**2. Работа с Git в терминале**

**Цель работы**

На основе уже полученных знаний о принципах работы СКВ получить более глубокое представление о работе Git при помощи команд терминала.

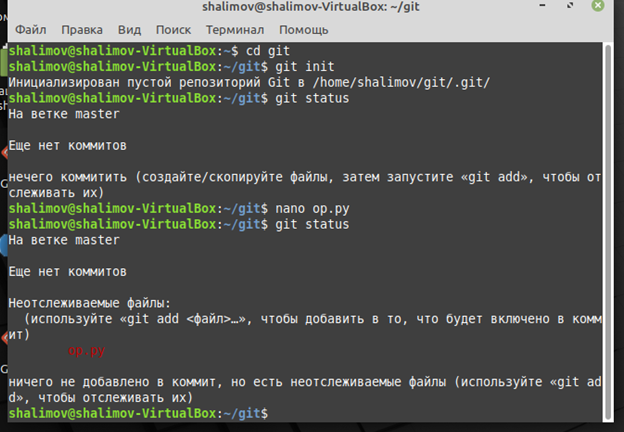
**Задания для выполнения**

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

Инициализируйте в рабочей директории репозиторий при помощи команды git init.

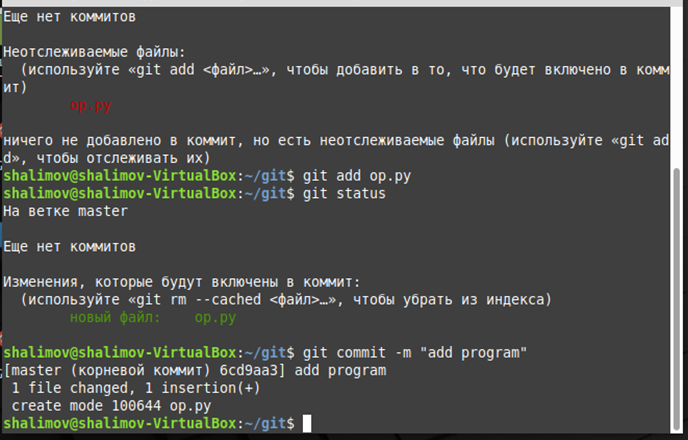
Выполните в репозитории команду git status. Проинтерпретируйте полученное сообщение.

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

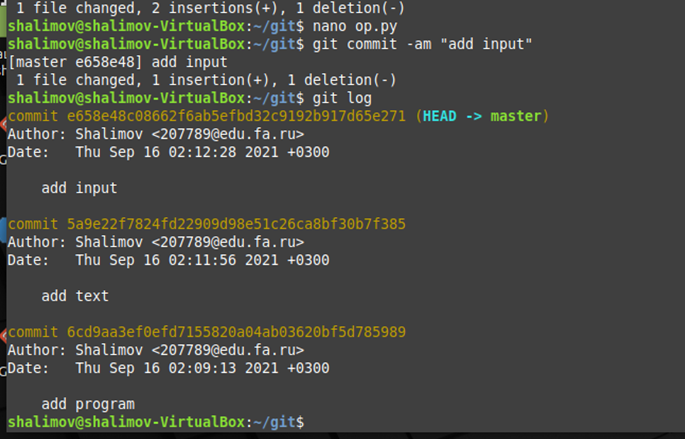


Добавьте созданный файл под версионный контроль при помощи команды git add. Еще раз выполните git status.

Сделайте начальный коммит при помощи команды git commit с опцией -m.



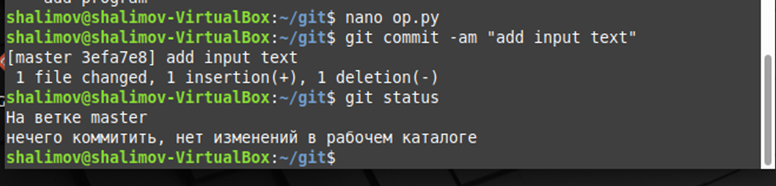
Сделайте еще несколько коммитов. Выполните команду git log для просмотра истории коммитов.



Добавьте еще несколько файлов с исходным текстом программы.

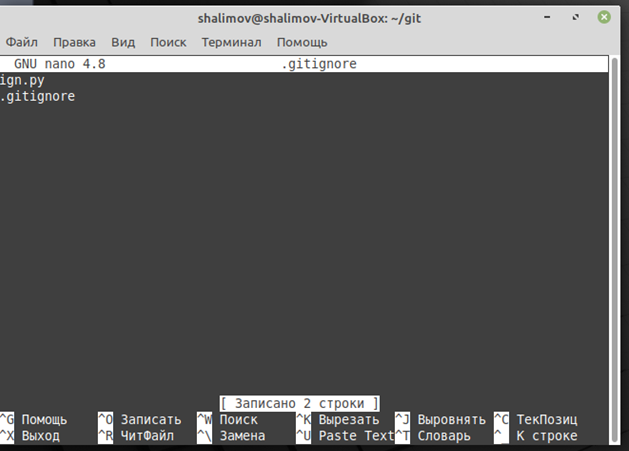
Добавьте все новые файлы под версионный контроль одной командой.

На всех стадиях работы пользуйтесь командой git status.



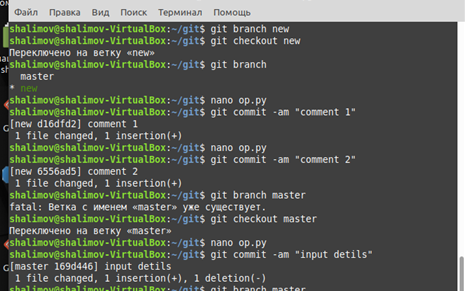
Инициализируйте в рабочей директории виртуальное окружение (Если вы пишите не на Python, то можете инициализировать какой-либо программный фреймворк, либо начать работать в IDE, которая создает скрытую папку с настройками в рабочем каталоге).

Добавьте созданную служебную папку в файл .gitignore. Проверьте, что они не добавляются в репозитории при добавлении новых файлов с исходным кодом.

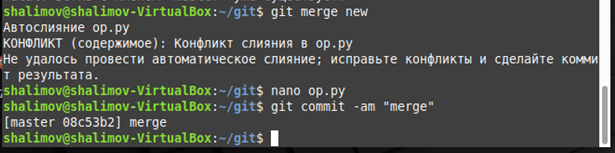


Создайте новую тематическую ветку git branch. Перейдите в нее с помощью git checkout. Выведите на экран список всех веток.

Сделайте несколько коммитов в основную и тематическую ветки.



Слейте изменения в основную ветку с помощью git merge. Если произошел конфликт слияния, разрешите его и завершите слияние с помощью git commit.



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.пше икфтср

**Методические указания**

Для работы в терминале изучите шпаргалки по основным командам git.

**Главные команды, которые вам понадобятся это**:

git init - создает репозиторий системы контроля версий в данной директории;

git add - добавляет указанный файл под версионный контроль;

git add . - добавляет все файлы текущей директории под версионный контроль;

git status - показывает состояние рабочей директории по сравнению с последним сохраненным состоянием:

git commit -m "(message)" - сохраняет текущее состояние рабочей директории как новое состояние (создает новый коммит); новый коммит получает сообщение, переданное как параметр;

git commit -am "(message)" - создает новый коммит и автоматически включает в него все изменившиеся отслеживаемые файлы;

git log - выводит историю коммитов репозитория;

git branch - показывает список веток репозитория;

git branch - создает новую ветку на основе текущего состояния с переданным названием;

git checkout -b - создает новую ветку и автоматически делает ее текущей;

git merge - сливает изменения, сделанные в ветке с переданным названием в текущую;

git branch -d - удаляет ветку с переданным названием;

git clone (repo URL) - клонирует удаленный репозиторий, находящийся по переданному адресу в текущую директорию; доступ обычно осуществляется по протоколам HTTP либо SSH;

git fetch - считывает изменения в удаленном репозитории, отсутствующие в локальной копии;

git pull - считывает новые изменения в удаленном репозитории и сливает их в соответствующие локальные ветки;

git push –all - отправляет изменения, сделанные в локальном репозитории в удаленный;

**Контрольные вопросы**

Что такое удаленный репозиторий? Каталоги и файлы общего проекта сохранённые в интернет сети.

Где нужно вводить команды git? В терминале ОС.

Для чего нужны ветки в системах контроля веток? Ветки нужны для параллельной работы нескольких программистов над разными частями кода в одном файле.

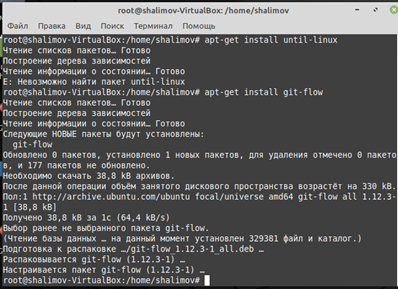
Как возникают конфликты слияния – конфликты слияния возникают когда гит обнаруживает изменения в одних и тех же строках в разных ветвях, в связи с чем не может определить какие из них необходимо внести в итоговую версию проекта.

Как разрешать конфликты слияния? – или прервать слияние командой git merge –abort, или зайти в текстовый редактор и вручную исправить конфликт

**Дополнительные задания**

Ознакомьтесь с методологией разработки GitFlow.

Установите на свой компьютер инструментальное средство для работы с GitFlow.



Выполните основные задания лабораторной работы с использованием команд git-flow.

