**Приёмы работы в современных системах контроля версий.**

**Цель работы:** познакомиться на практике с основными приемами работы в современных системах контроля версий.

**Задачи для выполнения:**

1. Работа с системой контроля версий Git.
2. Работа GUI git клиент GitKraken.
3. Работа с удаленными репозиториями и GitHub.

**Требования к отчету**

Отчет должен содержать:

* Описание каждого упражнения и его цели.
* Пошаговое объяснение реализации задачи, включая описание используемых функций и методов.
* Анализ полученных результатов и выводы.
* Любые трудности, с которыми студенты столкнулись при выполнении задания, и способы их решения.
* Размышления о том, как полученные знания могут быть применены в практических проектах.

**Упражнение 1. Работа с системой контроля версий Git.**

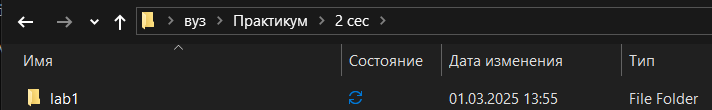
* 1. Установить на компьютер графический клиент Git.

После установки в главном меню появляются следующие приложения:



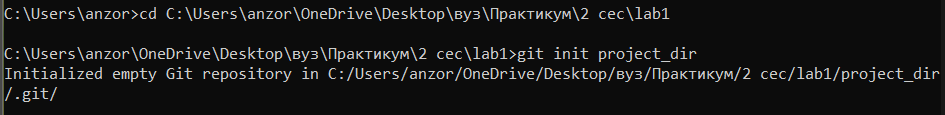
* 1. Создайте в своей домашней папке (или в любой другой на ваш выбор) каталог, который будет содержать файлы нового программного проекта.

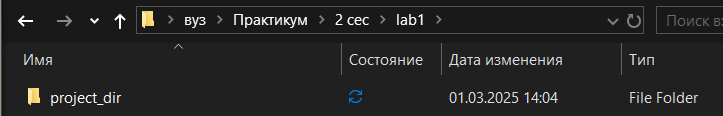
Создаем папку lab1, которая будет являться каталогом для нашего проекта.



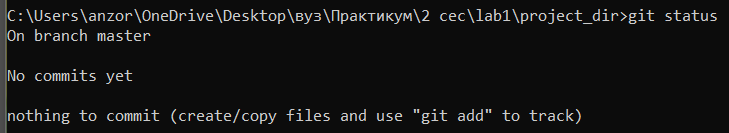
* 1. Инициализируйте в этой директории репозиторий гит при помощи команды git init. Обратите внимание на появление в этой папке скрытой подпапки с названием .git. Если вы ее не видите, то скорее всего, у вас отключено отображение скрытых папок.

Инициализируем репозиторий project\_dir, теперь она отображается в каталоге для проекта.





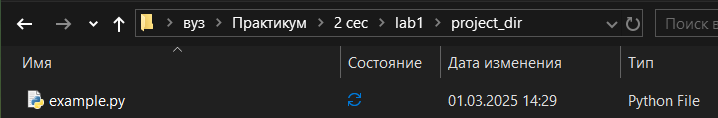
* 1. Выполните в репозитории команду git status. Проинтерпретируйте полученное сообщение.

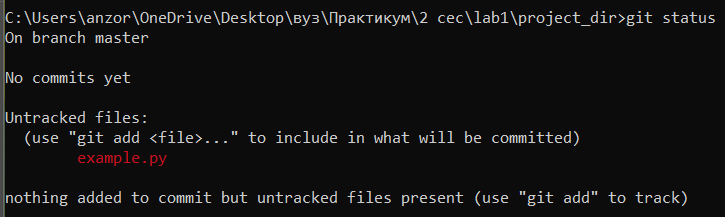


Полученное сообщение означает, что у нас пока что нет ни одного коммита.

* 1. Создайте файл для исходного текста программы. Выполните команду git status.

Создаем файл example.py:

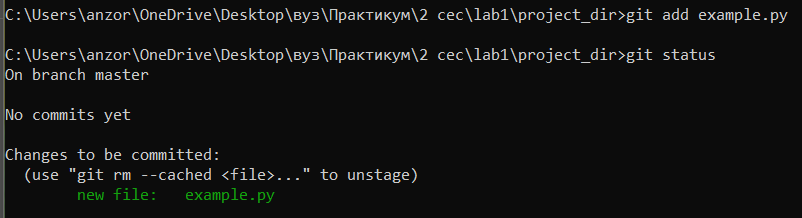




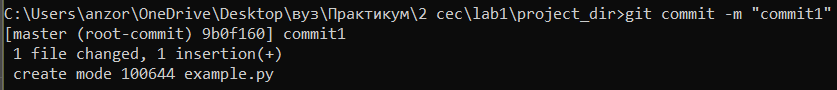
Прописываем команду git status, теперь у нас есть неотслеживаемый файл example.py – «Гит» видит его, но, если попытаться сохранить их версию сейчас, «Гит» этого не сделает. Для этого файлы нужно подготовить к сохранению.

* 1. Добавьте созданный файл под версионный контроль при помощи команды git add. Еще раз выполните git status.

Прописываем команду git add и вносим созданный файл под версионный контроль. Еще раз прописываем git status – теперь файл example.py индексирован.



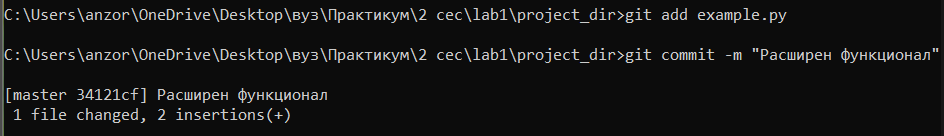
* 1. Когда все файлы готовы к сохранению, сделаем наш первый коммит — зафиксируем все сделанные изменения в «боевой версии». Делается это командой git commit c опцией -m. После -m идёт название коммита в кавычках.

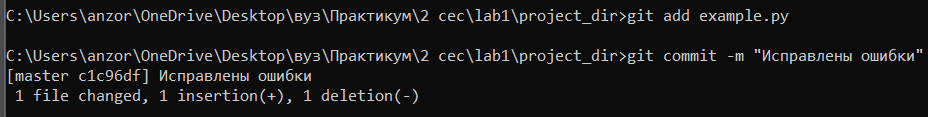


* 1. Сделайте еще несколько коммитов. Выполните команду git log для

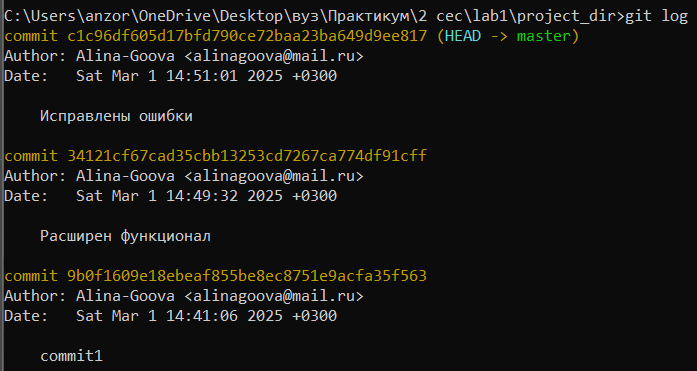
просмотра истории коммитов.

Вносим изменения в файл программы и создаем коммиты, перед этим добавляя файл под версионный контроль.



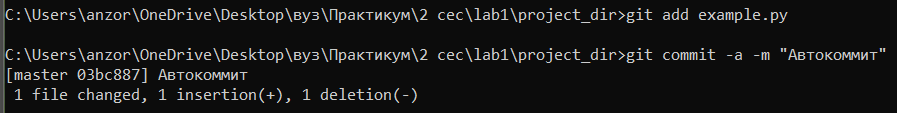


Выполняем команду:

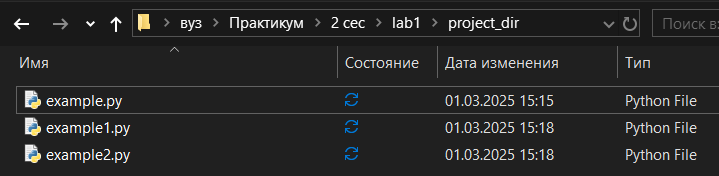


* 1. Сделайте так, чтобы при коммите измененные файлы автоматически добавлялись в коммит.

Чтобы при коммите измененные файлы автоматически добавлялись в коммит, используется команда git commit с опцией -a, которая автоматически индексирует все измененные и удаленные файлы, которые уже отслеживаются Git, но не добавляет новые файлы, о которых Git еще не знает.

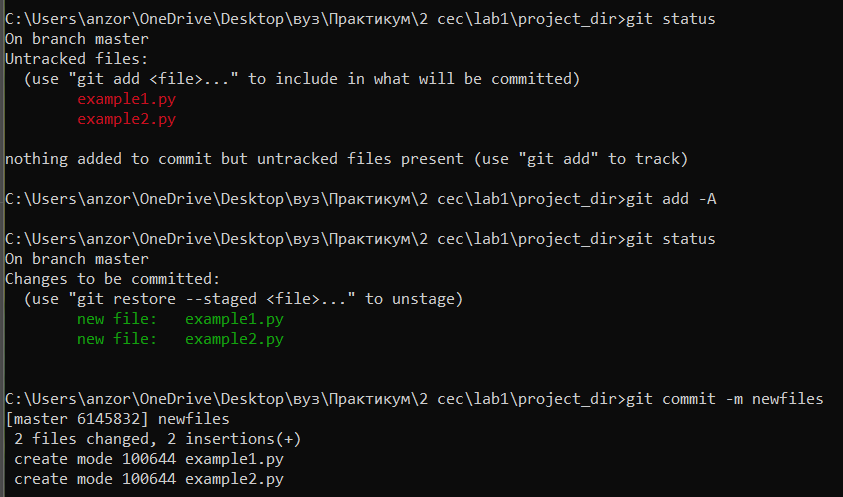


* 1. Добавьте еще несколько файлов с исходным текстом программы.



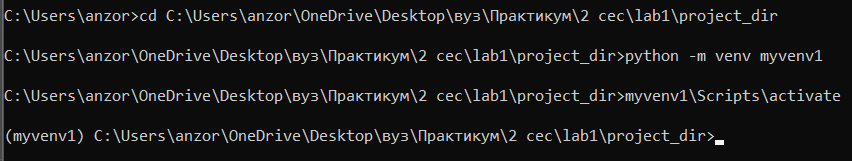
* 1. Добавьте все новые файлы под версионный контроль одной командой.

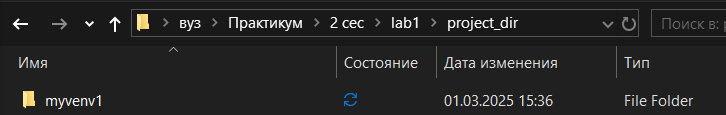
У нас появилось два неотслеживаемых файла, мы вносим их под версионный контроль с помощью команды git add с опцией -А, которая используется для добавления всех изменений, включая новые файлы, измененные файлы и удаленные файлы.



* 1. Инициализируйте в рабочей директории виртуальное окружение

Делаем это через терминал Windows PowerShell.

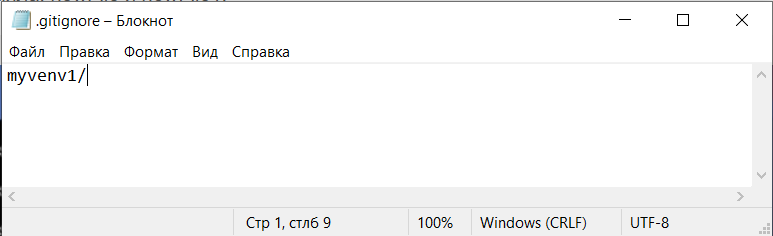




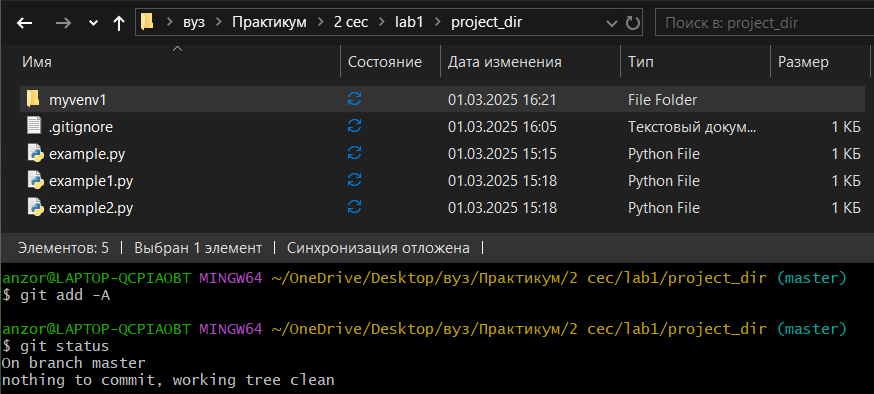
* 1. Добавьте созданную служебную папку в файл .gitignore. Проверьте,

что они не добавляются в репозитории при добавлении новых файлов с исходным кодом.

Вписываем название созданного виртуального окружения в файл .gitignore:

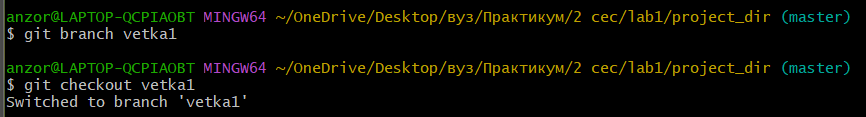


Выполняем команду git add -A и видим, что файлы виртуального окружения не добавляются в репозитории.

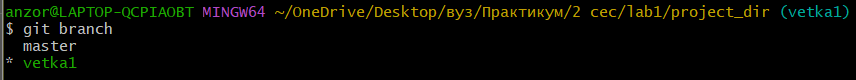


* 1. Создайте новую тематическую ветку git branch. Перейдите в нее с

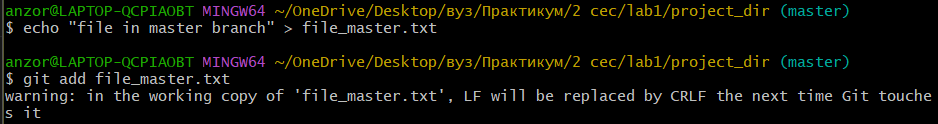
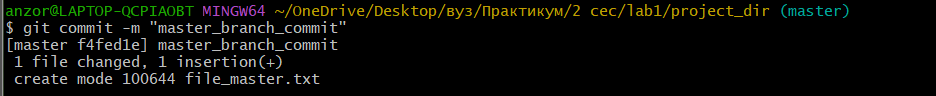
помощью git checkout. Выведите на экран список всех веток.

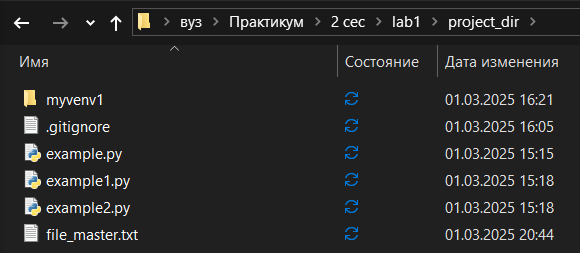


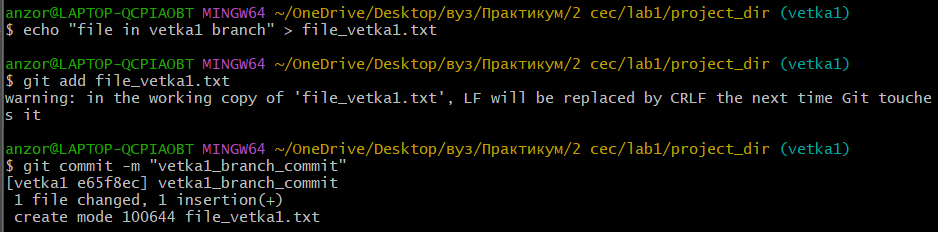
Список всех веток:

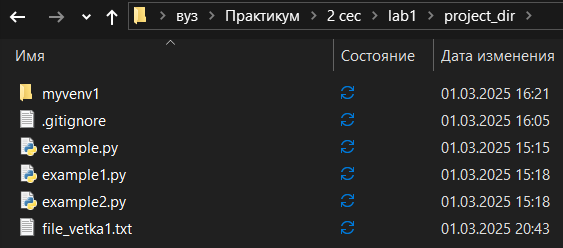


* 1. Сделайте несколько коммитов в основную и тематическую ветки





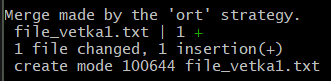


При переключении между ветками в папке отображаются только файлы, созданные в той ветке, в которой мы находимся.

* 1. Слейте изменения в основную ветку с помощью git merge. Если

произошел конфликт слияния, разрешите его и завершите слияние с

помощью git commit.

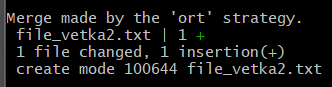


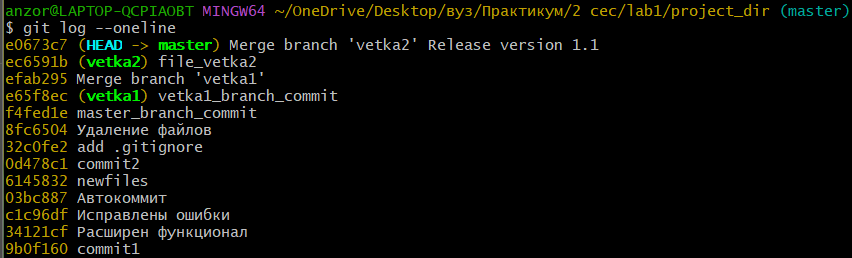
Конфликтов не возникло, слияние прошло успешно, и файл file\_vetka1.txt был добавлен в репозиторий.

* 1. При получении в процессе разработки программы в стабильно работающем состоянии, слейте это состояние в основную ветку и добавьте к коммиту слияния пометку с номером релиза.

В процессе слияния открывается файл в текстовом редакторе, прописываем в него пометку с номером релиза.



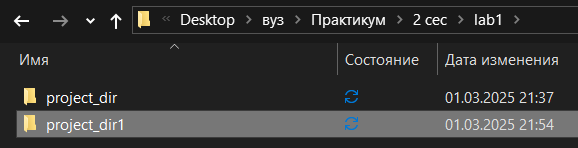




**Упражнение 2.** Работа GUI git клиент GitKraken.

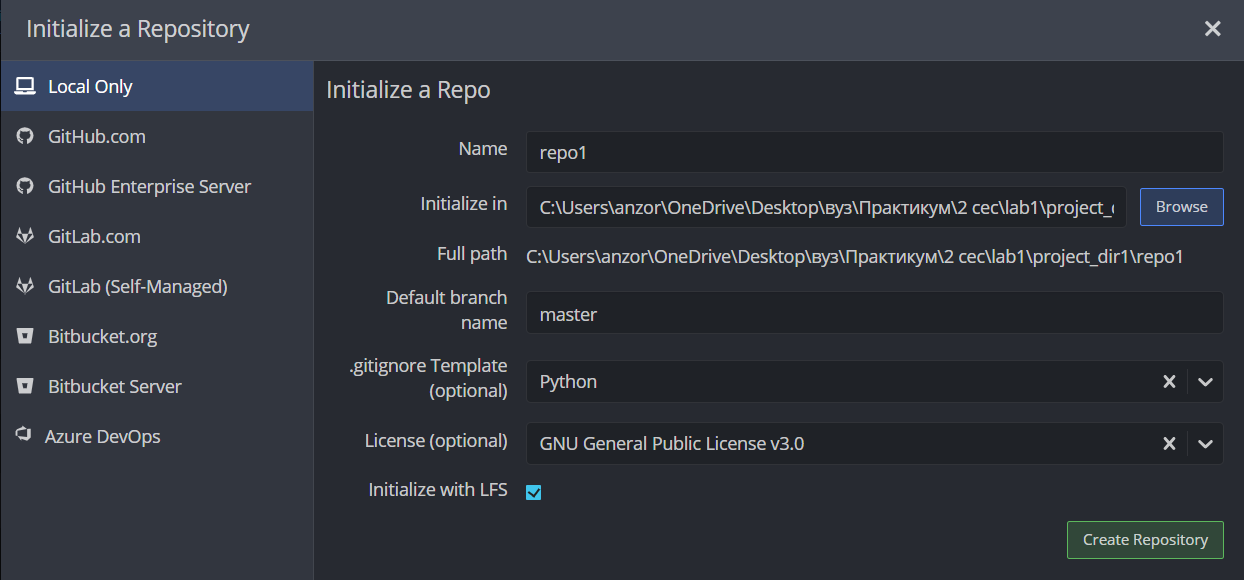
1. Выберите тематику программы, которую собираетесь написать. Создайте для нее рабочую директорию.

Создаем папку project\_dir1 в качестве рабочей директории для программы.



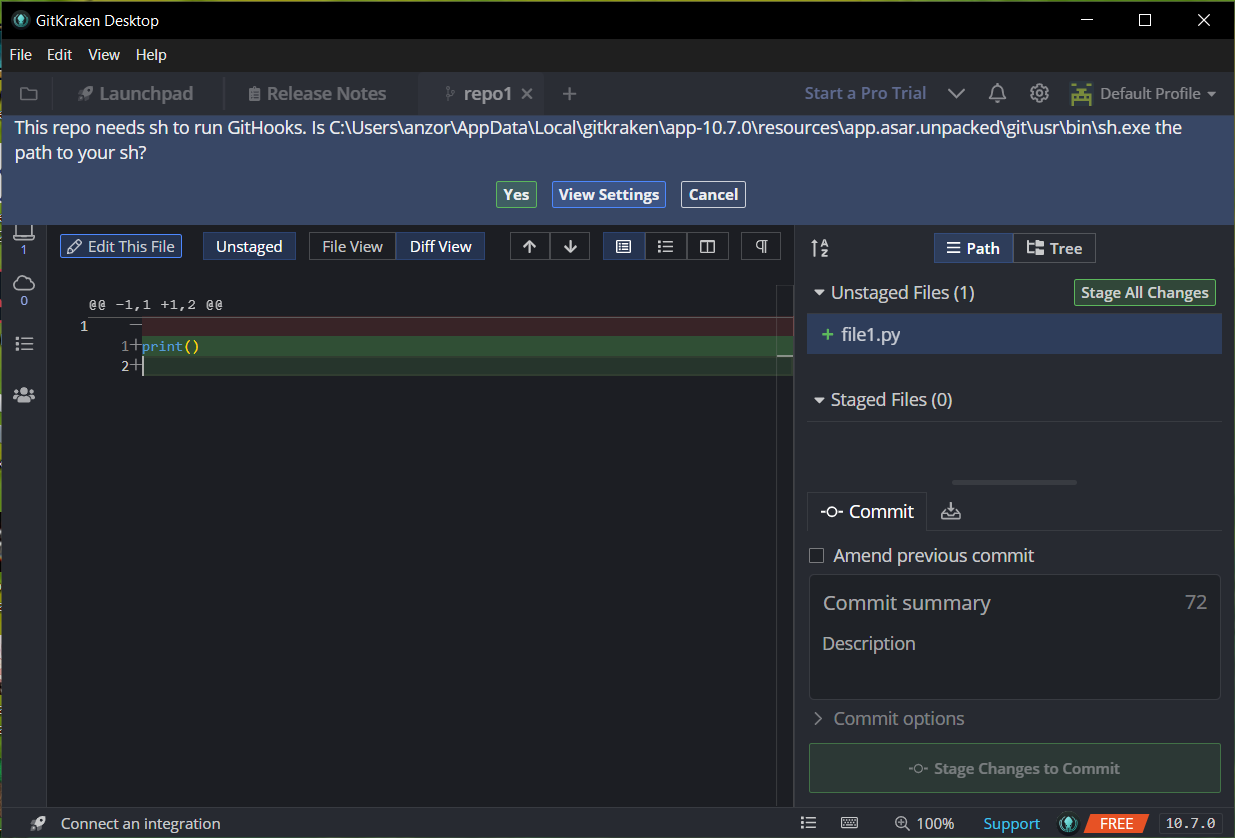
1. Инициализируйте в рабочей директории репозиторий.

Инициализируем репозиторий repo1 в директории project\_dir1, задаем название ветви по умолчанию master.



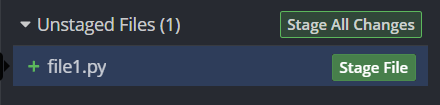
1. Создайте файл для исходного текста программы.

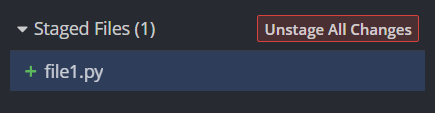
Создаем файл file1.py

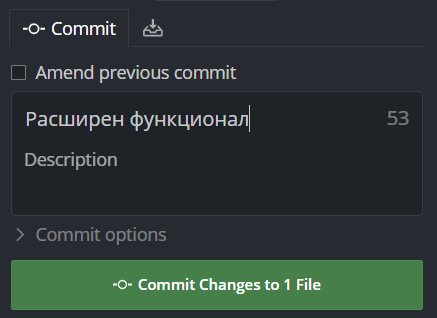


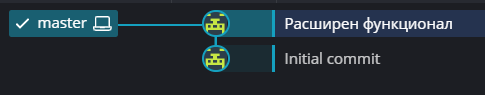
1. Добавьте созданный файл под версионный контроль.

Наводимся на наш файл и нажимаем “Stage File”:

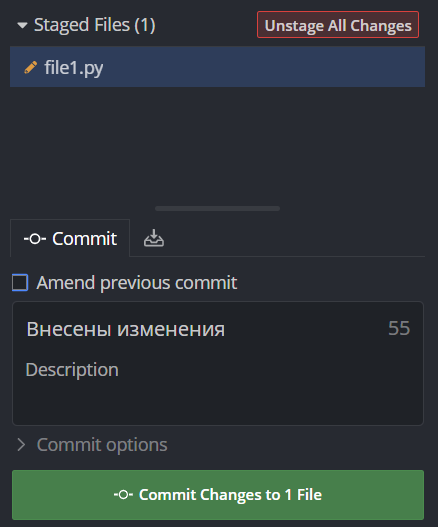


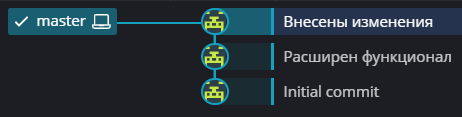


1. Сделайте начальный коммит.

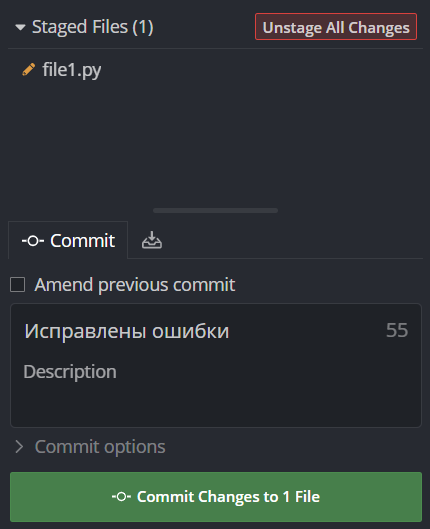


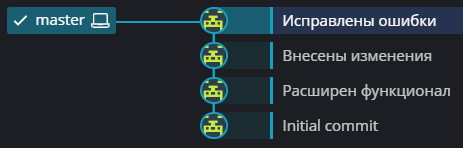
1. Сделайте еще несколько коммитов.



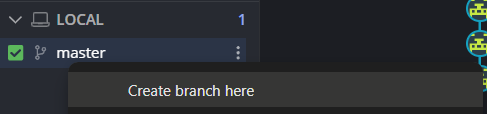


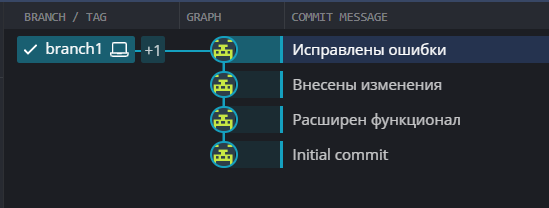
1. Добавьте еще несколько файлов с исходным текстом программы.



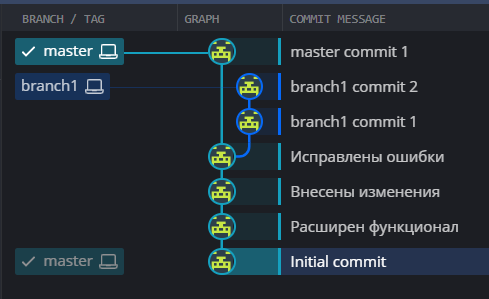


1. Создайте новую тематическую ветку.

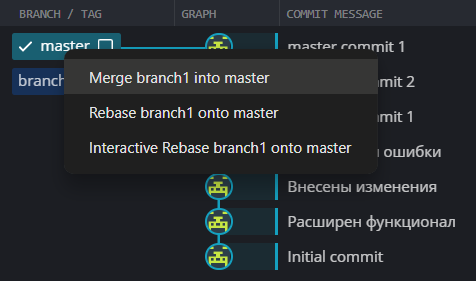




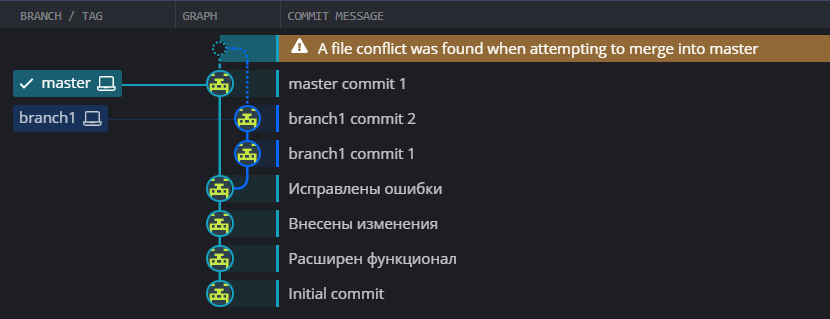
1. Сделайте несколько коммитов в основную и тематическую ветки.

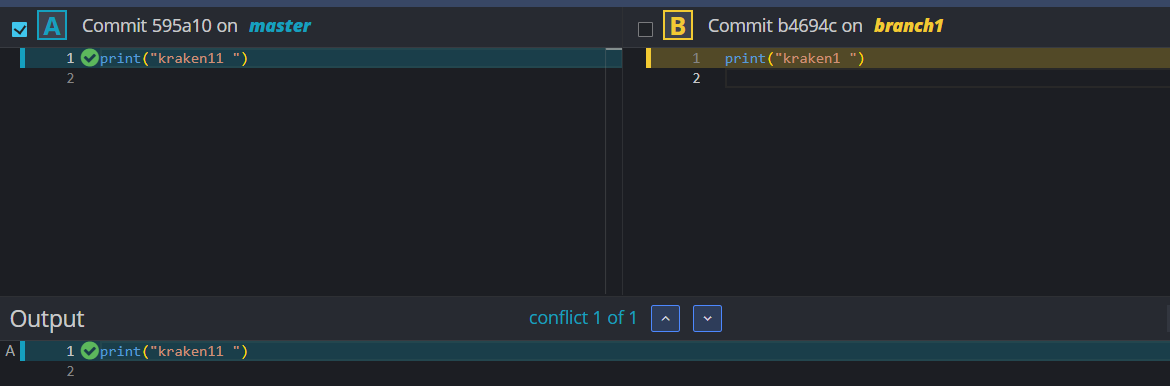


1. Слейте изменения в основную ветку. Если произошел конфликт слияния, разрешите его и завершите слияние.

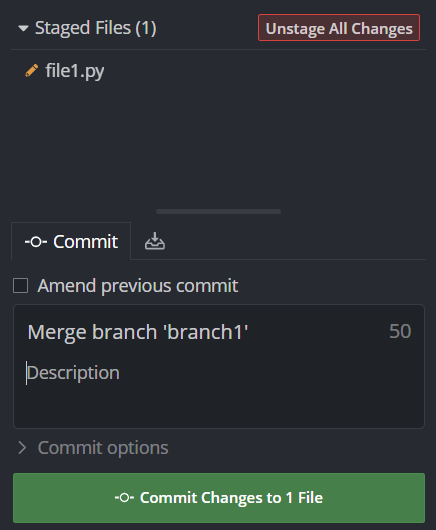
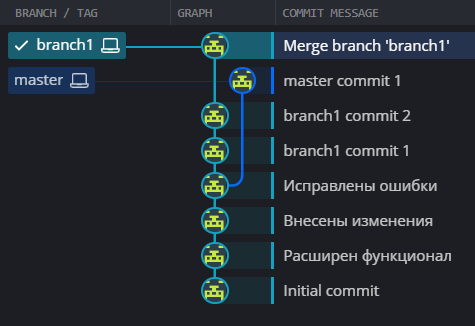


Во время слияния произошел конфликт:

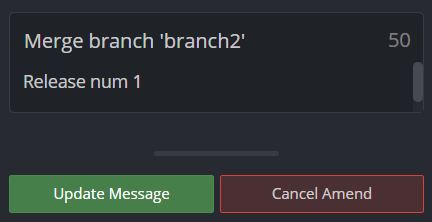




Он возник, так как в один и тот же файл в разных ветках внесены конкурирующие изменения. Используя редактор конфликтов слияния GitKraken, который наглядно визуализирует конфликтующие файлы, чтобы вы могли увидеть, где в коде существуют различия, разрешаем конфликт и выполняем успешное слияние веток:

1. При получении в процессе разработки программы в стабильно работающем состоянии, слейте это состояние в основную ветку и добавьте к коммиту слияния пометку с номером релиза.



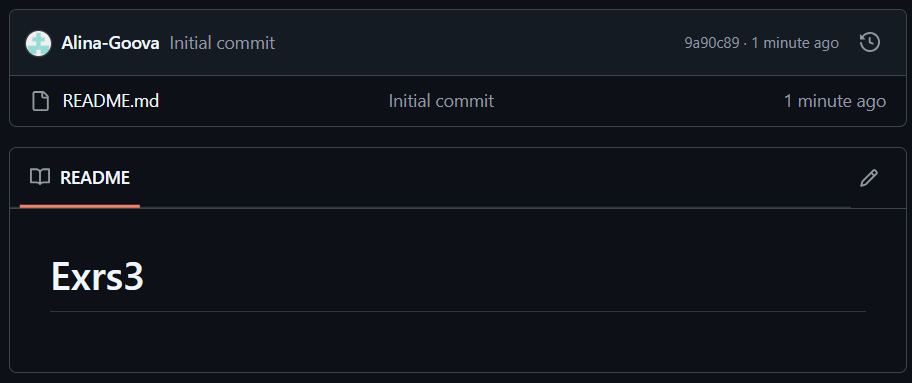


**Упражнение 3.** Работа с удаленными репозиториями и GitHub.

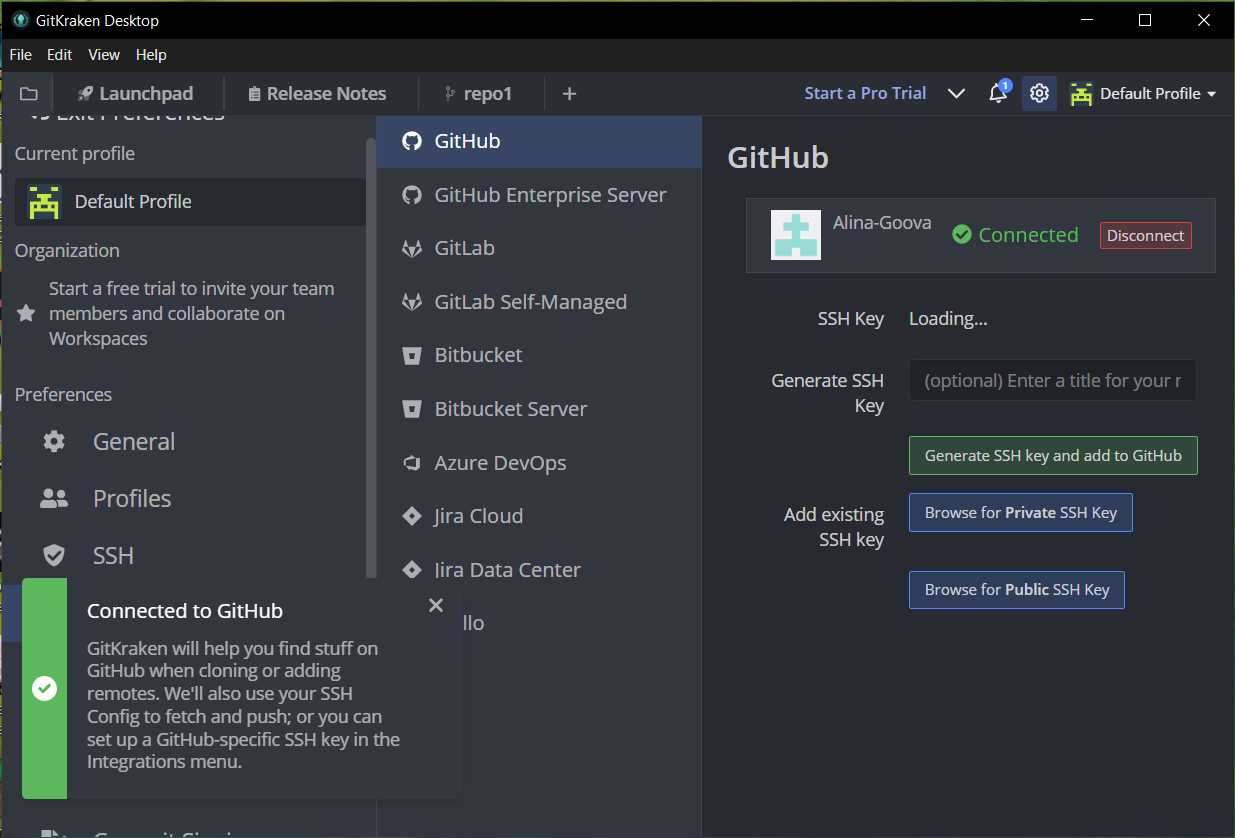
1. Зарегистрироваться на сайте github.com и создать новый пустой репозиторий



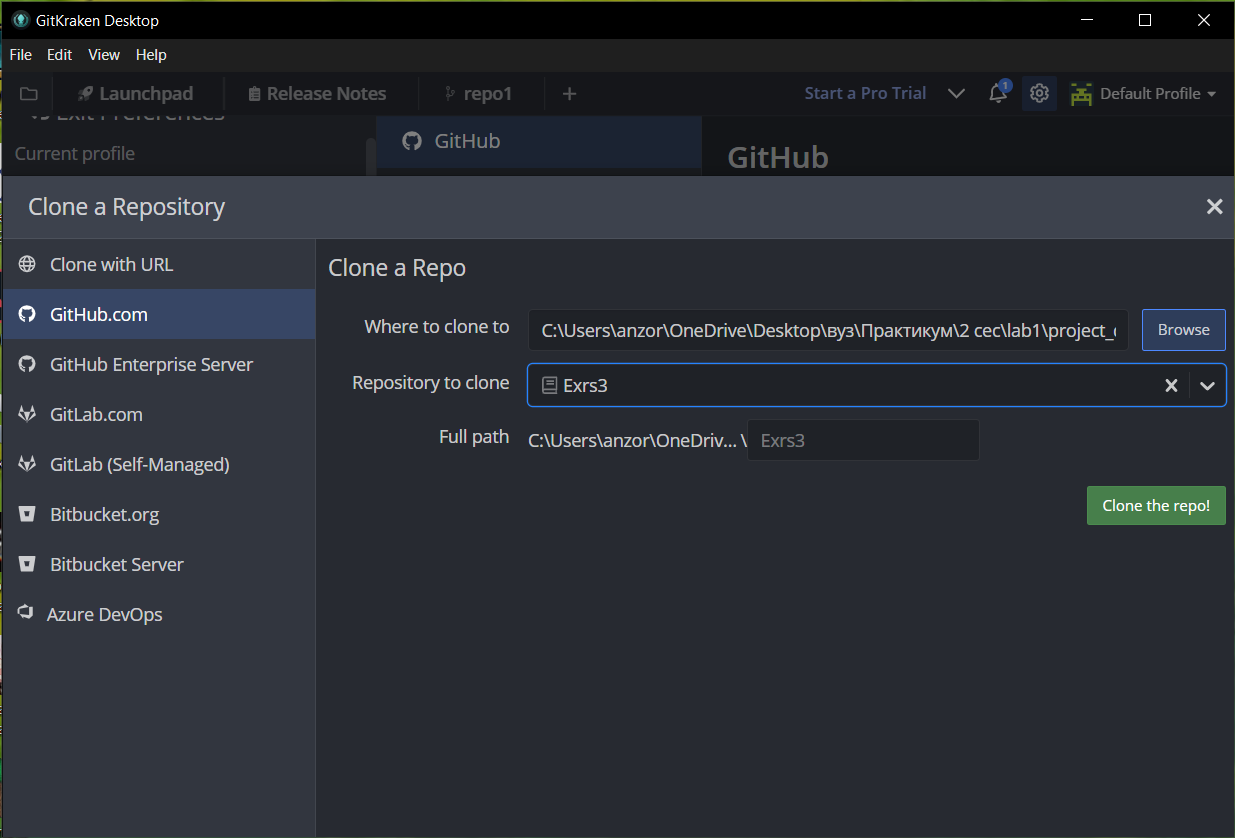
1. Создать в этом репозитории файл Readme



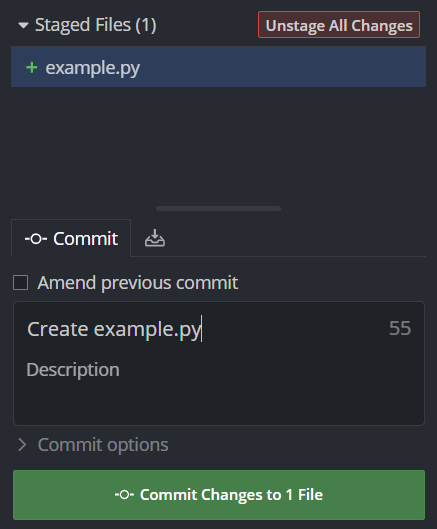
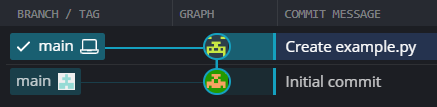
Привязываем свой аккаунт GitHub к GitKraken:



Клонируем заранее созданный репозиторий Exrs3:



– Фиксируем скрипт в репозитории (делаем коммит)

– Обновляем удаленный репозиторий (делаем пуш)

