

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

Управление процессами в Python

По дисциплине «Теории программирования и алгоритмизации»

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 многозадачных приложений на языке программирования Python версии 3.x.

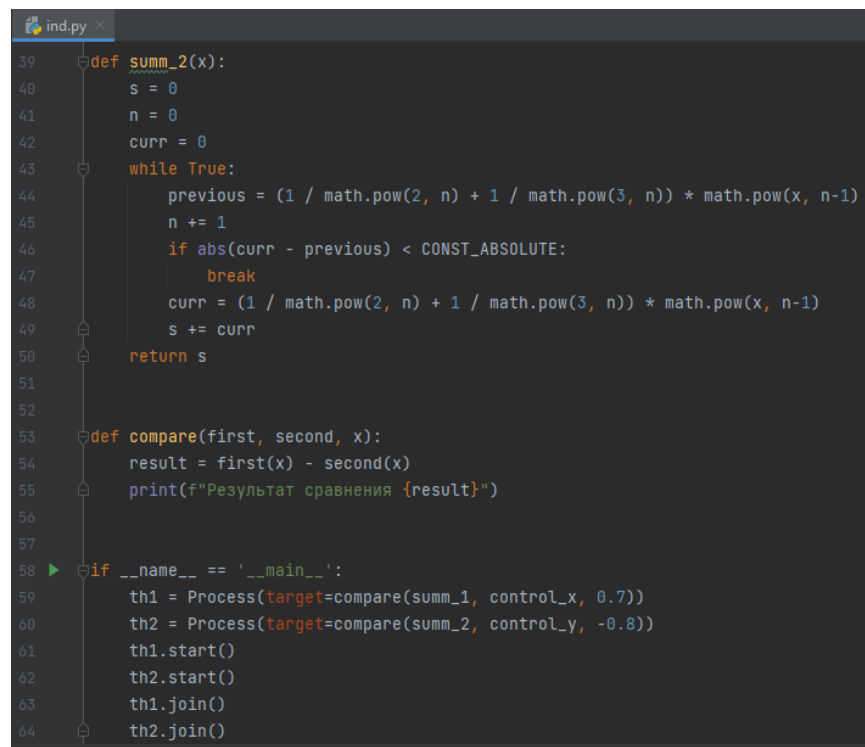
Ход работы

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

2. После ознакомления с теоретическим материалом приступил к выполнению задания.

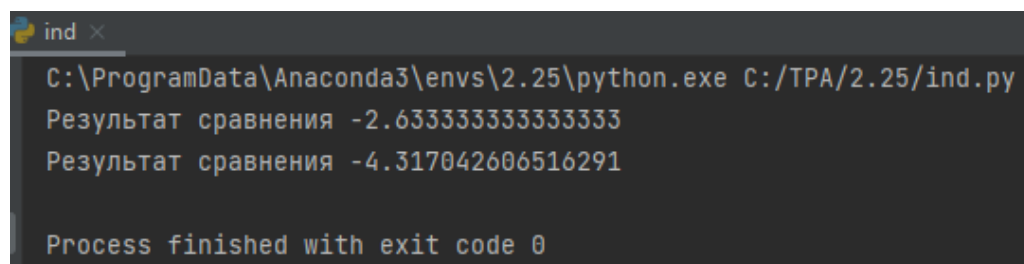
Условие задания: для своего индивидуального задания лабораторной работы 2.23 необходимо реализовать вычисление значений в двух функций в отдельных процессах.

3. Доработал код из прошлого задания, для выполнения задачи индивидуального задания.



```
39 def summ_2(x):
40     s = 0
41     n = 0
42     curr = 0
43     while True:
44         previous = (1 / math.pow(2, n) + 1 / math.pow(3, n)) * math.pow(x, n-1)
45         n += 1
46         if abs(curr - previous) < CONST_ABSOLUTE:
47             break
48         curr = (1 / math.pow(2, n) + 1 / math.pow(3, n)) * math.pow(x, n-1)
49         s += curr
50     return s
51
52
53 def compare(first, second, x):
54     result = first(x) - second(x)
55     print(f"Результат сравнения {result}")
56
57
58 if __name__ == '__main__':
59     th1 = Process(target=compare(summ_1, control_x, 0.7))
60     th2 = Process(target=compare(summ_2, control_y, -0.8))
61     th1.start()
62     th2.start()
63     th1.join()
64     th2.join()
```

Рисунок 1 – Фрагмент кода программы



```
ind x
C:\ProgramData\Anaconda3\envs\2.25\python.exe C:/TPA/2.25/ind.py
Результат сравнения -2.6333333333333333
Результат сравнения -4.317042606516291

Process finished with exit code 0
```

Рисунок 2 – Результат выполнения индивидуального задания

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 многопоточкового приложения к многопроцессному.

За ожидание завершения работы процесса(ов) отвечает метод *join*, со следующей сигнатурой: *join([timeout])*.

При выводе 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 отличие 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 выполнения лабораторной работы успешно приобретены навыки по написания многозадачных приложений на языке программирования Python версии 3.x.