Институт цифрового развития
Кафедра инфокоммуникаций

«Управление процессами в Python»

ОТЧЕТ
по лабораторной работе №25
дисциплины
«Основы программной инженерии»

Выполнил:
Мизин Глеб Егорович
2 курс, группа ПИЖ-б-о-21-1,
011.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

_____
(подпись)

Проверил:
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

```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()
```

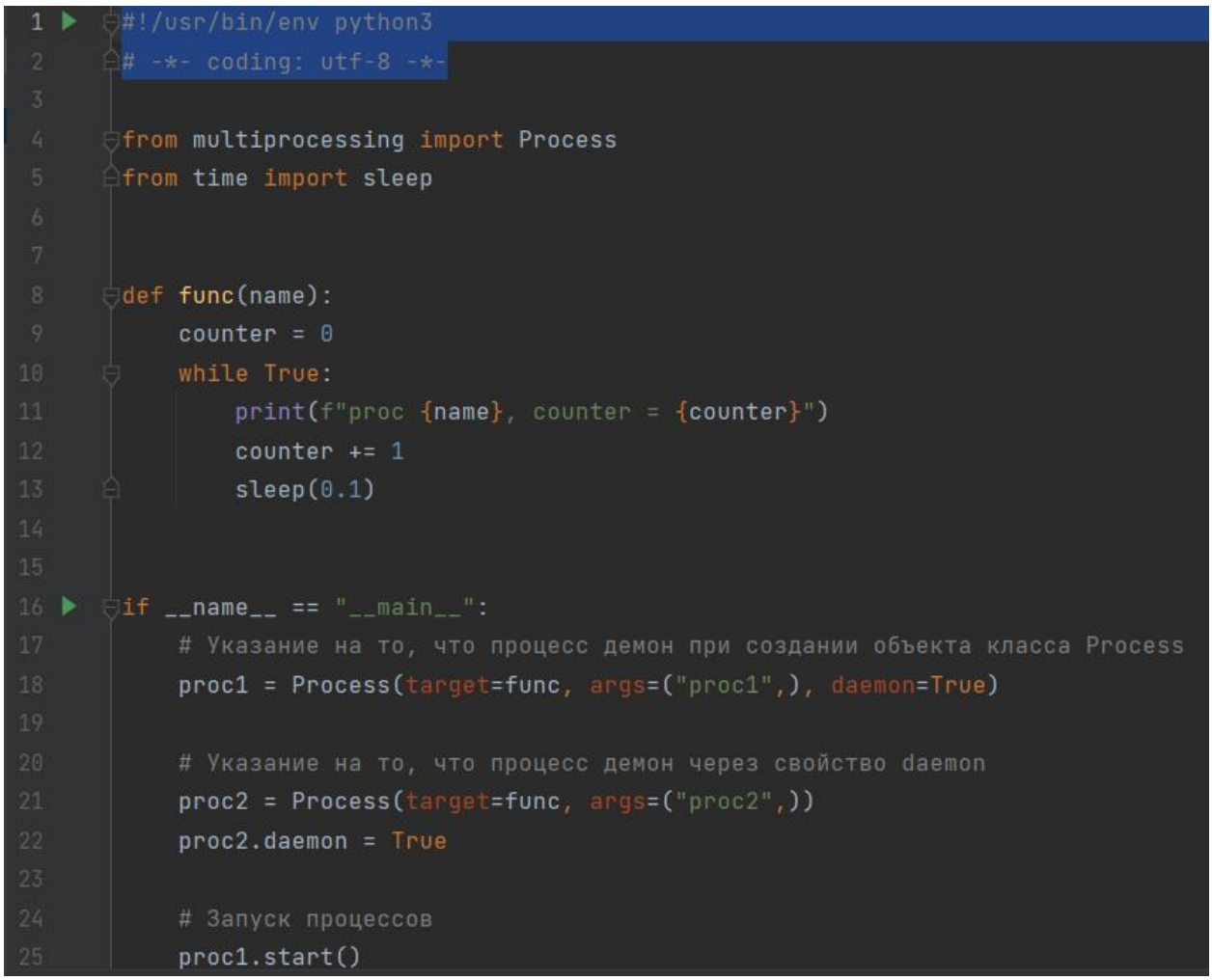
Рисунок 1 – Пример 1

```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()
```

Рисунок 2 – Пример 2

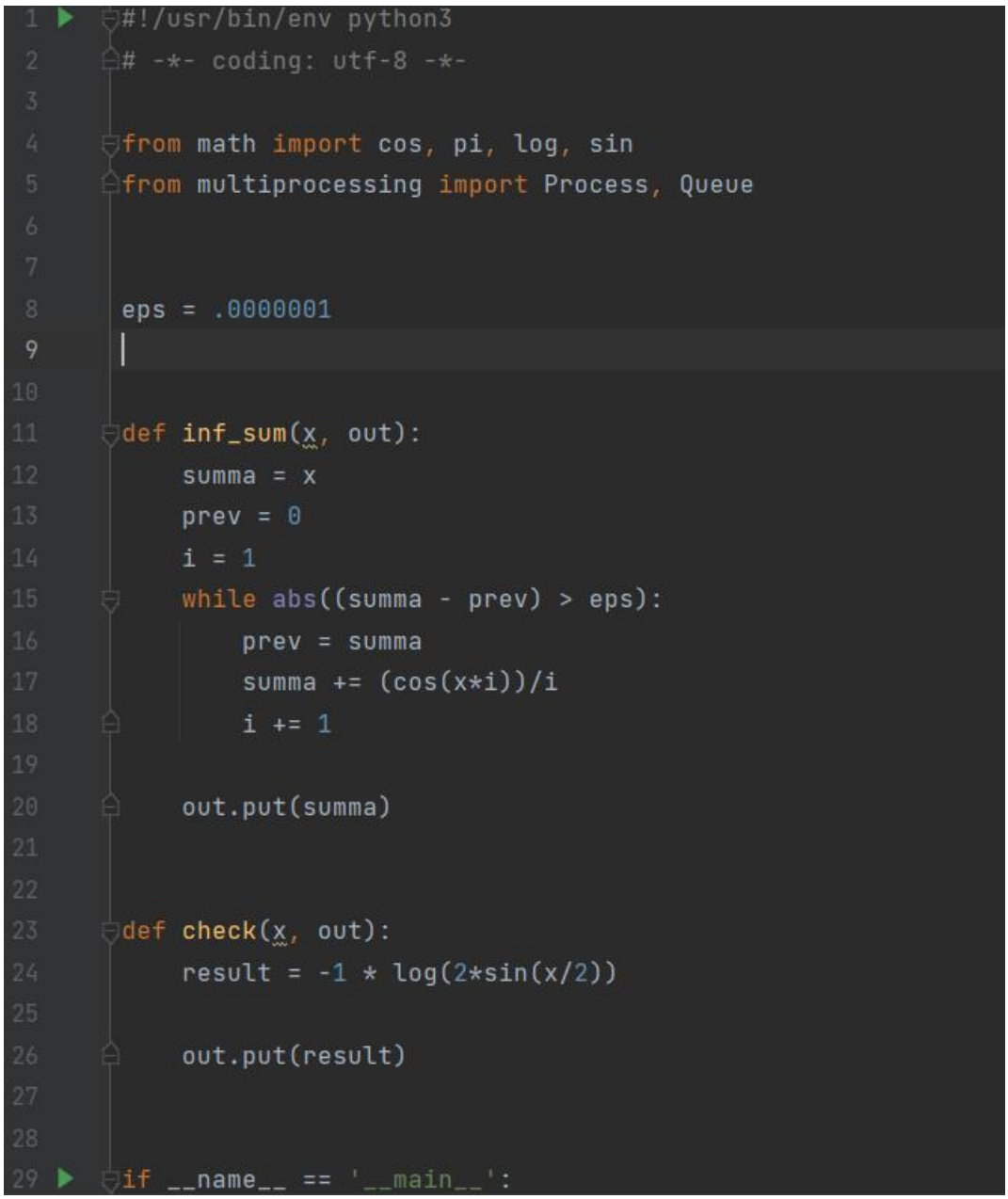
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

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()

```

Рисунок 3 – Пример 3



```python
#!/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__":
    # Указание на то, что процесс демон при создании объекта класса Process
    proc1 = Process(target=func, args=("proc1",), daemon=True)

    # Указание на то, что процесс демон через свойство daemon
    proc2 = Process(target=func, args=("proc2",))
    proc2.daemon = True

    # Запуск процессов
    proc1.start()
```

Рисунок 4 – Пример 4



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

from math import cos, pi, log, sin
from multiprocessing import Process, Queue


eps = .0000001

def inf_sum(x, out):
    summa = x
    prev = 0
    i = 1
    while abs((summa - prev) > eps):
        prev = summa
        summa += (cos(x*i))/i
        i += 1

    out.put(summa)


def check(x, out):
    result = -1 * log(2*sin(x/2))

    out.put(result)


if __name__ == '__main__':
```

Рисунок 5 – Индивидуальное задание №1

Контрольные вопросы

1. Как создаются и завершаются процессы в Python?

Процессы в Python создаются с помощью модуля multiprocessing. Для создания процесса необходимо создать объект класса Process, передав в конструктор функцию или метод класса, который будет выполняться в процессе. Для запуска процесса используется метод start(). Завершение процесса происходит автоматически при завершении выполнения функции или метода, переданного в конструктор Process.

2. В чем особенность создания классов-наследников от Process?

Особенность создания классов-наследников от Process заключается в том, что в методе run() необходимо указать код, который будет выполняться в процессе. Этот метод вызывается при запуске процесса. Также класс-наследник должен иметь свой конструктор, в котором нужно вызвать конструктор родительского класса и передать в него аргументы.

3. Как выполнить принудительное завершение процесса?

Для принудительного завершения процесса в Python можно использовать метод terminate(), который отправляет процессу сигнал SIGTERM. Обработка этого сигнала внутри процесса приводит к его завершению. Однако, не рекомендуется использовать terminate() без необходимости, так как это может привести к утечкам ресурсов.

4. Что такое процессы-демоны? Как запустить процесс-демон?

Процессы-демоны - это процессы, которые работают в фоновом режиме, не взаимодействуя с пользователем. Они не имеют своего терминала и не могут взаимодействовать с консолью. Запустить процесс-демон можно, используя методы из модуля daemonize. Кроме того, при создании процесса можно указать параметр daemon=True, что превращает процесс в демон.