

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

**Отчёт о лабораторной работе №13 по дисциплине основы программной
инженерии**

Выполнила:

Нестеренко Тамара Антоновна,
2 курс, группа ПИЖ-б-о-20-1,

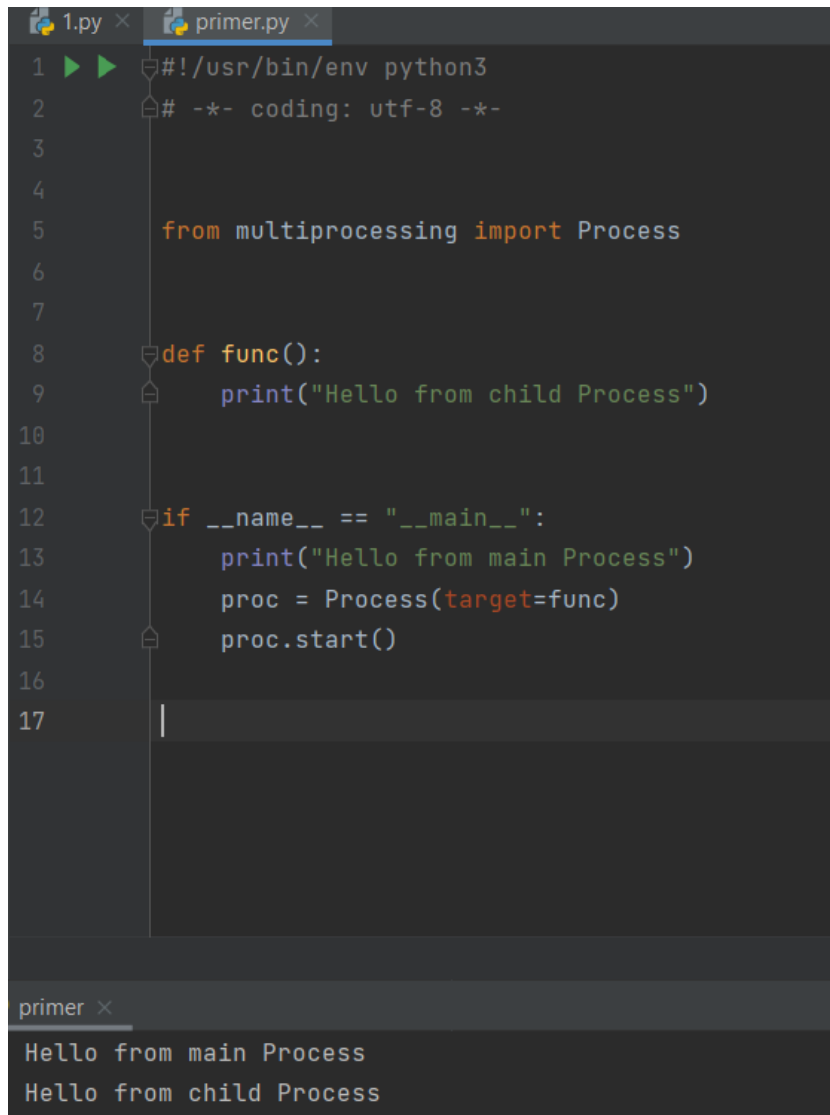
Проверил:

Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ

1. Практическая часть

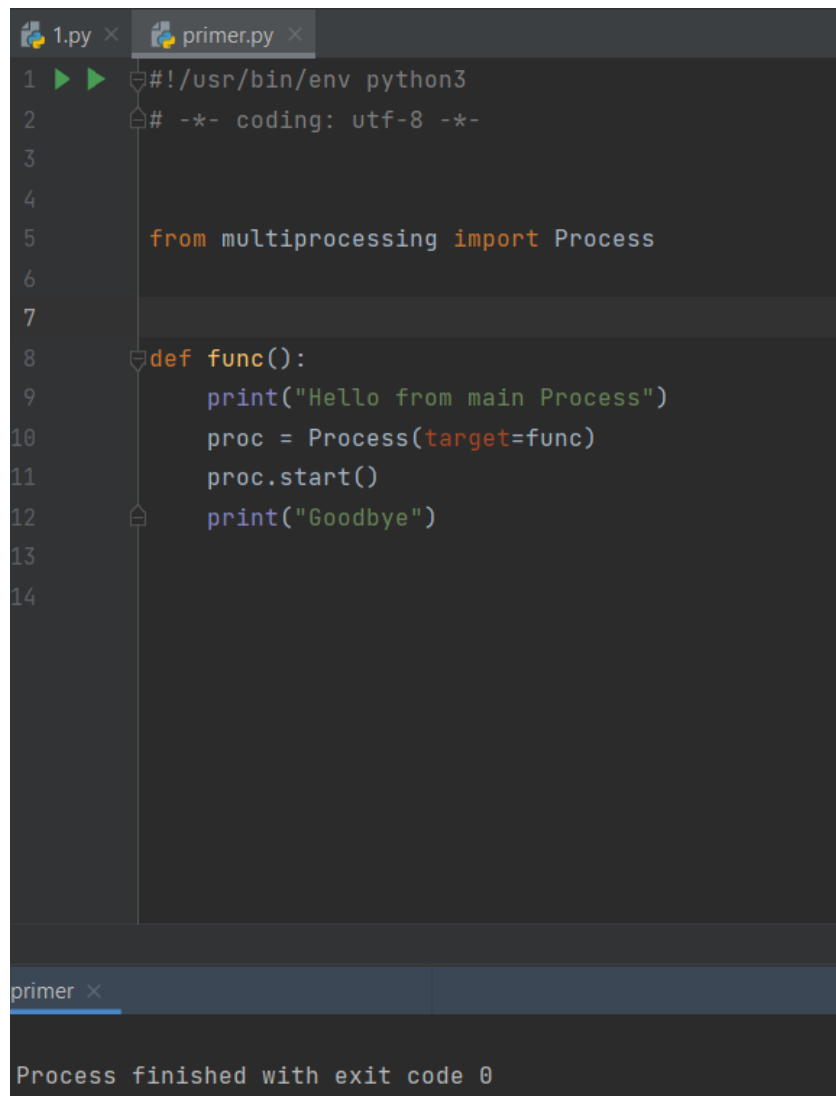


```
1.py x primer.py x
1  ▶▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4
5      from multiprocessing import Process
6
7
8      def func():
9          print("Hello from child Process")
10
11
12     if __name__ == "__main__":
13         print("Hello from main Process")
14         proc = Process(target=func)
15         proc.start()
16
17     |
```

primer x

Hello from main Process
Hello from child Process

Рисунок 1 – Пример создания и ожидания завершения работы процессов

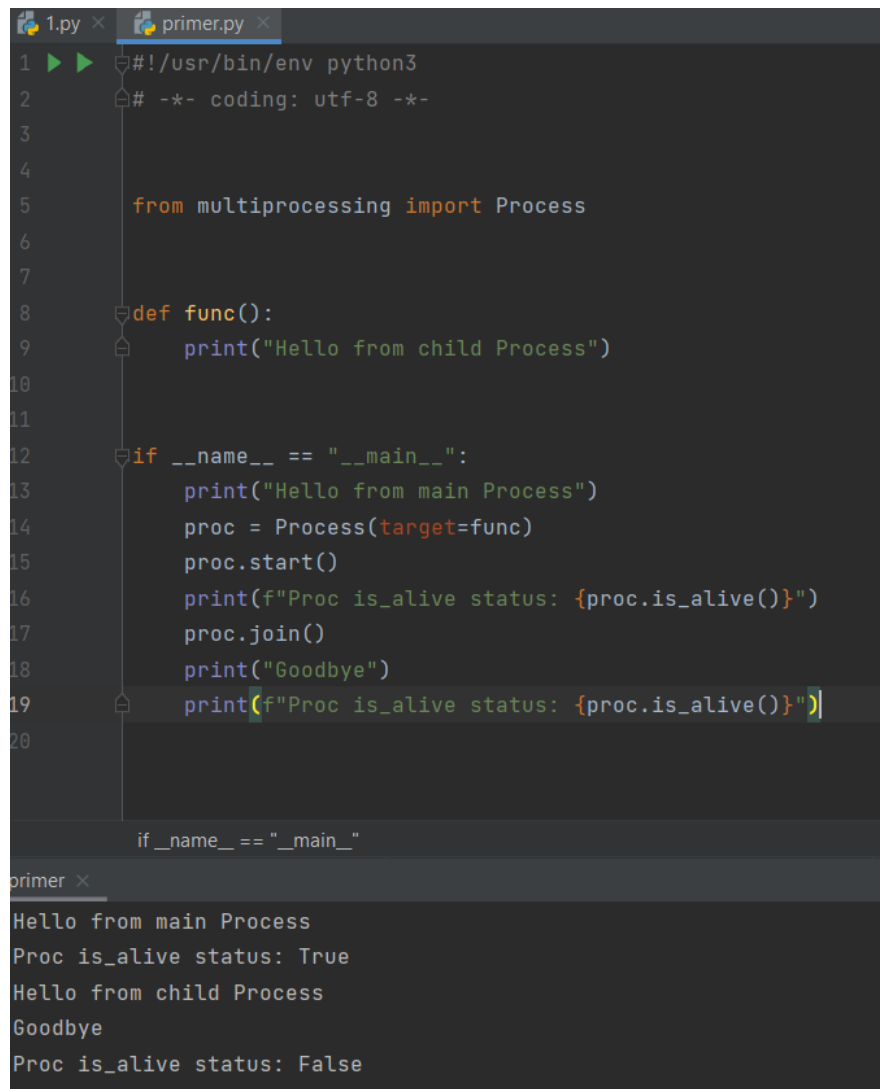


```
1.py x primer.py x
1  ▶▶ #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4
5  from multiprocessing import Process
6
7
8  def func():
9      print("Hello from main Process")
10     proc = Process(target=func)
11     proc.start()
12     print("Goodbye")
13
14
```

primer x

Process finished with exit code 0

Рисунок 2 – Пример создания и ожидания завершения работы процессов



```
1.py x primer.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
if __name__ == "__main__"
```

primer x

```
Hello from main Process
Proc is_alive status: True
Hello from child Process
Goodbye
Proc is_alive status: False
```

Рисунок 3 – Пример создания и ожидания завершения работы процессов

```
1.py x primer.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  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()

```

primer x

```
From CustomProcess: 0
From CustomProcess: 1
From CustomProcess: 2

```

Рисунок 4 – Пример создания классов наследников от Process


```
1.py x primer.py x
C:\Users\тома нестеренко\PycharmProjects\1 n3
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()
    if __name__ == "__main__"

primer x
proc proc1, counter = 0
proc proc2, counter = 0
proc proc1, counter = 1
proc proc2, counter = 1
proc proc1, counter = 2proc proc2, counter = 2
```

Рисунок 6 – Пример процессов-демонов

```
1.py x primer.py x
1  ▶▶ #!/usr/bin/env python3
2    # -*- coding: utf-8 -*-
3
4
5    from multiprocessing import Process
6    import math
7
8    eps = .0000001
9
10
11   def inf_sum(x, num):
12       a = 1
13       summa = math.cos(x)
14       i = 1
15       prev = 0
16       while abs(summa + prev) < eps:
17           a = a * (math.cos(2*x)) / 2
18           prev = summa
19           if i % 2 == 0:
20               summa += a
21           else:
22               summa += -1 * a
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
1 x
The sum number: 1 is: 1.0
Check: cos(0) = 1.0
The sum number: 2 is: -0.9899924966004454
Check: cos(3) = -0.9899924966004454
Process finished with exit code 0
```

Рисунок 7 – Пример решения индивидуального задания

2. Вопросы для защиты

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()
```

Ссылка на репозиторий: https://github.com/tamaranesterenko/Python_LR_13-2