

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №10
дисциплины «Программирование на языке Python»

Вариант_15_

Выполнила:
Маньшина Дарья Алексеевна
2 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., канд. тех. наук,
доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

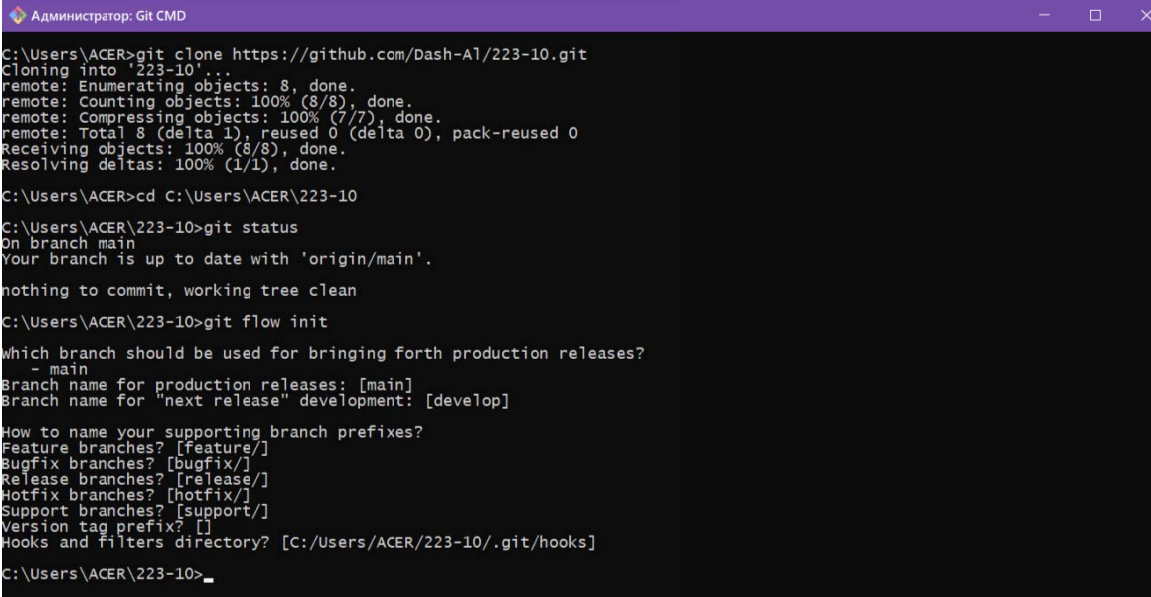
Ставрополь, 2023 г.

Тема: управление потоками в Python

Цель: приобретение навыков написания многопоточных приложений на языке программирования Python версии 3.x.

Ход работы:

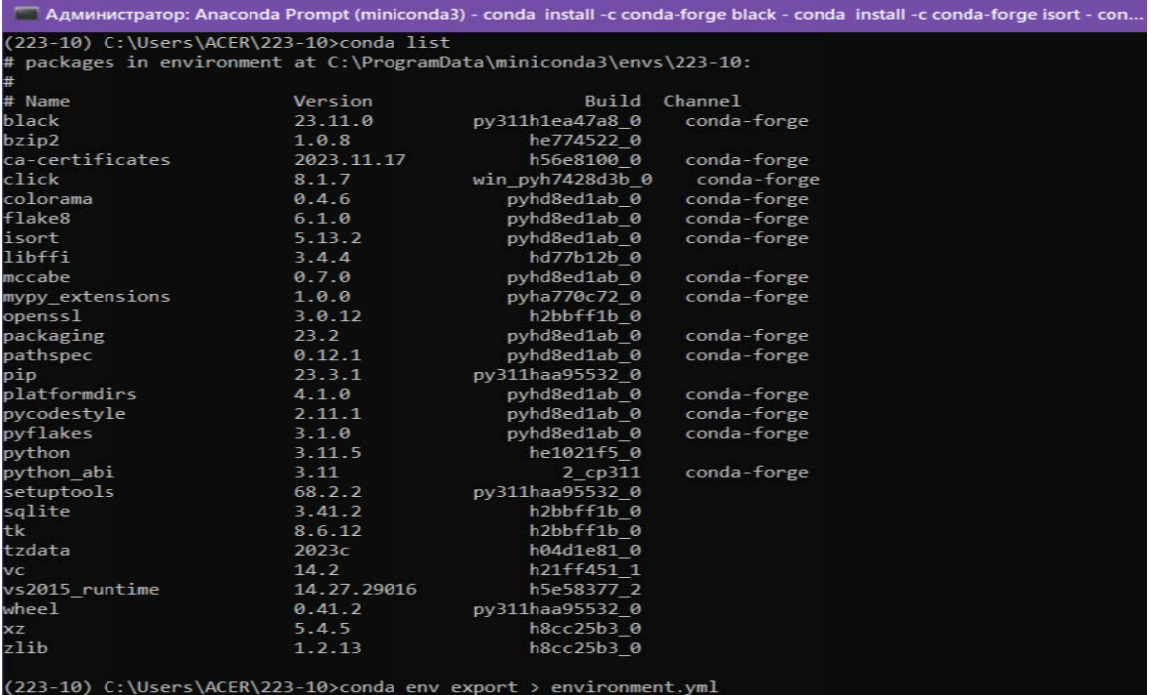
1. Подготовка к выполнению работы. Клонирование репозитория и добавление пакетов black, isort, flake8.



```
Администратор: Git CMD
C:\Users\ACER>git clone https://github.com/Dash-A1/223-10.git
Cloning into '223-10'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
C:\Users\ACER>cd C:\Users\ACER\223-10
C:\Users\ACER\223-10>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
C:\Users\ACER\223-10>git flow init
Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]
How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/ACER/223-10/.git/hooks]
C:\Users\ACER\223-10>
```

Рисунок 1. Клонирование репозитория



```
Администратор: Anaconda Prompt (miniconda3) - conda install -c conda-forge black - conda install -c conda-forge isort - con...
(223-10) C:\Users\ACER\223-10>conda list
# packages in environment at C:\ProgramData\miniconda3\envs\223-10:
#
# Name                    Version           Build    Channel
black                     23.11.0           py311h1ea47a8_0  conda-forge
bzip2                     1.0.8             he774522_0      conda-forge
ca-certificates          2023.11.17        h56e8100_0      conda-forge
click                     8.1.7             win_pyh7428d3b_0  conda-forge
colorama                  0.4.6             pyhd8ed1ab_0     conda-forge
flake8                    6.1.0             pyhd8ed1ab_0     conda-forge
isort                     5.13.2            pyhd8ed1ab_0     conda-forge
libffi                    3.4.4             hd77b12b_0       conda-forge
mccabe                    0.7.0             pyhd8ed1ab_0     conda-forge
mypy_extensions          1.0.0             pyha770c72_0     conda-forge
openssl                   3.0.12            h2bbff1b_0       conda-forge
packaging                 23.2              pyhd8ed1ab_0     conda-forge
pathspec                  0.12.1            pyhd8ed1ab_0     conda-forge
pip                       23.3.1            py311haa95532_0  conda-forge
platformdirs              4.1.0             pyhd8ed1ab_0     conda-forge
pycodestyle               2.11.1            pyhd8ed1ab_0     conda-forge
pyflakes                  3.1.0             pyhd8ed1ab_0     conda-forge
python                    3.11.5            he1021f5_0       conda-forge
python_abi                3.11              2_cp311          conda-forge
setuptools                68.2.2            py311haa95532_0  conda-forge
sqlite                    3.41.2            h2bbff1b_0       conda-forge
tk                         8.6.12            h2bbff1b_0       conda-forge
tzdata                    2023c             h04d1e81_0       conda-forge
vc                         14.2              h21ff451_1       conda-forge
vs2015_runtime            14.27.29016       h5e58377_2       conda-forge
wheel                     0.41.2            py311haa95532_0  conda-forge
xz                         5.4.5             h8cc25b3_0       conda-forge
zlib                      1.2.13            h8cc25b3_0       conda-forge
(223-10) C:\Users\ACER\223-10>conda env export > environment.yml
```

Рисунок 2. Установка пакетов

Проработаем примеры из методички.

Пример 1. Вариант создания потока на базе функции. В приведенном примере мы импортировали нужные модули. После этого объявили функцию func(), которая выводит пять раз сообщение с числовым маркером с задержкой в 500 мс. Далее создали объект класса Thread, в нем, через параметр target, указали, какую функцию запускать как поток и запустили его. В главном потоке добавили код вывода сообщений с интервалом в 1000 мс.

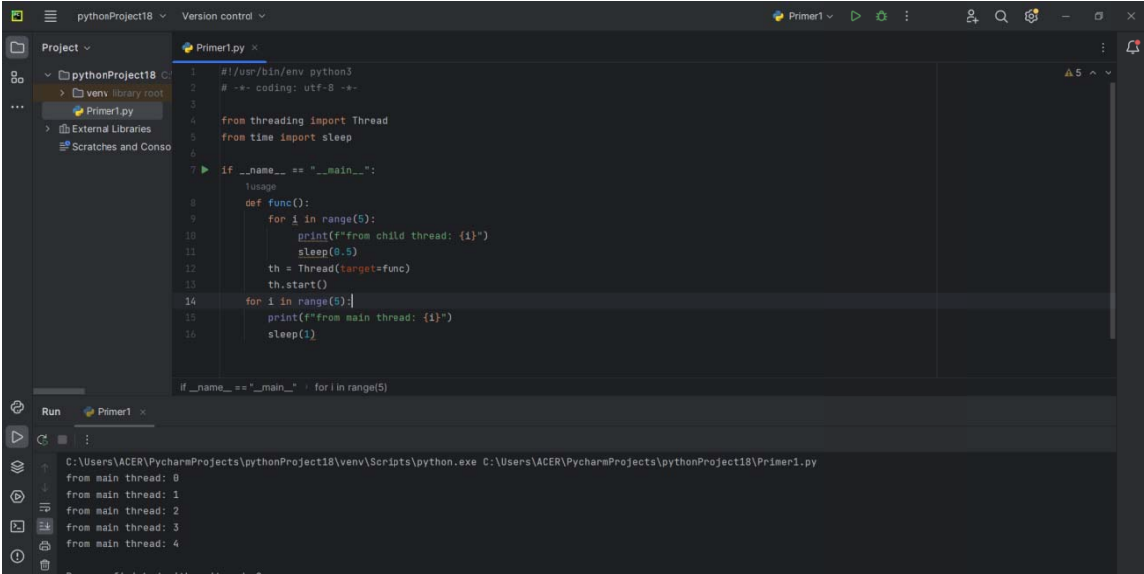
Программа:

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

from threading import Thread
from time import sleep

if __name__ == "__main__":
    def func():
        for i in range(5):
            print(f"from child thread: {i}")
            sleep(0.5)
        th = Thread(target=func)
        th.start()
    for i in range(5):
        print(f"from main thread: {i}")
        sleep(1)
```

Результат:



```
Run Primer1 x
C:\Users\ACER\PycharmProjects\pythonProject18\venv\Scripts\python.exe C:\Users\ACER\PycharmProjects\pythonProject18\Primer1.py
from main thread: 0
from main thread: 1
from main thread: 2
from main thread: 3
from main thread: 4
Process finished with exit code 0
```

Рисунок 3. Результат программы примера №1

Пример 2. В Python у объектов класса Thread нет методов для принудительного завершения работы потока. Один из вариантов решения этой задачи – это создать специальный флаг, через который потоку будет передаваться сигнал остановки. Доступ к такому флагу должен управляться объектом синхронизации.

Программа:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from threading import Thread, Lock
from time import sleep
if __name__ == "__main__":
    lock = Lock()
    stop_thread = False
    def infinit_worker():
        print("Start infinit_worker()")
    while True:
        print("--> thread work")
        lock.acquire()
        if stop_thread is True:
            break
        lock.release()
        sleep(0.1)
    print("Stop infinit_worker()")
    # Create and start thread
    th = Thread(target=infinit_worker)
    th.start()
    sleep(2)
    # Stop thread
    lock.acquire()
    stop_thread = True
    lock.release()
```

Результат:

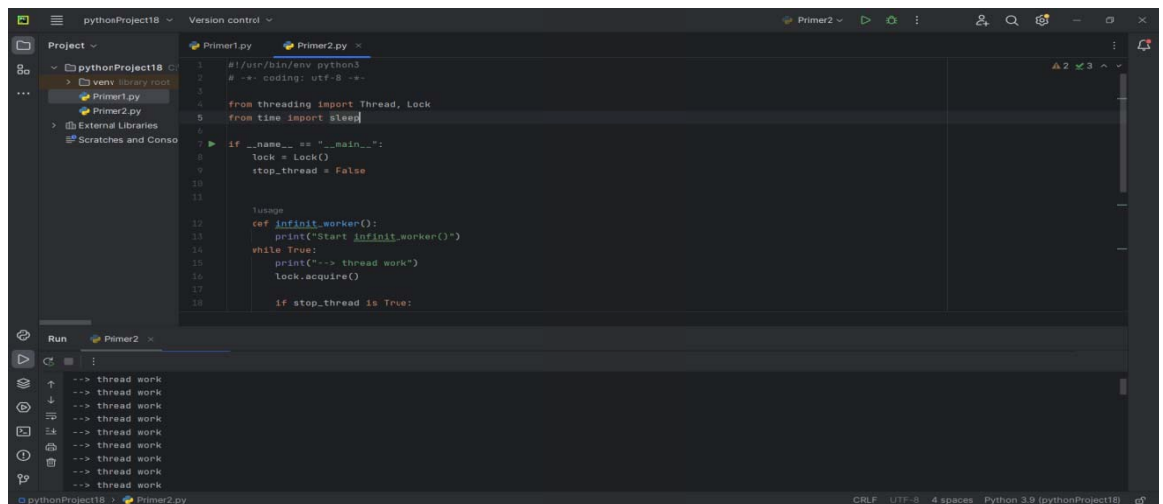
The image is a screenshot of a code editor window, likely PyCharm, showing a Python script and its execution output. The script, named 'Primer2.py', implements a thread that runs indefinitely until a stop signal is received. It uses a Lock to ensure thread safety when accessing a shared 'stop_thread' flag. The main thread sleeps for 2 seconds before setting the flag to True and releasing the lock. The console output shows the thread printing '--> thread work' multiple times, followed by 'Stop infinit_worker()'.

Рисунок 4. Результат программы примера №2

Пример 3. Есть такая разновидность потоков, которые называются демоны. Python приложение не будет закрыто до тех пор, пока в нем работает хотя бы один недемонический поток. Для того, чтобы потоки не мешали остановке приложения (т.е. чтобы они останавливались вместе с завершением работы программы) необходимо при создании объекта Thread аргументу `daemon` присвоить значение `True`, либо после создания потока, перед его запуском присвоить свойству `daemon` значение `True`.

Программа:

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

from threading import Thread
from time import sleep

if __name__ == "__main__":
    def func():
        for i in range(5):
            print(f"from child thread: {i}")
            sleep(0.5)

    th = Thread(target=func, daemon=True)
    th.start()
    print("App stop")
```

Результат:

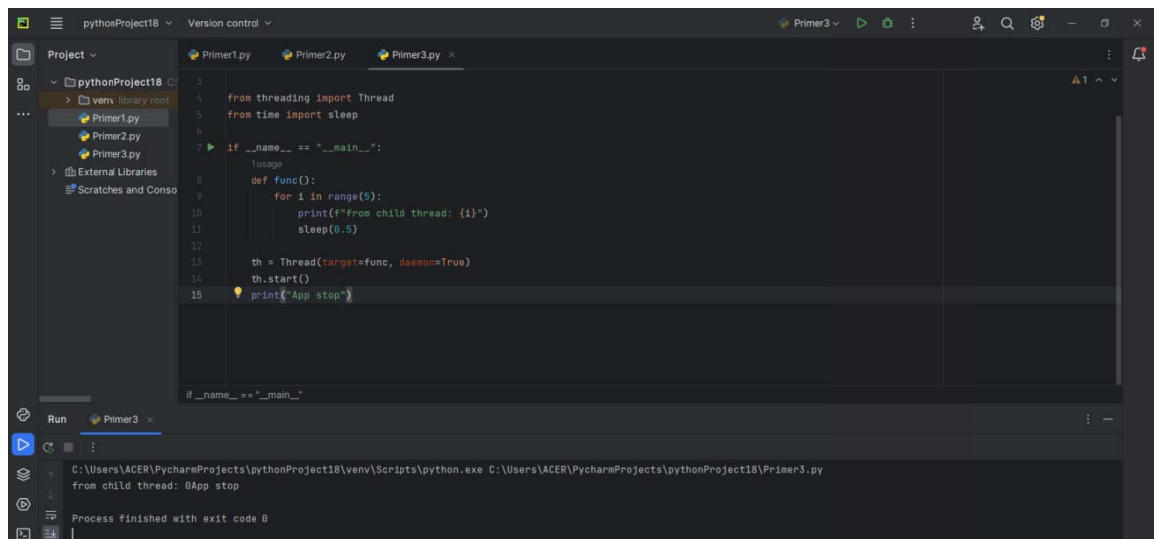


Рисунок 5. Результат программы примера №3

Решим индивидуальное задание

Условие: С использованием многопоточности для заданного значения найти сумму ряда с точностью члена ряда по абсолютному значению $\epsilon=10^{-7}$ и произвести сравнение полученной суммы с контрольным значением функции для двух бесконечных рядов.

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

Программа:

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

import threading
import math

class SeriesThread(threading.Thread):
    def __init__(self, x, epsilon):
        # Инициализация объекта
        threading.Thread.__init__(self)
        self.x = x
        # Заданная точность для вычислений
        self.epsilon = epsilon
        # Инициализация переменной для хранения суммы ряда
        self.result = 0

    def run(self):
        n = 0
        term = (self.x ** (2 * n)) / math.factorial(2 * n)

        # Вычисление суммы ряда до достижения заданной точности epsilon
        while abs(term) > self.epsilon:
            self.result += term
            n += 1
            term = (self.x ** (2 * n)) / math.factorial(2 * n)

def main():
    x = 1/2
    epsilon = 1e-7

    # Задаем контрольное значение
    control_value = (math.exp(x) + math.exp(-x)) / 2
    series_thread = SeriesThread(x, epsilon)
    series_thread.start()
    series_thread.join()
    series_sum = series_thread.result

    # Сравниваем результат с контрольным значением
    if abs(series_sum - control_value) < epsilon:
        print(f'Сумма ряда: {series_sum}')
        print(f'Контрольное значение: {control_value}')
        print("Результат совпадает с контрольным значением.")
```

```
else:  
    print("Результат не совпадает с контрольным значением.")
```

```
if __name__ == "__main__":  
    main()
```

Результат:

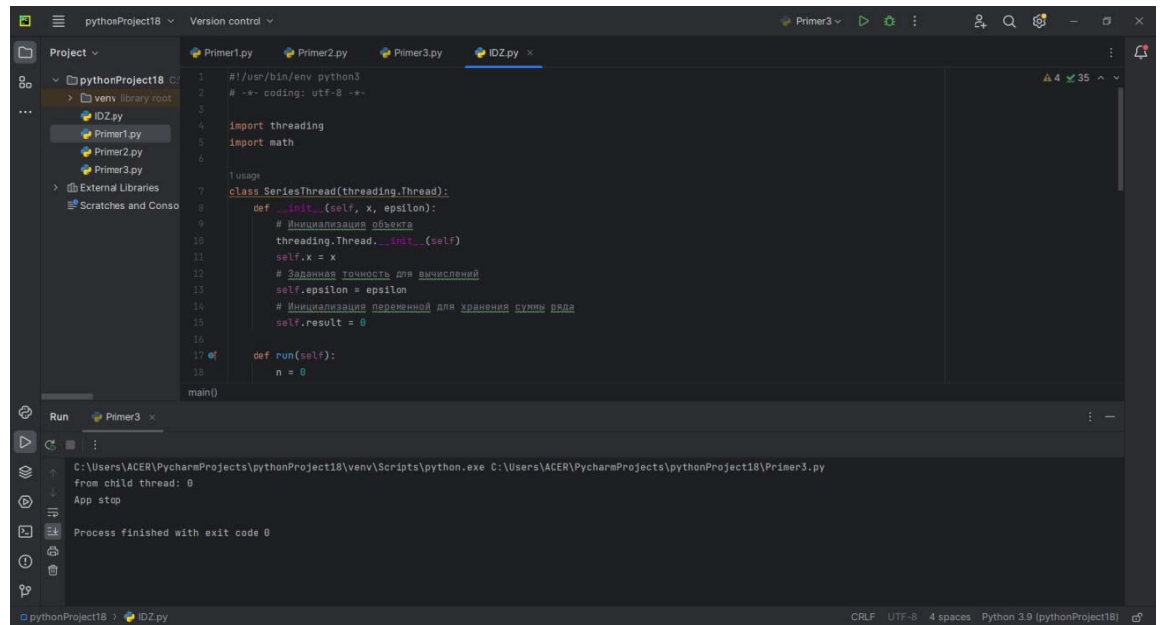


Рисунок 6. Результат работы программы ИДЗ

Вывод: в ходе лабораторной работы приобрела навыки написания многопоточных приложений на языке программирования Python версии 3.x.

Контрольные вопросы

1. Что такое синхронность и асинхронность?

Синхронность - это выполнение операций последовательно, одна за другой. Асинхронность же означает, что операции могут выполняться параллельно, одновременно.

2. Что такое параллелизм и конкурентность?

Параллелизм - это когда несколько операций выполняются одновременно, но могут использовать одни и те же ресурсы. Конкурентность - это когда операции конкурируют за использование ресурсов, например,

когда несколько потоков пытаются получить доступ к одному и тому же ресурсу одновременно.

3. Что такое GIL? Какое ограничение накладывает GIL?

GIL (Global Interpreter Lock) - это механизм, который гарантирует, что только один поток интерпретатора Python может быть активен в любой момент времени. Это накладывает ограничение на производительность многопоточных приложений на Python, поскольку только один поток может выполняться в любой момент.

4. Каково назначение класса Thread?

Класс Thread используется для создания новых потоков выполнения в Python. Он позволяет выполнять код параллельно с основным потоком и управлять временем жизни потока.

5. Как реализовать в одном потоке ожидание завершения другого потока?

Ожидание завершения другого потока может быть реализовано с помощью методов `join` или `lock`, доступных для потоков.

6. Как проверить факт выполнения потоком некоторой работы?

Факт выполнения потоком некоторой работы можно проверить с помощью методов `wait` и `notify`, доступных в классе Thread.

7. Как реализовать приостановку выполнения потока на некоторый промежуток времени?

Приостановка выполнения потока на некоторое время может быть реализована с помощью метода `sleep`.

8. Как реализовать принудительное завершение потока?

Принудительное завершение потока может быть выполнено с помощью метода `terminate`. Однако это может привести к нежелательным последствиям, поэтому обычно рекомендуется использовать более безопасные методы, такие как `join` или `exit`.

9. Что такое потоки-демоны? Как создать поток-демон?

Потоки-демоны - это потоки, которые автоматически завершаются при завершении основного потока. Создание потока-демона может быть выполнено с использованием ключевого слова `daemon` при создании потока.