

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

**Отчет о лабораторной работе №11 по дисциплине основы программной
инженерии**

Выполнил:
Выходцев Егор Дмитриевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г

1. Примеры из методических указаний

```
e1.py x
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
```

```
C:\Users\Evil\PycharmProjects\LR#13\ven
Hello from main Process
Hello from child Process

Process finished with exit code 0
```

```
e1.py x e2.py x
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()}")
20
```

```
C:\Users\Evil\PycharmProjects\LR#1
Hello from main Process
Proc is_alive status: True
Hello from child Process
Goodbye
Proc is_alive status: False

Process finished with exit code 0
```

```
e1.py × e2.py × e3.py ×
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()
23
```

```
C:\Users\Evil\PycharmProjects\LR#1
From CustomProcess: 0
From CustomProcess: 1
From CustomProcess: 2

Process finished with exit code 0
```

```
e4.py x
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()
22
```

```
C:\Users\Evil\PycharmProjects\LR#1
counter = 0
counter = 1
counter = 2
counter = 3
counter = 4
counter = 5

Process finished with exit code 0
```

```
e4.py × e5.py ×
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
```

```
C:\Users\Evil\PycharmProjects\LR#1
proc proc1, counter = 0
proc proc2, counter = 0
proc proc1, counter = 1
proc proc2, counter = 1
proc proc1, counter = 2
proc proc2, counter = 2

Process finished with exit code 0
```

2. Индивидуальное задание (рис. 1-2)

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

5.

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

```
idz.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5  from multiprocessing import Process
6  from math import cos
7
8
9  eps = .0000001
10
11
12  def inf_sum(x, num):
13      a = 1
14      summa = 1
15      i = 1
16      prev = 0
17      while abs(summa - prev) > eps:
18          a = a * x ** 2 / ((2 * i) * (2 * i - 1))
19          prev = summa
20          if i % 2 == 0:
21              summa += a
22          else:
23              summa += -1 * a
24          i += 1
25      print(f"The sum number: {num} is: {summa}")
26      print(f"Check: cos({x}) = {cos(x)}")
27
28
29  ▶  if __name__ == '__main__':
30      process1 = Process(target=inf_sum, args=(0, 1))
31      process1.start()
32      process2 = Process(target=inf_sum, args=(3, 2))
33      process2.start()
34
```

Рисунок 1 – Код программы

```
C:\Users\Evil\PycharmProjects\LR#13\venv\S
The sum number: 1 is: 1.0
Check: cos(0) = 1.0
The sum number: 2 is: -0.9899924980061545
Check: cos(3) = -0.9899924966004454

Process finished with exit code 0
```

Рисунок 2 – Результат выполнения программы

3. Ответы на 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()
```