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 №2.25 по дисциплине основы программной инженерии

Выполнил:
Шальнев Владимир Сергеевич,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:
Доцент кафедры
прикладной математики и
компьютерной безопасности,
Воронкин Р.А.

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2022 г.

Выполнение:

```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()
```

Пример 1

```
Hello from main Process
Hello from child Process

Process finished with exit code 0
```

Результат работы примера 1

```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()}")
```

Пример 2

```
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
```

Результат работы примера 2

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


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()
```

Пример 3

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

Process finished with exit code 0
```

Результат работы примера 3

```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()
```

Пример 4

```
counter = 0
counter = 1
counter = 2
counter = 3
counter = 4
counter = 5
counter = 6

Process finished with exit code 0
```

Результат работы примера 4

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


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)
```

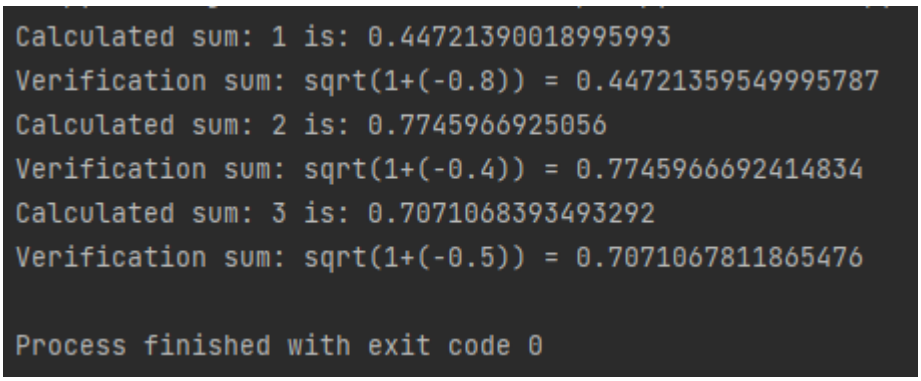
Пример 5

```
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
```

Результат работы примера 5

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


from multiprocessing import Process
from math import sqrt


def infinite_sum(x, check, num):
    eps = 10 ** -7
    a = 1
    summa = 1
    i = 1
    prev = 0
    while abs(summa - prev) > eps:
        prev = summa
        numerator = 1
        denominator = 1
        for j in range(1, i + 1):
            if 2 * j - 3 > 0:
                numerator *= 2 * j - 3
            denominator *= 2 * j
        temp = (numerator / denominator) * x**i
        if (-1) ** (i + 1) > 0:
            summa += temp
        else:
            summa -= temp
        i += 1
    print(f"Calculated sum: {num} is: {summa}")
    print(f"Verification sum: sqrt(1+({x})) = {check}")


if __name__ == '__main__':
    checksum1 = sqrt(1 - 0.8)
    process1 = Process(target=infinite_sum, args=(-0.8, checksum1, 1))
    process1.start()

    checksum2 = sqrt(1 - 0.4)
    process2 = Process(target=infinite_sum, args=(-0.4, checksum2, 2))
    process2.start()

    checksum3 = sqrt(1 - 0.5)
    process3 = Process(target=infinite_sum, args=(-0.5, checksum3, 3))
    process3.start()
```

Решение первой индивидуальной задачи



Результата работы программы

Ответы на вопросы:

1. Как создаются и завершаются процессы в Python?

from multiprocessing import Process

proc = Process(target=func)

proc.start()

join() – для завершения процесса.

kill(), terminate() – завершить процесс.

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.

Указать на то, что процесс является демоном можно при создании экземпляра класса через аргумент daemon, либо после создания через свойство daemon.

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