

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 №1**  
**дисциплины «Программирование на python»**

Выполнил:  
Кожуховский Виктор Андреевич  
1 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной  
техники и автоматизированных систем  
», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Роман Александрович

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2023 г.

**Тема:** исследование основных возможностей Git и GitHub

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

**Методика и порядок выполнения работы**

1. Создал общедоступный репозиторий на GitHub.

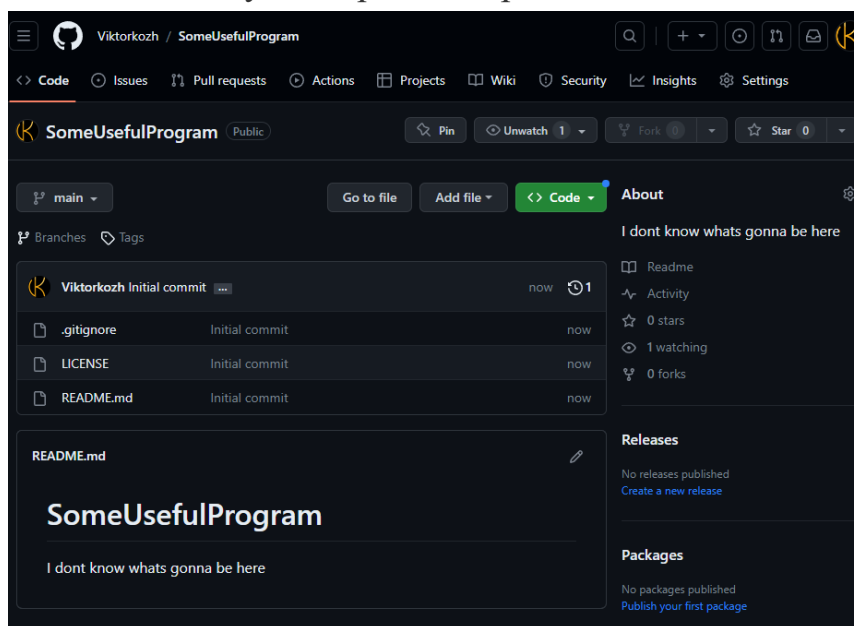


Рисунок 1. Созданный репозиторий

2. Выполнил клонирование созданного репозитория на рабочий компьютер.

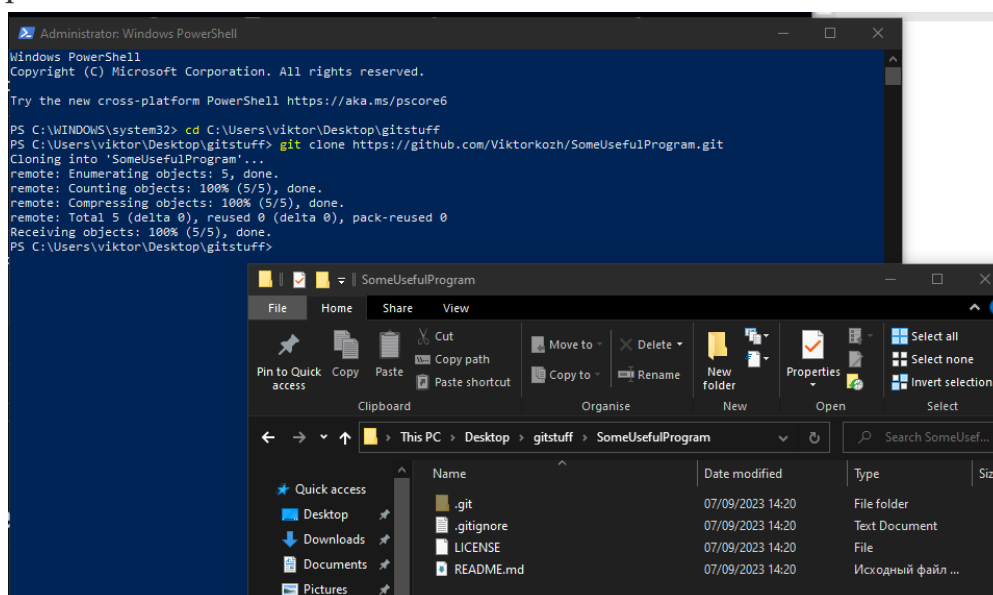
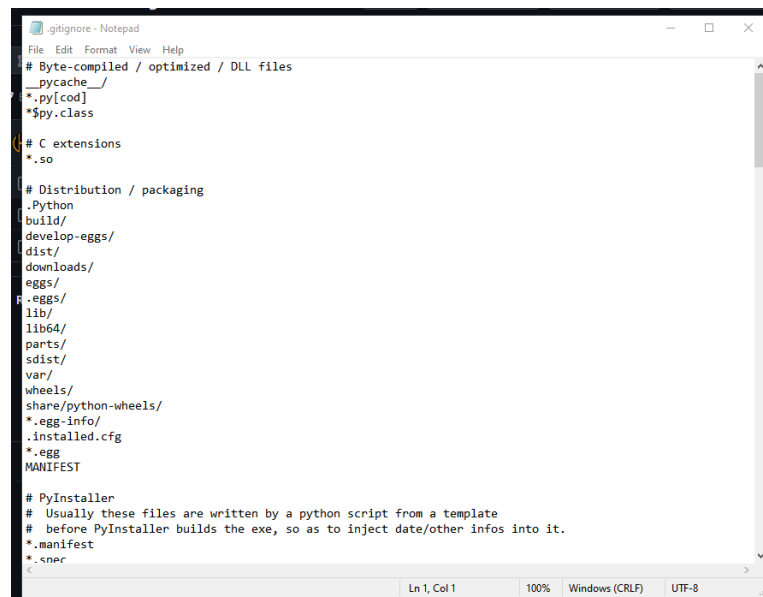


Рисунок 2. Клонирование репозитория

3. Дополнил файл .gitignore необходимыми правилами для выбранного языка программирования и интегрированной среды разработки.



```
.gitignore - Notepad
File Edit Format View Help
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

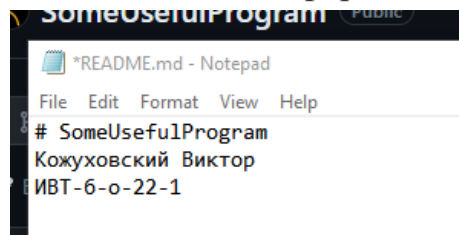
# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
*.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
share/python-wheels/
*.egg-info/
*.installed.cfg
*.egg
MANIFEST

# PyInstaller
# Usually these files are written by a python script from a template
# before PyInstaller builds the exe, so as to inject date/other infos into it.
*.manifest
*.spec
```

Рисунок 3. Информация в .gitignore

4. Добавил в файл README.md информацию о группе и ФИО.



```
*README.md - Notepad
File Edit Format View Help
# SomeUsefulProgram
Кожуховский Виктор
ИБТ-6-о-22-1
```

Рисунок 4. Информация в readme

5. Написал небольшую программу на python. Зафиксировал изменения при написании программы в локальном репозитории. Сделал 7 КОММИТОВ.

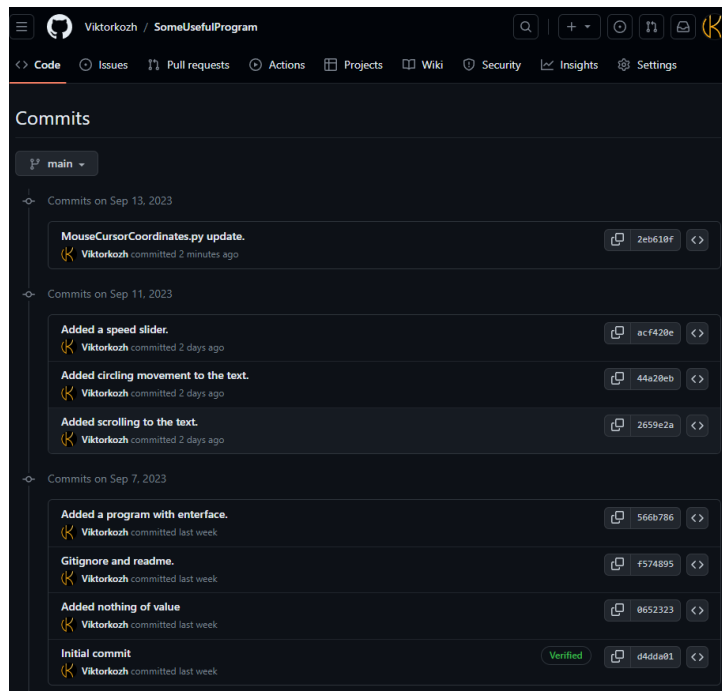


Рисунок 5. Демонстрация коммитов

6. Добавил файл README и зафиксировать сделанные изменения.

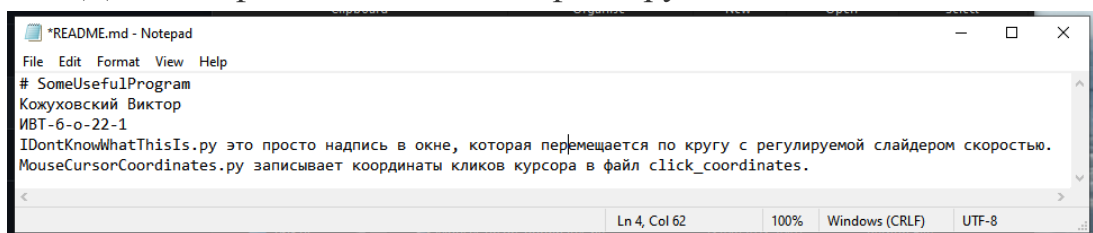


Рисунок 6. Дополненная информация в readme

7. Отправьте изменения в локальном репозитории в удаленный репозиторий GitHub.

```
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\SomeUsefulProgram> git add --all
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\SomeUsefulProgram> git commit
[main 9e28c54] Отчет в pdf.
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 doc/Python lab 1.pdf
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\SomeUsefulProgram> git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 320.73 KiB | 45.82 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Viktorkozh/SomeUsefulProgram.git
 258cbdf..9e28c54  main -> main
PS C:\Users\viktor\Desktop\скфу\python\gitstuff\SomeUsefulProgram>
```

Рисунок 7. Отправка изменений на гит

## Вопросы для защиты работы

1. Что такое СКВ и каково ее назначение?

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

Программисты обычно помещают в систему контроля версий исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа.

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

Локальных: можно легко забыть, в какой директории вы находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели. Невозможность удаленной работы. Когда вся история проекта хранится в одном месте, вы рискуете потерять всё.

Централизованных: единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками. Если жёсткий диск, на котором хранится центральная БД, повреждён, а своевременные бэкапы отсутствуют, вы потеряете всё — всю историю проекта, не считая единичных снимков репозитория, которые сохранились на локальных машинах разработчиков.

## **3. К какой СКВ относится Git?**

К распределенным.

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

Подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы делаете коммит, то есть сохраняете состояние своего проекта в Git, 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 для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом. Данная функциональность встроена в Git на низком уровне и является неотъемлемой частью его философии. Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git.

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

У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged). Они связаны командами git add и git commit.

**7. Что такое профиль пользователя в GitHub?**

Профиль - это ваша публичная страница на GitHub, как и в социальных сетях.

**8. Какие бывают репозитории в GitHub?**

В GitHub есть публичные, частные и форк репозитории.

**9. Укажите основные этапы модели работы с GitHub.**

Основные этапы работы с GitHub: создание/клонирование репозитория, работа с файлами, фиксация изменений, отправка изменений на сервер.

**10. Как осуществляется первоначальная настройка Git после установки?**

Необходимо войти в свой аккаунт при помощи команд:  
git config --global user.name <your\_name>  
git config --global user.email <email>

**11. Опишите этапы создания репозитория в GitHub.**

В правом верхнем углу, рядом с аватаром есть кнопка с плюсиком, нажимая которую мы переходим к созданию нового репозитория.

В результате будет выполнен переход на страницу создания репозитория.

Наиболее важными на ней являются следующие поля:

- **Имя репозитория.** Оно может быть любое, необязательно уникальное во всем github, потому что привязано к вашему аккаунту, но уникальное в рамках тех репозиторий, которые вы создавали.
- **Описание (Description).**
- **Public/private** открытость репозитория.
- **.gitignore** и **LICENSE.**

После заполнения этих полей нажимаем кнопку Create repository.

**12. Какие типы лицензий поддерживаются GitHub при создании репозитория?**

GitHub поддерживает разные типы лицензий, включая MIT, Apache, GPL и другие.

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

Клонирование репозитория с помощью `git clone <URL>`. Клонирование нужно для получения копии удаленного репозитория.

**14. Как проверить состояние локального репозитория Git?**

Состояние локального репозитория Git можно проверить с помощью `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 операций: добавление/изменение файла - файл переходит в состояние "измененных"; `git add` - файл переходит в состояние "подготовленных"; `git commit` - файлы фиксируются в текущей версии; `git push` - изменения отправляются на сервер.

**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`.

`git clone <URL>` (первичное клонирование)

- Внесите изменения в одном локальном репозитории

`git push` (отправка изменений на сервер)

Во втором локальном репозитории выполните `git pull` (получение изменений с сервера)

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

Кроме GitHub, есть GitLab и Bitbucket.

	GitHub	GitLab
Ценообразование	Бесплатно для общедоступных репозиторий; платные планы для частных репозиторий	Бесплатно для общедоступных и частных репозиторий
Функции	Хостинг кода, отслеживание проблем, запросы на включение изменений, вики	Хостинг кода, отслеживание проблем, запросы на включение изменений, вики
Совместная работа	Инструменты совместной работы, проверка кода, управление проектами	Инструменты совместной работы, проверка кода, управление проектами
Сообщество	Большое сообщество, популярность среди проектов с открытым исходным кодом	Растущее сообщество, набирающее популярность
Интеграция	Хорошо интегрируется с другими инструментами, обширная экосистема	Хорошо интегрируется с другими инструментами, растущая экосистема
Безопасность	Сильные функции безопасности, сканирование уязвимостей	Сильные функции безопасности, сканирование уязвимостей
Локальное развертывание	Возможность самостоятельного	Возможность самостоятельного



	развертывания (GitHub Enterprise)	развертывания (GitLab самостоятельное развертывание)
--	-----------------------------------	--

**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.**

Программные средства с графическим интерфейсом для Git: GitHub Desktop, Sourcetree, GitKraken. GitHub desktop позволяет настроить автоматическую синхронизацию, а также благодаря графическому интерфейсу избавляет пользователя от нужды печатанья команд для синхронизации, вместо этого предлагая сделать тоже самое, нажимая на кнопки.