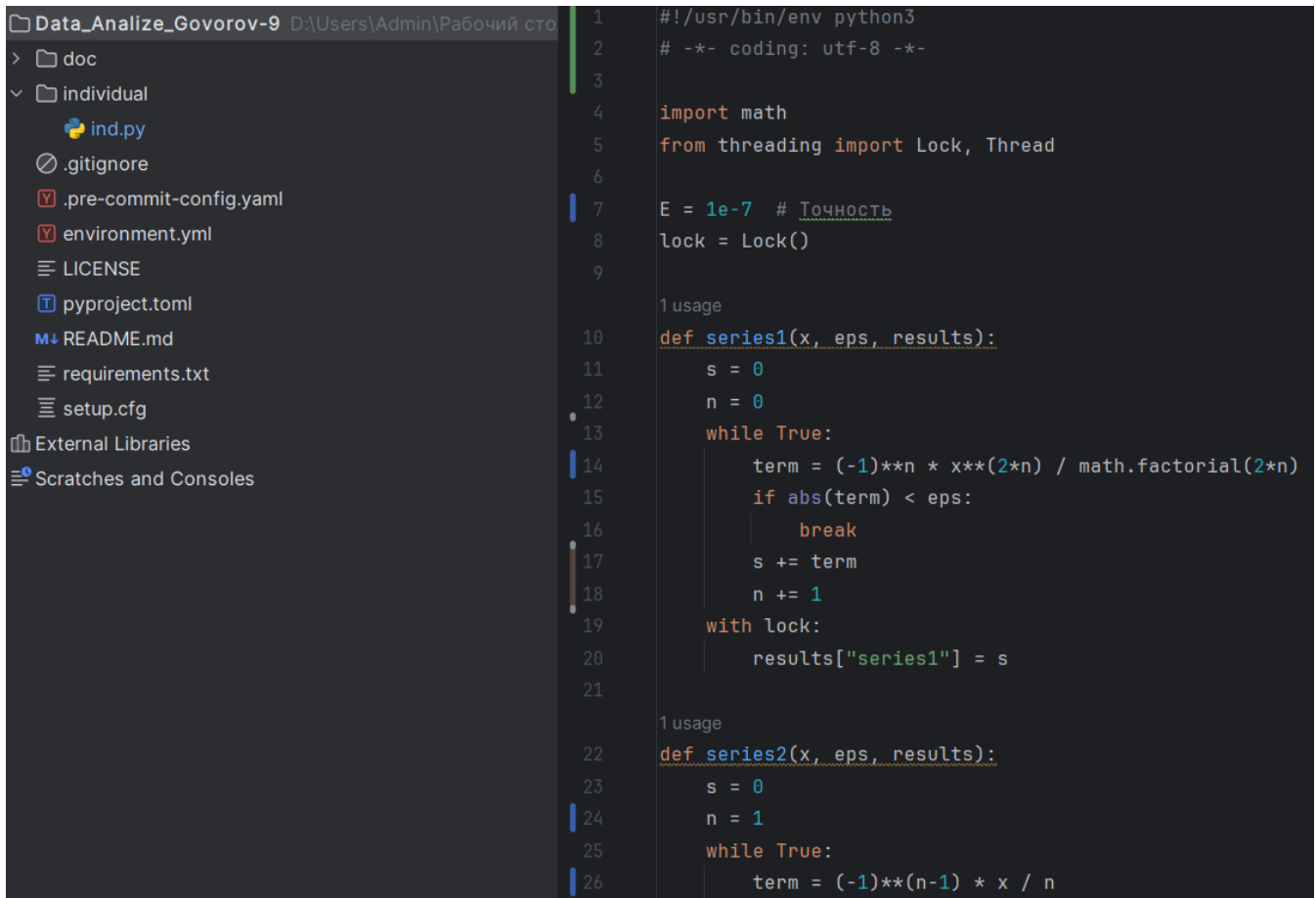
С использованием многопоточности для заданного значения x найти сумму ряда S с точностью члена ряда по абсолютному значению $E = 10e-7$ и произвести сравнение полученной суммы с контрольным значением функции бесконечного ряда. Вариант 5

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

Рисунок 1. Функция варианта 5

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

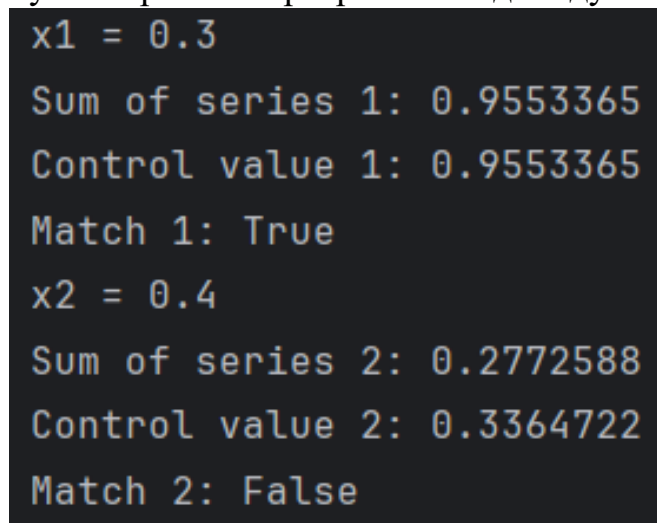
Рисунок 2. Функция варианта 6



The screenshot shows a code editor with a file explorer on the left and a Python script on the right. The file explorer shows a project named 'Data_Analyze_Govorov-9' with a subdirectory 'individual' containing 'ind.py'. The script 'ind.py' is open in the editor. It starts with a shebang and encoding declaration, followed by imports for 'math' and 'threading'. A constant 'E' is set to 1e-7, and a 'Lock' is created. Two functions, 'series1' and 'series2', are defined. 'series1' calculates the sum of the series $\sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}$ for a given 'x' and 'eps', and stores the result in 'results' under the key 'series1'. 'series2' calculates the sum of the series $\sum_{n=1}^{\infty} \frac{(-1)^{n-1} x^n}{n}$ for a given 'x' and 'eps'. The script uses a 'with lock:' block to ensure thread safety when updating 'results'.

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3
4import math
5from threading import Lock, Thread
6
7E = 1e-7 # Точность
8lock = Lock()
9
10usage
11def series1(x, eps, results):
12    s = 0
13    n = 0
14    while True:
15        term = (-1)**n * x**(2*n) / math.factorial(2*n)
16        if abs(term) < eps:
17            break
18        s += term
19        n += 1
20    with lock:
21        results["series1"] = s
22
23usage
24def series2(x, eps, results):
25    s = 0
26    n = 1
27    while True:
28        term = (-1)**(n-1) * x / n
```

Рисунок 2. Результат работы программы индивидуального задания



The screenshot shows the output of the program. It first sets x1 = 0.3 and calculates the sum of series 1 as 0.9553365. The control value is also 0.9553365, and the match is True. Then it sets x2 = 0.4 and calculates the sum of series 2 as 0.2772588. The control value is 0.3364722, and the match is False.

```
x1 = 0.3
Sum of series 1: 0.9553365
Control value 1: 0.9553365
Match 1: True
x2 = 0.4
Sum of series 2: 0.2772588
Control value 2: 0.3364722
Match 2: False
```

Рис 3. Результата вывода

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.