

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

**ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ**

**Отчёт по лабораторной работе №2.23 по дисциплине:**

**Основы программной инженерии**

Выполнила:

студент группы ПИЖ-б-о-20-1

Лазарева Дарья Олеговна

Проверил:

доцент кафедры инфокоммуникаций

Романкин Р.А.

Ставрополь, 2022 г.

## ВЫПОЛНЕНИЕ:

### 1. Создание и ожидание завершения работы процессов

```
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()
```

ex1 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.25/ex1.py

Hello from main Process

Hello from child Process

```
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()
16     print(f"Proc is_alive status: {proc.is_alive()}")
17     proc.join()
18     print("Goodbye")
19     print(f"Proc is_alive status: {proc.is_alive()}")
```

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

ex2 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.25/ex2.py

Hello from main Process

Proc is\_alive status: True

Hello from child Process

Goodbye

Proc is\_alive status: False

## 2. Создание классов-наследников

```
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      from multiprocessing import Process
6      from time import sleep
7
8
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()
```

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

ex3 x

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.25/ex3.py

From CustomProcess: 0

From CustomProcess: 1

From CustomProcess: 2

### 3. Принудительное завершение работы процессов

```
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      from multiprocessing import Process
6      from time import sleep
7
8
9      def func():
10         counter = 0
11         while True:
12             print(f"counter = {counter}")
13             counter += 1
14             sleep(0.1)
15
16
17  ▶  if __name__ == "__main__":
18         proc = Process(target=func)
19         proc.start()
20         sleep(0.7)
21         proc.terminate()
```


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

Run: ex4 ×

```
▶  ↑ counter = 1
   ↓ counter = 2
   ↻ counter = 3
   ⏏ counter = 4
   ⏏ counter = 5
```

## 4. Процессы-демоны

```
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      from multiprocessing import Process
6      from time import sleep
7
8
9      def func(name):
10         counter = 0
11         while True:
12             print(f"proc {name}, counter = {counter}")
13             counter += 1
14             sleep(0.1)
15
16
17  ▶  if __name__ == "__main__":
18         proc1 = Process(target=func, args=("proc1",), daemon=True)
19         proc2 = Process(target=func, args=("proc2",))
20         proc2.daemon = True
21         proc1.start()
22         proc2.start()
23         sleep(0.3)
24
25     if __name__ == "__main__"
```

Run:  ex5 ×

C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.25/ex5.py

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

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

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

from multiprocessing import Process
from math import factorial, sin

eps = .0000001

def inf_sum(x, num):
    a = 1
    summa = 1
    i = 1
    prev = 0
    while abs(summa - prev) > eps:
        a = ((-1)**i * x**(2*i + 1)) / factorial(2 * i + 1)
        prev = summa
        if i % 2 == 0:
            summa += a
        else:
            summa += -1 * a
        i += 1
    print(f"The sum number: {num} is: {summa}")
    print(f"Check: sin({x}) = {sin(x)}")

if __name__ == '__main__':
    process1 = Process(target=inf_sum, args=(1, 1))
    process1.start()
    process2 = Process(target=inf_sum, args=(4, 2))
    process2.start()
```

```
C:\Users\79616\anaconda3\python.exe D:/УЧЕБА/ОПИ/2.25/idz.py
The sum number: 1 is: 1.1752011934824436
Check: sin(1) = 0.8414709848078965
The sum number: 2 is: 24.289917194331494
Check: sin(4) = -0.7568024953079282
```

Контрольные 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()
```