**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное**

**образовательное учреждение высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Отчет по лабораторной работе №2.25**

**Работа с переменными окружения в Python3**

**по дисциплине «Управление потоками в Python»**

Выполнил студент группы ИВТ-б-о-20-1

Хашиев Х.М. «  »_____20__г.

Подпись студента_____

Работа защищена «  »_____20__г.

Проверил Воронкин Р.А. _____

(подпись

Ставрополь 2022

**Цель работы:** приобретение навыков с управлением потоками с помощью языка программирования Python версии 3.x.

## Ход работы: Примеры

**https://github.com/Mirror-Shard/L2.25**

1. Изучил теоретический материал и приступил к выполнению примеров:

```python
from multiprocessing import Process
from time import sleep


def func(name):
    counter = 0
    while True:
        print(f"proc {name}, counter = {counter}")
        counter += 1
        sleep(0.1)


if __name__ == "__main__":
    # Указание на то, что процесс демон при создании объекта класса Process
    proc1 = Process(target=func, args=("proc1",), daemon=True)
    # Указание на то, что процесс демон через свойство daemon
    proc2 = Process(target=func, args=("proc2",))
    proc2.daemon = True
    # Запуск процессов
    proc1.start()
    proc2.start()
    sleep(0.3)
    # Процессы proc1 и proc2 завершаться вместе с родительским процессом
    # ...
```

Рисунок 1 – Код примера

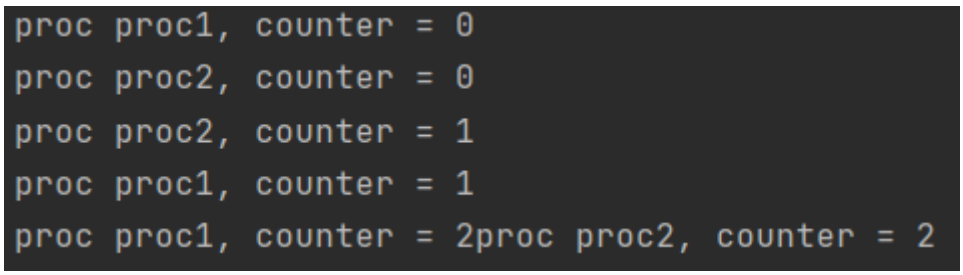


Рисунок 2 – Результат работы примера

**Задание 1**

С использованием многопоточности для заданного значения найти сумму ряда с точностью члена ряда по абсолютному значению и произвести сравнение полученной суммы с контрольным значением функции для двух бесконечных рядов.

1. Написал программу для вычисления суммы ряда указанной точности с использованием многопоточности:

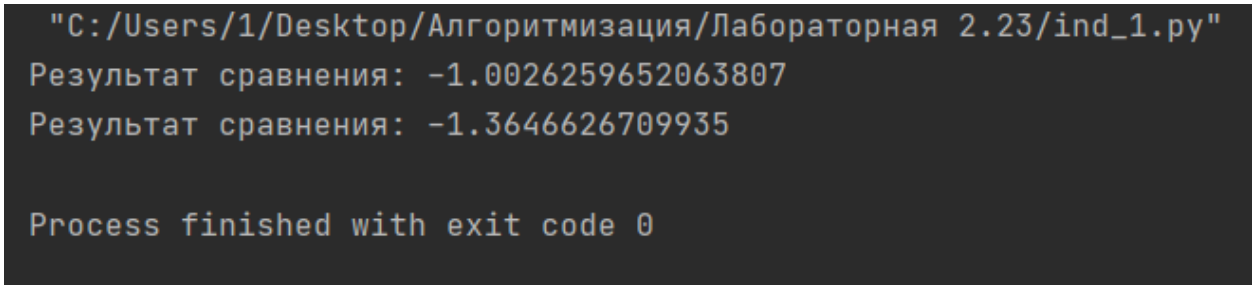


Рисунок 3 – Результат выполнения тестов

**Контрольные вопросы:**

1. Как создаются и завершаются процессы в Python?

Классом, который отвечает за создание и управление процессами является Process из пакета multiprocessing. Он совместим по сигнатурам методов и конструктора с threading.Thread, это сделано для более простого перехода от многопотокового приложения к многопроцессному. За ожидание завершения работы процесса(ов) отвечает метод join, со следующей сигнатурой: join([timeout]).

При вывозе метода join() выполнение программы будет остановлено до тех пор пока соответствующий процесс не завершит работу. Параметр timeout отвечает за время ожидания завершения работы процесса, если указанное время прошло, а процесс еще не завершился, то ожидание будет

прервано и выполнение программы продолжится дальше. В случае, если метод join() завершился по таймауту или в результате того, что процесс был завершен аварийно (терминирован), то он вернет None.

2. В чем особенность создания классов-наследников от Process?

В классе наследнике от Process необходимо переопределить метод run() для того, чтобы он (класс) соответствовал протоколу работы с процессами.

3. Как выполнить принудительное завершение процесса?

В отличии от потоков, работу процессов можно принудительно завершить, для этого класс Process предоставляет набор методов:

– terminate() - принудительно завершает работу процесса. В Unix отправляется команда SIGTERM, в Windows используется функция TerminateProcess().

– kill() - метод аналогичный terminate() по функционалу, только вместо SIGTERM в Unix будет отправлена команда SIGKILL.

4. Что такое процессы-демоны? Как запустить процесс-демон?

Процессы демоны по своим свойствам похожи на потоки-демоны, их суть заключается в том, что они завершают свою работу, если завершился родительский процесс.

Указание на то, что процесс является демоном должно быть сделано до его запуска (до вызова метода start()). Для демонического процесса запрещено самостоятельно создавать дочерние процессы. Эти процессы не являются демонами (сервисами) в понимании Unix, единственное их свойство – это завершение работы вместе с родительским процессом.

Указать на то, что процесс является демоном можно при создании экземпляра класса через аргумент daemon, либо после создания через свойство daemon.

**Вывод:** в ходе работы приобрёл навыки по работе с управлением потоками при написании программ с помощью языка программирования Python версии 3.