

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»  
ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчет по лабораторной работе № 13  
«Управление процессами в Python»**

**по дисциплине «Основы программной инженерии»**

Выполнила:  
Первых Дарья Александровна,  
2 курс, группа ПИЖ-б-о-20-1

Проверил:  
Доцент кафедры  
инфокоммуникаций, Воронкин Р.А.

Ставрополь, 2022 г.

## ВЫПОЛНЕНИЕ

```
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      from multiprocessing import Process
6
7
8      def func():
9          print("Hello from child Process")
10
11
12  ▶  if __name__ == "__main__":
13      print("Hello from main Process")
14      proc = Process(target=func)
15      proc.start()
```

if \_\_name\_\_ == "\_\_main\_\_"

main ×

C:\Users\podar\study\anaconda\envs\13LR\python  
Hello from main Process  
Hello from child Process

Рисунок 1 – Создание и ожидание завершения работы процессов

```
▶  #!/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()
    print(f"Proc is_alive status: {proc.is_alive()}")
    ...
    if __name__ == "__main__"
```

main ×

C:\Users\podar\study\anaconda\envs\13LR\python.exe "C:/Use  
Hello from main Process  
Proc is\_alive status: True  
Hello from child Process  
Goodbye  
Proc is\_alive status: False

Рисунок 2 – Создание и ожидание завершения работы процессов

```
9 class CustomProcess(Process):
10     def __init__(self, limit):
11         Process.__init__(self)
12         self._limit = limit
13
14     def run(self):
15         for i in range(self._limit):
16             print(f"From CustomProcess: {i}")
17             sleep(0.5)
18
19
20 if __name__ == "__main__":
21     cpr = CustomProcess(3)
22     cpr.start()
```

main x

C:\Users\podar\study\anaconda\envs\13LR\python.exe "C:\Users\podar\study\anaconda\envs\13LR\python.exe"

From CustomProcess: 0

From CustomProcess: 1

From CustomProcess: 2

Рисунок 3 – Создание классов-наследников от Process

```
10 def func():
11     counter = 0
12     while True:
13         print(f"counter = {counter}")
14         counter += 1
15         sleep(0.1)
16
17
18 if __name__ == "__main__":
19     proc = Process(target=func)
20     proc.start()
21     sleep(0.7)
22     proc.terminate()
```

if \_\_name\_\_ == "\_\_main\_\_"

main x

C:\Users\podar\study\anaconda\envs\13LR\python.exe "C:\Users\podar\study\anaconda\envs\13LR\python.exe"

counter = 0

counter = 1

counter = 2

counter = 3

counter = 4

counter = 5

Рисунок 4 – Принудительное завершение работы процессов

```
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)
    proc2 = Process(target=func, args=("proc2",))
    proc2.daemon = True
    proc1.start()
    proc2.start()
    sleep(0.3)

if __name__ == "__main__"
```

main x

```
C:\Users\podar\study\anaconda\envs\13LR\python.exe "C:/Users/podar/st
proc proc1, counter = 0
proc proc2, counter = 0
proc proc1, counter = 1
proc proc2, counter = 1
```

Рисунок 5 – Процессы-демоны

Индивидуальное задание.

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

17.

$$S = \sum_{n=0}^{\infty} (n+1)x^n = 1 + 2x + 3x^2 + 4x^3 + \dots; \quad x = -0,7; \quad y = \frac{1}{(1-x)^2}.$$

```
curr = curr * (n + 1) * x
previous = s
s += curr
n += 1
if n > 6:
    break

print(f"Сумма ряда для значения {x}: {s}")
print(f"Проверка: 1/(1 - {x})^2 = {1 / math.pow((1 - x), 2)}")

if __name__ == '__main__':
    process1 = Process(target=sum, args=(-0.7,))
    process1.start()
    process2 = Process(target=sum, args=(3,))
    process2.start()
```

sum() > while True > if n > 6

main ×

C:\Users\podar\study\anaconda\envs\13LR\python.exe "C:/Users/podar/study/P  
Сумма ряда для значения -0.7: 495.06055999999998  
Проверка: 1/(1 - -0.7)^2 = 0.34602076124567477  
Сумма ряда для значения 3: 3859549  
Проверка: 1/(1 - 3)^2 = 0.25

Рисунок 6 – Индивидуальное задание

## ВОПРОСЫ

1. Как создаются и завершаются процессы в Python?

`proc = Process(target=func)`

`proc.start()`

`join()` для того, чтобы программа ожидала завершения процесса.

Процессы завершаются при завершении функции, указанной в `target`, либо принудительно с помощью `kill()`, `terminate()`

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.

```
proc1 = Process(target=func, args=("proc1",), daemon=True)
```

```
proc2.daemon = True
```

```
proc1.start()
```

```
proc2.start()
```