ЛР 1. Знакомство с используемыми инструментами

# Вступление для тех, кто проходит работу без посещения вводной лекции

Идея курса - показать возможности

* информационных технологий в электроэнергетике,
* программирования в прикладных научных и инженерных задачах.

Лабораторные работы сфокусированы на

* обработке данных - это база цифровых энергетики;
* визуализации данных - это первый шаг к анализу;
* примерах кода на Python - этот язык программирования простой, универсальный и очень популярный для обработки и анализа данных;
* инструментах для совместной работы над кодом - это чтобы никто не говорил, что все сделал, а потом потерял флешку со всей работой.

## Ответы на часто задаваемые вопросы

1. Затем студентам не ИТ специальностей нужно программирование, почему нельзя просто использовать уже готовые программы?  
   **Ответ**

* Не бывает готовых программ на все случаи жизни.
* Минимальное понимание как пишутся программы позволяет лучше применять готовые инструменты.
* Даже небольшие навыки программирования могут очень сильно повысить продуктивность за счет автоматизации расчетов, математического моделирования, автогенерации графиков, диаграмм.
* Основа цифровых технологий - обработка данных. Без программирования невозможно обрабатывать большие объемы данных быстро и точно (даже в Excel, об это следующий вопрос).
* Изучение программирования дает возможность студенту не ИТ специальности начать карьеру в ИТ.

1. Почему нельзя просто использовать Excel для обработки данных?  
   **Ответ**

* Excel и его аналоги применять можно и нужно. Но не всякая задача обработки данных может быть продуктивно решена в Excel.
* Некоторые ситуации, когда табличные редакторы не подходят.
  1. Большой объем данных - табличные редакторы начинают тормозить, зависать и аварийно завершать работу.
  2. Нужно выполнять нестандартные действия с данными - в любом случае придется программировать или с помощью формул, или макросами. А уже если программировать, то лучше делать это в наиболее удобных инструментах.
  3. Задача не ограничивается обработкой данных. Например, нужно сделать что-либо из списка:
     + применить к данным сложный математический аппарат (вейвлет-анализ, нейронные сети и т.п.);
     + интегрировать решение в информационную систему, такую как сайт Интернет-магазина;
     + скачать данные из удаленных источников по определенному критерию;
     + найти в базе данных адреса e-mail людей и отправить по этим адресам сообщения;
     + на основе обработк цифрового сигнала посылать команды управления электрооборудованием.

1. Почему Python, если мы до этого изучали Matlab, или Делфи, или С, или Visal Basic?

**Ответ**

* Освоение программирования на начальном этапе должно быть мало привязано к конкретному языку, главное - понять основные принципы построения алгоритмов и их реализации.
* Изучение более чем одного языка программирования для начинающих полезно, потому что учит видеть отдельно алгоритм (непривязанный к языку) и реализацию алгоритма в коде.
* Автор курса считает Python наиболее подходящим для начинающих обучение по направлению “Цифровые технологии в энергетике” по совокупности критериев: простота, универсальность, востребованность в задачах обработки и анализа данных.

# 1. Цель работы

Познакомиться с инструментами, необходимыми для прохождения курса и полезными для дальнейшей работы в областях:

* развития алгоритмических навыков;
* развития навыков программирования;
* обработки, анализа, визуализации данных;
* машинного обучения;
* организации индивидуальной и коллективной работы над проектами.

Работу рекомендуется выполнять в бригаде из двух человек.

# 2. GitHub

GitHub (<https://github.com/>) - облачная платформа для размещения IT-проектов. “Облачная” потому что работает на удаленных серверах, это не программа, которую можно просто поставить на свой компьютер. “Платформа” потому что предназначена для решения многих задач. Символ GitHub - кот-осьминог (octocat), Рисунок 1.



Рисунок 1. Логотип GitHub

## 2.1. Актуальность GitHub для студентов

Для студентов IT-специальностей хотя бы поверхностное знакомство с системами контроля версий и платформами для совместной работы над проектами очень важно, так как этот навык требуется в любой профессиональной IT-компании. Привычка выкладывать свои проекты, пусть даже учебные на GitHub способна за несколько лет учебы создать портфолио, которое можно показать на собеседовании, чтобы показать, что начинающий свою карьеру специалист имеет определенные навыки.

Для студентов технических не IT-специальностей это менее важно, но, во-первых, использование таких платформ позволяет лучше организовывать работу над своими проектами в процессе обучения, во-вторых, минимальное владение GitHub часто требуется для прохождения онлайн-курсов, например, на Coursera практические задания во многих курсах нужно размещать на GitHub, чтобы их смогли проверить.

## 2.2. Возможности GitHub

GitHub используется для решения многих задач, основные перечислены ниже.

1. Контроль версий, хранилище файлов и возможностью отслеживать историю изменений. Так чтобы ни одна курсовая работа не исчезла бесследно при “случайном” форматировании диска и чтобы можно было не бояться испортить документ, переделав в нем какой-то раздел.
2. Совместная работа над проектами. В первую очередь над кодом IT-проектов. Чтобы в команде из 4-х человек не возникало ситуаций, когда Вася сделал что-то, переслал остальным, потом Петя половину изменил, Федя и Коля тоже что-то изменили, и теперь вместо одного файла есть его 4 разные варианта и никто не понимает, кто что менял и как это теперь объединить. В GitHub есть проекты, над которыми работают одновременно сотни людей.
3. Распространение своего проекта. Сделав что-то полезное или интересное можно разместить это на GitHub для общего пользования. Так можно получить признание или критику и найти в свой проект единомышленников. Многие компании выкладывают свои решения в так называемый open-source (весь программный код открыто публикуется) чтобы привлечь клиентов, партнеров и сотрудников. На GitHub можно найти проекты от Google, Facebook, Microsoft, Apple и других IT-гигантов.

Функционал GitHub постоянно расширяется. Например, с 2019 года стало возможно делать закрытые проекты, то есть такие, доступ к которым есть лишь у работающих над проектом людей. На GitHub можно бесплатно разместить свой собственный сайт (<https://pages.github.com/>), который будет доступен по ссылке вида https://user.github.io, где вместо будет имя вашего аккаунта.

Нужно различать понятия GitHub и Git. Первое - IT-компания и созданная ей облачная платформа. Второе - система контроля версий, задающая правила работы над проектами. GitHub использует Git, но Git существует сам по себе. Аналогично тому как сайты очень часто используют HTML для работы, но сам по себе HTML - это просто некие правила или язык разметки.

## 2.3. Инструкция и Задание 1

Задача - научиться создавать GitHub репозитории, работать с файлами в нем индивидуально и в команде.

1. **Создать индивидуальный GitHub аккаунт** на сайте <https://github.com/>. Нужно придумать для аккаунта более-менее осмысленное и приличное имя и указать почту, которой вы пользуетесь и доступ к которой вряд ли будет потерян.   
     
   Свой аккаунт должен создать каждый член бригады. При выполнении работы на одном компьютере можно сделать шаги 2-8 одному члену бригады, затем выйти из аккаунта и сделать эти же шаги другому.
2. **Создать открытый репозиторий**. Каждый создает свой репозиторий.  
   Репозиторий - это место для хранения и управления файлами проекта. Для создания репозитория нужно нажать на плюс в верхнем правом углу на сайте github.com и выбрать пункт “New repository” (Рисунок 2).

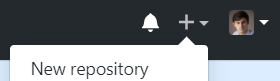


Рисунок 2. Пункт для создания репозитория

Далее нужно ввести имя репозитория в поле “Repository name” и описание в поле “Description”. Например: “My\_First\_Test\_Repo” и “Это мой первый учебный репозиторий”. Указать тип репозитория “Public” и отметить галочкой пункт “Initialize this repository with a README”, как показано на Рисунке 3. После этого ваш репозиторий будет доступен по адресу [https://github.com/имя-вашего-аккаунта/имя-вашего-репозитория](https://github.com/%D0%B8%D0%BC%D1%8F-%D0%B2%D0%B0%D1%88%D0%B5%D0%B3%D0%BE-%D0%B0%D0%BA%D0%BA%D0%B0%D1%83%D0%BD%D1%82%D0%B0/%D0%B8%D0%BC%D1%8F-%D0%B2%D0%B0%D1%88%D0%B5%D0%B3%D0%BE-%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D1%8F). Например, <https://github.com/Pavel-V/Test_Guide>.

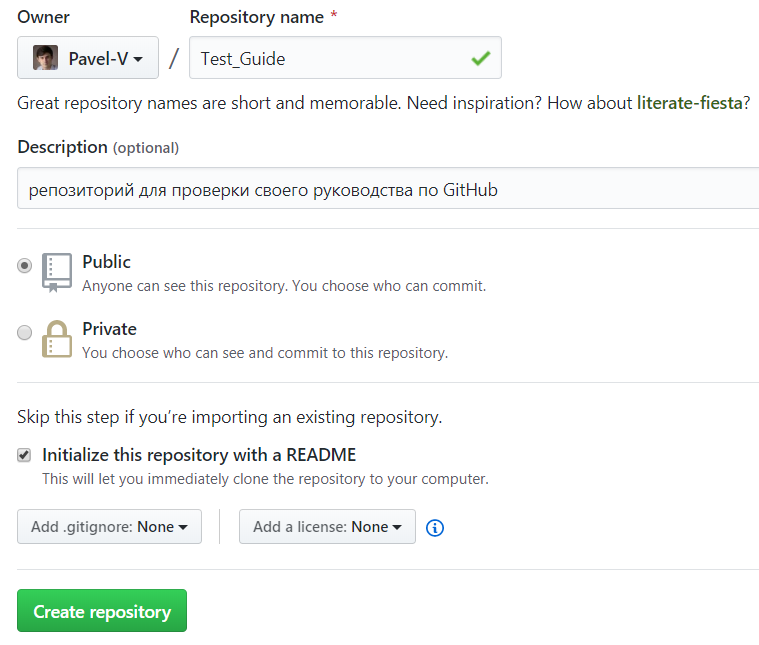


Рисунок 3. Создание репозитория

1. **Отредактировать файл README.md**. Указать группу и фамилии (без имен) всех членов бригады. Каждый в своем репозитории.  
     
   Опытные разработчики используют для работы с репозиториями специальные инструменты, обычно из терминала командной строки. Так что вся работа делается путем отправки команд. Но для начинающих это бывает непривычно, поэтому все работа будет вестись через web-интерфейс.  
     
   В своем репозитории нужно щелкнуть на иконку карандаша, как показано на Рисунке 4.

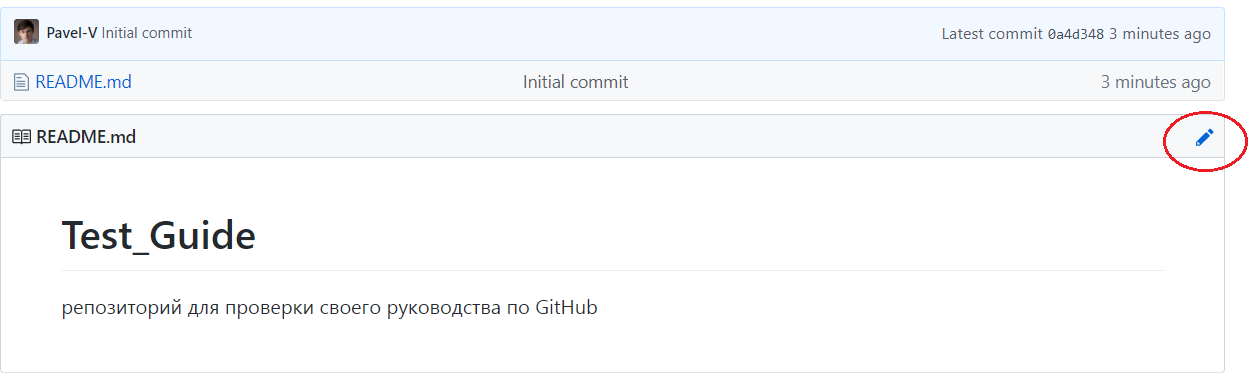


Рисунок 4. Редактирование файла в репозитории

Откроется окно для редактирования файла. Файлы README по умолчанию имеют формат MD - Markdown. Это специальный язык для упрощенного форматирования текстов. Более подробно он будет рассмотрен позже. Сейчас достаточно знать, что заголовки помечаются решеткой, а обычный текст пишется просто как в блокноте, только для перехода на новую строку нужно поставить пустую строку. В итоге должно получиться примерно как на Рисунке 5.

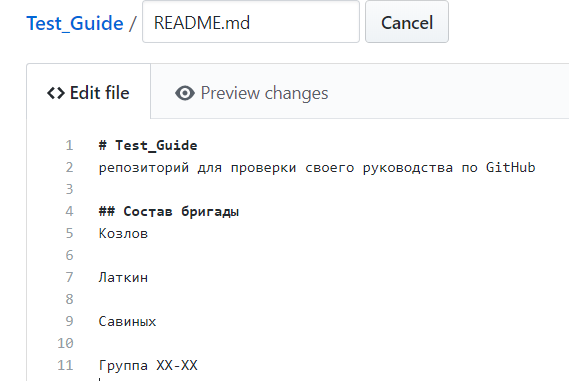


Рисунок 5. Редактирование README

Можно посмотреть результат, нажав на “Preview changes”.

1. **Сделать коммит**. Каждый в своем репозитории.

Чтобы сохранить изменения в файле README, нужно сделать коммит. Коммит (от англ. *commit* - зафиксировать) - это внесение изменений в содержимое репозитория, разработчики говорят “закоммитить”. У коммита должен быть комментарий, по которому можно потом быстро понять, что было в нем сделано. Пример показан на Рисунке 6. После ввода комментария нужно нажать “Commit changes”.

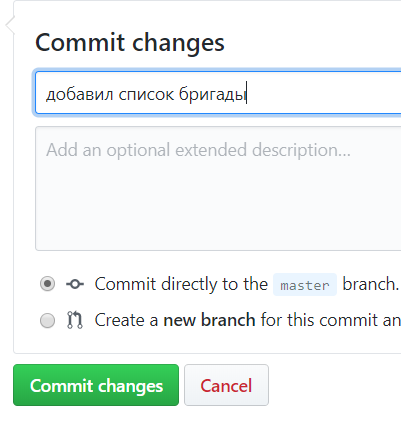


Рисунок 6. Коммит

После этого произойдет возврат на главную страницу репозитория и вы увидите внесенные изменения.

1. **Создать новую ветку.**  Каждый в своем репозитории.В работе над IT-проектам часто разработка делается не линейно шаг за шагом, а с параллельными процессами и ответвлениями, как показано на Рисунке 7.



Рисунок 7. Ветвление

Master - это основная, главная ветка. Чтобы работать над отдельной частью проекта, не мешая другим членам команды, принято создавать новые ветки, вести работу в них, а потом соединять с основной, как показано в синей ветке на Рисунке 7.

Но даже если над проектом работает один человек, удобно создавать ветки, чтобы попробовать в них нечто экспериментальное. То есть такие изменения, которые возможно будут полезны, а возможно вообще не получатся. Если все пройдет удачно, то можно будет добавить новое в основную ветку (оранжевая ветка на Рисунке 7). Если нет - просто оставить как тупиковую ветку (красная ветка на Рисунке 7).  
  
Благодаря этому сильно снижается риск внесения изменений, которые что-то испортят в основной ветке. Конечно, можно просто вернуться назад и взять версию до того, как она была испорчена. Но тогда, если были внесены и полезные изменения, то они тоже пропадут. Поэтому лучше использовать отдельные ветки.

Создание ветки показано на Рисунке 8.

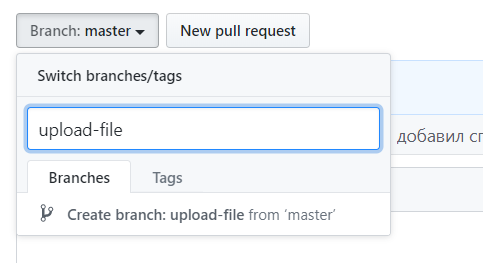


Рисунок 8. Создание ветки

Теперь с помощью раскрывающегося списка “Branch” можно выбирать, в какой ветке работать. Сейчас нужно выбрать новую созданную ветку.

1. **Загрузить каталог с файлом**. Каждый в своем репозитории.  
   Если бы все файлы нужно было создавать и редактировать прямо на сайте GitHub, это было бы совсем неудобно. GitHub позволяет работать с файлами на своем компьютере, а при необходимости отправлять (загружать) их в репозиторий или брать (скачивать) их на свой компьютер. Причем при получении некоторого опыта работы с системой контроля версия Git это делается очень легко без необходимости даже открывать сайт GitHub. Но для начального этапа лучше сделать более медленным способом, чтобы разобраться.  
     
   Для загрузки файлов можно нажать кнопку “Uploads files” на главной странице репозитория. Чтобы открыть главную страницу, достаточно перейти на панель Code, как показано на Рисунке 9 вверху, а если эта панель уже открыта, то нажать на ссылку с именем репозитория, как показано на Рисунке 9 внизу.

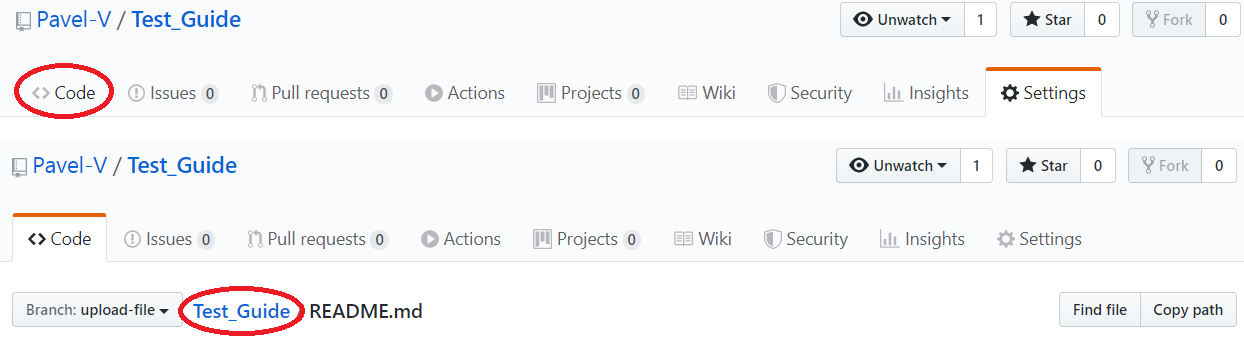


Рисунок 9. Переход на главную страницу репозитория

После перехода на главную можно просто мышкой перетащить файлы из проводника в окно браузера. Причем так можно загрузить сразу целый каталог со всем содержимым. После этого ввести комментарий к коммиту в поле и сделать коммит (Рисунок 10).  
  
Таким образом можно, работая над проектом на своем компьютере, время от времени переносить изменения в GitHub.

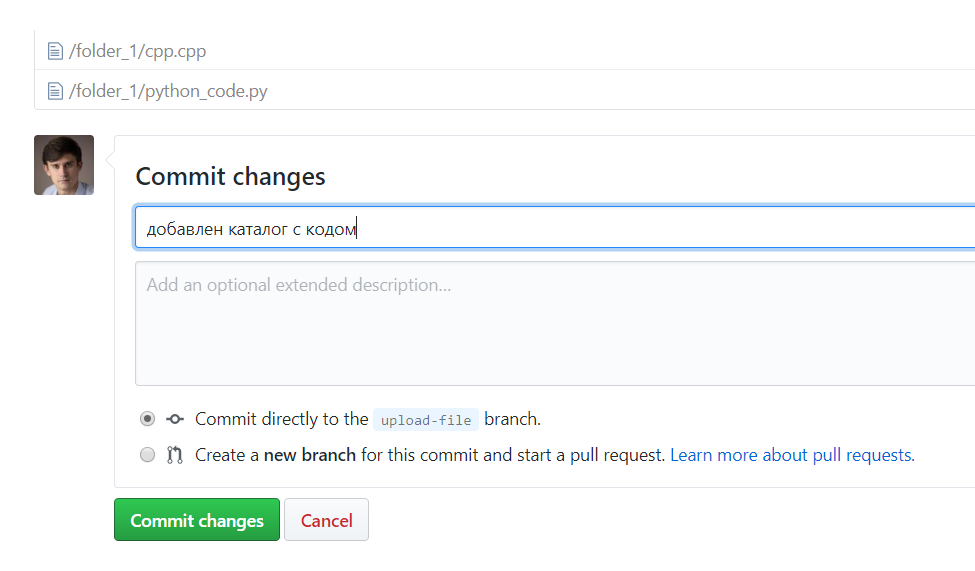


Рисунок 10. Загрузка файлов

Создайте в компьютере каталог с каким-либо файлом и загрузите в свой репозиторий. Например, можно взять код программ, созданных в лабораторных по курсу “Информатика”.

1. **Сделать слияние новой ветки в мастер**. Каждый в своем репозитории.  
   Нужно перейти на главную страницу своего репозитория, нажать на “branches” - откроется страница с ветками, затем нажать “New pull request” у вашей ветки (Рисунок 11 показывает вместе эти два действия, они делаются по-очереди).

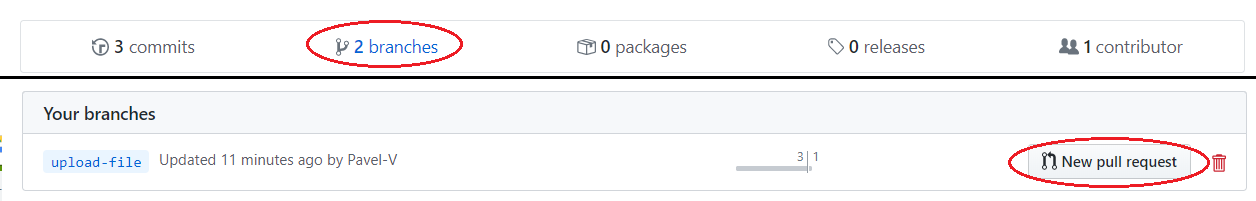


Рисунок 11. Слияние веток

Далее нужно выбрать в какую ветку внести изменения, в данном случае - в мастер. Затем нажать “Create pull request”.

После этого будет выполнен переход на новую страницу, на которой нужно подтвердить слияение (объединение) веток, нажава “Merge pull request”, потом “Confirm merge”.

Если теперь перейти на главную страницу репозитория (https://github.com/имя\_вашего\_аккаунта/имя\_вашего\_репозитория) и выбрать ветку master, то вы увидите, что в ней появились нужные файлы.

1. **Дать доступ к своему репозиторию всем членам бригады**.Каждый в своем репозитории.  
   Созданный репозиторий публичный, так что любой может видеть его содержимое. Но вносить в него изменения могут лишь те, кому дал это право владелец репозитория. Для этого нужно открыть настройки репозитория (“Settings”) и в меню выбрать пункт “Manage access” в нем нажать кнопку “Invite a collaborator” и указать адреса или логины членов своей бригады. Они получат на почту уведомление о приглашении и смогут вносить изменения в ваш репозиторий.
2. **Добавить свои имена в файлы README.md**. Каждый должен добавить только свое собственное имя к своей фамилии в каждом репозитории бригады. В соем репозитории имя можно добавить просто изменив README.md файл в master-ветке. В чужом репозитории нужно создать новую ветку add-name-***name***. Например, бригада Андрей Козлов, Кирилл Латкин. В файле README только фамилии. Андрей создает в репозитории Кирилла ветку add-name-andrey и добавляет свое имя в README. Кирилл создает в репозитории Андрея ветку add-name-kirill и добавляет своем имя.
3. **Соединить изменения в master-ветку**. Каждый в своем репозитории делает слияние веток для добавления имен с master-веткой. В итоге у каждого члена бригады в репозитории должен быть список имен и фамилий в README файле.
4. **Посмотреть историю изменений. Сделать скриншот для отчета.**Для просмотра истории всех изменений репозитория нужно на главной странице нажать на “commits”, как показано на Рисунке 12.

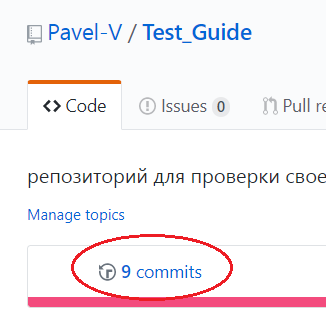


Рисунок 12. Коммиты

Пример показан на Рисунке 13. У каждого коммита есть свой уникальный номер - хэш, на рисунке хэши обведены красным. Если нажать на него, то можно увидеть изменения, внесенные данным коммитом.

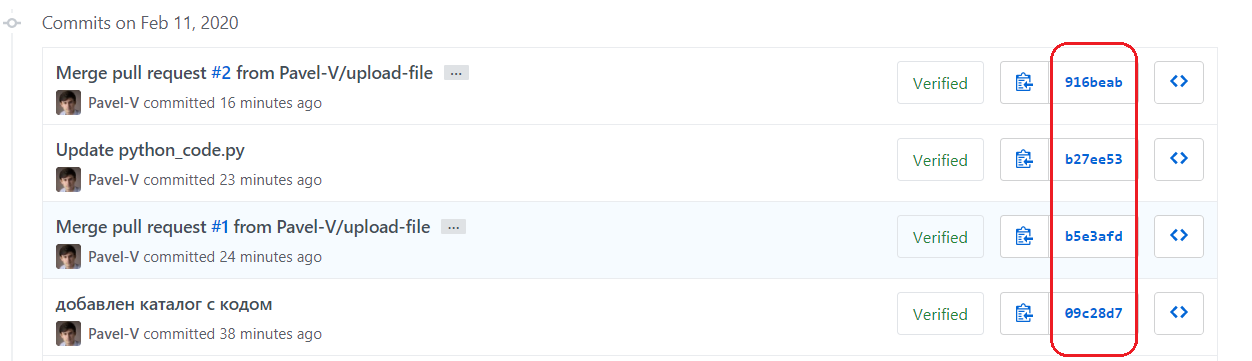


Рисунок 13. Список коммитов.

В начале работы с GitHub кажется, что нужно тратить очень много времени на эту систему. Со временем работа с Git входит в привычку и все делается быстро, к тому же есть инструменты для упрощения работы, такие как GitHub Desktop, Git GUI, Git Bash.

# 3. Google Colab + Jupyter

Google Colab - облачная платформа для создания и распространения решений в области обработки и визуализации данных, машинного обучения и просто программирования на языке Python.

## 3.1. Актуальность Google Colab для студентов

Во-первых, Google Colab удобен для студентов, чтобы начать программировать на Python и решать задачи связанные с данными, так как не требует ничего устанавливать и настраивать на своем компьютере и позволяет работать над проектами и на своем личном компьютере дома, и на разных компьютерах университета, и на компьютерах одногруппников и т.д. без необходимости что-то копировать и пересылать.

Во-вторых, Google Colab имеет ряд достоинств, которые привлекают к нему и начинающих специалистов, и профессионалов в математическом моделировании и Data Science.

## 3.2. Возможности Google Colab

Общее представление о Google Colab можно получить из видео “Get started with Google Colaboratory” (<https://www.youtube.com/watch?v=inN8seMm7UI>). Видео на английском языке, можно включить субтитры на английском или в крайнем случае включить автоматический перевод на русский.

На сайте Google Colab есть примеры для начинающих <https://colab.research.google.com/notebooks/intro.ipynb#scrollTo=5fCEDCU_qrC0>. Для работы с ними нужен Google аккаунт.

Работа в Google Colab построена на принципе блокнота Jupyter Notebook. Суть этого принципа в том, что программный код, пояснения к нему и результаты работы помещаются в одном пространстве, разбитом на ячейки (cells), пример показан на Рисунке 14. Такая логика особенно удобна для программирования в задачах моделирования, анализа данных, научных исследованиях, потому что все оказывается на одном экране. В то время как классический подход к созданию программного обеспечения подразумевает написание кода в одних файлах, документации в других, а для получения результатов работы нужно запустить созданную программу.

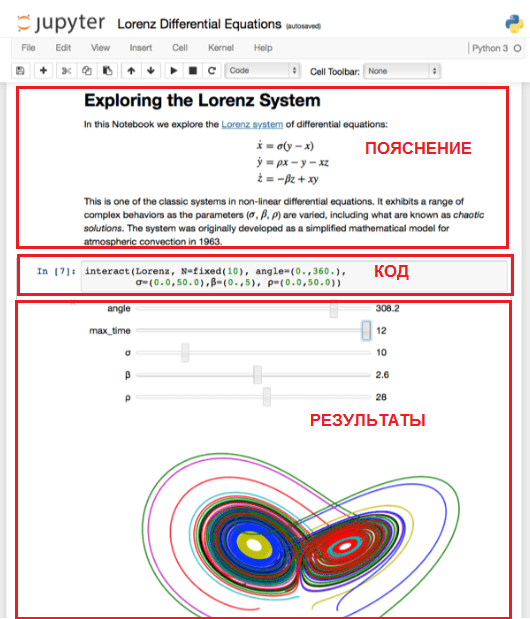


Рисунок 14. Пример Jupyter блокнота с jupyter.org

Возможности и достоинства Google Colab.

* Его не требуется устанавливать на свой компьютер и при этом можно сразу получить доступ к огромному количеству программных библиотек для решения задач от математических расчетов до распознавания видео.
* Функции для совместной работы над проектами.
* Созданный в Google Colab проект (блокнот) можно передать другому человеку для использования просто по ссылке.
* Возможность повышения скорости расчетов за счет использования предоставленной GPU видеокарты.
* Трудоемкие вычисления выполняются не на компьютере пользователя, а в облаке.
* Интеграция с Google Drive для работы со своими файлами.

Недостатки:

* требуется выход в Интернет;
* меню и документация только на английском (но технический английский является обязательным стандартом для IT-специалистов в любой стране);
* нет возможности работать с файлами и программами на своем компьютере, так как все будет исполнятся в облаке).

## 3.4. Инструкция и Задание 2

Задача - научиться создавать проекты (блокноты) в Google Colab, создавать описания на языке разметки Markdown и делать из Google Colab коммиты в GitHub.

1. **Создать Google-аккаунт**, если его нет. Достаточно у одного из членов бригады.
2. **Посмотреть видео и примеры** по ссылкам, указанным в пункте [3.1.](#_dshk9qx0b0ef)
3. **Создать один блокнот в Google Colab** на бригаду, в нем сделать ячейку Markdown c заголовком, маркированным списком, рисунком, таблицей и ссылкой на сайт. Содержание произвольное. Пример показан на Рисунке 15. Инструкцию или примеры для работы с Markdown нужно найти самостоятельно.

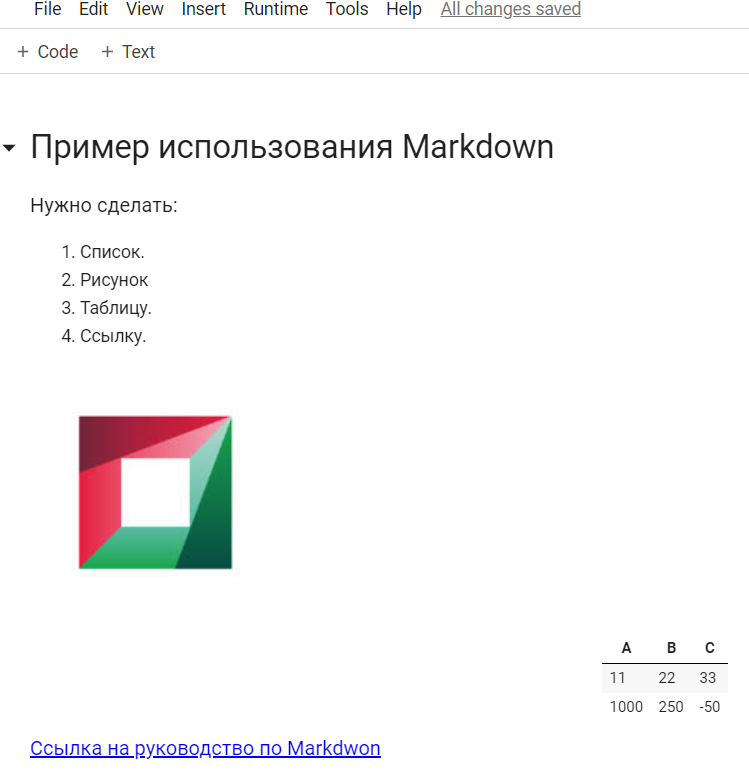


Рисунок 15. Пример ячейки с Markdown

1. **Отметить получившуюся версию и дать ей имя**.  
   В Google Colab есть своя простая система контроля версий. В меню нужно выбрать “File”, в выпавшем списке “Save and pin revision”. Это действие сохранит текущую версию блокнота с пометкой. Попробуйте внести какие-либо изменения и снова сохранить новую версию. Затем если еще раз выбрать “File” и потом “Revision history”, то откроется список версий, пример на Рисунке 16. В этом списке можно давать имена версиям (на рисунке дано имя “v.1.0”, можно отметить две версии и посмотреть что в одной изменилось относительно другой.

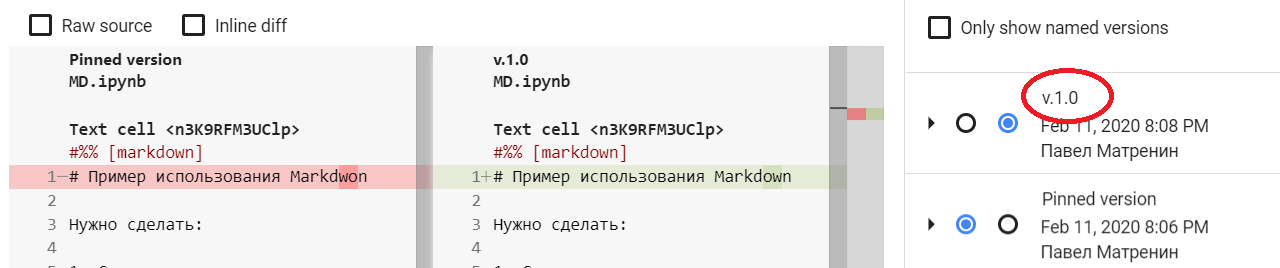


Рисунок 16. Версии в Google Colab

1. **Дать к блокноту доступ преподавателю** (для этого узнать адрес его почты, привязанной к Google-аккаунту, вероятно это будет pavel.matrenin@gmail.com).
2. **Добавить блокнот в созданный ранее GitHub аккаунт.**  
   Несмотря на наличие своей системы контроля версий, создатели Google Colab понимают, что разработчики очень любят GitHub и поэтому добавили возможность быстрой пересылки блокнота в GitHub репозиторий и обратно. В меню “File” нужно выбрать пункт “Save a copy in GitHub...”, в открывшемся окне нужно ввести свой GitHub аккаунт и согласиться дать доступ к нему. После чего можно будет выбрать свой репозиторий на GitHub, выбрать ветку (можно master) и сделать коммит из Google Colab в GitHub, как показано на Рисунке 17.

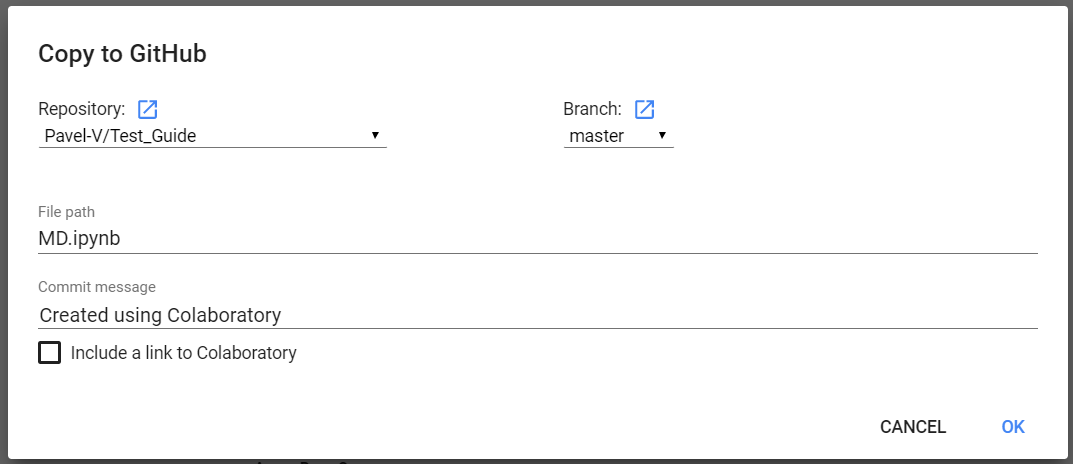


Рисунок 17. Коммит из Google Colab в GitHub

После этого откройте свой GitHub репозиторий и убедитель, что в него добавился ваш блокнот.

# 4. Отчет один на бригаду

В Google документах, MS Word или аналогах.

* Титульный лист.
* Цель работы.
* Имена созданных аккаунтов и репозиториев с указанием где чей. Пример в Таблице 1.

Таблица 1. GitHub аккаунты и репозитории членов бригады.

| **Аккаунт** | **Репозиторий** | **Студент** |
| --- | --- | --- |
| petr-ivanov | My\_test\_repo (ссылка на репозиторий на github) | Петр Иванов |

* Скриншот истории изменений репозитория (достаточно одного репозитория любого из бригады).
* Скриншот файла итогового вида файла README.md (достаточно одного репозитория любого из бригады).
* Код из ячейки на языке разметки Markdown, созданного в Google Colab.
* Скриншот внешнего вида блокнота, созданного в Google Colab и открытого через GitHub.

**Отчет в формат DOCX после проверки преподавателем и защиты прикрепить в DiSpace к мероприятию ЛР1. Каждый студент бригады прикрепляет отчет через свой аккаунт**