

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 №2.25**  
**дисциплины «Основы кроссплатформенного программирования»**

Выполнил:  
Евдаков Евгений Владимирович  
2 курс, группа ИТС-б-о-22-1,  
11.03.02 «Инфокоммуникационные  
технологии и системы связи»,  
направленность (профиль)  
«Инфокоммуникационные системы и  
сети», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р. А., доцент кафедры  
инфокоммуникаций

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

**Тема:** управление процессами в Python.

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

### Ход работы:

**Задание 1.** Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python, также добавил файл .gitignore с необходимыми правилами. Клонировал свой репозиторий на свой компьютер. Организовал свой репозиторий в соответствии с моделью ветвления git-flow, появилась новая ветка develop в которой буду выполнять дальнейшие задачи.

```
C:\Users\Gaming-PC>git clone https://github.com/EvgenyEvdakov/Laba_2.25.git
Cloning into 'Laba_2.25'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 1. Клонирование репозитория

**Задание 2.** Создал виртуальное окружение conda и активировал его, также установил необходимые пакеты isort, black, flake8.

```
(base) PS C:\Users\Gaming-PC> cd C:\Users\Gaming-PC\Laba_2.25
(base) PS C:\Users\Gaming-PC\Laba_2.25> conda create -n 2.24 python=3.10
Retrieving notices: ...working... done
WARNING: A conda environment already exists at 'C:\Users\Gaming-PC\conda\envs\2.24'
Remove existing environment (y/[n])? y

Collecting package metadata (current_repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 23.1.0
  latest version: 24.5.0

Please update conda by running

    $ conda update -n base -c defaults conda

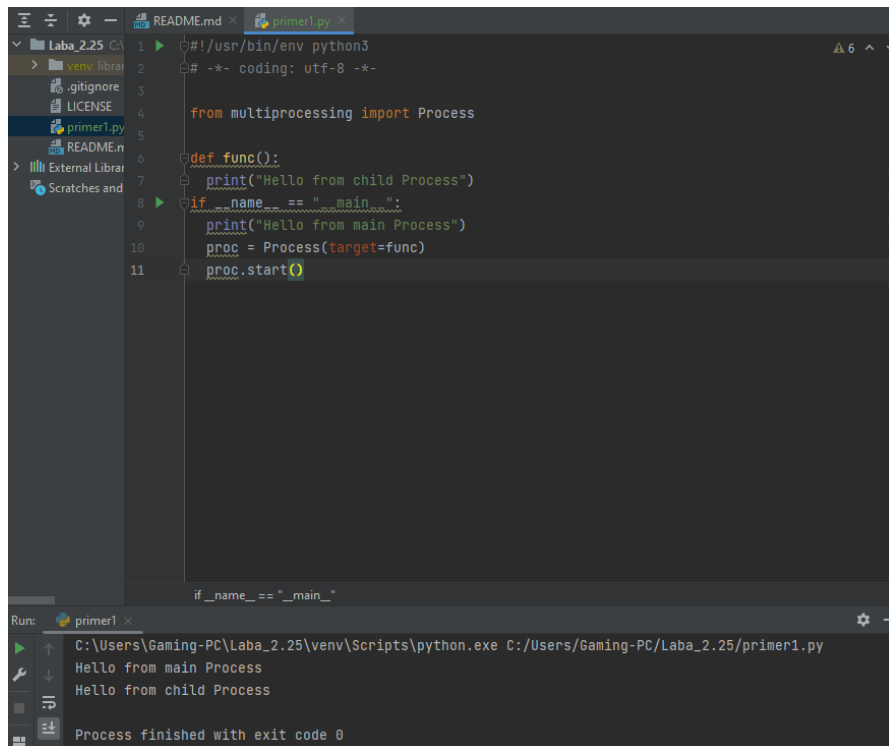
Or to minimize the number of packages updated during conda update use

    conda install conda=24.5.0
```

Рисунок 2. Создание виртуального окружения

**Задание 3.** Создал проект PyCharm в папке репозитория. Приступил к работе с примером. Добавил новый файл primer1.py.

**Условие примера:** создание и ожидание завершения работы процессов. Для запуска процесса используется метод `start()`.



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

from multiprocessing import Process

def func():
    print("Hello from child Process")

if __name__ == "__main__":
    print("Hello from main Process")
    proc = Process(target=func)
    proc.start()

if __name__ == "__main__":
```

Run: primer1

C:\Users\Gaming-PC\Laba\_2.25\venv\Scripts\python.exe C:\Users\Gaming-PC\Laba\_2.25\primer1.py

Hello from main Process

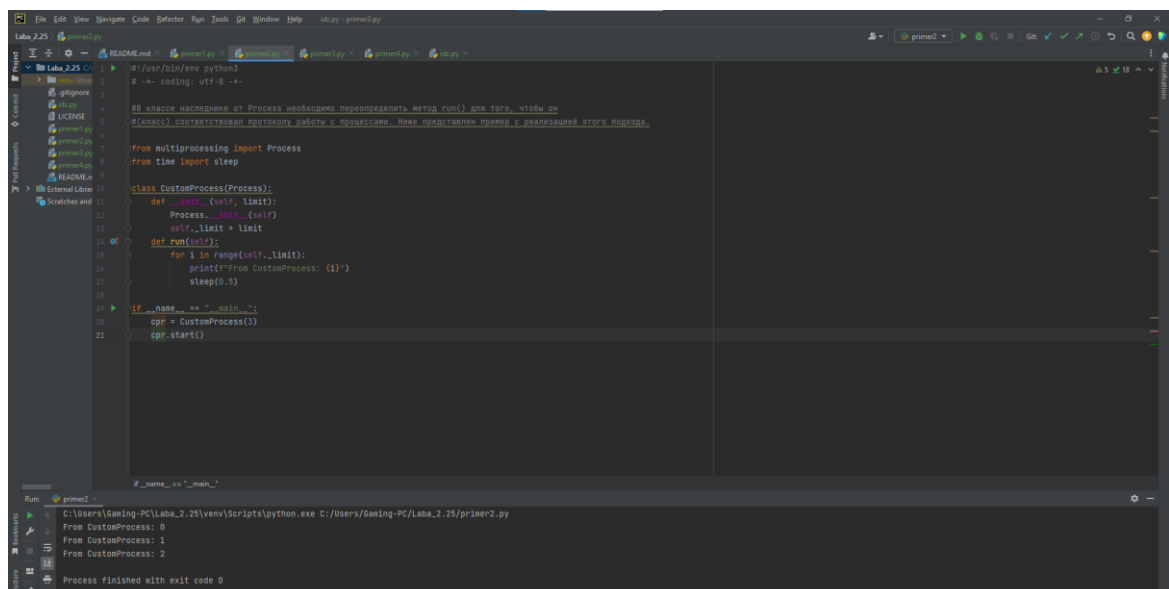
Hello from child Process

Process finished with exit code 0

Рисунок 3. Выполнение первого примера

Добавил новый файл `primer2.py`.

**Условие примера:** 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.



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

from multiprocessing import Process
from time import sleep

class CustomProcess(Process):
    def __init__(self, limit):
        Process.__init__(self)
        self.limit = limit

    def run(self):
        for i in range(self.limit):
            print(f"From CustomProcess: {i}")
            sleep(0.5)

if __name__ == "__main__":
    cpr = CustomProcess(3)
    cpr.start()

if __name__ == "__main__":
```

Run: primer2

C:\Users\Gaming-PC\Laba\_2.25\venv\Scripts\python.exe C:\Users\Gaming-PC\Laba\_2.25\primer2.py

From CustomProcess: 0

From CustomProcess: 1

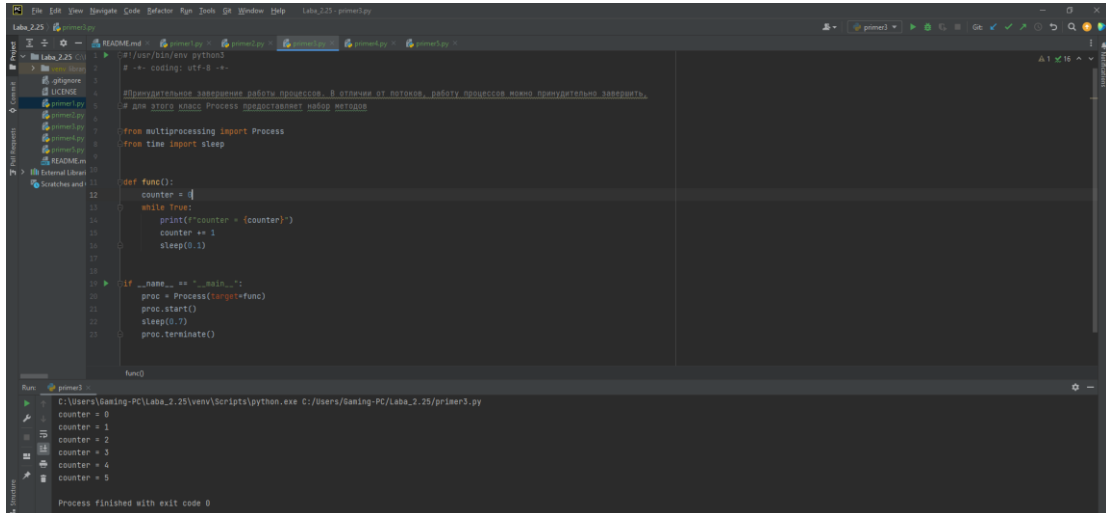
From CustomProcess: 2

Process finished with exit code 0

Рисунок 4. Выполнение второго примера

Добавил новый файл primer3.py.

**Условие примера:** Принудительное завершение работы процессов. В отличие от потоков, работу процессов можно принудительно завершить, для этого класс Process предоставляет набор методов.



```
# primer3.py
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# multiprocessing module provides a way to spawn processes, but unlike the thread module it does not support global variables.
# This script shows how to use the Process class to create a process.

from multiprocessing import Process
from time import sleep

def func():
    counter = 0
    while True:
        print('counter = (counter)')
        counter += 1
        sleep(0.1)

if __name__ == '__main__':
    proc = Process(target=func)
    proc.start()
    sleep(0.7)
    proc.terminate()

func()
```

Run: primer3

C:\Users\Gaming-PC\Labo\_2.25\venv\Scripts\python.exe C:/Users/Gaming-PC/Labo\_2.25/primer3.py

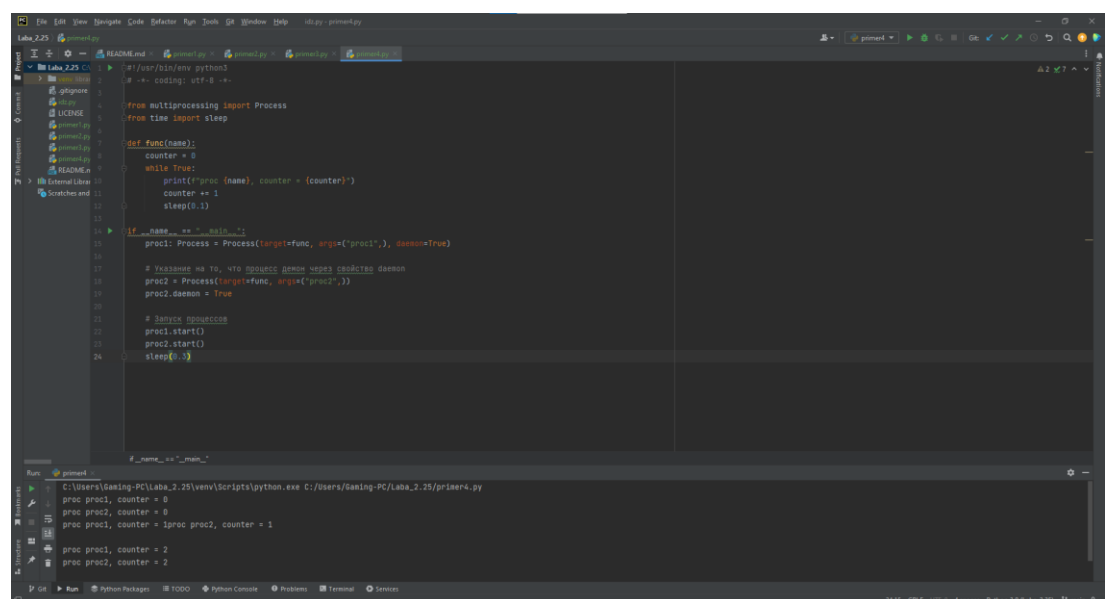
counter = 0  
counter = 1  
counter = 2  
counter = 3  
counter = 4  
counter = 5

Process finished with exit code 0

Рисунок 5. Выполнение третьего примера

Добавил новый файл primer4.py.

**Условие примера:** процессы-демоны. Процессы демоны по своим свойствам похожи на потоки-демоны, их суть заключается в том, что они завершают свою работу, если завершился родительский процесс. Указать на то, что процесс является демоном можно при создании экземпляра класса через аргумент daemon, либо после создания через свойство daemon.



```
# primer4.py
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

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__':
    proc1 = Process(target=func, args=('proc1',), daemon=True)
    # Указание на то, что данный демон должен работать как демон
    proc2 = Process(target=func, args=('proc2',), daemon=True)
    # Запуск процессов
    proc1.start()
    proc2.start()
    sleep(0.5)

if __name__ == '__main__':
```

Run: primer4

C:\Users\Gaming-PC\Labo\_2.25\venv\Scripts\python.exe C:/Users/Gaming-PC/Labo\_2.25/primer4.py

proc proc1, counter = 0  
proc proc2, counter = 0  
proc proc1, counter = 1  
proc proc1, counter = 2  
proc proc2, counter = 2

Рисунок 6. Выполнение четвертого примера

## Задание 4.

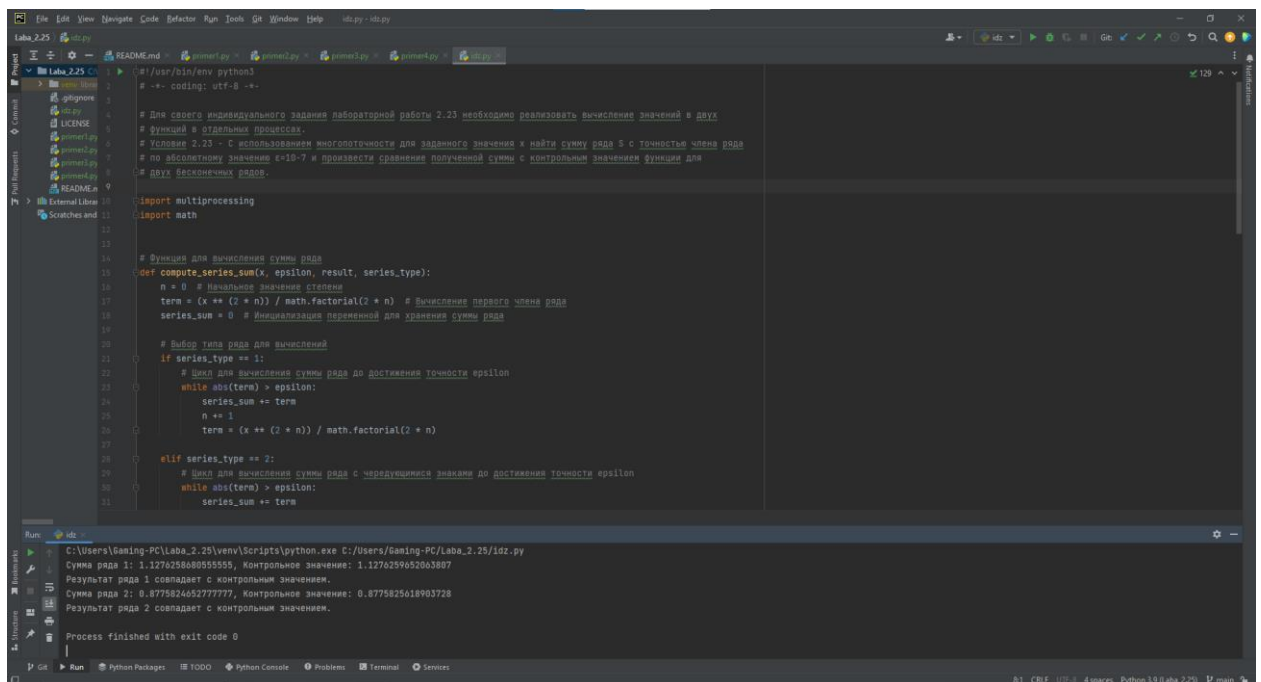
### Индивидуальное задание

#### Вариант 10

Создал новый файл под названием idz.py.

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

Условие 2.23 - 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.



```
1 # coding: utf-8 -*-
2
3 # Для своего индивидуального задания лабораторной работы 2.23 необходимо реализовать вычисление значений в двух
4 # функций в отдельных процессах.
5 # Условие 2.23 - с использованием многопоточности для заданного значения x найти сумму ряда S с точностью члена ряда
6 # по абсолютному значению  $\varepsilon=10^{-7}$  и произвести сравнение полученной суммы с контрольным значением функции для
7 # двух бесконечных рядов.
8
9
10 import multiprocessing
11 import math
12
13
14 # Функция для вычисления суммы ряда
15 def compute_series_sum(x, epsilon, result, series_type):
16     n = 0 # Начальное значение столбца
17     term = (x ** (2 * n)) / math.factorial(2 * n) # Вычисление первого члена ряда
18     series_sum = 0 # Инициализация переменных для хранения суммы ряда
19
20     # Выбор типа ряда для вычисления
21     if series_type == 1:
22         # Цикл для вычисления суммы ряда до достижения точности epsilon
23         while abs(term) > epsilon:
24             series_sum += term
25             n += 1
26             term = (x ** (2 * n)) / math.factorial(2 * n)
27
28     elif series_type == 2:
29         # Цикл для вычисления суммы ряда с чередующимися знаками до достижения точности epsilon
30         while abs(term) > epsilon:
31             series_sum += term
32             n += 1
33             term = (-1) ** n * (x ** (2 * n)) / math.factorial(2 * n)
34
35
36 # Запуск функций в отдельных процессах
37 if __name__ == '__main__':
38     x = 1.1276258680955555
39     epsilon = 1e-7
40     result = {}
41
42     # Вычисление суммы ряда 1
43     p1 = multiprocessing.Process(target=compute_series_sum, args=(x, epsilon, result, 1))
44     p1.start()
45
46     # Вычисление суммы ряда 2
47     p2 = multiprocessing.Process(target=compute_series_sum, args=(x, epsilon, result, 2))
48     p2.start()
49
50     p1.join()
51     p2.join()
52
53     # Вывод результатов
54     print("Сумма ряда 1: 1.1276258680955555, Контрольное значение: 1.1276259652063867")
55     print("Результат ряда 1 совпадает с контрольным значением.")
56     print("Сумма ряда 2: 0.8775824432777777, Контрольное значение: 0.8775825618903728")
57     print("Результат ряда 2 совпадает с контрольным значением.")
58
59     # Завершение процесса
60     exit(0)
```

Рисунок 7. Выполнение индивидуального задания

## Задание 5.

После выполнения работы на ветке develop, слил ее с веткой main и отправил изменения на удаленный сервер. Создал файл environment.yml и деактивировал виртуальное окружение.

```
(base) PS C:\Users\Gaming-PC> cd C:\Users\Gaming-PC\Laba_2.25
(base) PS C:\Users\Gaming-PC\Laba_2.25> conda env export > environment
(base) PS C:\Users\Gaming-PC\Laba_2.25> conda deactivate
```

Рисунок 8. Деактивация ВО

Ссылка: [https://github.com/EvgenyEvdakov/Laba\\_2.25](https://github.com/EvgenyEvdakov/Laba_2.25)

**Ответы на контрольные вопросы:**

**Вывод:** приобрел навыки написания многозадачных приложений на языке программирования Python версии 3.x.