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 №13 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

```
#!/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()

```

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

Process finished with exit code 0
```

e1.py e2.py

```python
#!/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()}")
    proc.join()
    print("Goodbye")
    print(f"Proc is_alive status: {proc.is_alive()}")

```

```
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

```python

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()

```

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

Process finished with exit code 0
```

e4.py

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


from multiprocessing import Process
from time import sleep


def func():
    counter = 0
    while True:
        print(f"counter = {counter}")
        counter += 1
        sleep(0.1)


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

```

```
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

```python

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

```

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

idz.py

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


from multiprocessing import Process
from math import cos


eps = .0000001


def inf_sum(x,  num):
    a = 1
    summa = 1
    i = 1
    prev = 0
    while abs(summa - prev) > eps:
        a = a * x ** 2 / ((2 * i) * (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: cos({x}) = {cos(x)}")


if __name__ == '__main__':
    process1 = Process(target=inf_sum, args=(0, 1))
    process1.start()
    process2 = Process(target=inf_sum, args=(3, 2))
    process2.start()

```

Рисунок 1 – Код программы

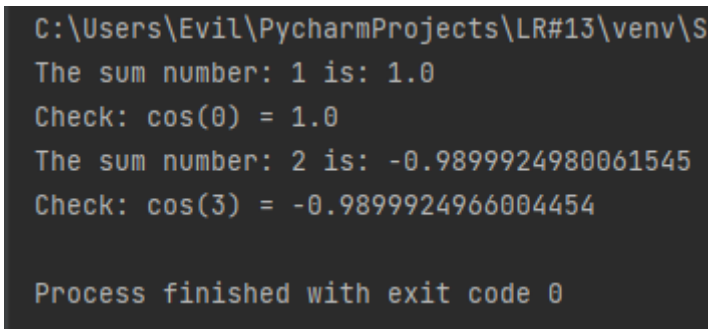

Рисунок 2 – Результат выполнения программы

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 отличии 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()
```