Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Перспективной инженерии

Департамент цифровых и робототехнических систем и электроники

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1**

**дисциплины «Программирование на Python»**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил: Мендеш Пашкоал Педру  1 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 Роман 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 ІТ-проектов GitHub.

**Этапы выполнения лабораторных работ**

#### ****1. Создание репозитория на GitHub****

* Перешел на github.com, создал публичный репозиторий с именем Lab-git-Mendes-Pascoal.
* Выбрал лицензию **MIT** и добавил .gitignore для **Python**.

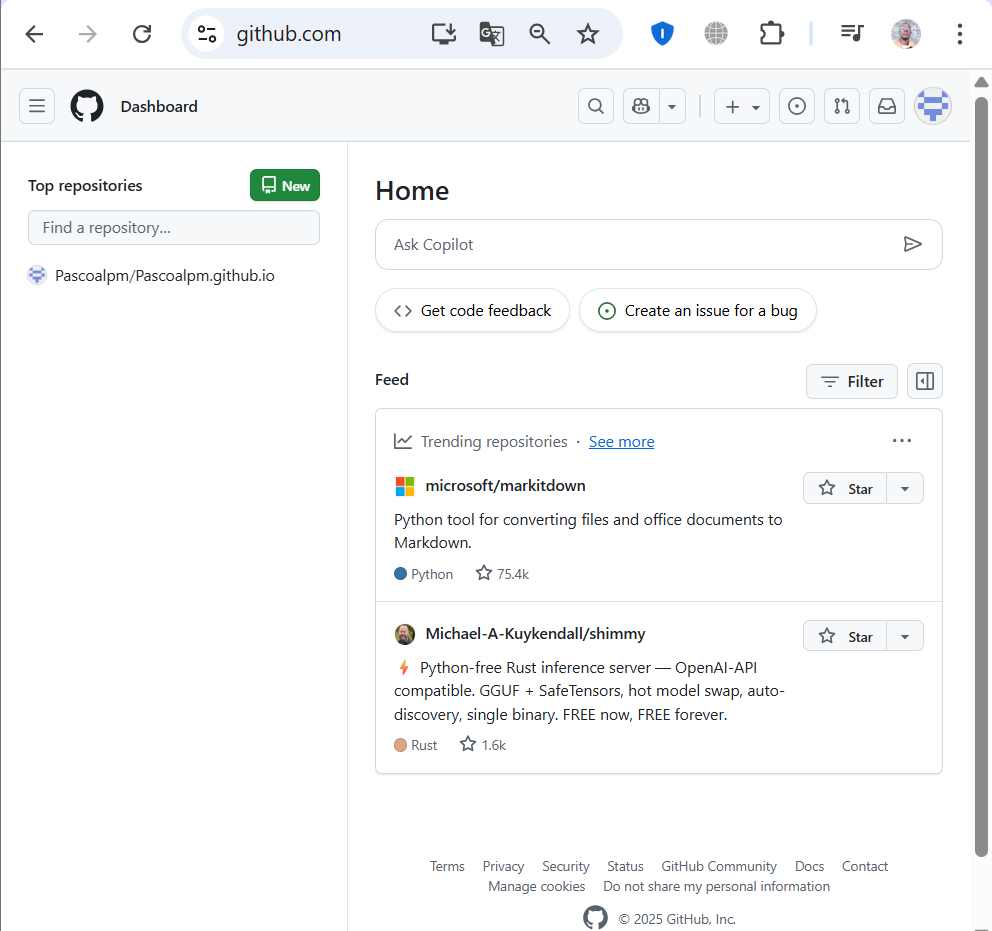


Рисунок 1: Домашняя страница GitHub

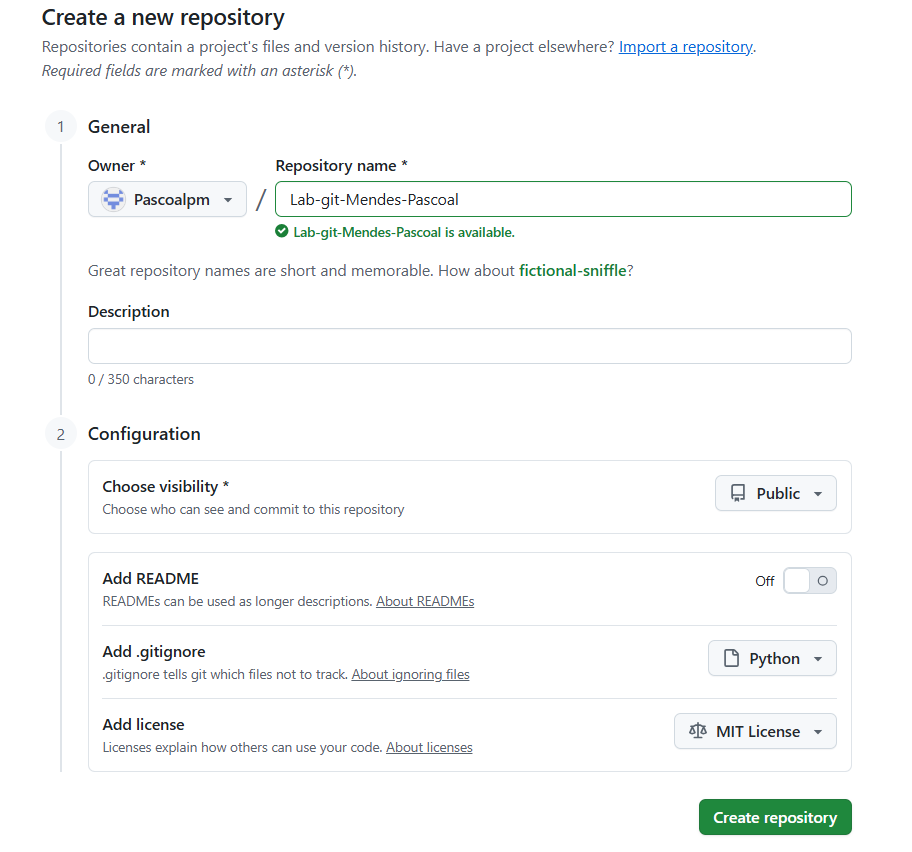


Рисунок 2: Создание репозитория на GitHub

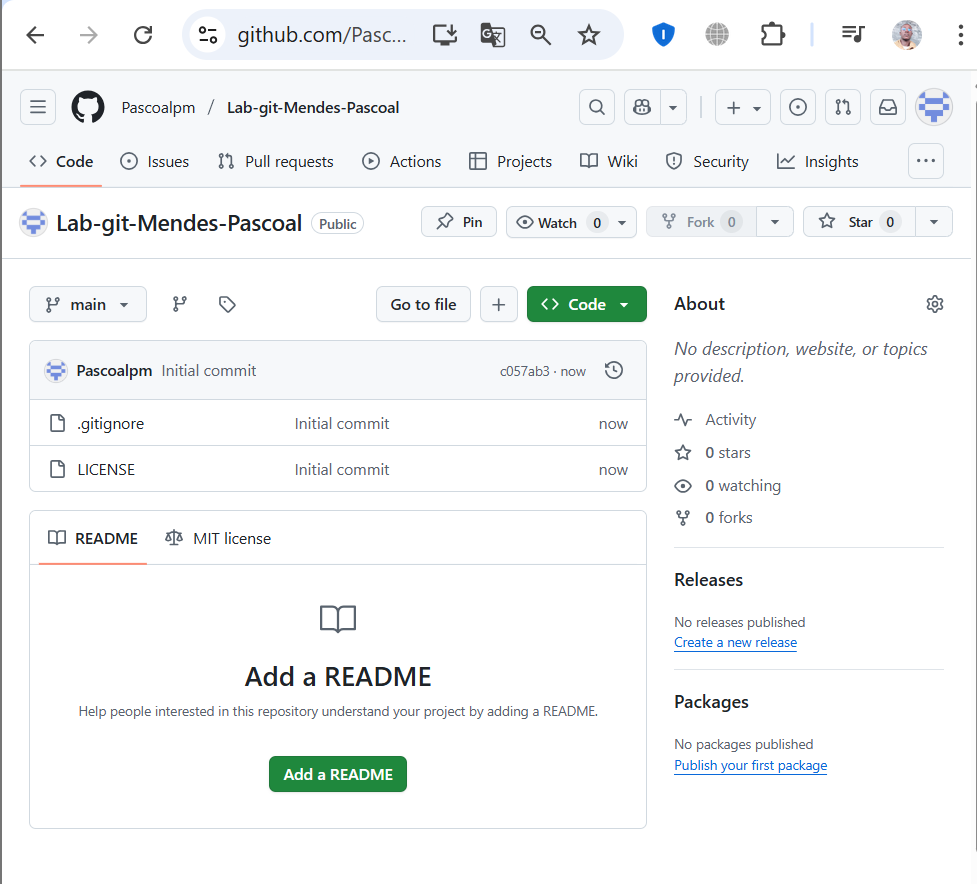


Рисунок 3: Созданный репозиторий (Lab-git-Mendes-Pascoal)

#### ****2. Клонирование репозитория на локальный компьютер****

Использовал команду git clone для загрузки репозитория: <https://github.com/Pascoalpm/Lab-git-Mendes-Pascoal.git>

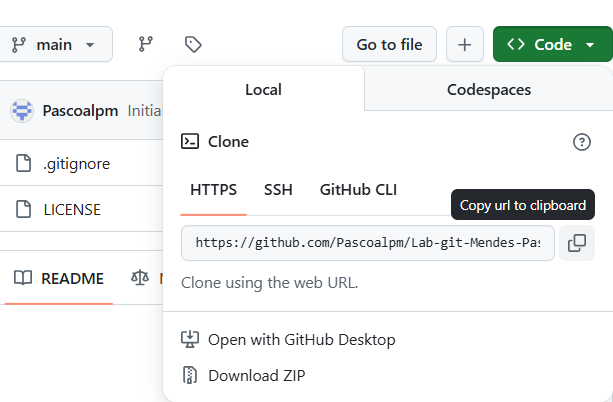


Рисунок 4. Копирование URL-адреса репозитория

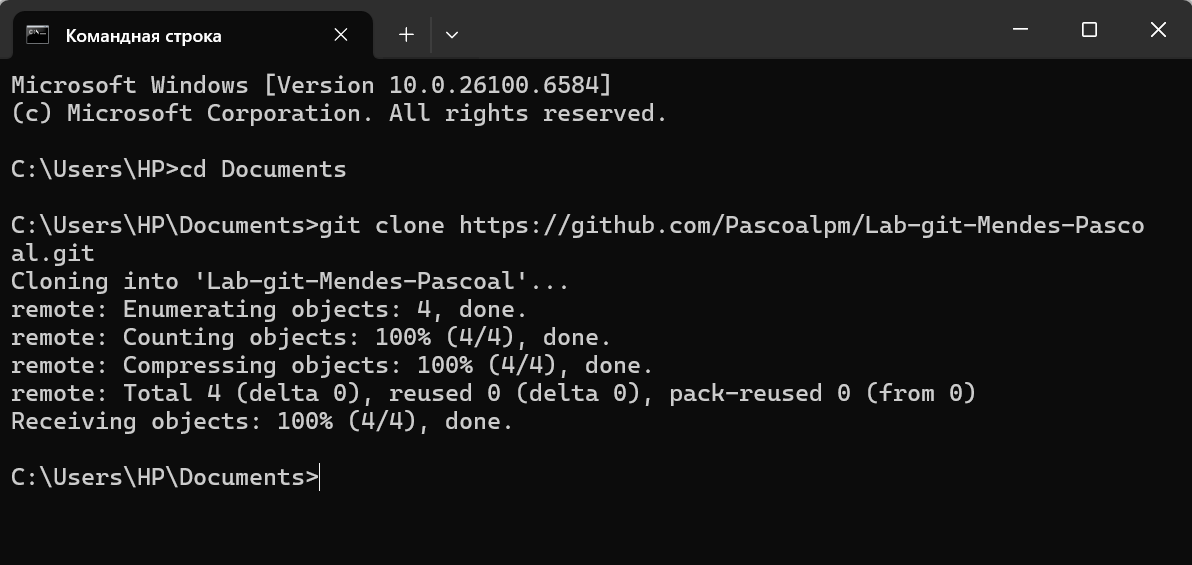


Рисунок 5. Клонирование репозитория в cmd

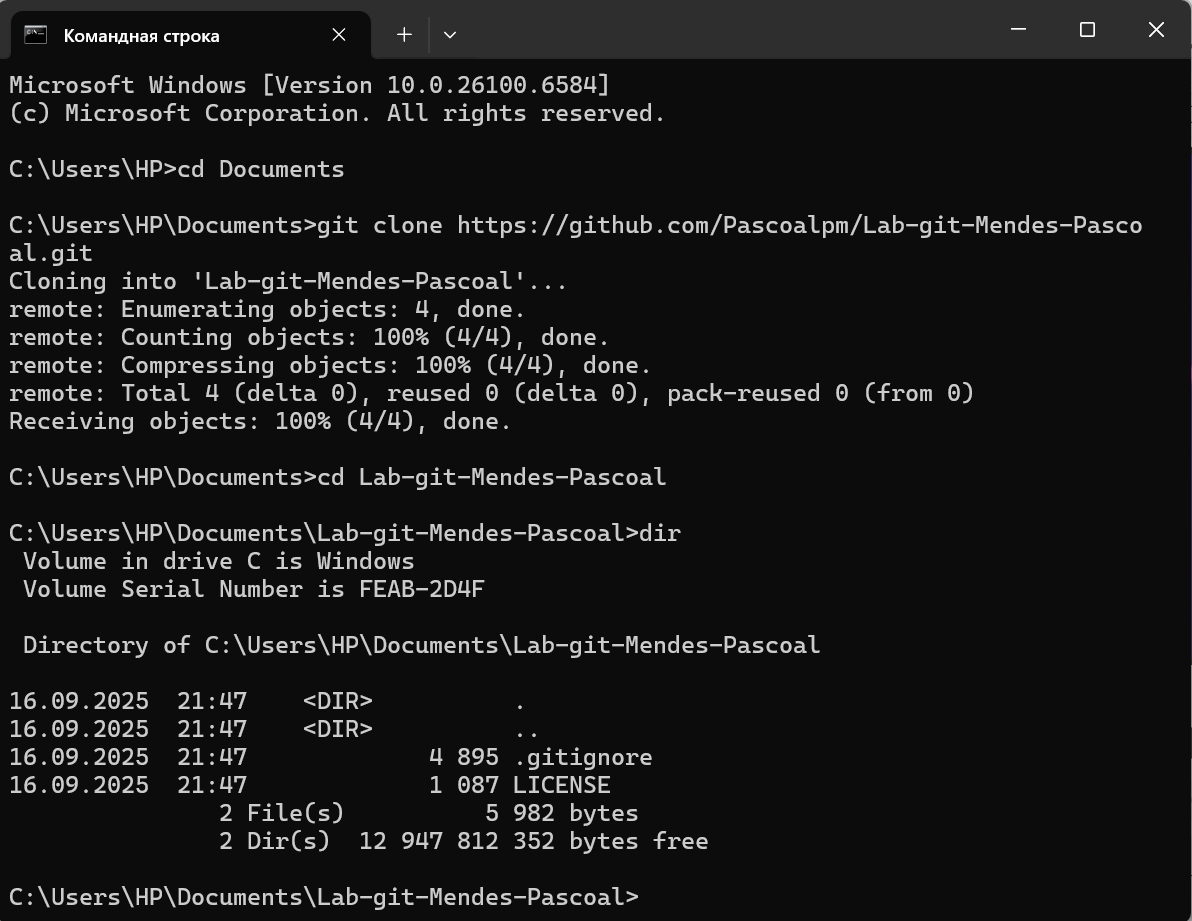


Рисунок 6. Сообщение об успешном клонировании в cmd

#### ****3. Настройка идентификации в Git****

* Настроил глобальное имя и email в Git:

git config --global user.name "Mendes Pascoal"

git config --global user.email [pascoalp.mendes@gmail.com](mailto:pascoalp.mendes@gmail.com)

#### ****4. Редактирование файла .gitignore****

* Добавил правила для IDE (PyCharm или VSCode) в файл .gitignore.
* Команды:

git add .gitignore

git commit -m "Add IDE rules to .gitignore"

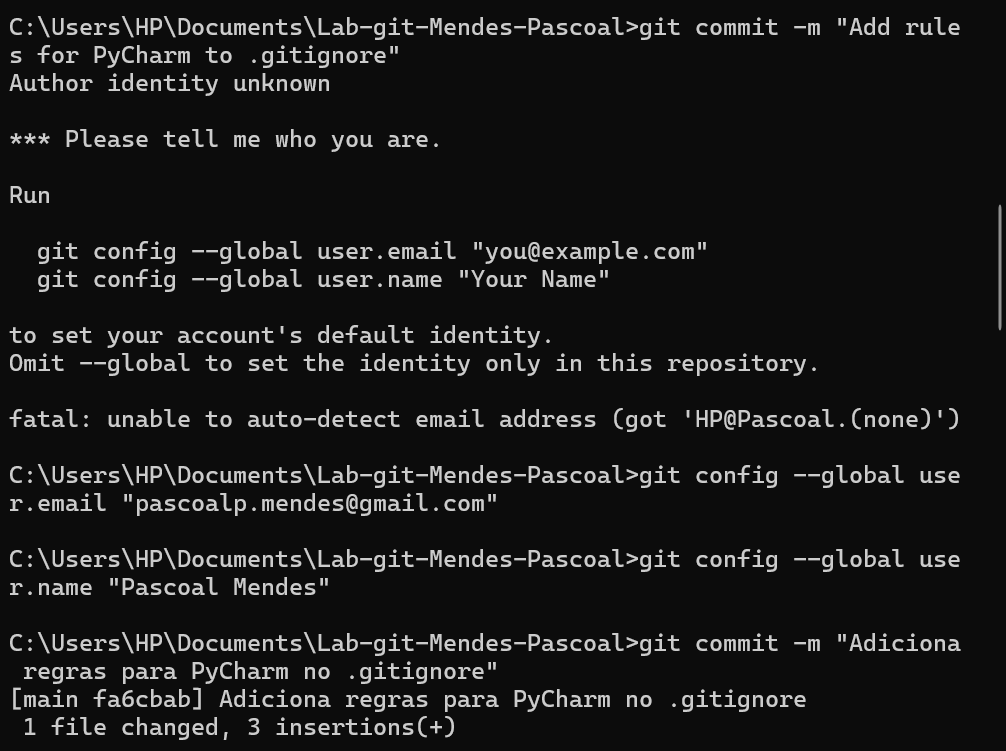


Рисунок 7. Создание идентификации в git и правилах

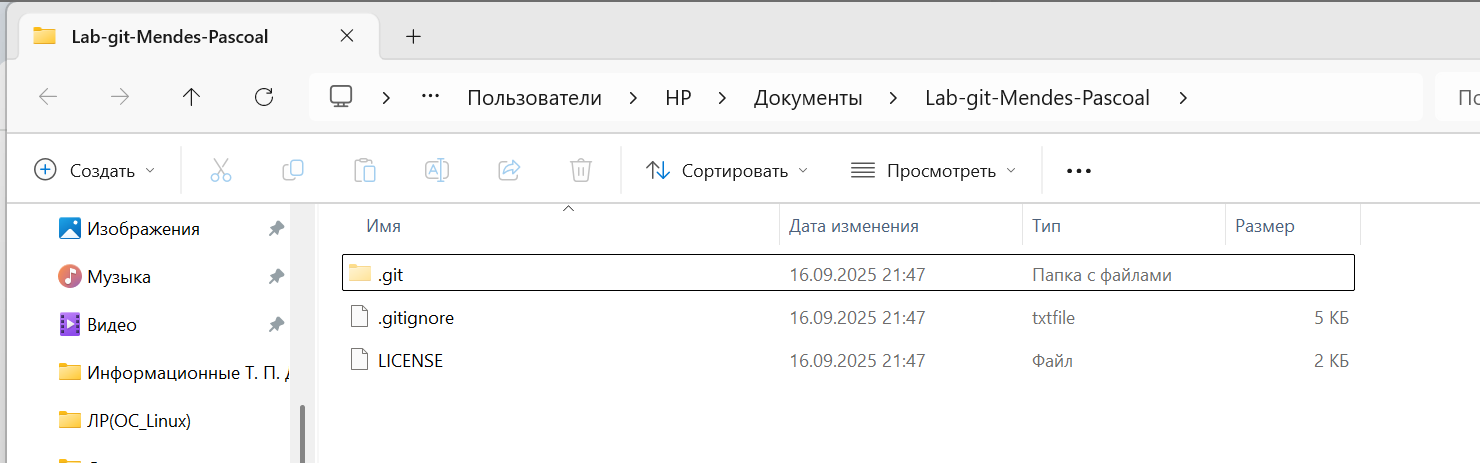


Рисунок 8. Создание папки Lab-git-Mendes-Pascoal в файлах с помощью команды cmd

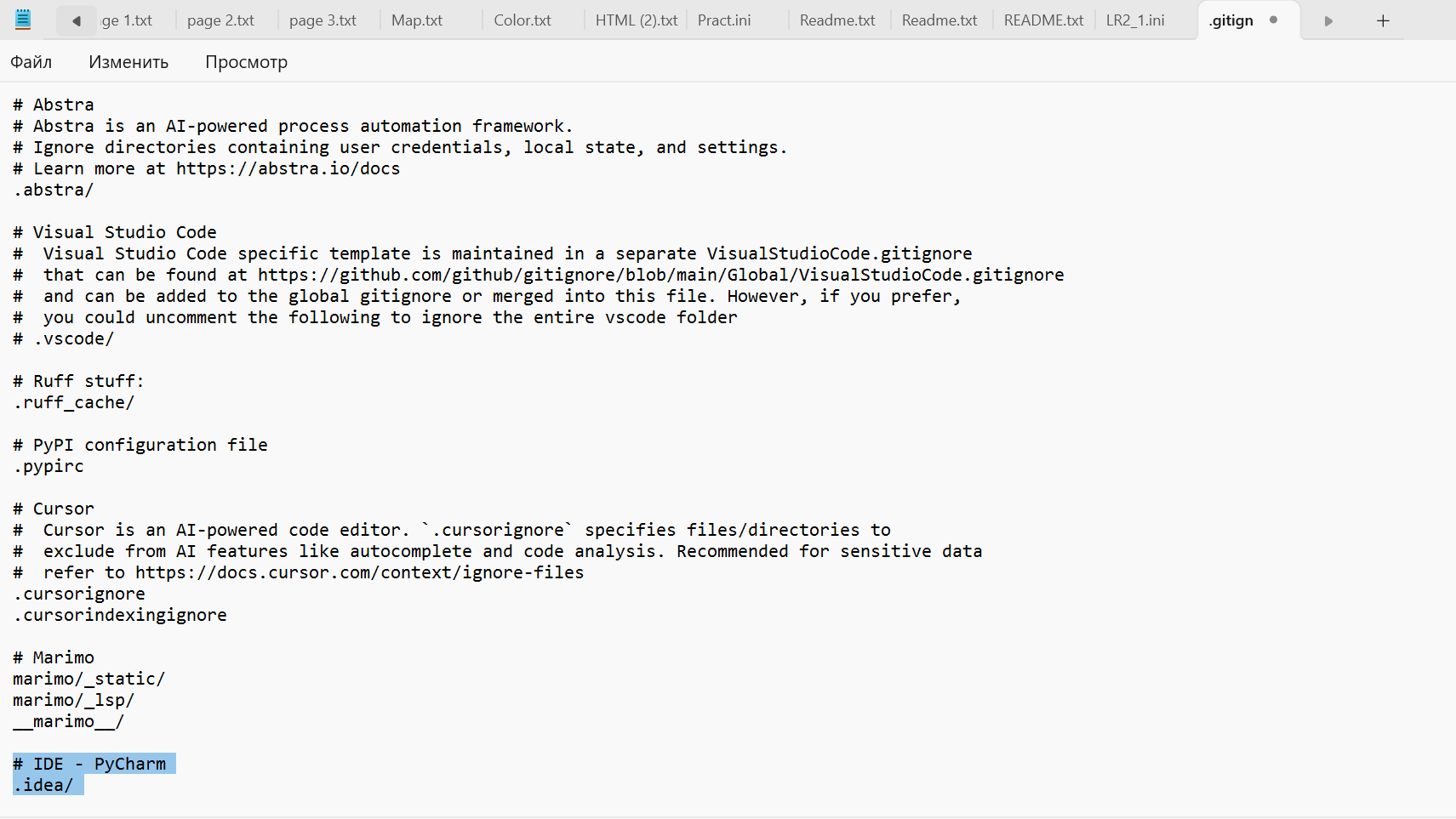


Рисунок 9. Изменение кода Python .gitignore в блокноте

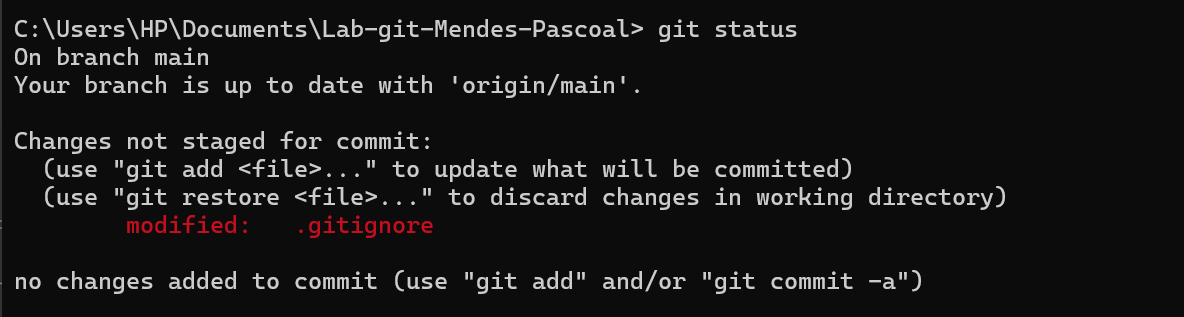


Рисунок 10. Подтверждение изменения в .gitignore

#### ****5. Создание и редактирование файла README.md****

* Создал файл README.md с личной информацией (имя, группа, курс).
* Команды:

notepad README.md

git add README.md

git commit -m "Add README with student info"

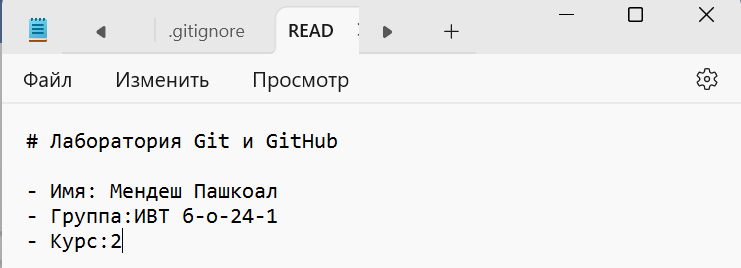


Рисунок 11. Редактирование README в Блокноте

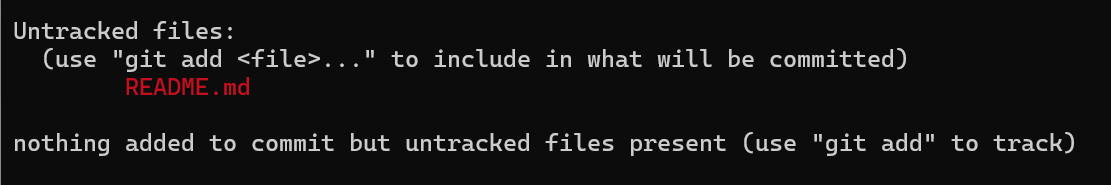


Рисунок 12. Подтверждение изменения README

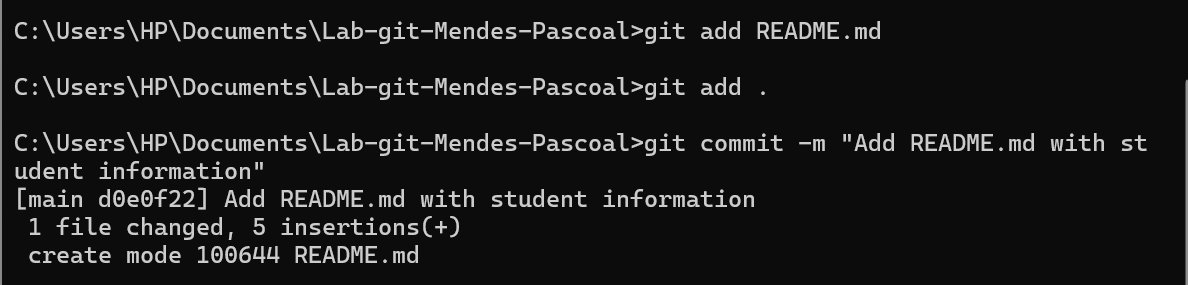


Рисунок 13. Добавление изменения README к данным

#### ****6. Разработка программы на Python с несколькими коммитами****

* Создал файл programa.py и выполнил **7 коммитов** с постепенным добавлением функционала:
  + Git Commit -m 1: Создание программы с "Hello World".
  + Git Commit -m 2: Добавление переменной.
  + Git Commit -m 3: Создание функции.
  + Git Commit -m 4: Ввод данных от пользователя.
  + Git Commit -m 5: Добавление цикла.
  + Git Commit -m 6: Добавление условия.
  + Git Commit -m 7: Финальное сообщение.

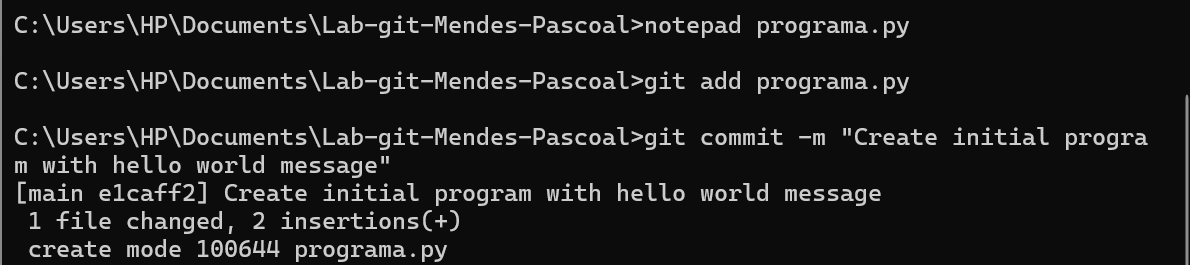


Рисунок 14. Git Commit -m 1: Создание программы с "Hello World".

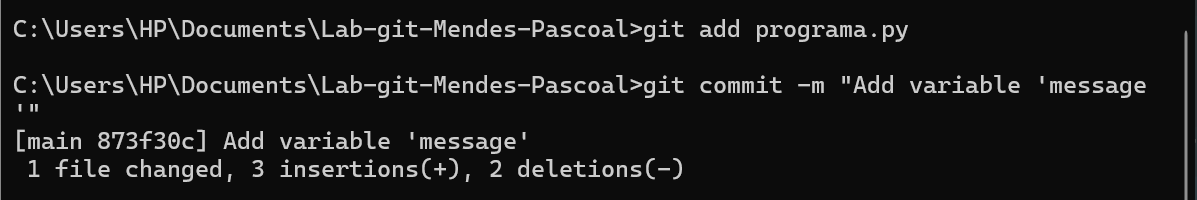


Рисунок 15. Git Commit -m 2: Добавление переменной.

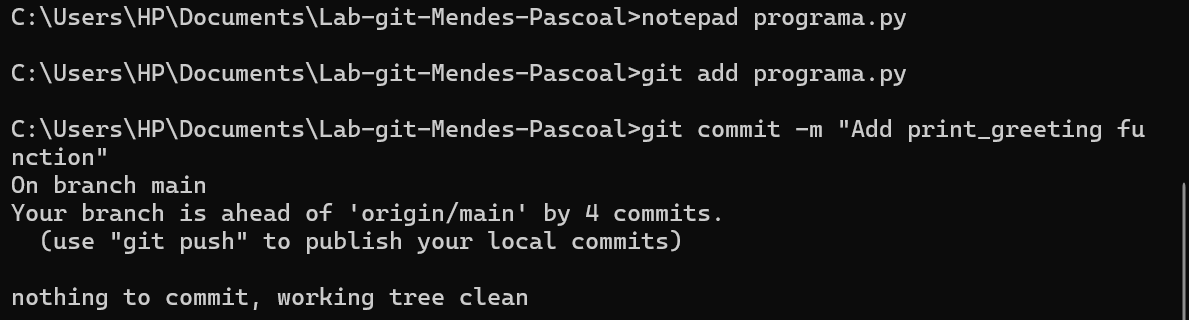


Рисунок 16. Git Commit -m 3: Создание функции.

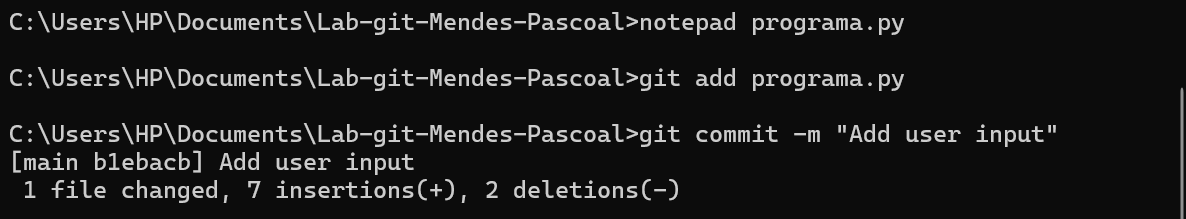


Рисунок 17. Git Commit -m 4: Ввод данных от пользователя.

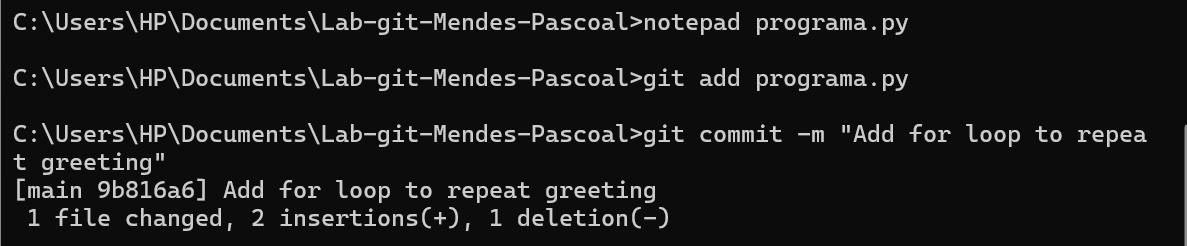


Рисунок 18. Git Commit -m 5: Добавление цикла.

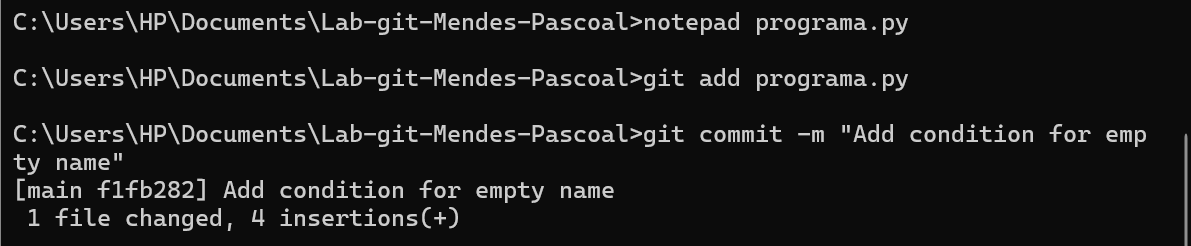


Рисунок 19. Git Commit -m 6: Добавление условия.

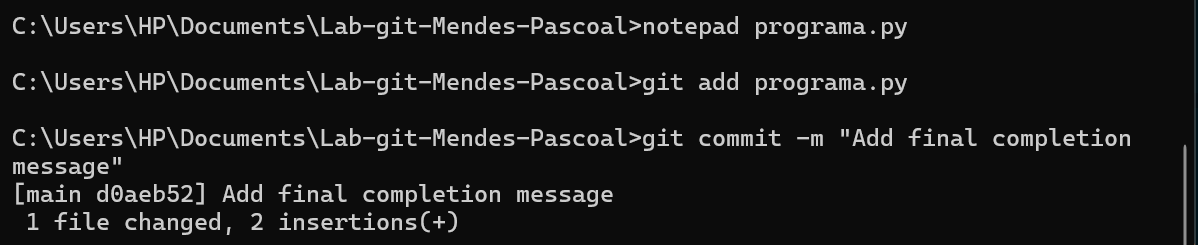


Рисунок 20. Git Commit -m 7: Финальное сообщение.

#### ****7. Создание папки****doc****и добавление отчета****

* Создал папку doc и добавил отчет в PDF:

mkdir doc

git add doc/

git commit -m "Add lab report PDF"

#### ****8. Отправка изменений на GitHub (Push)****

* Использовал git push для отправки всех коммитов:

#### ****9. Проверка на GitHub****

* Убедился, что все файлы и коммиты находятся в удаленном репозитории.

### **1. Что такое СКВ и каково ее назначение?**

**СКВ** (Система Контроля Версий) — это система, регистрирующая изменения в файлах over времени, позволяя возвращаться к более ранним их версиям.  
**Назначение:**

* Хранение истории изменений файлов.
* Возможность отката к предыдущим версиям.
* Совместная работа нескольких людей над одними файлами без потери данных.
* Отслеживание авторства изменений.

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

**Локальные СКВ (например, RCS):**

* Нет возможности collaborate с другими разработчиками.
* Риск потери всей истории при повреждении локального хранилища.

**Централизованные СКВ (например, Subversion, CVS):**

* Единая точка отказа — при недоступности сервера работа невозможна.
* Риск полной потери истории проекта при повреждении сервера без бэкапов.

### **3. К какой СКВ относится Git?**

Git относится к **распределённым СКВ** (РСКВ).

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

В отличие от систем, хранящих данные как последовательность изменений (дельты), Git хранит данные как **набор снимков (snapshots)** файловой системы при каждом коммите.

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

Git использует **хеш-суммы SHA-1** для идентификации всех объектов (коммитов, файлов и т.д.). Любое изменение содержимого приводит к изменению хеша, что гарантирует обнаружение нарушений целостности.

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

Файлы в Git могут находиться в трёх состояниях:

1. **Изменённый (modified)** — файл изменён, но не помещён в staging area.
2. **Подготовленный (staged)** — изменённый файл помечен для включения в следующий коммит.
3. **Зафиксированный (committed)** — файл сохранён в локальной базе данных Git.

**Связь:**  
modified → git add → staged → git commit → committed

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

Профиль пользователя — это его **публичная страница** на GitHub (например, github.com/username), где отображаются его репозитории, вклад в проекты, активность и личная информация. Используется как портфолио разработчика.

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

* **Публичные (public)** — видимы всем, любой пользователь может просматривать и клонировать их.
* **Приватные (private)** — доступны только владельцу и выбранным collaborators.

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

1. **Fork** репозитория (создание копии в своём аккаунте).
2. **Clone** репозитория на локальную машину (git clone).
3. Создание **ветки (branch)** для новой функциональности 1. (git checkout -b new-feature).
4. Внесение изменений и их **коммит** (git add, git commit).
5. **Push** изменений в свой репозиторий на GitHub (git push origin new-feature).
6. Создание **Pull Request** для предложения изменений в оригинальный репозиторий.
7. **Синхронизация** с оригинальным репозиторием череp gitt pull.

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

Настройка имени и email пользователя:

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

git config --global user.email "ваш@email.com"

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

1. Нажать кнопку **"+"** в верхнем правом углу и выбрать **"New repository"**.
2. Ввести **имя репозитория**.
3. Выбрать **видимость** (public или private).
4. Добавить **.gitignore** (опционально, выбрать язык).
5. Выбрать **лицензию** (например, MIT).
6. Нажать **"Create repository"**.

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

GitHub поддерживает множество лицензий, включая:

* **MIT License**
* **GNU General Public License (GPL)**
* **Apache License 2.0**
* **BSD License**

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

**Клонирование:**

bash

git clone https://github.com/username/repository.git

**Зачем нужно:** Чтобы получить полную локальную копию репозитория, включая всю историю изменений, для работы без постоянного доступа к интернету и последующей синхронизации с удалённым репозиторием.

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

Команда:

“git status”

Показывает:

* Изменённые файлы.
* Файлы в staging area.
* Файлы, не отслеживаемые Git.

### **15. Как изменяется состояние локального репозитория Git после выполнения следующих операций:**

* **Изменение файла:** Файл переходит в состояние **modified**.
* **git add:** Файл перемещается в **staged**.
* **git commit:** Файл становится **committed** (сохранён в локальной БД).
* **git push:** Изменения отправляются в удалённый репозиторий (например, GitHub).

### **16. У Вас имеется репозиторий на GitHub и два рабочих компьютера. Опишите последовательность команд...**

**На Компьютере 1:**

git clone https://github.com/user/repo.git

cd repo

*# Внести изменения*

git add .

git commit -m "Commit from Computer 1"

git push origin main

**На Компьютере 2:**

bash

git clone https://github.com/user/repo.git

cd repo

*# Для синхронизации после изменений с Компьютера 1:*

git pull origin main

### **17. GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны?**

**Известные сервисы, работающие с Git:**

* GitLab
* Bitbucket
* Azure DevOps

**Сравнительный анализ GitLab и GitHub:**

**GitLab:**

* **CI/CD:** Имеет встроенный мощный CI/CD, который не требует дополнительных настроек.
* **СамоХостирование:** Предлагает бесплатную возможность self-hosting (установки на собственный сервер).
* **Лицензия:** Есть бесплатная версия с широкими возможностями.
* **Фокус:** Более ориентирован на DevOps и полный жизненный цикл проекта.

**GitHub:**

* **CI/CD:** Использует GitHub Actions (требует настройки).
* **СамоХостирование:** Нет бесплатного варианта self-hosting.
* **Лицензия:** Бесплатный план с ограничениями для private-репозиториев.
* **Фокус:** Более ориентирован на open-source и сообщество.
* **Популярность:** Наиболее популярная платформа с огромным сообществом.

**Общее:** Оба сервиса предоставляют возможность hosting Git-репозиториев, Issue Tracking, Code Review и Collaboration Tools.

**Основное отличие:** GitLab предлагает более комплексное решение «всё в одном» для DevOps, в то время как GitHub сильнее в сообществе и интеграциях с сторонними сервисами.

**18. Какие Вам известны программные средства с графическим интерфейсом пользователя для работы с Git?**

Популярные GUI-инструменты:

* **GitKraken**
* **Sourcetree**
* **GitHub Desktop**
* **Fork**

**Реализация операций в GitHub Desktop:**

* **Клонирование:** File → Clone repository → выбор репозитория.
* **Коммит:** Вкладка Changes → выбор файлов → ввод сообщения → кнопка "Commit to main".
* **Пуш:** Кнопка "Push origin".
* **Пулл:** Кнопка "Fetch origin" → "Pull origin".

**Вывод**

В ходе работы были освоены основы Git и GitHub: создание репозитория, клонирование, коммиты и push. Git доказал свою эффективность для контроля версий, а GitHub — для удобного collaboration. Работа подтвердила важность СКВ в современной разработке.