Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЁТ

по лабораторной работе №2.24

Дисциплина: «Анализ данных»
Тема: «Управление процессами в Python»

Выполнил:
Епифанов Алексей Александрович
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.

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

Порядок выполнения работы:

1. Создал новый репозиторий, клонировал его, в нем создал ветку developer и перешел на нее.

2. Выполнил индивидуальное задание: Для своего индивидуального задания лабораторной работы 2.23 необходимо организовать конвейер, в котором сначала в отдельном потоке вычисляется значение первой функции, после чего результаты вычисления должны передаваться второй функции, вычисляемой в отдельном потоке. Потоки для вычисления значений двух функций должны запускаться одновременно.

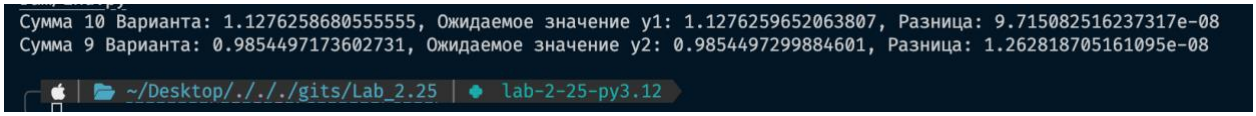

Рисунок 1. Результат работы

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


import math
from multiprocessing import Barrier, Manager, Process


# 10 V
def sum1(x, eps, s_dict, br, lock):
    s = 0
    n = 0
    while True:
        k = 2 * n
        term = x**k / math.factorial(k)
        if abs(term) < eps:
            break
        else:
            s += term
            n += 1
    with lock:
        s_dict["s1"] = s
    br.wait()
```

```
# 9 V
def sum2(x, eps, s_dict, br, lock):
    s = 0
    n = 0
    while True:
        k = 2 * n + 1
        term = (-1) ** n * x**k / math.factorial(k)
        if abs(term) < eps:
            break
        else:
            s += term
            n += 1
    with lock:
        s_dict["s2"] = s
    br.wait()


def compair(s, y1, y2, br):
    br.wait()
    s1 = s["s1"]
    s2 = s["s2"]

    print(
        f"Сумма 10 Варианта: {s1},"
        f" Ожидаемое значение y1: {y1}, Разница: {abs(s1 - y1)}"
    )
    print(
        f"Сумма 9 Варианта: {s2},"
        f" Ожидаемое значение y2: {y2}, Разница: {abs(s2 - y2)}"
    )


def main(m):
    s = m.dict()

    br = Barrier(3)
    lock = m.Lock()

    eps = 10**-7
    # 10 V
    x1 = 1 / 2
    y1 = (math.e**x1 + math.e**-x1) / 2
    # 9 V
    x2 = 1.4
```

```
    y2 = math.sin(x2)

    process1 = Process(target=sum1, args=(x1, eps, s, br, lock))
    process2 = Process(target=sum2, args=(x2, eps, s, br, lock))
    process3 = Process(target=compair, args=(s, y1, y2, br))

    # Запуск потоков
    process1.start()
    process2.start()
    process3.start()

    process1.join()
    process2.join()
    process3.join()


if __name__ == "__main__":
    with Manager() as manager:
        main(manager)
```

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. Помимо одноименных с Thread методов, класс Process дополнительно предоставляет ряд своих.

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. Ниже представлен пример с реализацией этого подхода.

3. 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 результате выполнения работы были приобретены навыки написания многозадачных приложений на языке программирования Python версии 3.x.