

## Лабораторная работа 2.23


Тема: Управление потоками в Python

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

Ссылка на GitHub: [https://github.com/RomanGorchakov/Py3\\_10](https://github.com/RomanGorchakov/Py3_10)

### Порядок выполнения работы

1. Создаём новый общедоступный репозиторий, в котором будет использована лицензия MIT и язык программирования Python.

 Start writing code ...

**Start a new repository for RomanGorchakov**

A repository contains all of your project's files, revision history, and collaborator discussion.

Repository name \*

Py3\_10

✓ Py3\_10 is available.

☒ **Public**  
Anyone on the internet can see this repository

☐ **Private**  
You choose who can see and commit to this repository

Create a new repository

**Introduce yourself with a profile README**

Share information about yourself by creating a profile README, which appears at the top of your profile page.

RomanGorchakov / README.md 

Create

1 - 👋 Hi, I'm @RomanGorchakov

2 - 🧐 I'm interested in ...

3 - 🌱 I'm currently learning ...

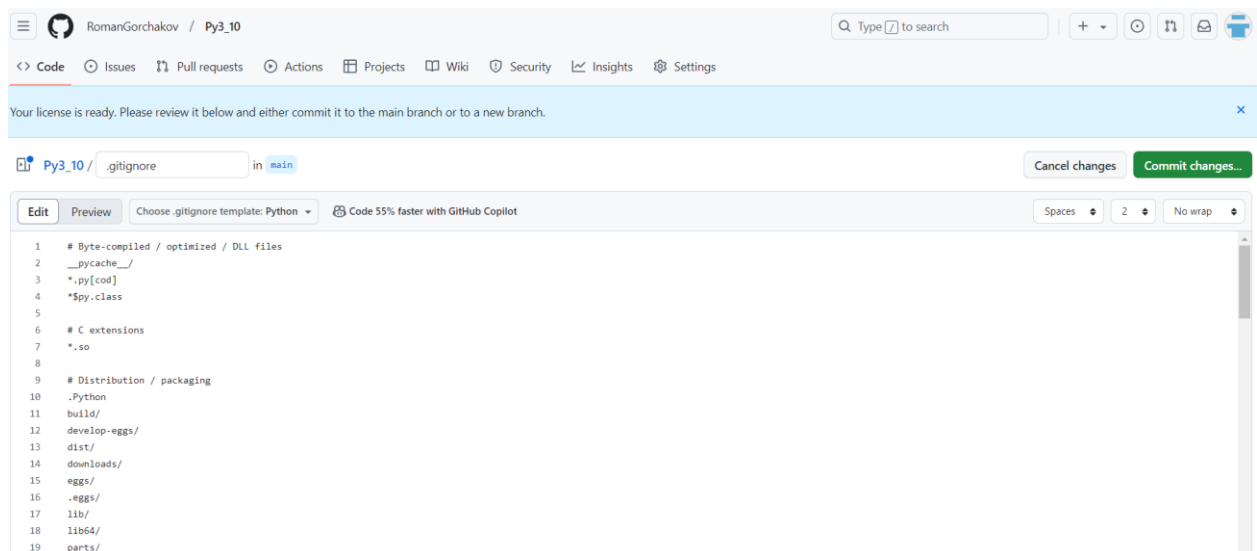
4 - ❤️ I'm looking to collaborate on ...

5 - 💬 How to reach me ...

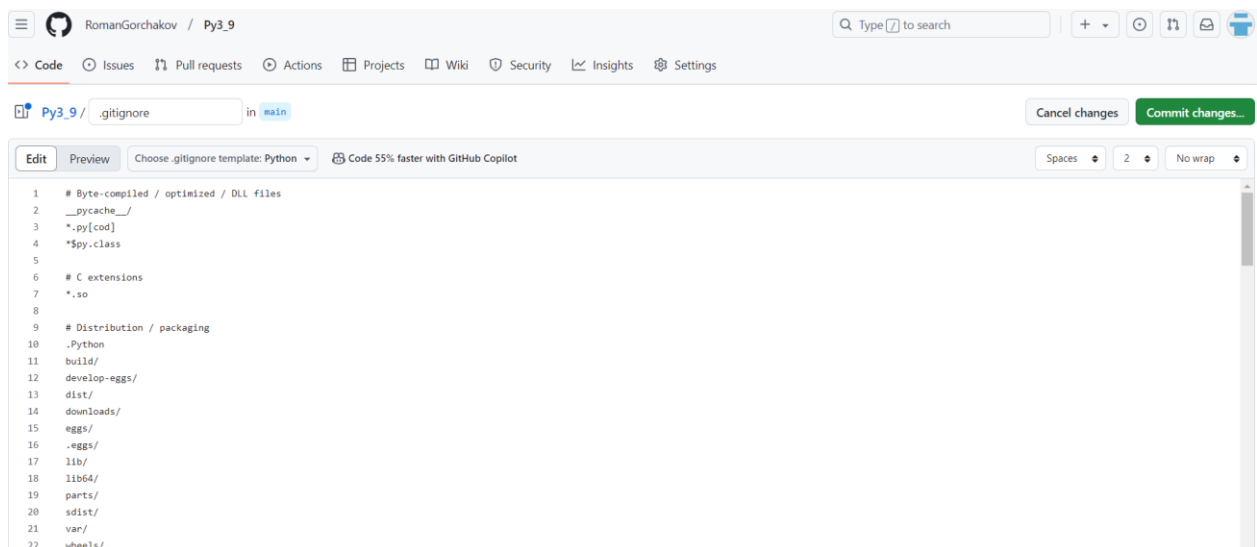
6 - 🗨️ Pronouns: ...

7 - ⚡ Fun fact: ...

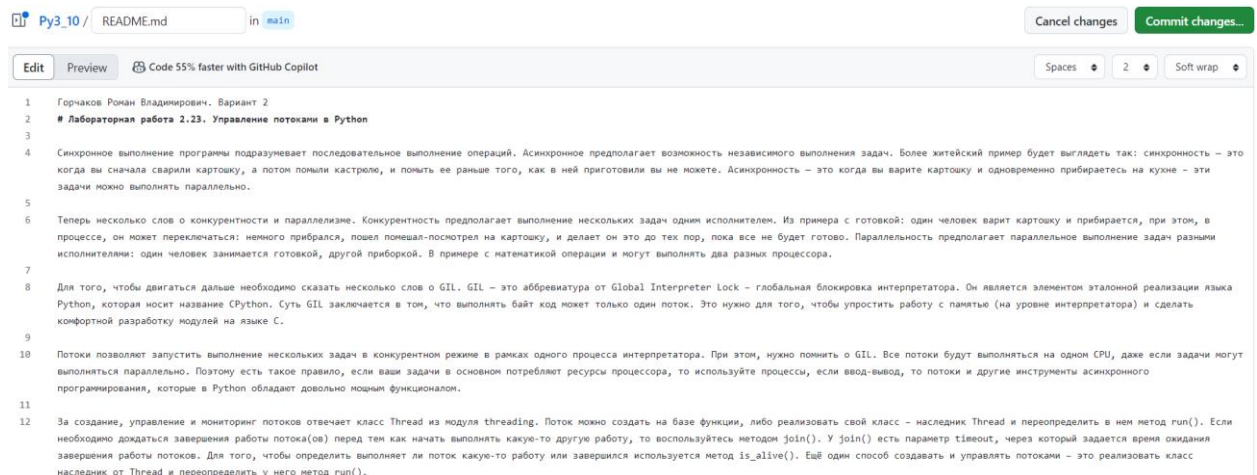
8



2. Теперь необходимо дополнить файл `.gitignore` с необходимыми правилами для языка программирования Python. Для этого переходим по ссылке «<https://github.com/github/gitignore>» и скачиваем оттуда файл «Python.gitignore».



3. Теперь создаём файл «README.md», где вносим ФИО и теоретический конспект лекции. Сохраняем набранный текст через кнопку «Commit changes».



4. В окне «Codespace» выбираем опцию «Create codespace on main». Откроется терминал, куда мы введём команду «git clone», чтобы клонировать свой репозиторий. После этого организуем репозиторий в соответствии с моделью ветвления Git-flow. Для этого введём в терминал команды: «git checkout –b develop» для создания ветки разработки; «git branch feature\_branch» для создания ветки функций; «git branch release/1.0.0» для создания ветки релиза; «git checkout main» и «git branch hotfix» для создания веток hotfix. Устанавливаем библиотеки isort, black и flake8 и создаём файлы .pre-commit-config.yaml и environment.yml.

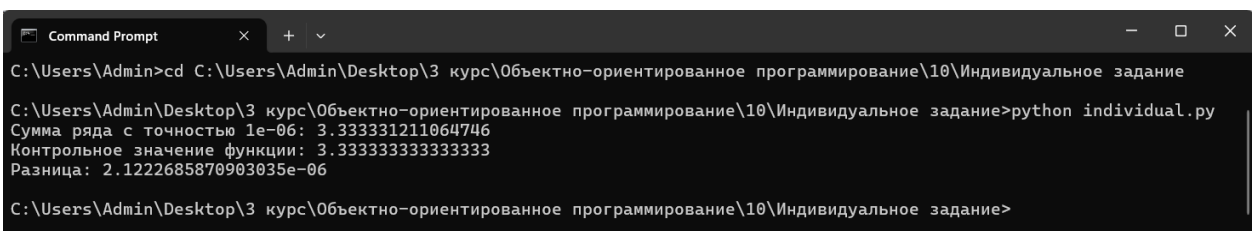
```
@RomanGorchakov →/workspaces/Py3_10 (main) $ git checkout -b develop
Switched to a new branch 'develop'
• @RomanGorchakov →/workspaces/Py3_10 (develop) $ git branch feature_branch
• @RomanGorchakov →/workspaces/Py3_10 (develop) $ git branch release/1.0.0
• @RomanGorchakov →/workspaces/Py3_10 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
• @RomanGorchakov →/workspaces/Py3_10 (main) $ git branch hotfix
• @RomanGorchakov →/workspaces/Py3_10 (main) $ git checkout develop
Switched to branch 'develop'
○ @RomanGorchakov →/workspaces/Py3_10 (develop) $
```

```

@RomanGorchakov → /workspaces/Py3_10 (develop) $ pip install black
Collecting black
  Downloading black-24.10.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl.metadata (79 kB)
Collecting click>=8.0.0 (from black)
  Downloading click-8.1.8-py3-none-any.whl.metadata (2.3 kB)
Collecting mypy_extensions>=0.4.3 (from black)
  Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: packaging>=22.0 in /home/codespace/.local/lib/python3.12/site-packages (from black) (24.2)
Collecting pathspec>=0.9.0 (from black)
  Downloading pathspec-0.12.1-py3-none-any.whl.metadata (21 kB)
Requirement already satisfied: platformdirs>=2 in /home/codespace/.local/lib/python3.12/site-packages (from black) (4.3.6)
Downloading black-24.10.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (1.8 MB)
  1.8/1.8 MB 55.6 MB/s eta 0:00:00
Downloading click-8.1.8-py3-none-any.whl (98 kB)
Downloading mypy_extensions-1.0.0-py3-none-any.whl (4.7 kB)
Downloading pathspec-0.12.1-py3-none-any.whl (31 kB)
Installing collected packages: pathspec, mypy_extensions, click, black
Successfully installed black-24.10.0 click-8.1.8 mypy_extensions-1.0.0 pathspec-0.12.1
@RomanGorchakov → /workspaces/Py3_10 (develop) $ pip install flake8
Collecting flake8
  Downloading flake8-7.1.1-py2.py3-none-any.whl.metadata (3.8 kB)
Collecting mccabe<0.8.0,>=0.7.0 (from flake8)
  Downloading mccabe-0.7.0-py2.py3-none-any.whl.metadata (5.0 kB)
Collecting pycodestyle<2.13.0,>=2.12.0 (from flake8)
  Downloading pycodestyle-2.12.1-py2.py3-none-any.whl.metadata (4.5 kB)
Collecting pyflakes<3.3.0,>=3.2.0 (from flake8)
  Downloading pyflakes-3.2.0-py2.py3-none-any.whl.metadata (3.5 kB)
Downloading flake8-7.1.1-py2.py3-none-any.whl (57 kB)
Downloading mccabe-0.7.0-py2.py3-none-any.whl (7.3 kB)
Downloading pycodestyle-2.12.1-py2.py3-none-any.whl (31 kB)
Downloading pyflakes-3.2.0-py2.py3-none-any.whl (62 kB)
Installing collected packages: pyflakes, pycodestyle, mccabe, flake8
Successfully installed flake8-7.1.1 mccabe-0.7.0 pycodestyle-2.12.1 pyflakes-3.2.0
@RomanGorchakov → /workspaces/Py3_10 (develop) $ pre-commit sample-config > .pre-commit-config.yaml
@RomanGorchakov → /workspaces/Py3_10 (develop) $ conda env export > environment.yml

```

5. Создаём файл «individual.py», в котором нужно написать программу, состоящую из двух списков Listbox. В первом будет, например, перечень товаров, заданный программно. Второй изначально пуст, пусть это будет перечень покупок. При клике на одну кнопку товар должен переходить из одного списка в другой. При клике на вторую кнопку – возвращаться (человек передумал покупать).



```

C:\Users\Admin>cd C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\10\Индивидуальное задание
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\10\Индивидуальное задание>python individual.py
Сумма ряда с точностью 1e-06: 3.333331211064746
Контрольное значение функции: 3.333333333333333
Разница: 2.1222685870903035e-06
C:\Users\Admin\Desktop\3 курс\Объектно-ориентированное программирование\10\Индивидуальное задание>

```

6. Выполняем коммит файлов в репозиторий Git в ветку разработки, сливаем её с веткой main и отправляем изменения на сервер GitHub.

```

@RomanGorchakov →/workspaces/Py3_10 (main) $ git merge develop
Updating 819a420..0adf598
Fast-forward
 .pre-commit-config.yaml      | 38 ++++++
 .python-version              | 1 +
 edu.pyoop.code-workspace     | 47 ++++++
 environment.yml              | 80 ++++++
 pyproject.toml               | 88 ++++++
 setup.cfg                    | 43 ++++++
 uv.lock                      | 303 ++++++
 .../individual.py"           | 46 ++++++
 ...\\320\\2404.8_\\320\\223\\320\\276\\321\\200\\321\\207\\320\\260\\320\\272\\320\\276\\320\\262\\320\\240\\320\\222.pdf" | Bin 0 -> 1071047 bytes
9 files changed, 646 insertions(+)
create mode 100644 .pre-commit-config.yaml
create mode 100644 .python-version
create mode 100644 edu.pyoop.code-workspace
create mode 100644 environment.yml
create mode 100644 pyproject.toml
create mode 100644 setup.cfg
create mode 100644 uv.lock
create mode 100644 "...\\320\\230\\320\\275\\320\\264\\320\\270\\320\\262\\320\\270\\320\\264\\321\\203\\320\\260\\320\\273\\321\\214\\320\\275\\320\\276\\320\\265_\\320\\267\\320\\260\\320\\264\\320\\260\\320\\275\\320\\270\\320\\265\\individual.py"
create mode 100644 "...\\320\\236\\321\\202\\321\\221\\321\\202\\320\\233\\320\\2404.8_\\320\\223\\320\\276\\321\\200\\321\\207\\320\\260\\320\\272\\320\\276\\320\\262\\320\\240\\320\\222.pdf"
@RomanGorchakov →/workspaces/Py3_10 (main) $ git push -u
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 2 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (13/13), 1000.00 KiB | 22.73 MiB/s, done.
Total 13 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/RomanGorchakov/Py3_10
 819a420..0adf598  main -> main
branch 'main' set up to track 'origin/main'.
@RomanGorchakov →/workspaces/Py3_10 (main) $

```

Py3\_10 Public

Pin
Unwatch 1
Fork 0
Star 0

main 1 Branch 0 Tags

Go to file Add file Code

RomanGorchakov Threads 0adf598 · 9 minutes ago 4 Commits

Индивидуальное задание	Threads	9 minutes ago
Отчёт	Threads	9 minutes ago
.gitignore	Create .gitignore	53 minutes ago
.pre-commit-config.yaml	Threads	9 minutes ago
.python-version	Threads	9 minutes ago
LICENSE	Create LICENSE	53 minutes ago
README.md	Create README.md	23 minutes ago
edu.pyoop.code-workspace	Threads	9 minutes ago
environment.yml	Threads	9 minutes ago
pyproject.toml	Threads	9 minutes ago
setup.cfg	Threads	9 minutes ago
uv.lock	Threads	9 minutes ago

README MIT license

Горчаков Роман Владимирович. Вариант 2

## Лабораторная работа 2.23. Управление потоками в Python

About

No description, website, or topics provided.

Readme
MIT license
Activity
0 stars
1 watching
0 forks

Releases
No releases published
[Create a new release](#)

Packages
No packages published
[Publish your first package](#)

Languages
Python 100.0%

Suggested workflows
Based on your tech stack

Django
Configure
Build and Test a Django Project

Python package
Configure
Create and test a Python package on multiple Python versions.

## Контрольные вопросы

### 1. Что такое синхронность и асинхронность?

Синхронное выполнение программы подразумевает последовательное выполнение операций. Асинхронное предполагает возможность независимого выполнения задач.

## 2. Что такое параллелизм и конкурентность?

Конкурентность предполагает выполнение нескольких задач одним исполнителем. Параллельность предполагает параллельное выполнение задач разными исполнителями: один человек занимается готовкой, другой приборкой.

## 3. Что такое GIL? Какое ограничение накладывает GIL?

GIL – аббревиатура от Global Interpreter Lock – глобальная блокировка интерпретатора. Он является элементом эталонной реализации языка Python, которая носит название CPython. Суть GIL заключается в том, что выполнять байт код может только один поток. Это нужно для того, чтобы упростить работу с памятью (на уровне интерпретатора) и сделать комфортной разработку модулей на языке C.

Если вы запустили в одном интерпретаторе несколько потоков, которые в основном используют процессор, то скорее всего получите общее замедление работы, а не прирост производительности. Пока выполняется одна задача, остальные простаивают (из-за GIL), переключение происходит через определенные промежутки времени. Таким образом, в каждый конкретный момент времени, будет выполняться только один поток, несмотря на то что у вас может быть многоядерный процессор (или многопроцессорный сервер), плюс ко всему, будет тратиться время на переключение между задачами

## 4. Каково назначение класса Thread?

За создание, управление и мониторинг потоков отвечает класс Thread из модуля `threading`. Поток можно создать на базе функции, либо реализовать свой класс – наследник Thread и переопределить в нем метод `run()`.

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

Если необходимо дождаться завершения работы потока(ов) перед тем как начать выполнять какую-то другую работу, то воспользуйтесь методом `join()`.

## 6. Как проверить факт выполнения потоком некоторой работы?

Для того, чтобы определить выполняет ли поток какую-то работу или завершился используется метод `is_alive()`.

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

Для реализации приостановки выполнения потока на определённый промежуток времени в Python используется функция `sleep()`.

8. Как реализовать принудительное завершение потока?

В Python у объектов класса `Thread` нет методов для принудительного завершения работы потока. Один из вариантов решения этой задачи – создание специального флага, через который потоку будет передаваться сигнал остановки. Доступ к такому флагу должен управляться объектом синхронизации.

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

Есть такая разновидность потоков, которые называются демоны (терминология взята из мира Unix-подобных систем). 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`.