

# Лабораторная работа 1.1

Тема: Исследование основных возможностей Git и GitHub.

Цель работы: исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

## Порядок выполнения работы


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

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).

Owner \*

 RomanGorchakov ▾

Repository name \*

/ Test

✓ Test is available.

Great repository names are short and memorable. Need inspiration? How about [automatic-octo-chainsaw](#) ?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

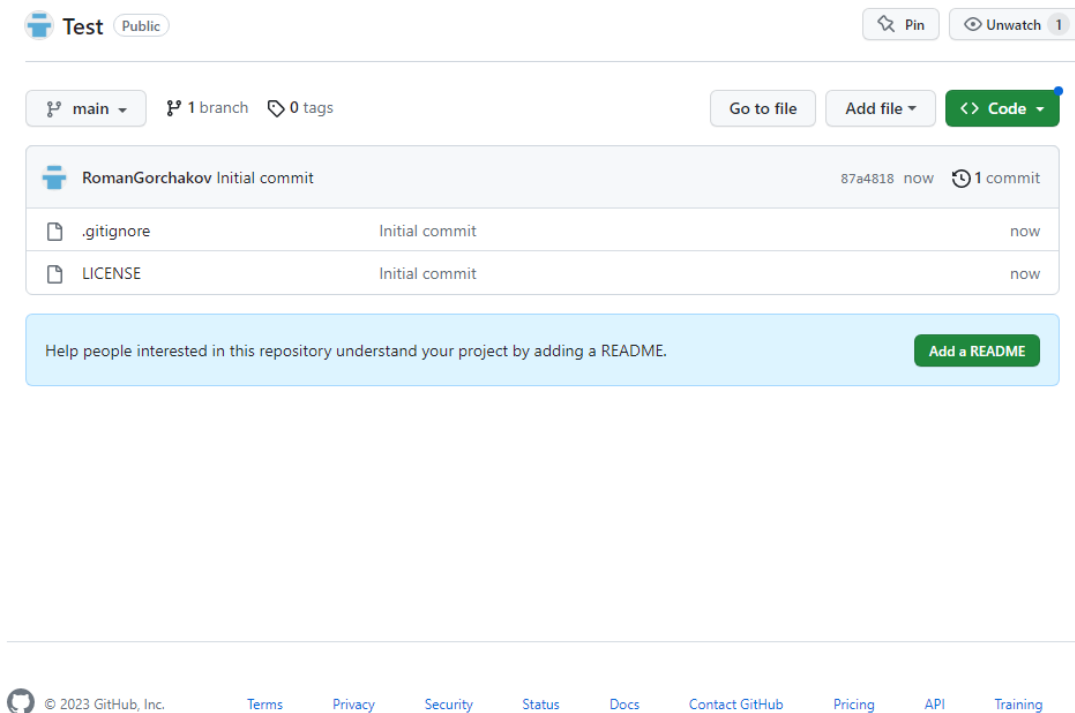
Choose a license

License: MIT License ▾

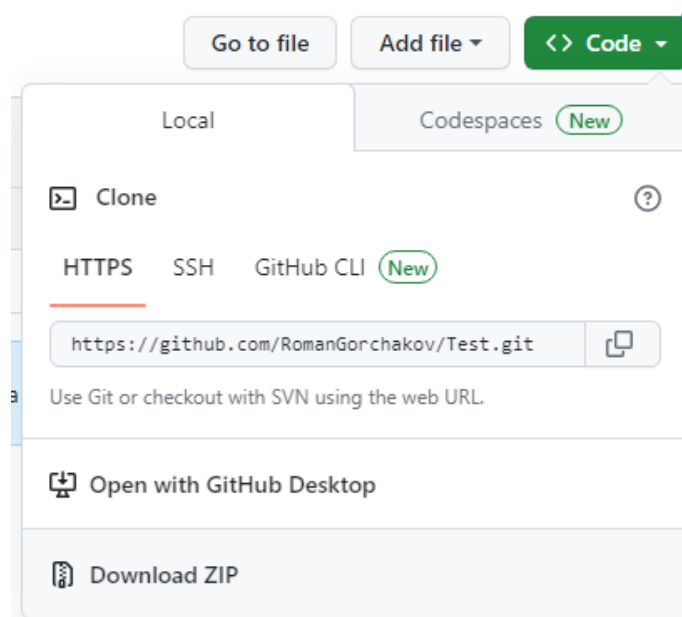
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.


















Create repository



2. В созданном репозитории нажимаем на зелёную кнопку Code, чтобы скопировать репозиторий на компьютер.



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

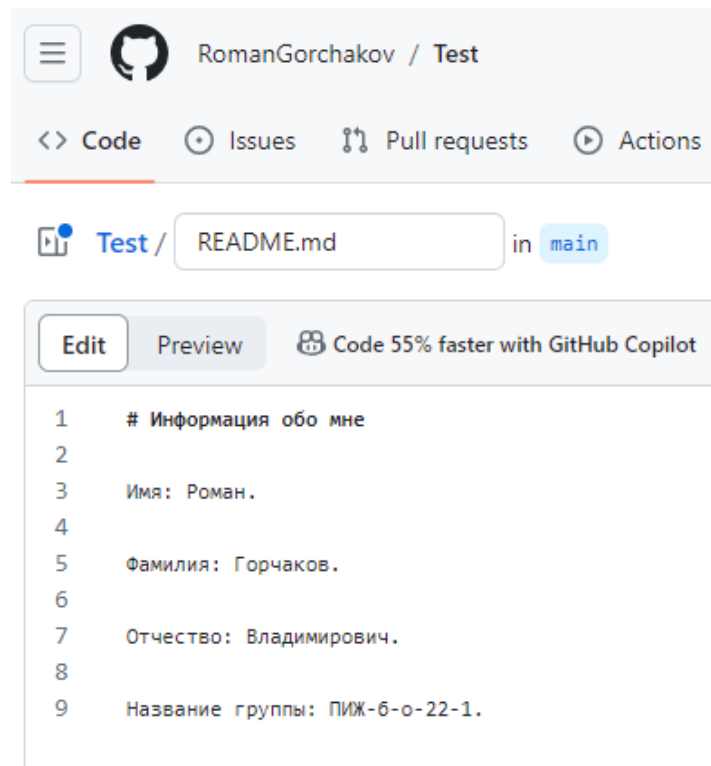
 Phalcon.gitignore	Remove trailing asterisks in Phalcon rules	10 years ago
 PlayFramework.gitignore	Added /project/project to PlayFramework.gitignore	7 years ago
 Plone.gitignore	Covered by global vim template	10 years ago
 Prestashop.gitignore	Update for Prestashop 1.7 (#3261)	3 years ago
 Processing.gitignore	Ignore transpiled .java and .class files (#3016)	4 years ago
 PureScript.gitignore	Update PureScript adding .spago (#3278)	3 years ago
 <a href="#">Python.gitignore</a>	Update Python.gitignore	last year
 Qooxdoo.gitignore	Add gitignore for qooxdoo apps	13 years ago
 Qt.gitignore	Remove trailing whitespace	2 years ago
 R.gitignore	Merge pull request #3792 from jl5000/patch-1	2 years ago
 README.md	Merge pull request #3854 from AnilSeervi/patch-1	2 years ago
 ROS.gitignore	Added ignore for files created by <code>catkin_make_isolated</code>	6 years ago
 Racket.gitignore	Update Racket.gitignore	2 years ago
 Rails.gitignore	Ignore Rails .env according recommendations	2 years ago
 Raku.gitignore	Changes the name of Perl 6 to Raku (#3312)	3 years ago
 RhodesRhomobile.gitignore	Add Rhodes mobile application framework gitignore	13 years ago
 Ruby.gitignore	Ruby: ignore RuboCop remote inherited config files (#3197)	4 years ago

```

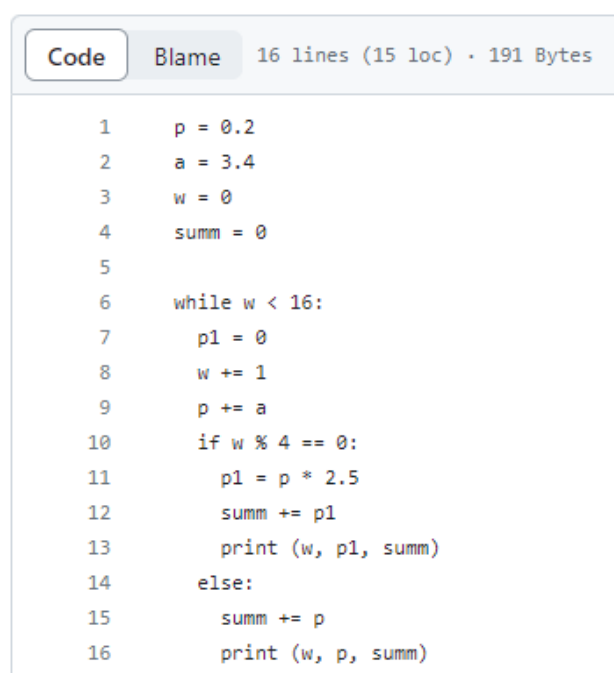
1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
16 .eggs/
17 lib/
18 lib64/
19 parts/
20 sdist/
21 var/
22 wheels/
23 share/python-wheels/
24 *.egg-info/
25 .installed.cfg
26 *.egg
27 MANIFEST
28

```

4. Теперь создаём файл «README.md», где вносим информацию о своей группе и ФИО. Сохраняем набранный текст через кнопку «Commit changes».




5. Создаём новый файл «code», в которую запишем код для программы. Сохраняем изменения с помощью кнопки «Commit changes».




6. Сохраняем отчёт по лабораторной работе в качестве PDF-файла. Заходим в репозиторий на GitHub и нажимаем на кнопку «Add file». Выбираем опцию «Upload files», вставляем отчёт по лабораторной работе и нажимаем на кнопку «Commit changes».

Test /

  
Drag additional files here to add them to your repository  
[Or choose your files](#)

ЛР1\_ГорчаковРВ.pdf x




### Commit changes

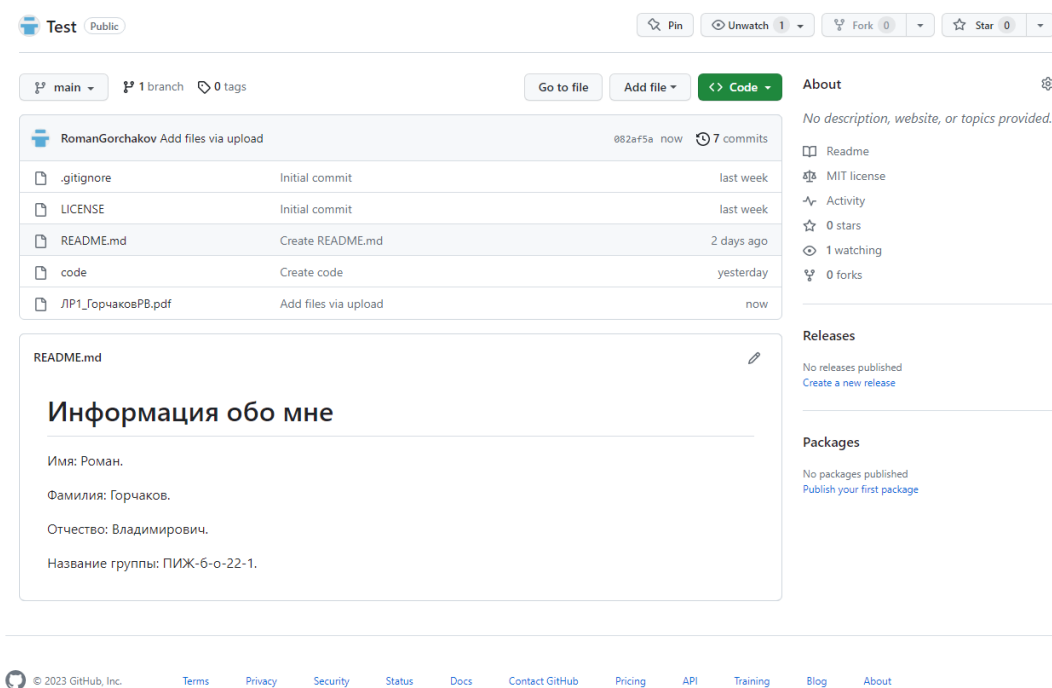
Add files via upload

Add an optional extended description...

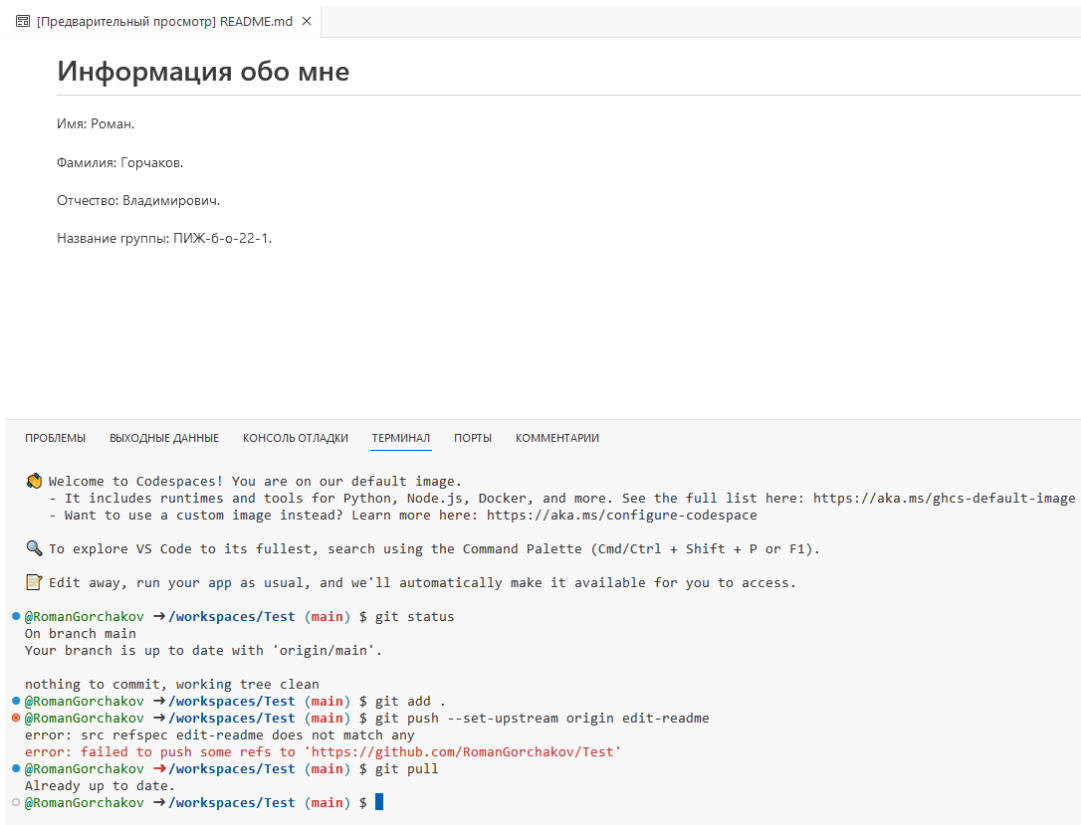
☒ Commit directly to the `main` branch.  
☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes Cancel

 © 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)



7. В окне «Codespace» выбираем опцию «Create codespace on main», где введём команды «git status», «git add .», «git push» и «git pull».



## Контрольные вопросы

### 1. Что такое СКВ и каково её назначение?

Система контроля версий – система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

СКВ даёт возможность возвращать отдельные файлы к прежнему виду, возвращать к прежнему состоянию весь проект, просматривать происходящие со временем изменения, определять, кто последним вносил изменения во внезапно переставший работать модуль, кто и когда внёс в код какую-то ошибку, и многое другое.

### 2. В чем недостатки локальных и централизованных СКВ?

Локальные СКВ: возможность потери данных вследствие возникновения физических поломок оборудования, отсутствие возможности совместной разработки.

Централизованные СКВ: отсутствие доступа к данным при сбое работы сервера, довольно низкая скорость работы из-за возникновения сетевых задержек.

### 3. К какой СКВ относится Git?

Git относится к распределённым систем контроля версий.

### 4. В чем концептуальное отличие Git от других СКВ?

Основное отличие Git от любой другой СКВ – подход к работе со своими данными. Концептуально, большинство других систем представляет хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени. Git же представляет свои данные как поток снимков. Каждый раз при совершении коммита, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок.

### 5. Как обеспечивается целостность хранимых данных в Git?

В Git целостность хранимых данных обеспечивается тем, что обращение к сохранённым объектам в нём происходит по хеш-сумме.

6. В каких состояниях могут находиться файлы в Git? Как связаны эти состояния?

Файлы в Git могут находиться в трёх состояниях: зафиксированном, изменённом и подготовленном.

Зафиксированный файл уже сохранён в локальной базе. К изменённым относят файлы, которые были изменены, но не были зафиксированы. Подготовленные файлы – изменённые файлы, отмеченные для включения в следующий коммит.

7. Что такое профиль пользователя в GitHub?

Профиль – публичная страница пользователя на GitHub.

8. Какие бывают репозитории в GitHub?

Репозитории в GitHub бывают локальными и удалёнными. Локальный репозиторий – подкаталог `.git`, создаётся (в пустом виде) командой `git init` и (в непустом виде с немедленным копированием содержимого родительского удалённого репозитория и простановкой ссылки на родителя) командой `git clone`. Удалённые репозитории представляют собой версии проекта, сохранённые в интернете или ещё где-то в сети.

9. Укажите основные этапы модели работы с GitHub.

Рассмотрим область GitHub. В нем есть два хранилища (`upstream` и `origin`). Чтобы перенести изменения с вашей копии в исходный репозиторий проекта, нужно сделать запрос на извлечение. А чтобы внести небольшие изменения в свою копию (`fork`), необходимо создать локальный клон удаленного репозитория и работать с ним локально, периодически внося изменения в удаленный репозиторий.

10. Как осуществляется первоначальная настройка Git после установки?

Чтобы убедиться, что Git был успешно установлен, нужно ввести в терминале команду `«git version»`, чтобы отобразить текущую версию Git, и, если она сработала, добавить в настройки Git имя, фамилию и адрес электронной почты пользователя через команду `«git config –global»`.

11. Опишите этапы создания репозитория в GitHub.



В правом верхнем углу, рядом с аватаром есть кнопка с плюсиком, нажав на которую происходит переход к созданию нового репозитория. В открывшейся странице после заполнения полей «Имя репозитория», «Описание» и «Public/private» нажать на кнопку «Create repository».

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Academic Free, Apache, Artistic, Boost Software, BSD, Creative Common, Educational Community, Eclipse Public, European Union Public, GNU, ISC, LaTeX Project Public, Microsoft Public, MIT, Mozilla Public, Open Software, PostgreSQL, SIL Open Font, University of Illinois/NCSA Open Source, The Unlicense, zLib.

13. Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий?

После создания репозитория его необходимо клонировать на ваш компьютер. Для этого на странице репозитория необходимо найти кнопку Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования. Открываем командную строку или терминал и перейдите в каталог, куда нужно скопировать хранилище, затем пишем команду `git clone` и вводим адрес.

При клонировании Вы получаете последнюю версию всех файлов и папок репозитория, плюс всю его историю. Таким образом, каждый клон – это полноценный Git-репозиторий.

14. Как проверить состояние локального репозитория Git?

Состояние локального репозитория Git можно проверить с помощью команды «`git status`».

15. Как изменяется состояние локального репозитория Git после выполнения следующих операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/измененного файла под версионный контроль с помощью команды `git add`; фиксации (коммита) изменений с помощью команды `git commit` и отправки изменений на сервер с помощью команды `git push`?

После добавления/изменения файла в локальный репозиторий Git:

On branch main

Your branch is up to date with 'origin/main'.

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file:

После добавления нового/изменённого файла под версионный контроль с помощью команды «git add»:

On branch main

Your branch is up to date with 'origin/main'.

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file:

После фиксации изменений с помощью команды «git commit» и отправки изменений на сервер с помощью команды «git push»:

On branch main

Your branch is up to date with 'origin/main'.

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file:

16. У Вас имеется репозиторий на GitHub и два рабочих компьютера, с помощью которых Вы можете осуществлять работу над некоторым проектом с использованием этого репозитория. Опишите последовательность команд, с помощью которых оба локальных репозитория, связанных с репозиторием GitHub будут находиться в синхронизированном состоянии. Примечание: описание необходимо начать с команды git clone.

17. GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub.

Codebase, SourceForge, SourceHut, Gitea, Bitbucket, GitLab.

18. Интерфейс командной строки является не единственным и далеко не самым удобным способом работы с Git. Какие Вам известны программные средства с графическим интерфейсом пользователя для работы с Git? Приведите как реализуются описанные в лабораторной работе операции Git с помощью одного из таких программных средств.

GitHub Desktop, Fork, Tower, Sourcetree, SmartGit, Sublime Merge, GitKraken, GitUp, Auress Git Client, GitBlade, Git Cola, GitEye, UnGit.