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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №11
дисциплины «анализ данных»

Выполнил:
Середа Кирилл Витальевич
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Управление процессами в Python

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

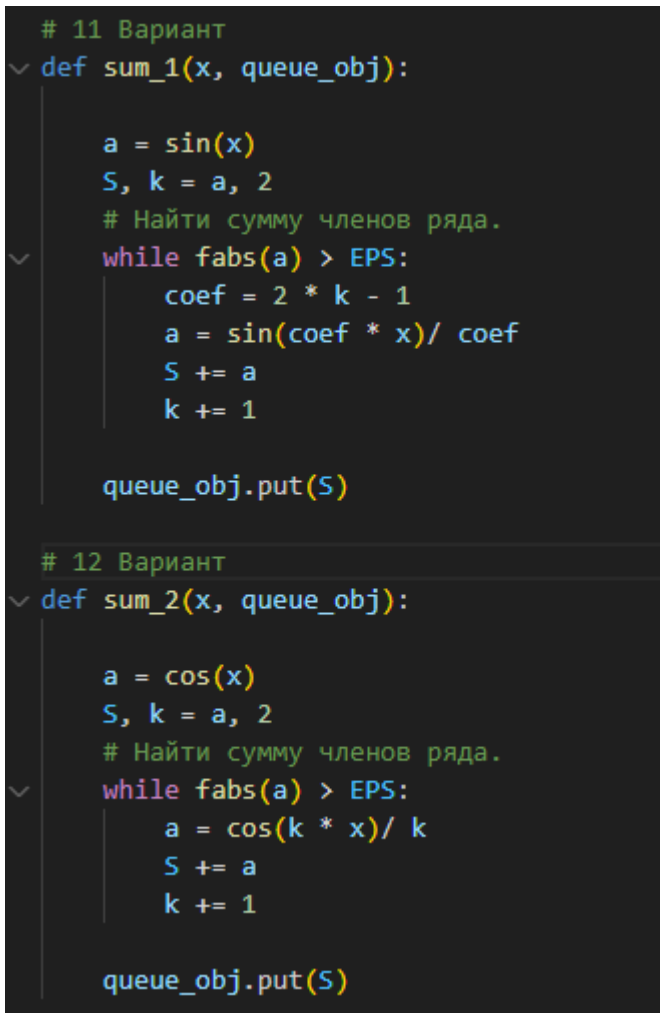
Ход работы:

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

Создал файл (ind.py) в котором реализовал вычисление разных рядов в разных процессах, на рис. 1 изображена реализация появления процессов, на рис. 2 изображены функции расчета суммы рядов.

```
def main():
    queue_obj = Queue()

    pc1 = Process(target=sum_1, args=(pi/2, queue_obj))
    pc2 = Process(target=sum_2, args=(pi, queue_obj))

    pc1.start()
    pc2.start()

    pc1.join()
    pc2.join()

    sum1, sum2 = queue_obj.get(), queue_obj.get()
    control_value_1 = pi / 4
    control_value_2 = - log(2 * sin(pi / 2))

    print(f"Сумма ряда 1: {round(sum1, 4)}")
    print(f"Контрольное значение ряда 1: {round(control_value_1, 4)}")
    print(f"Проверка: {round(sum1, 4) == round(control_value_1, 4)}")

    print(f"x2 = {pi}")
    print(f"Сумма ряда 2: {round(sum2, 4)}")
    print(f"Контрольное значение ряда 2: {round(control_value_2, 4)}")
    print(f"Проверка: {round(sum2, 4) == round(control_value_2, 4)}")
```

Рисунок 1 – Функция main()



```python
# 11 Вариант
def sum_1(x, queue_obj):

    a = sin(x)
    S, k = a, 2
    # Найти сумму членов ряда.
    while fabs(a) > EPS:
        coef = 2 * k - 1
        a = sin(coef * x)/ coef
        S += a
        k += 1

    queue_obj.put(S)

# 12 Вариант
def sum_2(x, queue_obj):

    a = cos(x)
    S, k = a, 2
    # Найти сумму членов ряда.
    while fabs(a) > EPS:
        a = cos(k * x)/ k
        S += a
        k += 1

    queue_obj.put(S)
```

Рисунок 2 – Функции для расчета суммы рядов

Вывод: в ходе выполнения практической работы были приобретены навыки написания многозадачных приложений на языке программирования Python версии 3.x.

Контрольные вопросы:

1. **Создание и завершение процессов в Python**

Процессы в Python создаются с использованием модуля **multiprocessing**. Для создания процесса необходимо создать экземпляр класса **Process** и вызвать метод **start()** для запуска процесса. Завершение процесса происходит автоматически после выполнения всей работы внутри процесса.

2. **Особенность создания классов-наследников от Process**

Особенность создания классов-наследников от **Process** заключается в том, что они могут переопределять методы, такие как **run()**, для определения поведения процесса.

3. **Принудительное завершение процесса**

Принудительное завершение процесса в Python может быть осуществлено путем вызова метода **terminate()** для объекта процесса.

4. **Процессы-демоны и их запуск**

Процессы-демоны (daemon processes) в Python – это процессы, которые работают в фоновом режиме и завершаются, когда основной процесс завершает свою работу. Для запуска процесса-демона, необходимо установить атрибут **daemon** объекта процесса в значение **True** перед его запуском.