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 потокамив 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 соответствие 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

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

Рисунок 1. Функция варианта 5

```python
import math
from threading import Lock, Thread

E = 10e-7
lock = Lock()

1 usage  new *
def series1(x, eps, results):
    s = 0
    n = 0
    x_pow = 1  # x^0
    factorial = 1  # 0!
    while True:
        term = ((-1)**n * x_pow) / factorial
        if abs(term) < eps:
            break
        else:
            s += term
            n += 1
            x_pow *= x * x  # x^(2*n)
            factorial *= (2*n) * (2*n - 1)  # (2*n)!
    with lock:
        results["series1"] = s

1 usage  new *
def series2(x, eps, results):
    s = 0
    n = 0
    x_pow = 1  # x^0
    factorial = 1  # 0!
    while True:
        term = ((-1)**n * x_pow) / factorial
        if abs(term) < eps:
```

Рисунок 2. Результат работы программы индивидуального задания

```
x = 0.3
Sum of series 1: 0.9553365
Control value: 0.9553365
Match 1: True
Sum of series 2: 0.9553365
Control value: 0.9553365
Match 2: True
```

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 deamon значение True.

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