МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчёт по лабораторной работе №2.23 по дисциплине:

Основы программной инженерии

Выполнила:
студент группы ПИЖ-б-о-20-1
Лазарева Дарья Олеговна

Проверил:
доцент кафедры инфокоммуникаций
Романкин Р.А.

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ:

1. Потоки в Python

```python
def func():
    for i in range(5):
        print(f"from child thread: {i}")
        sleep(0.5)


if __name__ == '__main__':
    th = Thread(target=func)
    th.start()

    for i in range(5):
        print(f"from main thread: {i}")
    sleep(1)
```

if __name__ == '__main__'

ex1

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.23/ex1.py
from child thread: 0from main thread: 0

from main thread: 1
from main thread: 2
from main thread: 3
from main thread: 4
from child thread: 1
from child thread: 2
from child thread: 3
from child thread: 4
```

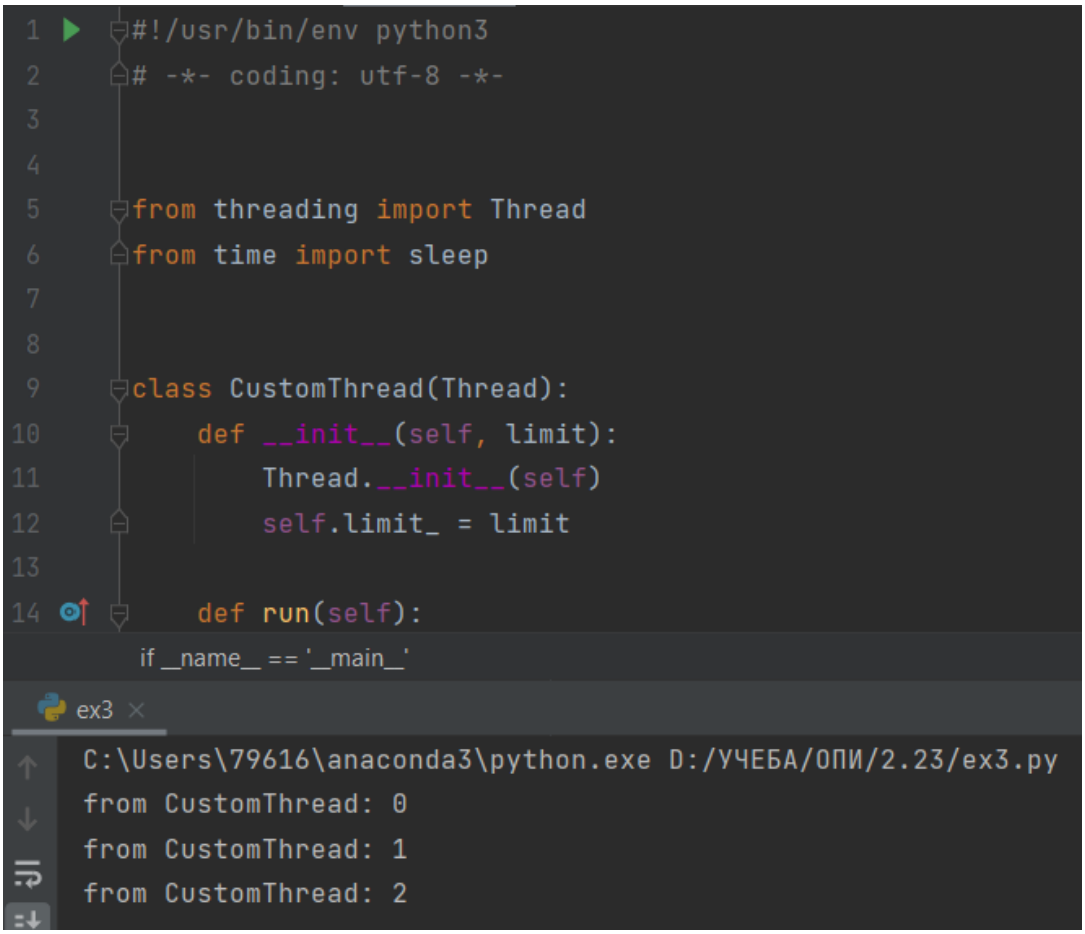
```python
def func():
    for i in range(5):
        print(f"from child thread: {i}")
        sleep(0.5)


if __name__ == '__main__':
    th = Thread(target=func)
    print(f"thread status: {th.is_alive()}")
    th.start()
    print(f"thread status: {th.is_alive()}")
    sleep(5)
    print(f"thread status: {th.is_alive()}")
```

if __name__ == '__main__'

ex2

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.23/ex2.py
thread status: False
from child thread: 0thread status: True

from child thread: 1
from child thread: 2
from child thread: 3
from child thread: 4
thread status: False
```

2. Создание классов наследников от Thread

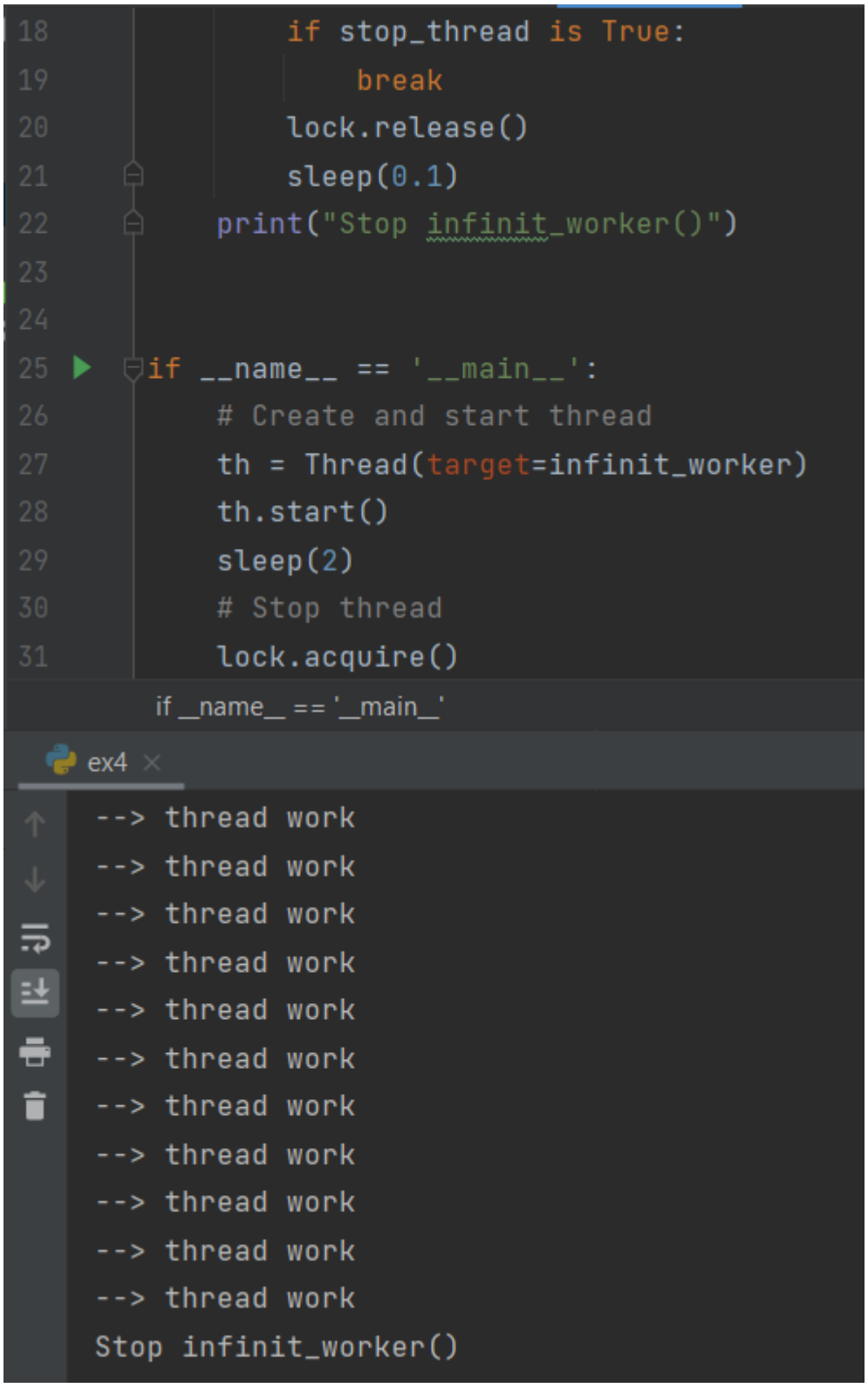


```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


from threading import Thread
from time import sleep


class CustomThread(Thread):
    def __init__(self, limit):
        Thread.__init__(self)
        self.limit_ = limit

    def run(self):
```

```
if __name__ == '__main__'
```

```
ex3
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.23/ex3.py
from CustomThread: 0
from CustomThread: 1
from CustomThread: 2
```

3. Принудительное завершение работы потока



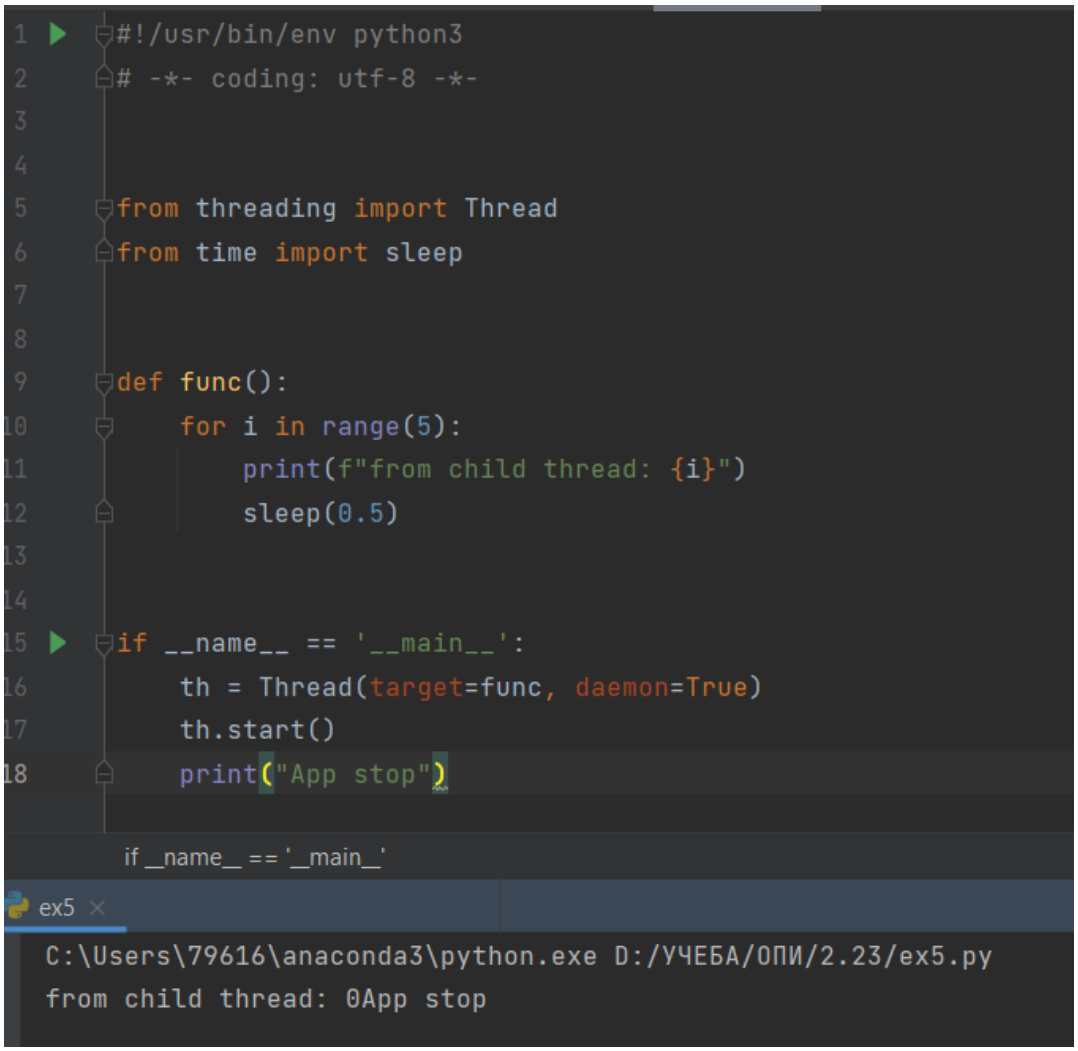
```
            if stop_thread is True:
                break
            lock.release()
            sleep(0.1)
        print("Stop infinit_worker()")


if __name__ == '__main__':
    # Create and start thread
    th = Thread(target=infinit_worker)
    th.start()
    sleep(2)
    # Stop thread
    lock.acquire()
```

if __name__ == '__main__'

ex4

```
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
--> thread work
Stop infinit_worker()
```

4. Потоки-демоны



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


from threading import Thread
from time import sleep


def func():
    for i in range(5):
        print(f"from child thread: {i}")
        sleep(0.5)


if __name__ == '__main__':
    th = Thread(target=func, daemon=True)
    th.start()
    print("App stop")
```

```
if __name__ == '__main__'
ex5
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.23/ex5.py
from child thread: 0App stop
```

Выполнение индивидуального задания (вариант 9):

С использованием многопоточности для заданного значения x найти сумму ряда S с точностью члена ряда по абсолютному значению $\varepsilon = 10^{-7}$ и произвести сравнение полученной суммы с контрольным значением функции для двух бесконечных рядов.

$$S = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots; \; x = 1,4; \; y = \sin x.$$

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


from threading import Thread
from math import factorial, sin


eps = .0000001


def inf_sum(x, check, num):
    summa = 1
    i = 1
    prev = 0
    while abs(summa - prev) > eps:
        prev = summa
        summa += ((-1)**i * x**(2*i + 1)) / factorial(2 * i + 1)
        i += 1
    print(f"The sum number {num} is: {summa}")
    print(f"Check: sin{x} = {-check}")


if __name__ == '__main__':
    checksum1 = sin(0)
    thread1 = Thread(target=inf_sum, args=(0, checksum1, 1))
    thread1.start()
```

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.23/idz.py
The sum number 1 is: 1.0
Check: sin0 = -0.0
```

Ответы на 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 — глобальная блокировка интерпретатора. Суть GIL заключается в том, что выполнять байт код может только один поток. Это нужно для того, чтобы упростить работу с памятью и сделать комфортной разработку модулей на языке C. В CPython все стандартные библиотечные функций, которые выполняют блокирующий ввод-вывод, освобождают GIL, это дает возможность поработать другим потокам, пока ожидается ответ от ОС.

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 другогопотока?

Если необходимо дождаться завершения работы потока(-ов) перед тем как начать выполнять какую-то другую работу, то воспользуйтесь методом join().У join() есть параметр timeout, через который задается время ожидания завершения работы потоков.

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?

С помощью метода sleep() из модуля time.

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.

```
lock.acquire()
if stop_thread is True:
break lock.release()
```

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

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.

```
th = Thread(target=func, daemon=True)
```