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:  Кожуховский Виктор Андреевич  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.

Тема: Управление потоками в Python

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

Порядок выполнения работы:

1. Изучил теоретический материал работы.

2. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

3. Выполнил клонирование созданного репозитория.

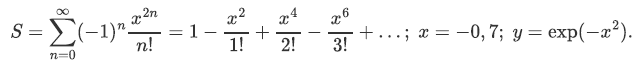
4. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.

5. Организовал свой репозиторий в соответствие с моделью ветвления git-flow.

6. Создал проект в папке репозитория.

8. Выполнил индивидуальное задание.

Используя многопоточность для заданного значения x, най­­­ти сумму ряда S с точностью до члена ряда по абсолютному значению = 10-7 и произвести сравнение полученной суммы с контрольным значением функции y для двух бесконечных рядов. Номер варианта 16 и 17.





Код:  
#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

import math

from threading import Thread, Lock

epsilon = 1e-7

lock = Lock()

def func(x, result):

sum = 0

n = 0

term = 1

while abs(term) > epsilon:

sum += term

n += 1

term = (-1)\*\*n \* x\*\*(2 \* n) / math.factorial(n)

with lock:

result.append(sum)

def func2(x, result):

sum = 0

n = 1

while True:

term = 1 / (2 \* n - 1) \* ((x - 1) / (x + 1))\*\*(2 \* n - 1)

if abs(term) < epsilon:

break

else:

sum += term

n += 1

with lock:

result.append(sum)

def main():

result1 = []

result2 = []

thread1 = Thread(target=func, args=(-0.7, result1))

thread2 = Thread(target=func2, args=(0.6, result2))

thread1.start()

thread2.start()

thread1.join()

thread2.join()

sum\_func = result1[0]

sum\_func2 = result2[0]

test1 = math.exp(-(-0.7)\*\*2)

test2 = 1/2 \* math.log(0.6)

print(f"Результат функции 1: {sum\_func}")

print(f"Контрольное значение для функции 1: {test1}")

print(f"Результат функции 2: {sum\_func2}")

print(f"Контрольное значение для функции 2: {test2}")

if abs(sum\_func - test1) < epsilon:

print("func: Верно.")

else:

print("func: Неверно.")

if abs(sum\_func2 - test2) < epsilon:

print("series\_solution: Верно.")

else:

print("series\_solution: Неверно.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

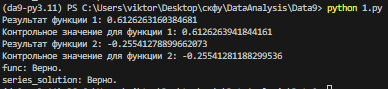


Рисунок 1. Выполнение кода индивидуального задания

9. Зафиксировал сделанные изменения в репозитории.

10. Добавил отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксировал изменения.

11. Выполнил слияние ветки для разработки с веткой master/main.

12. Отправил сделанные изменения на сервер GitHub.

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 deamon значение True.