**1. Основы работы с СКВ в графическом режиме**

**Цель работы**

Познакомиться на практике с основными приемами работы в современных системах контроля версий.

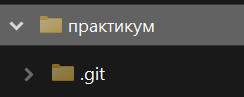
**Задания для выполнения**

1. Установить на компьютер графический клиент Git.
2. Создайте в своей домашней папке (или в любой другой на ваш выбор) каталог, который будет содержать файлы нового программного проекта.

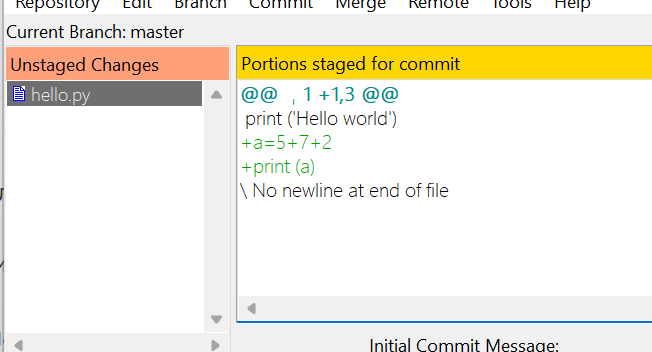
Изображение выглядит как текст

Автоматически созданное описание

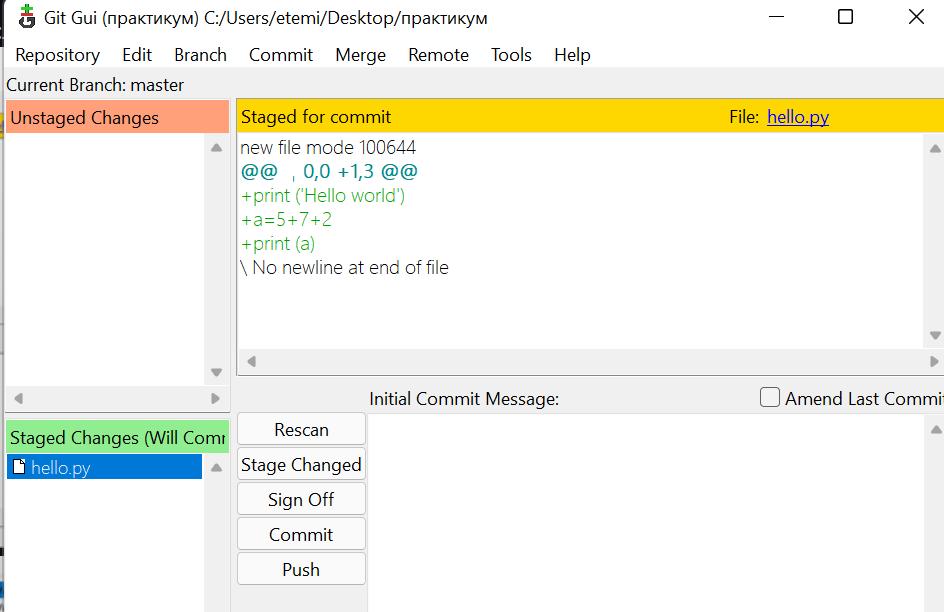
1. Выберите тематику программы, которую собираетесь написать. Язык программирования и используемые инструменты разработки сейчас не важны.
2. Инициализируйте в этой директории репозиторий гит.
3. Обратите внимание на появление в этой папке скрытой подпапки с названием .git. Если вы ее не видите, то скорее всего, у вас отключено отображение скрытых папок.



1. Создайте новый файл для исходного текста программы. Если вы используете программный фреймворк, инициализируйте его в рабочий каталог.
2. Напишите несколько строк вашей программы.
3. Добавьте файл с исходным текстом (несколько файлов, если необходимо) в индекс вашего репозитория.

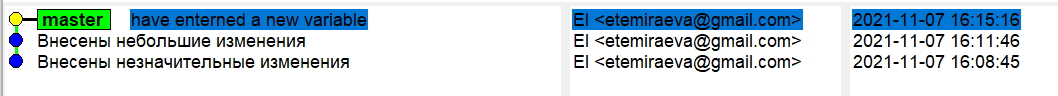


1. Совершите ваш первый коммит. Напишите осмысленное сообщение коммита.





1. Повторите несколько раз. Каждый раз, завершая определенный этап работы, выполняйте коммит и описывайте проделанные изменения в сообщении коммита.
2. Просмотрите историю коммитов. Попробуйте перейти на один из прошлых коммитов. Вернитесь в актуальное состояние программы.



**Методические указания**

Для выполнения лабораторной работы необходимо воспользоваться любым консольным клиентом системы контроля версий Git, подходящую для рабочей операционной системы. Для Windows подойдет, например, дистрибутив [git-scm](https://git-scm.com/). Этот же пакет включает в себя и консольную утилиту git, необходимую для выполнения следующих работ. Для Linux существует, например, пакет [Git Cola](https://git-cola.github.io/).

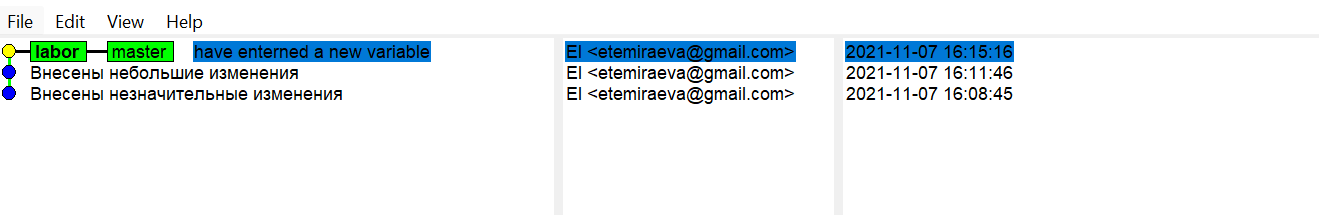
Для написания сообщений коммита необходимо придерживаться следующего общепринятого правила: в первой строке сообщения следует кратко описать произведенные изменения; если необходимо подробное описание, состоящее из многих строк, то его приводят, отступив от первой строки одну пустую. Помните, что заголовок описания коммита - это то, что будете видеть вы и ваши коллеги в истории изменений проекта.

**Контрольные вопросы**

1. Опишите своими словами значение следующих терминов:
   1. рабочий каталог - каталог в котором ведется отслеживание файлов
   2. репозиторий – виртуальное хранилище созданного проекта в котором хранятся данные о отслеживаемых файлах
   3. коммит - некоторое сохранение, контрольная точка которая хранит состояние файлов
   4. ветка - указатель на сохраненное состояние файлов
2. Ознакомьтесь с гайдом по выбранной вами программе-клиенту Git.

**Дополнительные задания**

1. Представьте, что вы начинаете большой раздел работы. Для изоляции изменений создайте новую ветку. Назовите ее, чтобы было понятно, что вы в ней будете делать.



1. Перейдите в новую ветку и сделайте несколько коммитов.

Изображение выглядит как текст

Автоматически созданное описание

1. Перейдите в основную ветку и обратите внимание на состояние рабочей директории.
2. Создайте еще одну ветку для работы над другим направлением в вашей программе. Обычно так работают в команде, каждый участник в собственной ветке. Либо в ветках может идти параллельная работа над разными возможностями программы. В таком случае эти ветки называются тематическими.
3. Сделайте несколько коммитов во вновь созданную ветку.

Изображение выглядит как текст

Автоматически созданное описание

1. Перейдите в основную ветку и слейте в нее первую тематическую ветку.

Изображение выглядит как текст

Автоматически созданное описание

1. Слейте в основную ветку вторую тематическую. Если возникли конфликты слияния, разрешите их и завершите слияние.

Изображение выглядит как текст

Автоматически созданное описание

1. Удалите более не нужные тематические ветки. Обратите внимание в истории, что даже при удалении веток никакие коммиты не теряются.

Изображение выглядит как текст

Автоматически созданное описание

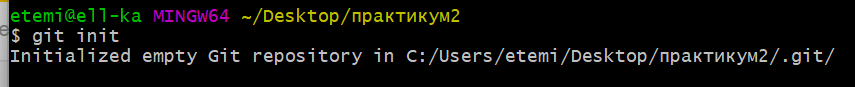
**2. Работа с Git в терминале**

**Цель работы**

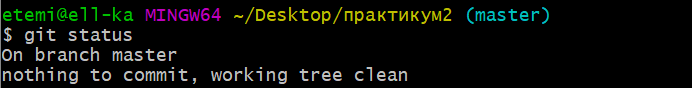
На основе уже полученных знаний о принципах работы СКВ получить более глубокое представление о работе Git при помощи команд терминала.

**Задания для выполнения**

1. Выберите тематику программы, которую собираетесь написать. Создайте для нее рабочую директорию
2. Инициализируйте в рабочей директории репозиторий при помощи команды git init.



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



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

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

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

Изображение выглядит как текст

Автоматически созданное описание

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

Изображение выглядит как текст

Автоматически созданное описание

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

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

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



1. Добавьте еще несколько файлов с исходным текстом программы.

Изображение выглядит как текст

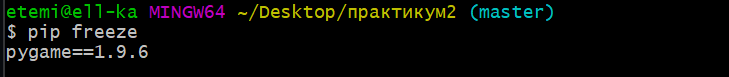
Автоматически созданное описание

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

Изображение выглядит как текст

Автоматически созданное описание

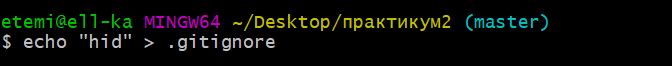
1. На всех стадиях работы пользуйтесь командой git status.
2. Инициализируйте в рабочей директории виртуальное окружение (Если вы пишите не на Python, то можете инициализировать какой-либо программный фреймворк, либо начать работать в IDE, которая создает скрытую папку с настройками в рабочем каталоге).



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

Изображение выглядит как текст

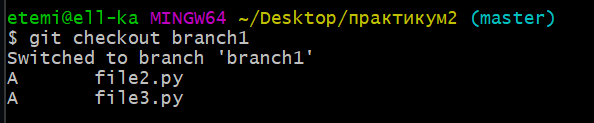
Автоматически созданное описание

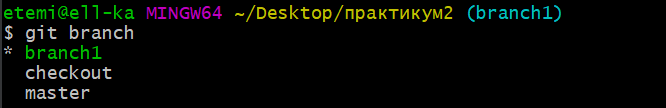
 Изображение выглядит как текст, монитор, экран, снимок экрана

Автоматически созданное описание

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

Изображение выглядит как текст

Автоматически созданное описание 



1. Сделайте несколько коммитов в основную и тематическую ветки.

**Изображение выглядит как текст

Автоматически созданное описание**

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

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

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.

**Контрольные вопросы**

1. Что такое удаленный репозиторий?

Удалённые репозитории — это модификации проекта, которые хранятся в интернете или ещё где-то в сети. Их может быть несколько, каждый из которых, как правило, доступен для вас либо только на чтение, либо на чтение и запись.

1. Где нужно вводить команды git?

В любую консоль\терминал. Я вводила в Git Bash

1. Для чего нужны ветки в системах контроля веток?

Ветки нужны для того, чтобы программисты могли вести совместную работу над проектом и не мешать друг другу при этом. При создании проекта, Git создает базовую ветку.

1. Как возникают конфликты слияния?

Обычно конфликты возникают, когда два человека изменяют одни и те же строки в файле или один разработчик удаляет файл, который в это время изменяет другой разработчик. В таких случаях Git не может автоматически определить, какое изменение является правильным. Конфликты затрагивают только того разработчика, который выполняет слияние, остальная часть команды о конфликте не знает. Git помечает файл как конфликтующий и останавливает процесс слияния. В этом случае ответственность за разрешение конфликта несут разработчики.

1. Как разрешать конфликты слияния?

Только сам разработчик вручную, который проводит слияние

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

**Цель работы**

Освоить основные навыки работы с облачными и распределенными системами контроля версий, получить навыки работы с инструментальными средствами, обеспечивающими командную работу над разработкой ПО.

**Задания для выполнения**

1. Зарегистрироваться на сайте github.com
2. Установить на компьютере программу Git
3. Форкнуть данный репозиторий в свой аккаунт

Изображение выглядит как текст, монитор, внутренний, экран

Автоматически созданное описание

1. Склонировать созданный удаленный репозиторий в директорию ~/git/test
2. На локальной машине пишем скрипт ~/git/test/backup.sh, с произвольным содержанием
3. Фиксируем скрипт в репозитории (делаем коммит)

Изображение выглядит как текст

Автоматически созданное описание

1. Обновляем удаленный репозиторий репозиторий (делаем пуш)



1. Через текстовый редактор добавить любую новую строку с комментарием
2. Сделать коммит



1. Внeсти синтаксическую ошибку в скрипт
2. Сделать коммит ошибочного скрипта



1. Откатываем до последней рабочей версии



Добавить несколько коммитов произвольного содержимого

1. Создать пулл реквест в данный репозиторий

**Контрольные вопросы**

1. Зачем нужен облачный хостинг репозиториев?

Для совместной удаленной работы в пределах глобальной сети интернет

1. Какими основными функциями обладает сайт github.com?

Создание репозиториев, веток, возможностью переносить и забирать репозитории с локальной машины, возможность предлагать изменения в чужие проекты

1. Как организовать командную работу над открытым проектом?

Например можно создать проект где будет ведущий разработчик а остальные будут работать параллельно в своих ветках, ведущий же разработчик будет сводить все в одно Дополнительные задания