**2023/09/26 00:00|Лекция по GIT.**

Что такое GIT. Создание репозитория. Коммиты.

### Что такое Git и зачем он нужен?

**Git - это консольная утилита для отслеживания и ведения истории изменения файлов**. Иначе говоря, система контроля версий файлов. Чаще всего его используют для кода, но можно и для других файлов. Например, для картинок, что полезно для дизайнеров.

С помощью Git-a вы можете откатить свой проект до более старой версии, сравнивать, анализировать или сливать свои изменения в репозиторий.  
Репозиторием называют хранилище вашего кода и историю его изменений. Git работает локально и все ваши репозитории хранятся в определенных папках на жестком диске.  
Для чего нужен Git? К примеру, когда два программиста работают над одним файлом, позже с помощью Git можно совмещенно работать над обеими версиями файла и объединить их.

### Жизнь текстового файла (File Status Lifecycle)

Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Автоматически созданное описание

(Рис.1)

Здесь представлены три версии текста (Рис.1). Мы будем рассматривать их во время процесса, визуально наблюдая за каждым коммитом, сопровождаемым определенным сообщением от раннее просмотревшего его человека.

### Практика Git

**План**:

1. Создать файл repka.txt

2. Создать git repo

3. Добавить repka.txt в Git

4. Первый коммит

5. Изменение файла, второй коммит

Переключимся на вкладку **Project** и создадим новый файл repka.txt (Рис.2).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

(Рис.2)

Далее пробуем симулировать. Для этого откроем вкладку **VCS** (Version Control System), а затем **VCS Operations Popup** и **Create Git Repository**. Находим папку нашего проекта (Рис.3).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

(Рис.3)

Изменение цвета названия файла на красный говорит о том, что этот файл не находится под контролем версии (Рис.4).

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Автоматически созданное описание

(Рис.4)

Чтобы сохранить файл, щелкнем по нему правой кнопкой мыши и наведем на строчку **Git**, а затем нажимаем **+Add**, как показано на картинке (Рис.5), чтобы добавить файл в репозиторий.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

(Рис.5)

Теперь приступим к коммиту этих файлов. **Pycharm** поможет открыть окно с содержимым коммита.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

(Рис.6)

Обратите внимание, **Commit Message** – это поле, где программист описывает внесенные изменения. Как мы видим, начальный коммит здесь initial (Рис.6), чтобы все сработало, нужно указать коммиту, кто оставил комментарий. Вводим свои данные (Рис.7).

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

(Рис.7)

Вносим изменения и видим, как файл меняет свой цвет на синий, а также подсвечиваются измененные строчки (Рис.8). Если убрать изменения подсветка пропадет.

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

(Рис.8)

Так же можно посмотреть предыдущее состояние файла (Рис.9).

Изображение выглядит как текст, Мультимедийное программное обеспечение, программное обеспечение, снимок экрана

Автоматически созданное описание

(Рис.9)

Перейдя к разделу журнал (находится под Commit Message), мы можем наблюдать первоначальный коммит и все последующие (Рис.10). А также файлы, связанные с каждым из них.

Изображение выглядит как текст, Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

(Рис.10)

Теперь у нас есть запись изменений, внесенных в файл, начиная с первоначального коммита.

### Жизнь нескольких файлов

Git сохраняет лишь разницу между версиями текста, а не весь файл целиком. Здесь наглядно показано, как происходят сохранения, белым выделены файлы, оставшиеся без изменений, а синим измененные (Рис.11). Git запоминает лишь измененные строчки, что особенно удобно при работе с большим объёмом информации.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

(Рис.11)

### Практика Git для нескольких файлов

**План**:

1. Создаем несколько файлов

2. Правим файлы, коммитим

3. Сообщение коммита (Важно!)

4. Просмотр истории

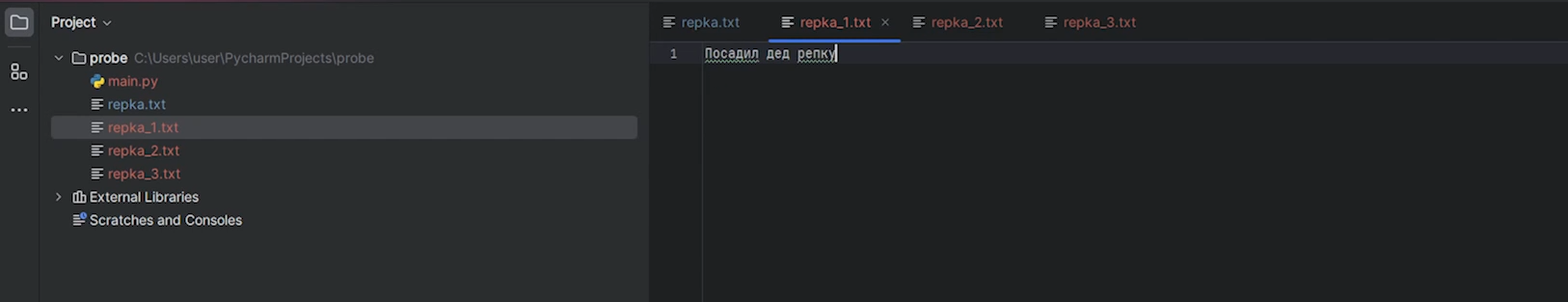
Создадим три файла, пронумеровав от 1 до 3 (Рис.12).

Изображение выглядит как текст, снимок экрана, Шрифт

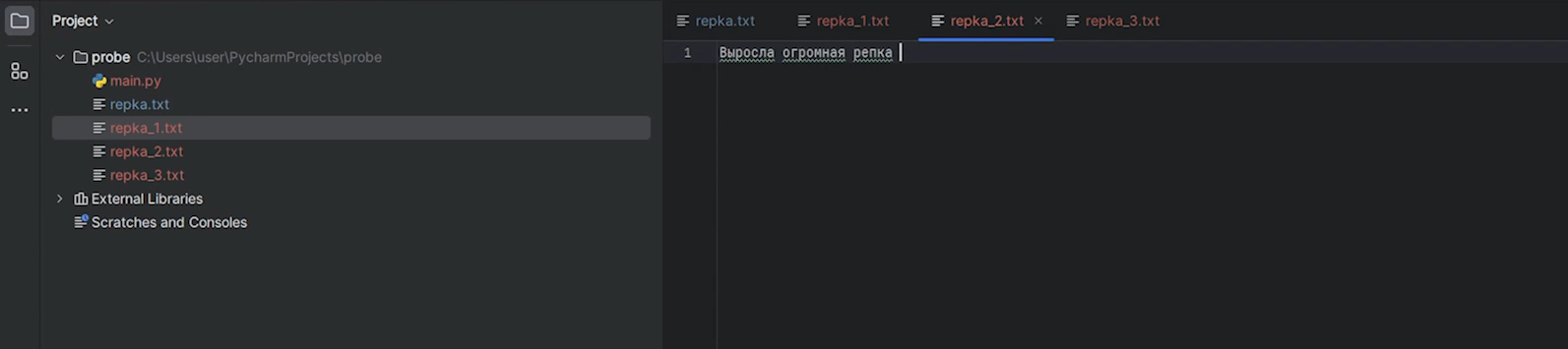
Автоматически созданное описание

(Рис.12)

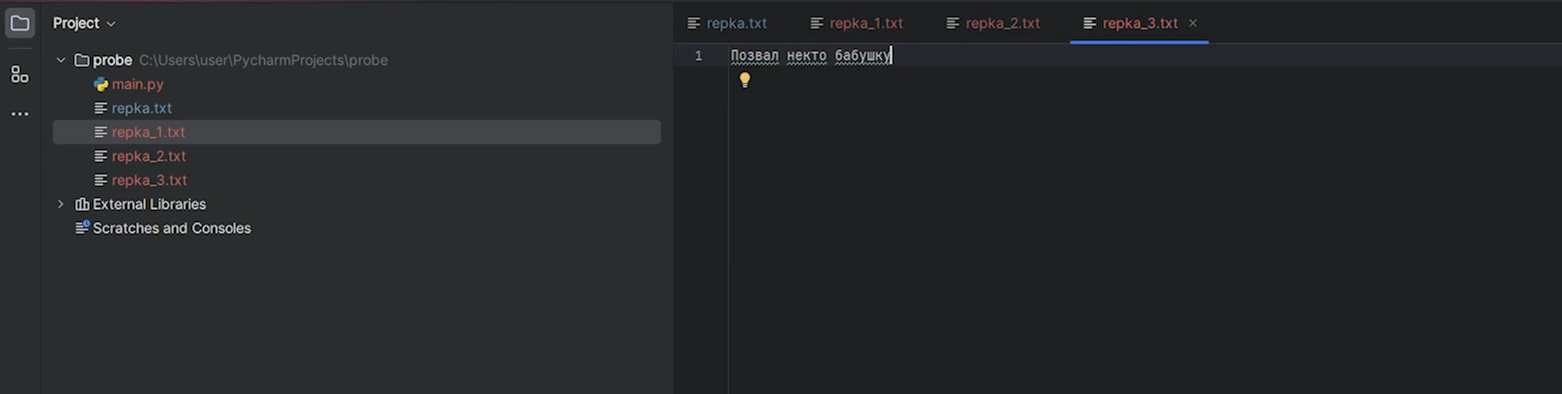
Делаем запись в первом, втором и третьем файлах соответственно (Рис.13, 14, 15).



(Рис.13)

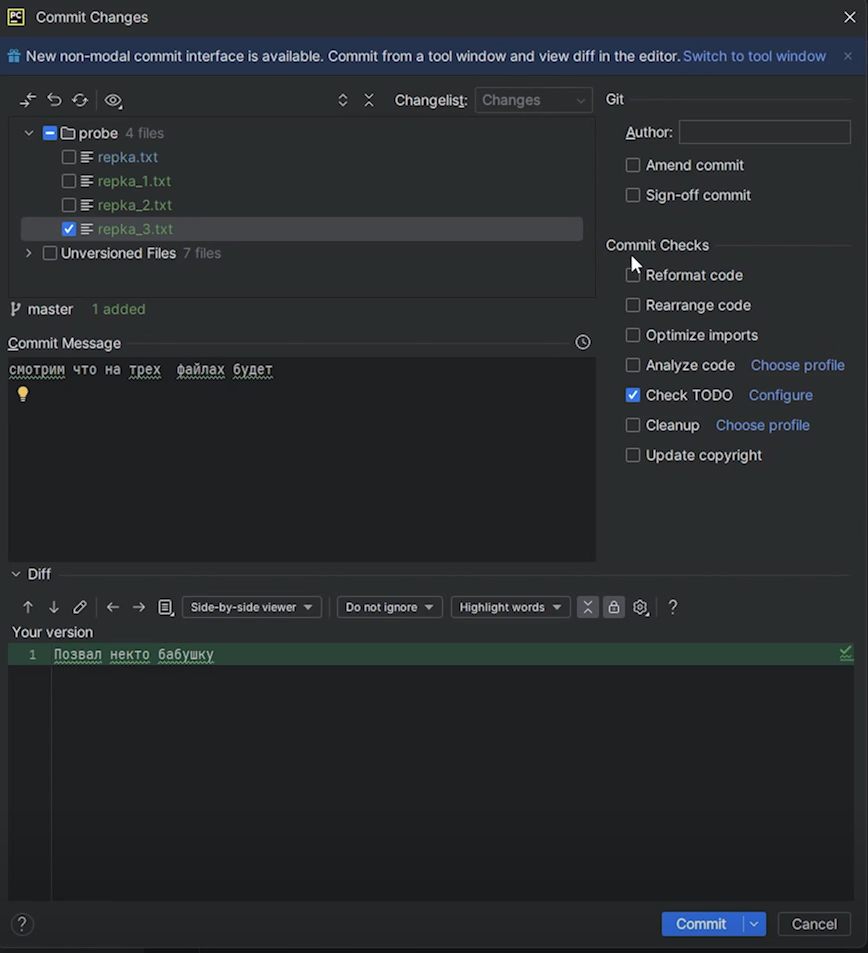


(Рис.14)



(Рис.15)

Проделываем уже известные нам действия, чтобы сохранить файлы в репозиторий. Теперь коммитим их. Нажимаем правую кнопку мыши и открываем окно содержимого коммита. Пишем коммит и помечаем все три файла (Рис.16).



(Рис.16)

Вносим изменения в третий файл (Рис.17).

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

(Рис.17)

Пишем коммит к третьему файлу (Рис.18).

Изображение выглядит как снимок экрана, Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

(Рис.18)

Теперь можем просмотреть все изменения файлов (Рис.19).

Изображение выглядит как текст, Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

(Рис.19)

А также коммиты к ним (Рис.20).

Изображение выглядит как текст, Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

(Рис.20)

Теперь вносим изменения во второй файл (Рис.21).

Изображение выглядит как текст, Мультимедийное программное обеспечение, программное обеспечение, снимок экрана

Автоматически созданное описание

(Рис.21)

И пишем коммит к нему (Рис.22).

Изображение выглядит как Мультимедийное программное обеспечение, программное обеспечение, Графическое программное обеспечение, снимок экрана

Автоматически созданное описание

(Рис.22)

Коммит должен содержать одно логическое изменение. А сообщение коммита обязательно должно содержать описание изменения. Чем больше коммитов, тем подробнее можно проследить прогресс проекта.

### Копирование изменений Git

При переносе изменений с одного компьютера на другой, мы соединяем два репозитория. У Git есть свой сервер со множеством репозиториев – Gitlab.

Изображение выглядит как текст, Шрифт, снимок экрана, чек

Автоматически созданное описание

(Рис.23)

С помощью параметра **Push** можно отправить изменения, а для загрузки изменений из другого репозитория используется параметр **Pull**(Рис.23).

### Полезные ссылки

<https://git-scm.com/downloads>**-**Скачать Git  
  
<https://about.gitlab.com/> - Gitlab