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

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1
дисциплины
«Программирование на Python»

Выполнил:
Поляков Никита Александрович
2 курс, группа ИВТ-б-о-24-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Тема: Исследование основных возможностей Git и GitHub

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

Практическая часть:

Перед началом работы необходимо создать или войти в свою учётную запись GitHub. В данном случае учётная запись уже существует, поэтому был произведён только вход в ранее созданную учётную запись:

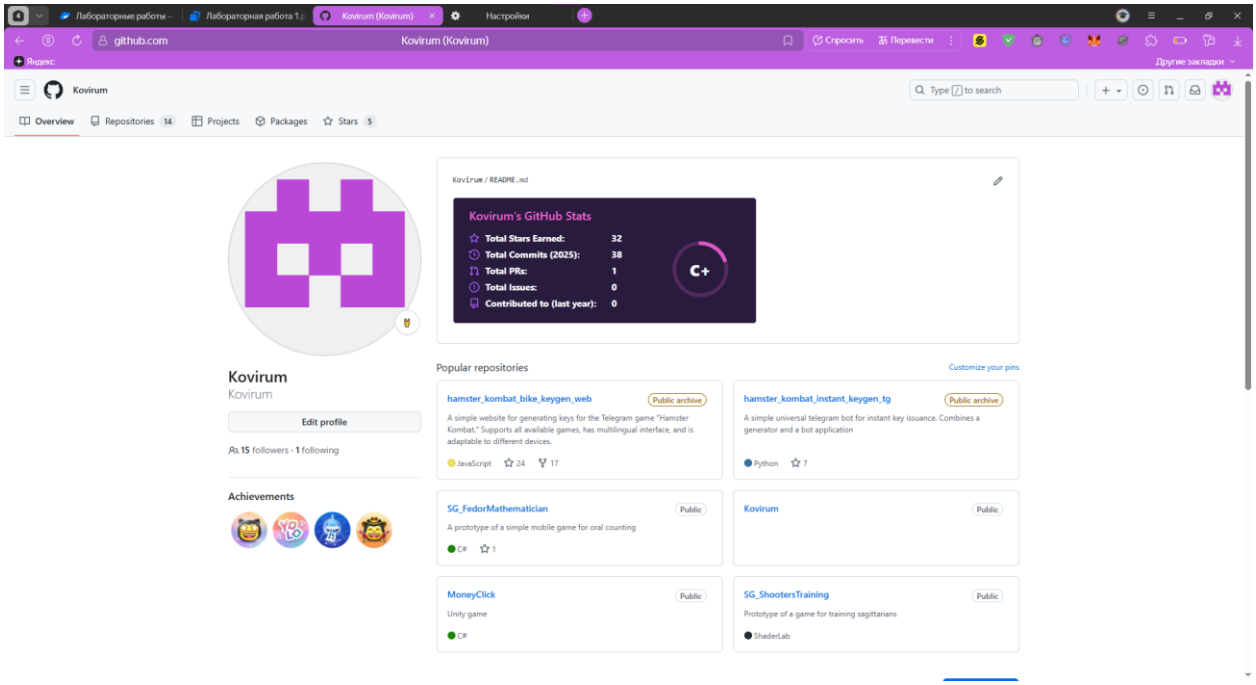

Рисунок 1. Главная страница учётной записи GitHub

Далее был создан новый общедоступный репозиторий с использованием лицензии MIT и выбранным языком программирования:

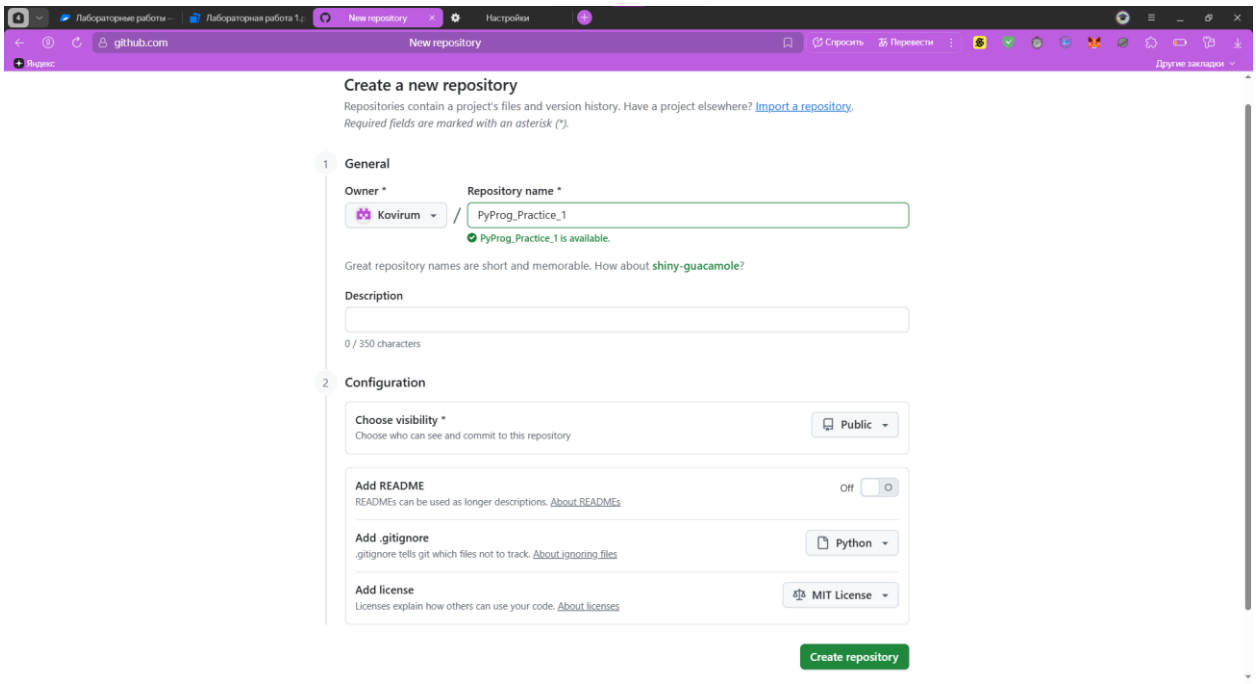

Рисунок 2. Страница создания нового репозитория

Важно правильно установить флажки при создании репозитория. При создании были выбраны следующие:

– Choose visibility: Public – общедоступный репозиторий

– Add .README: off – добавление ридми согласно задания требуется после создания репозитория

– Add .gitignore: Python – добавление .gitignore с настройками для языка Python

– Add license: MIT license – добавление лицензии MIT

Результат создания репозитория:

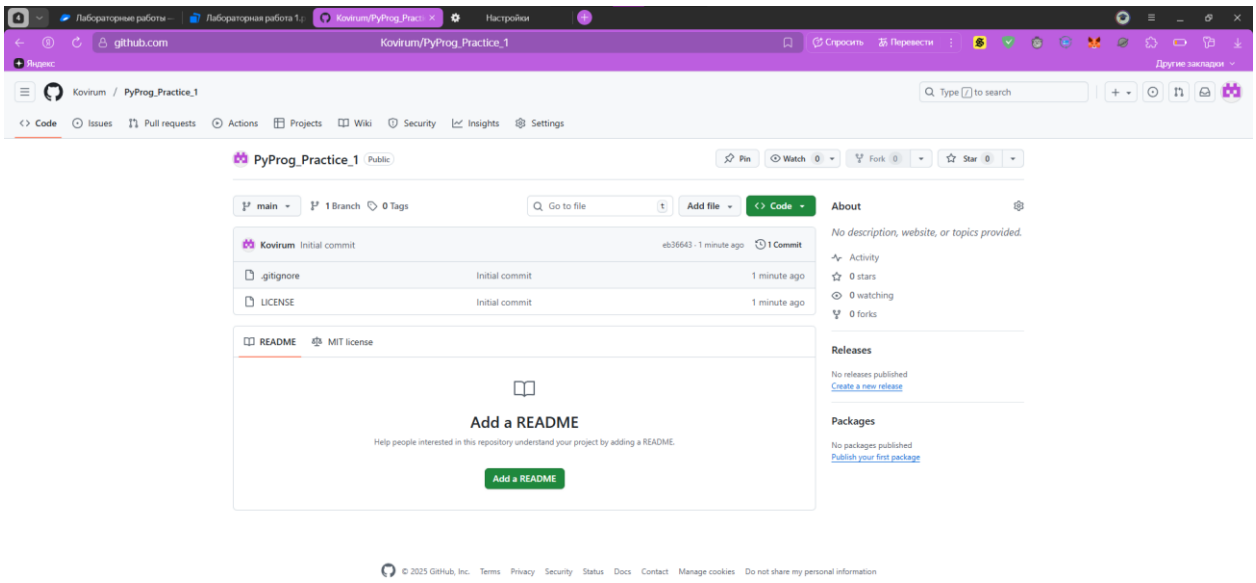

Рисунок 3. Главная страница созданного репозитория

Далее репозиторий был клонирован на рабочий компьютер. Для этого с помощью консоли произведён в переход в отдельную директорию для лабораторной работы, а также выполнена команда «git clone» с ссылкой на репозиторий:

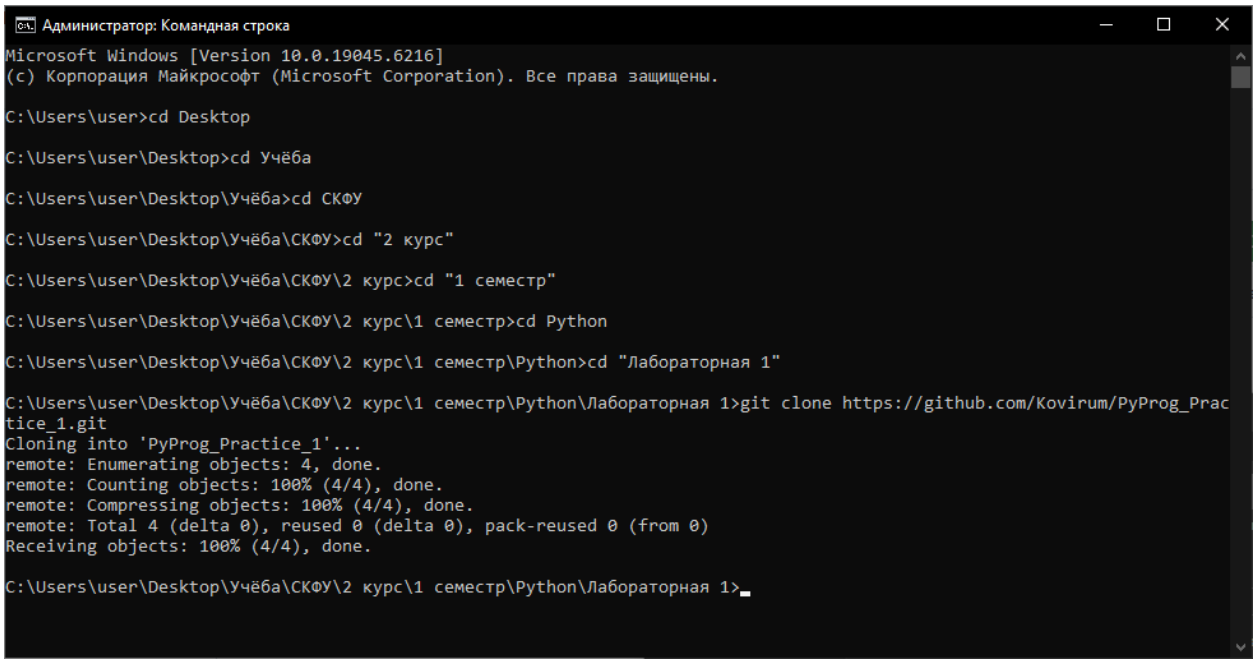

Рисунок 3. Результат выполнения консольной команды клонирования репозитория

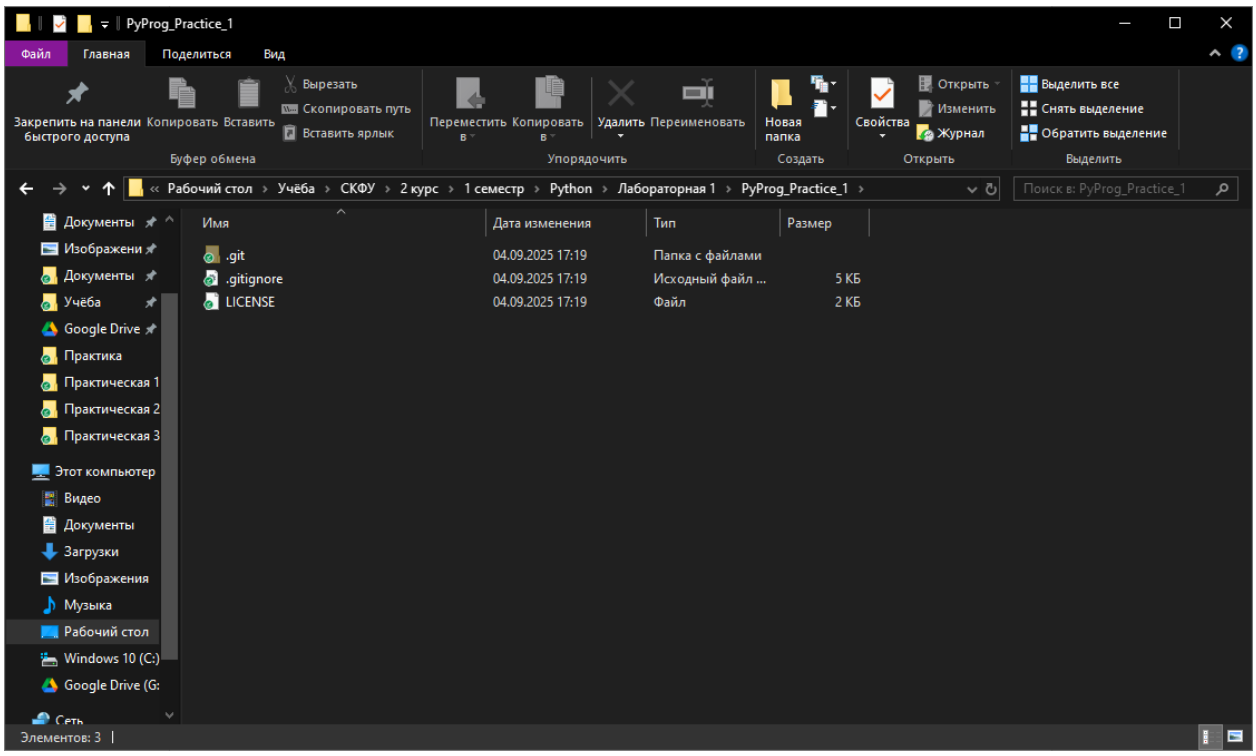

Рисунок 4. Созданная git директория с репозиторием

Далее в данной директории был создан файл «README.md». В данном файле содержится необходимая согласно заданию информация:

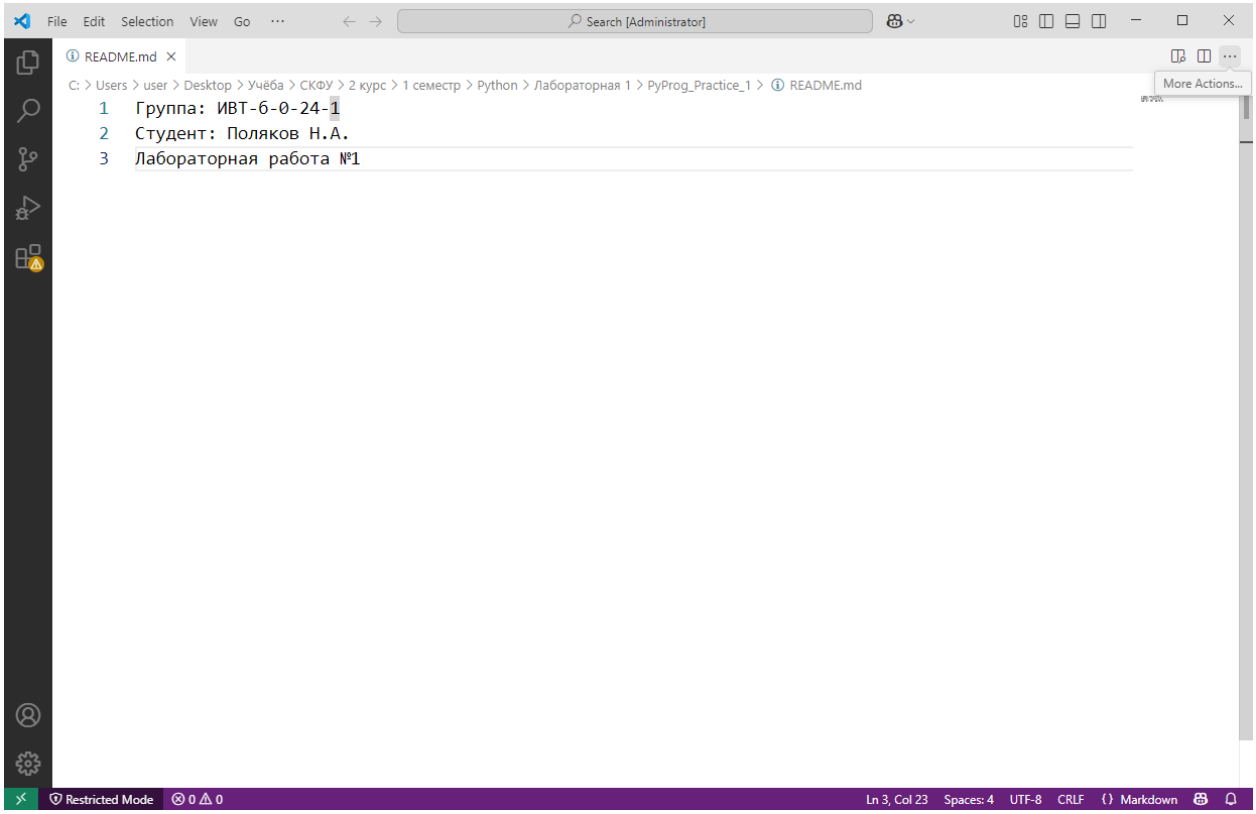

Рисунок 5. Содержимое созданного файла «README.md»

Далее была написана небольшая программа на языке программирования Python. В процессе написания было создано 7 коммитов, отражающих фиксацию изменений при написании программы в локальном репозитории:

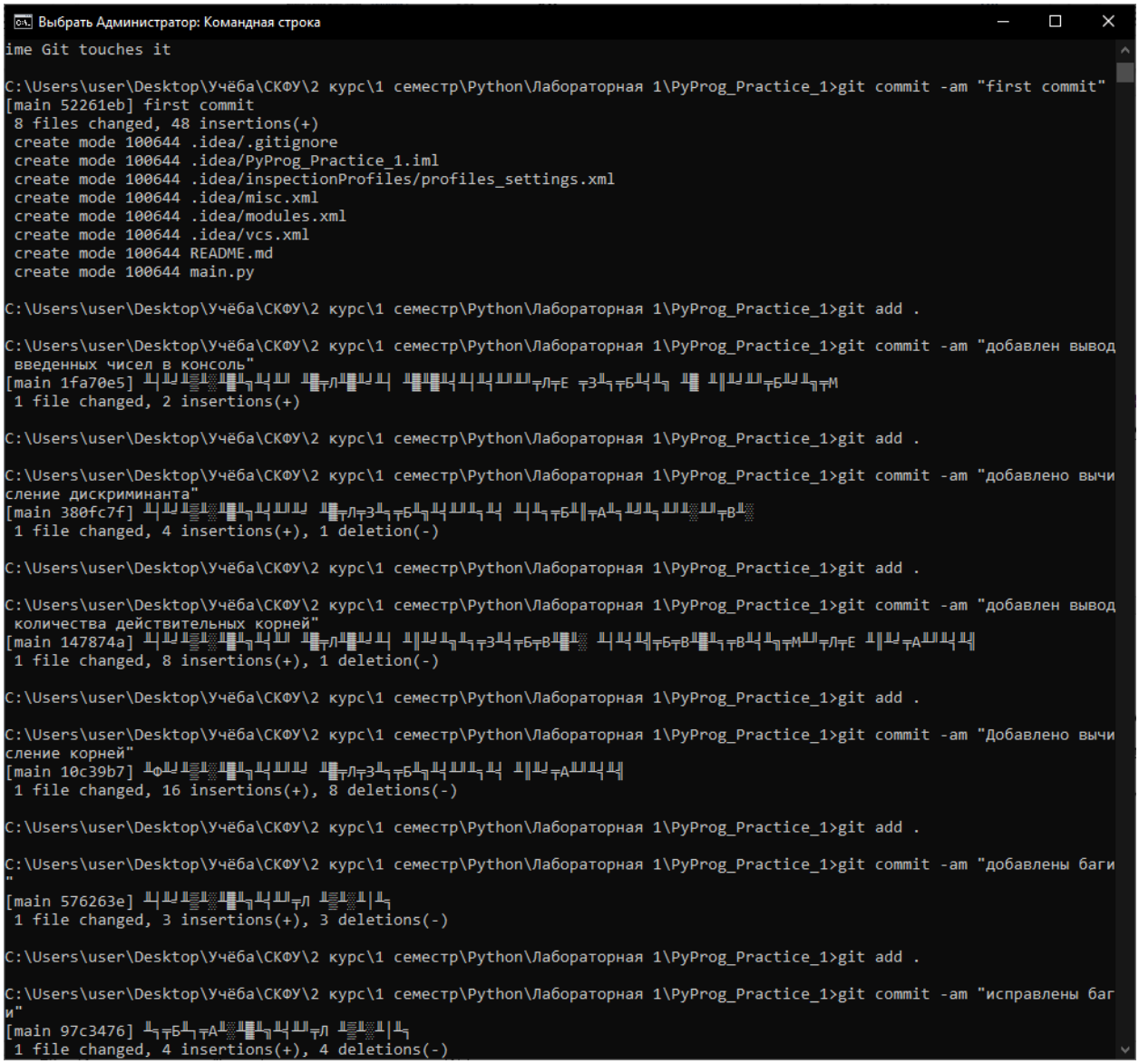

Рисунок 6. Процесс создания коммитов

Далее в директории репозитория была создана новая папка «doc», в которую был помещен отчёт в формате «.pdf»:

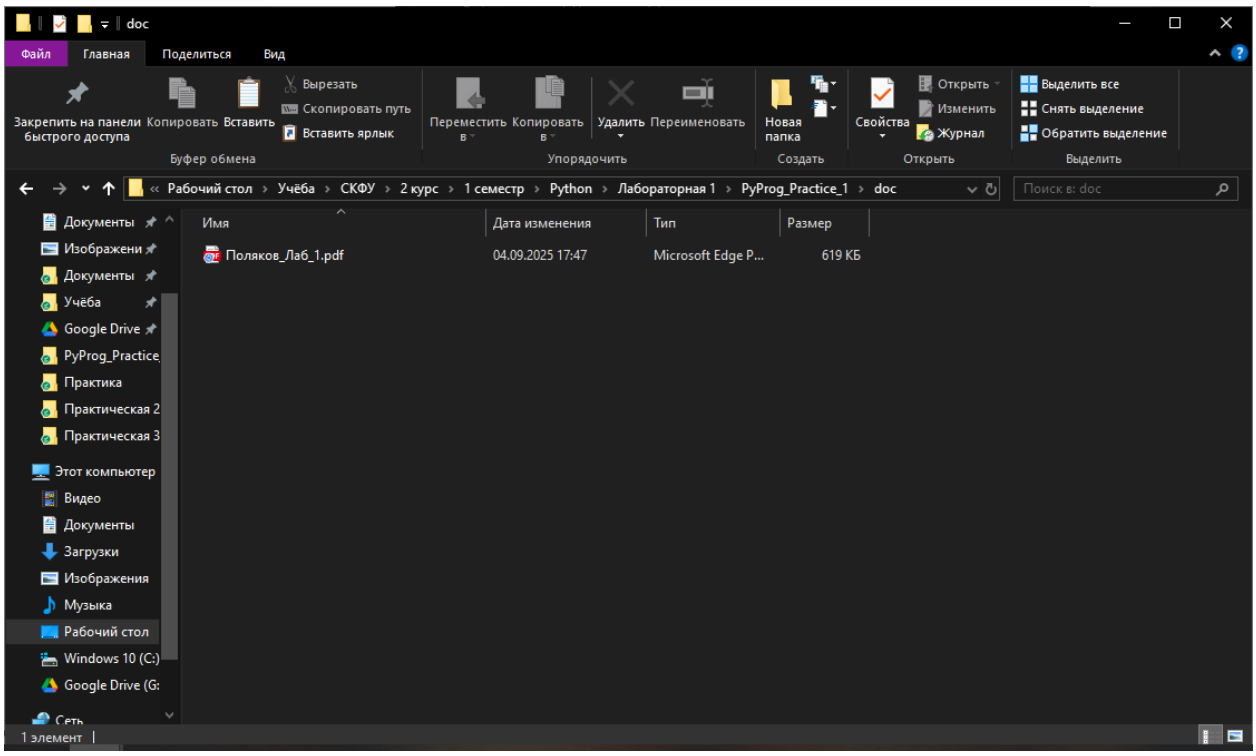

Рисунок 7. Созданная директория «doc» с отчётом внутри

После создания директории и помещения в нее отчета изменения были зафиксированы, затем все локальные изменения были отправлены в удалённый репозиторий GitHub:

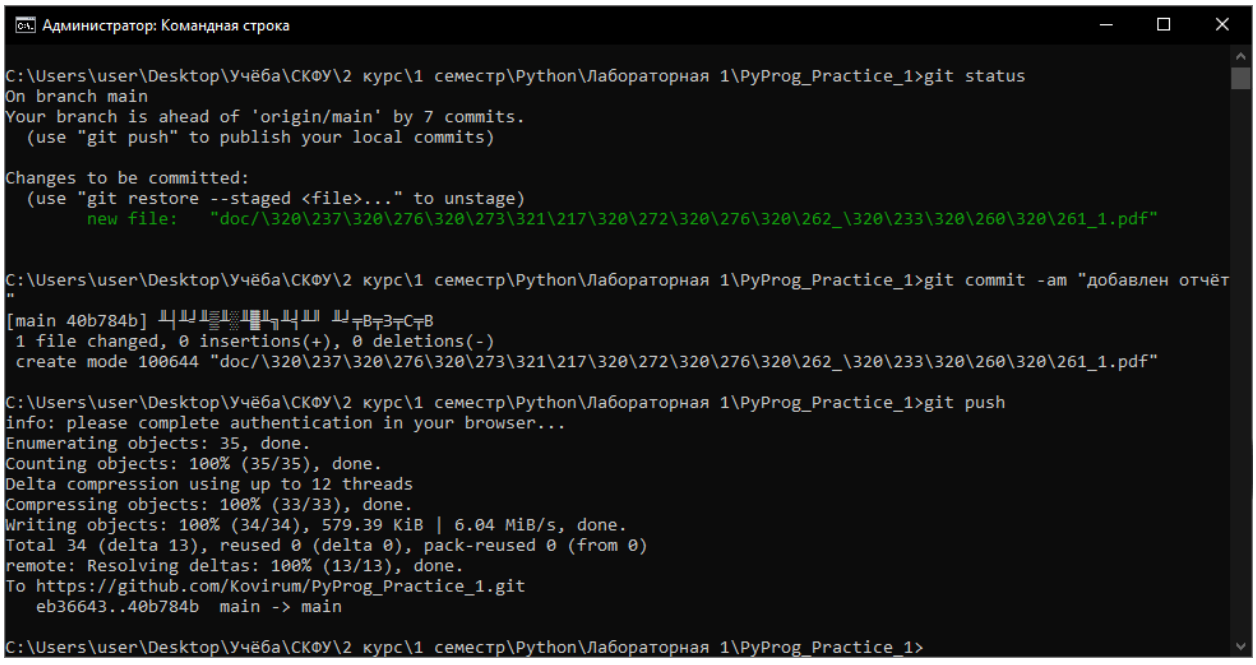

Рисунок 8. Результат отправки локальных изменений на удалённый репозиторий

Результат отправки изменений можно увидеть, зайдя на сайт GitHub на страницу репозитория:

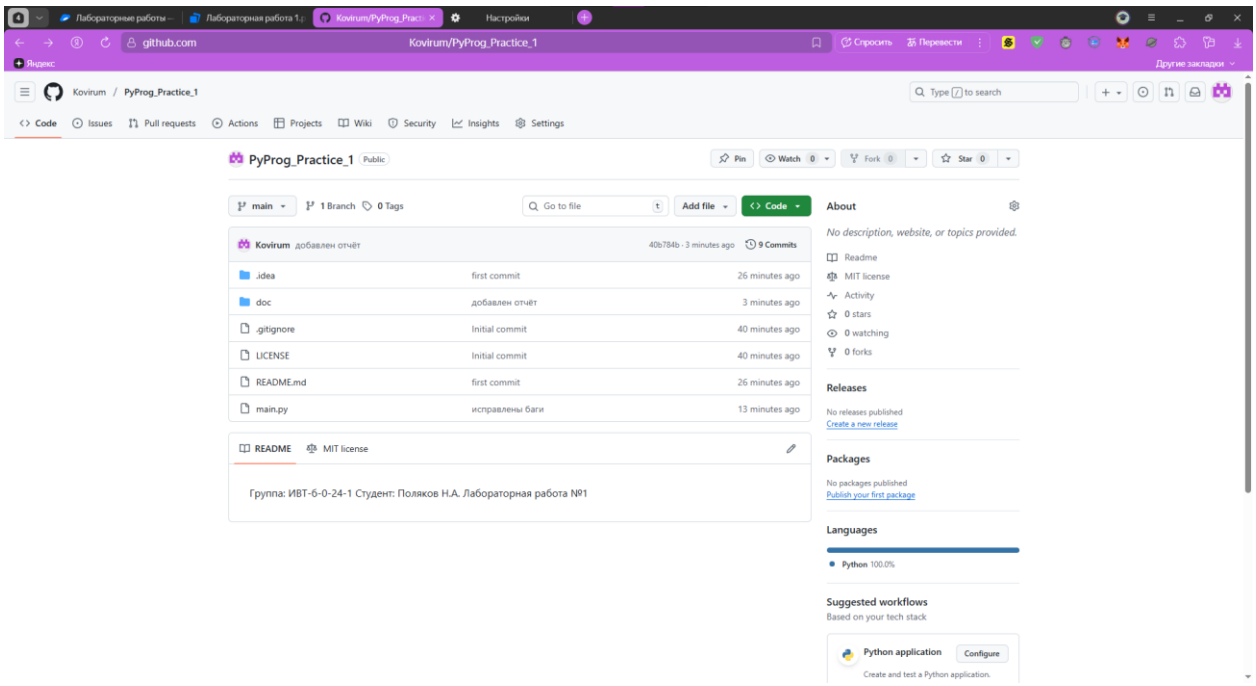

Рисунок 9. Страница репозитория после отправки изменений

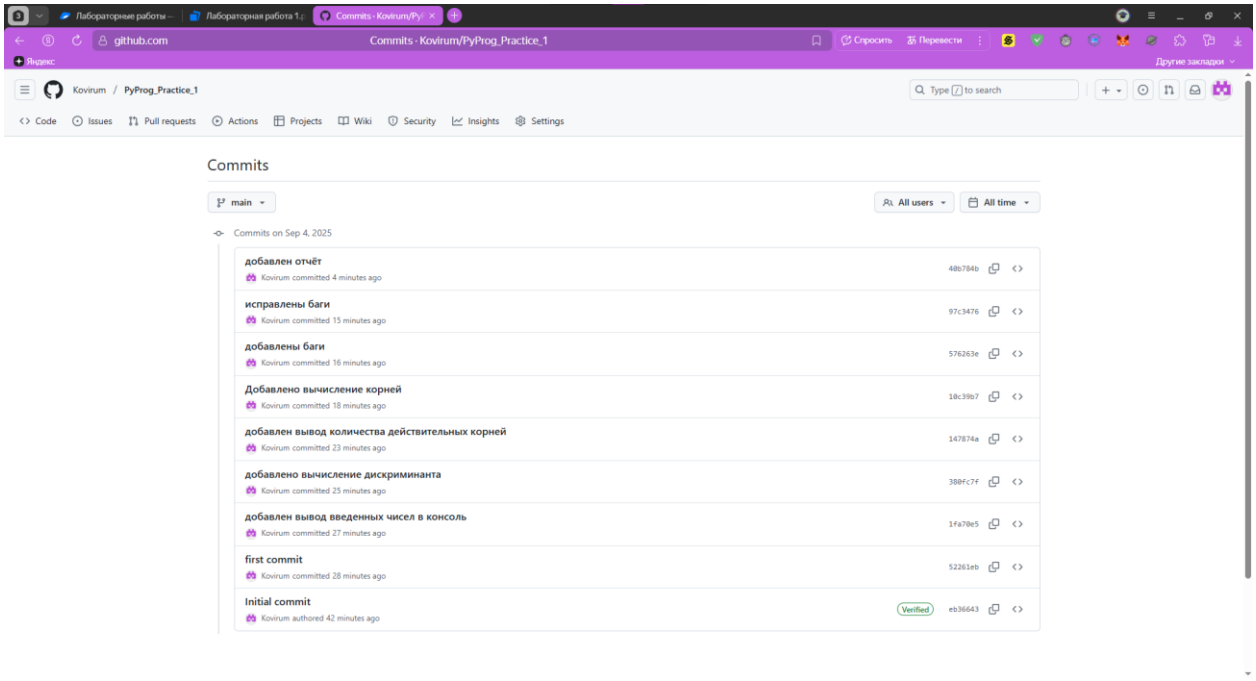

Рисунок 10. История коммитов репозитория

Таким образом, все изменения, произведённые в локальном репозитории, были успешно отправлены на удалённый репозиторий GitHub.

Контрольные вопросы

1. Что такое СКВ и каково ее назначение?

Система контроля версий (СКВ). Назначение: отслеживание изменений в файлах, возможность отката к предыдущим версиям, совместная работа над кодом.

2. В чем недостатки локальных и централизованных СКВ?

Локальные: нет возможности командной работы.
Централизованные: единая точка отказа (сервер). Невозможно работать без сети.

3. К какой СКВ относится git?

Git — это распределенная (децентрализованная) система контроля версий.

4. В чем концептуальное отличие git от других СКВ?

Каждый разработчик имеет полную локальную копию всего репозитория, включая всю его историю. Работа не зависит от центрального сервера.

5. Как обеспечивается целостность хранимых данных в Git?

Через хеши SHA-1. Каждый коммит и файл идентифицируются уникальным хеш-кодом. Любое изменение данных изменит хеш, что легко обнаружится.

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

Три состояния:

Modified (измененный): Файл изменен, но не помечен для коммита.

Staged (подготовленный): Файл добавлен в индекс (git add) и будет частью следующего коммита.

Committed (зафиксированный): Файл сохранен в локальной БД (git commit).

Связь: modified -> (git add) -> staged -> (git commit) -> committed.

7. Что такое профиль пользователя в GitHub?

Это личная страница-визитка пользователя на GitHub, которая содержит репозитории, вклад в другие проекты, настройки и личную информацию.

8. Какие бывают репозитории в GitHub?

Public (Публичные): Видны всем, можно клонировать.

Private (Приватные): Видны и доступны только владельцу и выбранным со-авторам

9. Укажите основные этапы модели работы с GitHub

Клонирование удаленного репозитория (git clone).

Внесение изменений в файлы.

Добавление изменений в индекс (git add).

Фиксация изменений (коммит, git commit).

Получение обновлений с сервера (git pull).

Отправка изменений на сервер (git push).

10. Как осуществляется первоначальная настройка Git после установки?

Задаются имя пользователя и email:

git config --global user.name "Ваше Имя"

git config --global user.email "your.email@example.com"

11. Опишите этапы создания репозитория в GitHub.

Нажать "+" в GitHub -> "New repository".

Ввести имя репозитория.

(Опционально) Добавить описание.

Выбрать тип (Public/Private).

(Опционально) Добавить README, .gitignore, лицензию.

Нажать "Create repository".

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

MIT, Apache 2.0, GPLv3, LGPLv3, AGPLv3, Mozilla Public License 2.0, Unlicense и другие.

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

Как: Команда git clone <URL-репозитория>.

Зачем: Чтобы получить полную локальную копию проекта со всей историей для начала работы.

14. Как проверить состояние локального репозитория GitHub

Командой git status. Она покажет текущую ветку, изменения, готовые к коммиту, и неизменённые файлы.

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?

изменение файла -> файл переходит в состояние modified.

git add -> файл переходит в состояние staged.

git commit -> изменения фиксируются, файлы переходят в состояние committed.

git push -> зафиксированные изменения отправляются на удаленный сервер (GitHub).

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 находится в синхронизированном состоянии. Примечание: описание необходимо начать с команды git clone.

На обоих компьютерах: выполнить git clone <URL-репозитория>.

На Компьютере 1:

– Внести изменения.

– git add .

– git commit -m "Сообщение"

– git push (отправляем изменения на GitHub)

На Компьютере 2:

– Перед началом работы всегда выполнять git pull (получаем изменения с GitHub, сделанные на Компьютере 1).

– Теперь можно работать. После изменений: git add ., git commit -m "...", git push.

На Компьютере 1:

– Перед следующей работой снова выполнить git pull для получения изменений с Компьютера 2.

– Главное правило: перед любым git push всегда делать git pull, чтобы забрать чужие изменения и избежать конфликтов.

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

Известные сервисы-конкуренты GitHub:

– GitLab
– Bitbucket
– Azure DevOps Repos (от Microsoft)
– SourceForge (более старый, в основном для open-source)

Сравнительный анализ GitHub и GitLab представлен ниже:

Таблица 1. Сравнительный анализ GitHub и GitLab

Критерий	GitHub	GitLab
Основная бизнес-модель	Публичные репозитории и комьюнити.	Полный жизненный цикл DevOps (CI/CD, мониторинг и т.д.) "из коробки".
Бесплатные приватные репозитории	Есть (без ограничения по collaborators).	Есть (также без ограничения).
Встроенные CI/CD	GitHub Actions: Настраивается через YAML-файлы в репозитории.	GitLab CI/CD: Более зрелое и глубоко интегрированное решение. Также настраивается через .gitlab-ci.yml.
Установка	В основном облачный SaaS-сервис. Есть корпоративная версия для собственного сервера (GitHub Enterprise).	Сильное преимущество: Существует полнофункциональная Community Edition с открытым исходным кодом, которую можно бесплатно установить на свой собственный сервер (self-hosted).

Интерфейс и удобство	Очень чистый, популярный и привычный для миллионов разработчиков интерфейс.	Интерфейс более "перегруженный" из-за огромного количества встроенных функций (CI/CD, Issues, Wiki, Kubernetes integration и пр.).
Фокус	Социальное кодирование, открытый исходный код, collaboration.	Полная DevOps-платформа "все в одном" для компаний, желающих развернуть у себя весь процесс.

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.

Известные GUI-клиенты для Git:

– GitHub Desktop (кроссплатформенный, простой)

– GitKraken (мощный, кроссплатформенный)

– Sourcetree (от Atlassian, бесплатный)

– Fork (современный, растущий в популярности)

– Встроенные в IDE: JetBrains (IntelliJ IDEA, PyCharm), Visual Studio Code.

Реализация операций через GitHub Desktop:

Клонирование репозитория (git clone):

– Меню File -> Clone repository.

– Выбрать репозиторий из списка на GitHub или вставить URL.

– Нажать "Clone".

Проверка состояния репозитория (git status):

– Состояние отображается визуально в главном окне:

– Список измененных файлов (с иконками: M - modified, ? - untracked).

– Список файлов в индексе (Staged Changes).

– Текущая ветка.

Добавление файлов в индекс (git add):

– В списке "Changes" напротив каждого файла есть галочка.

– Чтобы добавить файл в коммит (проставить на stage), нужно поставить галочку рядом с ним. Чтобы добавить все, можно поставить галочку в заголовке списка.

Создание коммита (git commit):

– После того как файлы добавлены (поставлены галочки), они перемещаются в список "Staged Changes".

– В поле внизу вводится сообщение коммита (Commit message).

– Нажимается кнопка "Commit to main" (или к другой ветке).

Отправка изменений на сервер (git push):

– После создания коммита в интерфейсе появляется кнопка "Push origin" (или кнопка с цифрой, показывающая количество неотправленных коммитов).

– Нажатие на эту кнопку выполняет команду git push.

Получение изменений с сервера (git pull):

– Кнопка "Fetch origin" проверяет, есть ли новые изменения на сервере.

– Если изменения есть, кнопка меняется на "Pull origin". Нажатие на нее выполняет git pull.

Вывод

В результате выполнения данной лабораторной работы были исследованы базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub