МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе № 18

Управление потоками в Python

По дисциплине «Технологии программирования и алгоритмизация»

Выполнил студент группы ИВТ-б-о-20-1

Бобров Н. В. « »_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил Воронкин Р. А. _____

(подпись)

Ставрополь 2022

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

Ход работы:

1. Создал общедоступный репозиторий и клонировал его на своей локальный сервер.

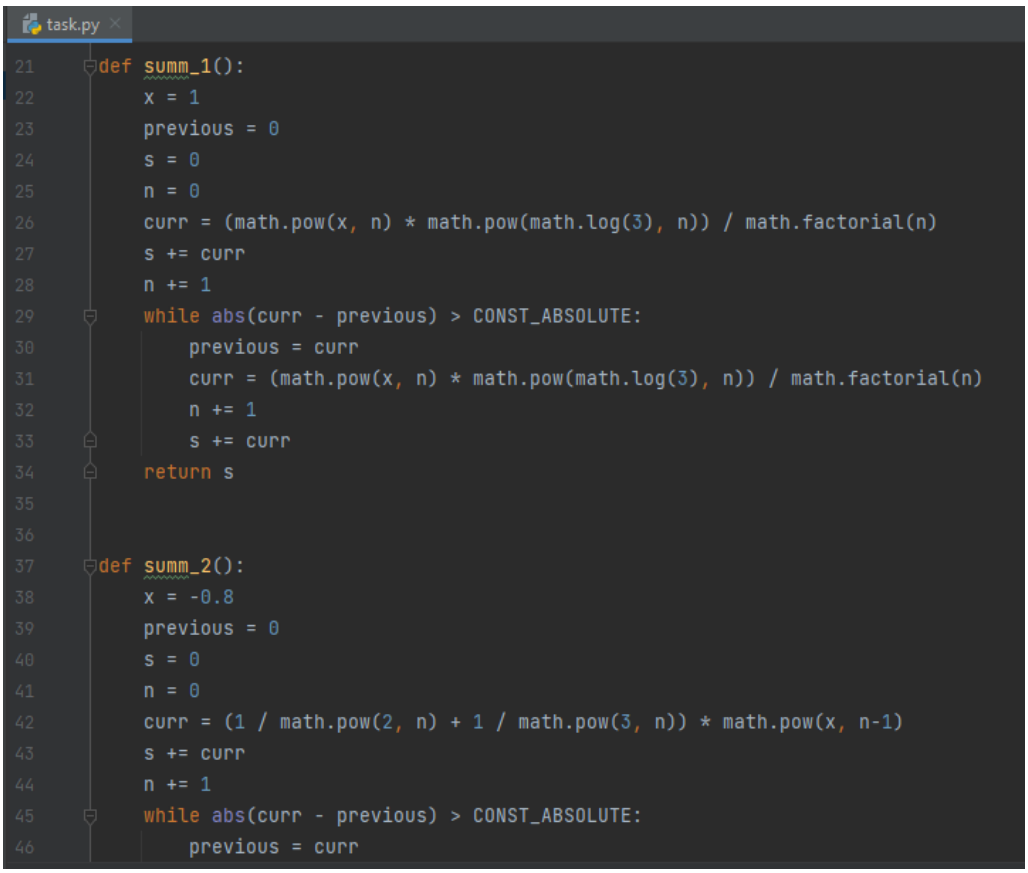
2. Изучив методичку к лабораторной работе, приступил непосредственно к её выполнению.

3. Проработал пример.

```
def infinit_worker():
    print("Start infinit_worker()")
    while True:
        print("--> thread work")
        lock.acquire()

        if stop_thread is True:
            break
        lock.release()
        sleep(0.1)

    print("Stop infinit_worker()")


# Create and start thread
th = Thread(target=infinit_worker)
th.start()
sleep(2)
```

Рисунок 1 – Отработка примера

4. Затем приступил к выполнению индивидуального задания.



```python
def summ_1():
    x = 1
    previous = 0
    s = 0
    n = 0
    curr = (math.pow(x, n) * math.pow(math.log(3), n)) / math.factorial(n)
    s += curr
    n += 1
    while abs(curr - previous) > CONST_ABSOLUTE:
        previous = curr
        curr = (math.pow(x, n) * math.pow(math.log(3), n)) / math.factorial(n)
        n += 1
        s += curr
    return s


def summ_2():
    x = -0.8
    previous = 0
    s = 0
    n = 0
    curr = (1 / math.pow(2, n) + 1 / math.pow(3, n)) * math.pow(x, n-1)
    s += curr
    n += 1
    while abs(curr - previous) > CONST_ABSOLUTE:
        previous = curr
```

Рисунок 2 – Код, реализующий работу программы

```
PS C:\TPA\2.23> python task.py
Результат сравнения -5.915392620181592e-10
Результат сравнения -4.879699252047998
PS C:\TPA\2.23>
```

Рисунок 3 – Результат работы программы

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. Из примера с готовкой: один человек варит картошку и прибирается, при этом, в процессе, он может переключаться: немного прибрался, пошел помешал-посмотрел на картошку, и делает он это до тех пор, пока все не будет готово.

Параллельность предполагает параллельное выполнение задач разными исполнителями: один человек занимается готовкой, другой приборкой.

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.

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 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?

У метода join() есть параметр timeout, через который задается время ожидания завершения работы потоков.

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?

Поток демона – это тип потока, который может работать независимо в фоновом режиме. Эти типы потоков выполняются независимо от основного потока. Поэтому они называются неблокирующими потоками.

Чтобы создать такой поток необходимо при создании объекта Thread аргументу daemon присвоить значение True, либо после создания потока, перед его запуском присвоить свойству deamon значение True.

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