Устимова Снежана, ПИ19-3

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

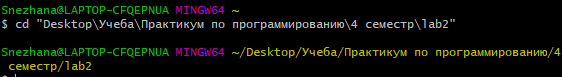
Тематику программы я менять не стала. Все тот же «Hello, world!».

Создание директории:

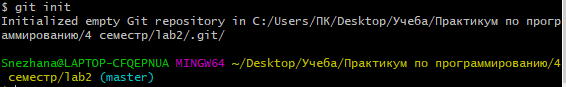


2. Инициализируйте в рабочей директории репозиторий при помощи команды git init.

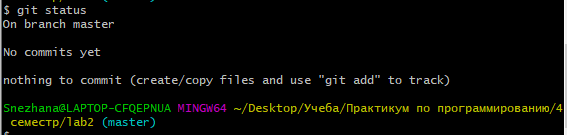
Сначала перейдем в эту папку:



И сделаем init:

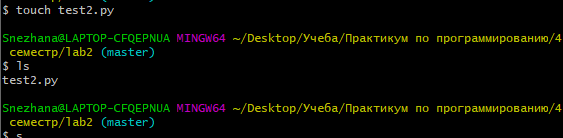


3. Выполните в репозитории команду git status. Проинтерпретируйте полученное сообщение.

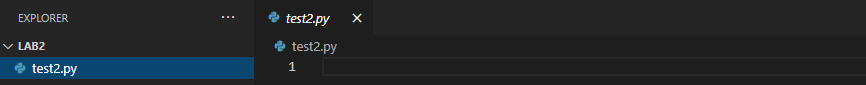


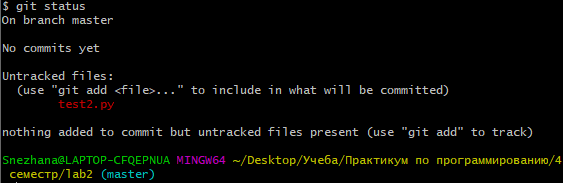
Так как наша папка пока пуста, следовательно, никаких коммитов еще не происходило, да и нам пока нечего коммитить. А первое сообщение говорит о тои, что мы находимся на ветке master.

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

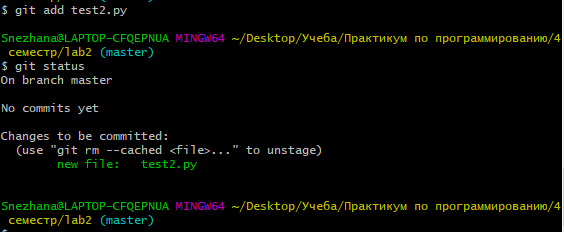


Файл создался в папке lab2:





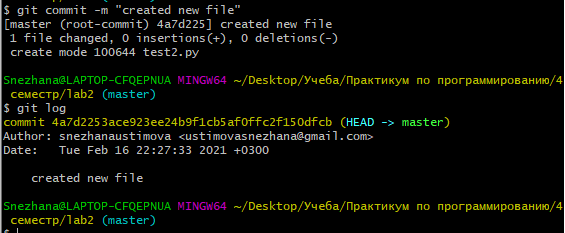
5. Добавьте созданный файл под версионный контроль при помощи команды git add. Ещё раз выполните git status.



Мы видим, что файл из красного стал зеленым, т.е. он добавился под версионный контроль и готов к коммиту.

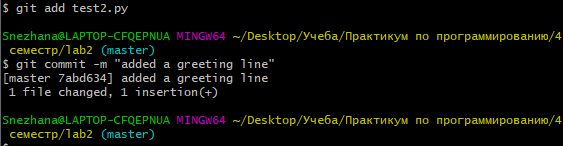
6. Сделайте начальный коммит при помощи команды git commit с опцией –m.

Добавили к коммиту текст «created new file», выполнили и посмотрели историю.



7. Сделайте ещё несколько коммитов. Выполните команду git log для просмотра истории коммитов.

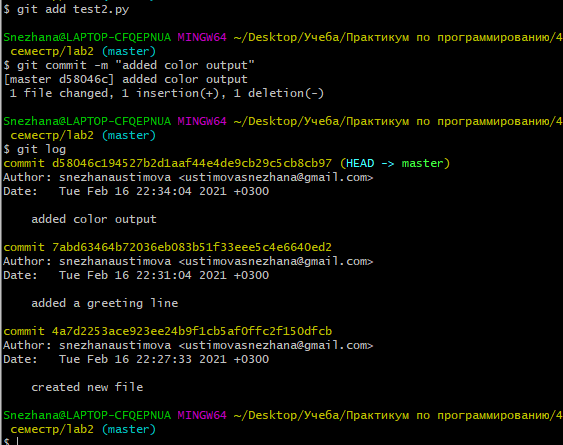




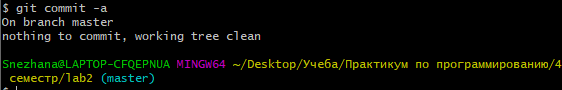
Теперь сделаем еще одно изменение – поменяем цвет:



Коммитим и смотрим историю коммитов:



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

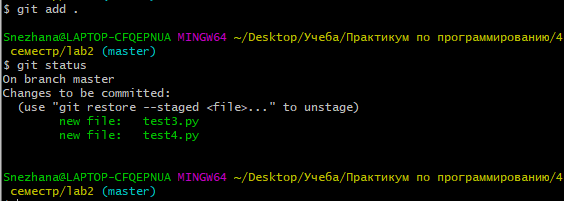


9. Добавьте ещё несколько файлов с исходным текстом программы.





10. Добавьте все новые файлы под версионный контроль одной командой.



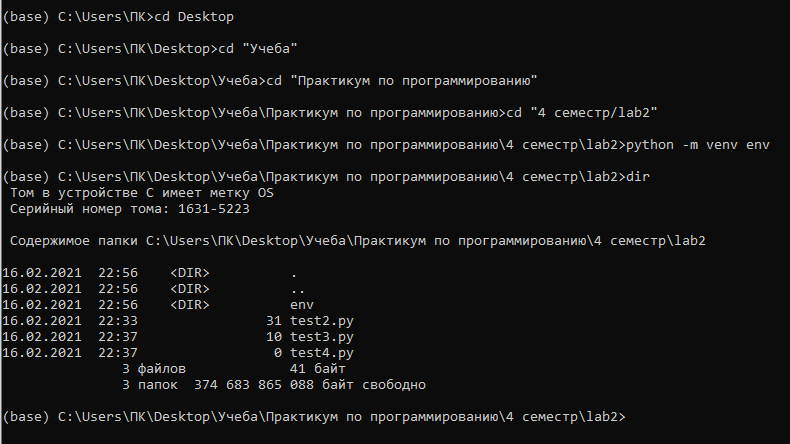
. добавляет все файлы, находящиеся в директории под версионный контроль.

11. На всех стадиях работы пользуйтесь командой git status.

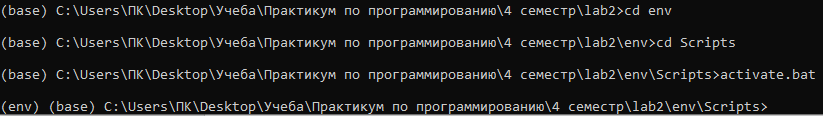
Ну, ладно…

12. 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 создаёт скрытую папку с настройками в рабочем каталоге).

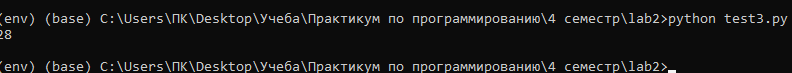
Сначала установим виртуальное окружение:



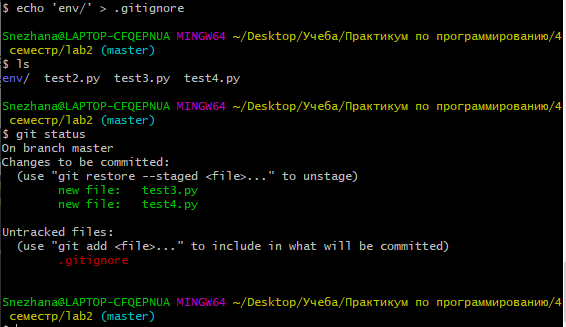
Теперь активируем его:



Проверим работу:

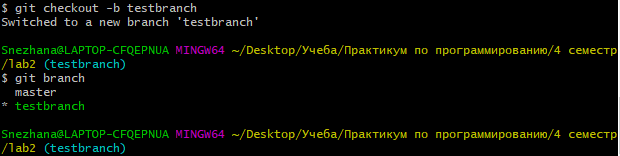


13. Добавьте созданную служебную папку в файл .gitignore. Проверьте, что они не добавляются в репозитории при добавлении новых файлов с исходным кодом.



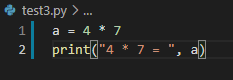
Убеждаемся, что мы не видим папку env.

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

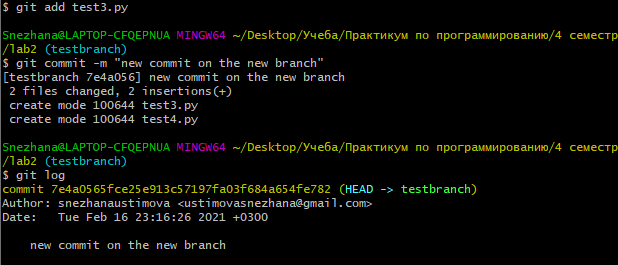


15. Сделайте несколько коммитов в основную и тематическую ветки.

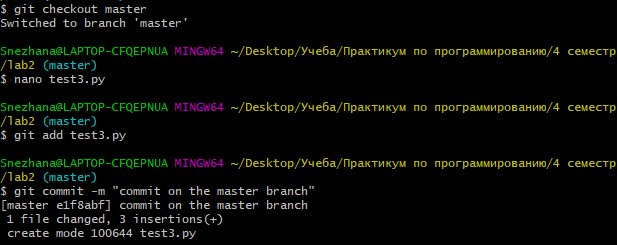
Изменим файл test3.py.



И сделаем коммит:

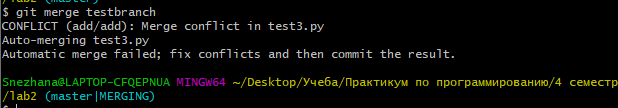


Перейдем в master, изменим test3.py и сделаем коммит:



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

Сольем изменения из testbranch в master:



К сожалению, произошел конфликт слияния, и его нужно решить вручную. Для этого перейдем в режим nano и отредактируем проблемный файл:

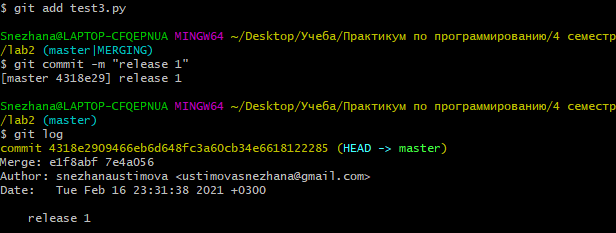


Изменим файл до такого вида, не забыв удалить метки:



Теперь нам осталось только самостоятельно сделать коммит, что и будет соответствовать 17 пункту лабораторной работы:

17. Сделаем последний коммит с пометкой «релиз 1» и напоследок выведем историю коммитов.



Конец!