

ВАЛЕНТИН ЮЛЬЕВИЧ АРЬКОВ

---

# Цифровые технологии и искусственный интеллект

УЧЕБНОЕ ПОСОБИЕ



Валентин Юльевич Арьков

# **Цифровые технологии и искусственный интеллект**

**Учебное пособие**

Шрифты предоставлены компанией «ПараТайп»

© Валентин Юльевич Арьков, 2024

Если к любому названию добавить слово «цифровые», получается что-то новое, заманчивое. Обещающее хорошую зарплату. А если к цифровизации добавить искусственный интеллект, то зарплата должна подняться до неприличных высот. Что скрывают от нас цифровизаторы и оптимизаторы? Как выглядит под капотом тот самый искусственный интеллект, которому почему-то «нет оснований не доверять»? Разбираемся и удивляемся.



Создано в интеллектуальной издательской системе Ridero

# ОГЛАВЛЕНИЕ

[Цифровые технологии и искусственный интеллект](#)

[Картинка на обложке](#)

[Введение](#)

## [1. Конструирование запросов](#)

[Поисковый запрос](#)

[Оформляем отчёт](#)

[Запрос к чат боту.](#)

[Программируем без программирования](#)

[Опыты с запросами](#)

[Ссылки](#)

## [2. Markdown](#)

[Введение](#)

[Оформление отчёта](#)

[Установка редактора](#)

[Markup и Markdown](#)

[Основные конструкции Markdown](#)

[Ещё о разметке](#)

[Markdown в веб-сервисах](#)

[Docker + Anaconda](#)

[Дополнительные инструменты форматирования](#)

[Формулы TeX](#)

[Заключение](#)

[Ссылки](#)

## КАРТИНКА НА ОБЛОЖКЕ

Котик на обложке нашего пособия сгенерирован с помощью нейросети Kandinsky от Сбера. В соответствии с правилами использования этого сервиса, исключительные права на использование сгенерированного контента принадлежат автору промпта — то есть запроса к нейронке.

Кстати, вот и сам запрос:

Пушистый котик работает за компьютером с большим экраном, нажимает лапками на клавиатуру, смотрит в экран и держит в зубах компьютерную мышь.

Как оказалось, не каждый котик может взять компьютерную мышку за хвостик. Тут будет над чем поработать.

# ВВЕДЕНИЕ

В рамках лабораторных работ мы будем знакомиться с возможностями интеллектуальных систем и сразу же будем рассматривать практические примеры применения этих возможностей. Область искусственного интеллекта охватывает самые разные методы и инструменты. На сегодняшний день чаще всего говорят о нейросетях. Вот эту тему мы и будем обсуждать на протяжении нескольких занятий.

Материалы по нашему курсу будут постепенно появляться на страничках проекта на GitHub и GitVerse.

Каждое занятие — это ещё один полезный навык. За считанные часы вы научитесь и сможете

- грамотно составлять подробные запросы к нейросети
- создавать персонального интеллектуального бота в Телеграм
- работать с файлами Markdown в Colab и на GitHub

# 1. КОНСТРУИРОВАНИЕ ЗАПРОСОВ

В этой работе мы рассматриваем основные правила построения запросов к нейронной сети. Контент, который будет сгенерирован в ответ на наш запрос, зависит от того, насколько подробно и тщательно мы сформулировали наше задание. Один из самых популярных интеллектуальных инструментов на сегодняшний день — это большие языковые модели — Large Language Models (LLM). Они действительно большие, в буквальном смысле слова, потому что внутри содержат миллиарды параметров (коэффициентов). При обучении таких моделей подбирают значение этих коэффициентов, а для этого используют огромное количество материалов, в основном, взятых из интернет.

Интеллектуальные системы, которые создают новые объекты (текст, изображение, звук, видео и так далее), в настоящее время обычно называют генеративным искусственным интеллектом (Generative Artificial Intelligence, Gen-AI). Технология составления заданий для таких систем называется конструирование запросов (Prompt Engineering).

Интеллектуальные системы — это не какая-то очередная заумная, абстрактная теория. Это практичные, работающие инструменты, причём иногда даже неплохо работающие инструменты. Так что в процессе знакомства с искусственным интеллектом мы будем не просто искать информацию в интернете, но и сразу же применять полученные знания.

Если где-то в интернете, на каком-то сайте что-то написано, наша задача будет проверить и убедиться в этом лично. Одно дело, когда нам советуют: «Запрос к нейросети надо писать вот так...» Мы

посмотрим на эту инструкцию и сразу же проверим, насколько это правда. Насколько это полезно и удобно.

Всё, что пишут в интернете, пишут люди. А в последнее время даже и не люди, а нейросети или боты. И нет здесь никакой ответственности — за правильность и за результаты.

Специалисты по информационной безопасности предупреждают и объясняют... Проблема в том, что интернет — это не то же самое, что водопровод. Если открыть водопроводный кран, из него потечет вода. И во многих наших городах её даже можно пить — прямо из крана. Есть города, где сначала надо бы её прокипятить. Так вот, из интернет «потечёт» всё, что угодно. Может потечь питьевая вода, а может потечь какая-нибудь отравка. И когда мы выходим в интернет, мы сталкиваемся с тем, что не всем и не всему можно верить. Наша задача научиться отличать — где правда и где неправда. И сразу же проверять. Как говорится, доверяй, но проверяй.

Кстати говоря, поскольку большие языковые модели обучают на материалах из интернета, они осваивают всё — в том числе вредное, неправильное и нежелательное. Это примерно как учить ребёнка читать по надписям на заборе. После такого «обучения» он будет удивлять своих родителей и радовать сверстников. Поэтому при обучении нейросетей есть дополнительный этап, на котором сгенерированные ответы проверяют специально подготовленные люди. Это нужно, чтобы обучить нейросеть «хорошим манерам».

В рамках этого занятия нас будет интересовать такой вопрос: как писать запросы к нейросети? Запросы по-английски называются промты или промпты — prompt.

## **ПОИСКОВЫЙ ЗАПРОС**

Первое действие очень простое. Открываем поисковую машину, например, Яндекс. А потом Google. И задаем ей такой вопрос: «Как писать запросы к нейросети?» Естественно, на экране появится



множество ссылок. Мы просматриваем хотя бы первую страницу и пытаемся перейти по некоторым ссылкам, которые нам приглянулись. Наша цель — вначале понять основную идею, посмотреть на основные моменты — что нам советуют. Мы должны просмотреть результаты поиска и извлечь для себя основные, ключевые моменты. В интернете мы находим много всякого текста, но наша задача извлечь из этого что-то полезное.

Затем мы возвращаемся к началу. А зачем мы это спрашивали? Какая была конечная цель? И тогда можно будет наш запрос постепенно уточнять.

Задание. Используя поисковые машины Яндекс и Google, сформируйте список основных правил составления запросов к нейросети. Это будет начало нашего отчёта. Отчёт у нас будет в электронном виде.

## **ОФОРМЛЯЕМ ОТЧЁТ**

Попутно мы с вами проходим весь цикл работы. От начала до оформления результата.

По каждой работе мы составляем неформальный отчёт. В этом отчёте мы описываем то, что мы сделали и что нам удалось узнать.

Результаты будем оформлять тоже в облаке. Как вы знаете, есть облачные офисные продукты. Есть документы Google. Есть аналогичные отечественные сервисы, например, документы Яндекс — в них и будем работать. Опять же, это не самый лучший и не самый худший вариант. По крайней мере, отечественный. И есть надежда, что он не заблокируется и не исчезнет в ближайшее время.

В отдельной вкладке открываем Яндекс. Документы. Для этого понадобится учётная запись Яндекса — такая же, как для почты Яндекса. Создаём новый документ. Интерфейс чем-то напоминает Word. Вот в этом облачном документе нам и предстоит начать писать отчет по лабораторной работе. Затем мы можем просто передать

ссылку на отчёт и не пересылать сам файл: Поделиться – Просмотр – Скопировать ссылку.

Вначале мы создаём новый документ. Как его назвать? Обычное решение: «лаба», или «отчет», или «мой отчет». Это хорошо, если такой документ у нас один-единственный. Теперь представим себе два-три таких файла с названием «Отчет» и «мой отчет»... Становится понятно, что в названии должен быть смысл. А ещё у нас могут быть разные дисциплины, а по ним могут быть разные виды занятий, так что «отчеты» могут быть разные.

Теперь подумайте, какие сведения в названии будут самые главные? То, что это отчет? Или то, что это «лаба»? И если вы этот отчет кому-то еще отправляете, как его отличить от остальных, от 20 – 30 таких же документов – тоже с названием «отчет»?

Напомним, что отчет – это такое произведение, у которого есть название и автор. Так что, на самом деле, разница будет в названии предмета и самого документа. Ещё у нас есть фамилия автора документа – конкретного студента. У нас редко бывает одинаковые фамилии в одной группе – тогда добавим инициалы. И, конечно, есть номер группы. Вот это ключевые моменты для названия документа. Теперь расположим их в порядке важности: фамилия, группа, название предмета. Название предмета можно сокращенно. Можно указать номер лабораторной работы. Получаем что-то вроде «Иванов-ЛР1-ЦТИИ-ИБТ123».

В нашем ВУЗе есть некоторые традиции – как оформляют отчеты, курсовые и прочие документы. Вначале идет особая, первая страничка. У книги бывает обложка, а у отчёта бывает титульный лист.

Итак, документ начинается с титульного листа. Есть такое понятие, титульный лист. Это первый лист документа, на котором говорится, что это за произведение такое такое, на какую тему это произведение и кто его автор. Эту традицию вы обнаружите во многих учреждениях.

На самом верху титульного листа пишется название министерства, к которому мы относимся. По какому министерству или ведомству мы с вами проходим. Попробуйте выяснить в интернете: Министерство чего? Кому подчиняется наш университет? Можно даже на сайте вуза попытаться это обнаружить. Наверное что-то, связанное с образованием. Вот эта фраза должна быть первой строчкой титульного листа.

Теперь, когда мы начинаем печатать, нужно сразу выбрать размер шрифта. Какой высоты будут наши буквы, чтобы потом это другой человек ещё и смог прочитать... Например 14 или 16 пунктов – кому как нравится. Слишком мелко не надо, слишком крупно тоже не надо.

Периодически название министерства меняется. Ваша цель проверить, правда ли, что оно теперь называется Минобрнауки. Находим сайт и убеждаемся, что это действующий сайт и что это действующее министерство. Находим, проверяем. То, что мы с вами делаем, – это мы проверяем информацию. В области нейросетей это действие называется «факт-чекинг» – fact checking. Check – это проверка. Мы проверяем факты – facts.

Вторая строчка – это название нашего учебного заведения. Как оно правильно называется? Обычно на главной веб-страничке указывают полное название вуза – и мы пишем полное, а не сокращённое.

Дальше у нас должно быть указано название кафедры, которая проводит занятия. Например, если вы проходите занятия на кафедре иностранных языков, а учитесь на кафедре АСУ, мы пишем кафедра иностранных языков. Если занятие проводит кафедра АСУ, тогда мы пишем кафедра АСУ.

И только после этого мы уже начинаем объяснять, а что это за документ. И документ этот – отчет по лабораторной работе. Не лабораторная номер один, а отчет по лабораторной работе №1. Потому что кроме отчета бывают и другие документы.

Дальше — тема нашего занятия. Слово «тема» писать не обязательно. А вот название лабораторной работы хорошо бы указать. Для нашей текущей работы темой будет «Конструирование запросов».

По-английски это «пром프트-инжиниринг» — prompt engineering.

Название предмета тоже надо куда-то вставить.

Видимо, после строчки «отчет по лабораторной работе» нужно будет сказать по какой дисциплине. Как у нас предмет называется? Длинное красивое название, мы указываем название нашего предмета. У кого-то лабораторные по физике, у кого-то по химии, а у нас по нашему предмету.

Затем мы указываем группу, фамилию и инициалы студента.

В самой нижней строчке — город и год.

Когда мы справились с оформлением титульного листа, можно переходить к работе по существу. Это внутреннее содержание отчета. Нас в этой работе что вы делали и что вам удалось выяснить. То есть какой запрос вы отправили и куда. Какие ключевые, полезные моменты вы для себя вынесли.

Действие первое было — поисковая машина Яндекс. Как вы сформулировали запрос к поисковой машине? Какие полезные советы вы для себя определили? Здесь полностью переписывать не нужно. Здесь ценность в том, что вы через себя это пропускаете. Какие полезные правила составления запросов вы для себя обнаружили? Нам нужно свои мысли отразить.

На всякий случай поясним. Зачем мне нужно эти бумажки писать, эти документы и отчёты составлять? В любой работе вам придется составлять какие-нибудь документы, какие-нибудь тексты. Пусть даже заявление на отпуск или о приеме на работу. Или резюме. А в рамках нашей профессии — информатика и вычислительная техника — приходится разные документы составлять.

Здесь художественный талант не требуется. Здесь нужно очень коротко написать, чего вы сделали и что вы получили. Кому

отправили запрос, как звучал запрос и что вы для себя из этого вынесли.

## **ЗАПРОС К ЧАТ БОТУ**

Второе действие. Мы можем спросить и у самой нейросети то же самое, что мы пишем в поисковой машине.

Открываем на новой вкладке Яндекс GPT — просто в качестве примера. Мы не говорим, что этот сервис лучше всех или хуже всех. Это просто очередной инструмент. Причём отечественный. На главной странице сервиса читаем описание и смотрим просветительские ролики. Просвещаемся. И вот мы задаём ему вопрос — точно такую же фразу. Фиксируем в отчёте результаты.

Берём следующий инструмент: GigaChat от Сбера. Тоже просвещаемся по материалу главной страницы. Задаем ему тот же самый запрос: «Как писать запросы к нейросети?» Можно работать без регистрации. Можно войти с авторизацией для сохранения истории запросов. Чтобы войти в гигачат, потребуется учетная запись или SberID по номеру телефона. В отчете тоже нужно отразить, что нового узнали.

По каждому полезному совету подумайте и приведите примеры. Например, «контекст». В каком контексте нам нужен ответ? Придумайте пример, как объяснить это другому человеку. Это может быть описание проблемы, которую мы решаем: мы хотим научиться писать качественные запросы и для этого выясняем, какие есть советы и рекомендации, какие есть элементы/составные части в таких запросах и т. д.

По стилю и форме ответа тоже можно сформулировать требование: «Объясни простым языком, как для пятилетнего ребенка» — и она объяснит максимально простым языком. Одно дело — скопировать, скачать страничку текста и совсем другое дело — объяснить своими словами.

Можно встретить совет избегать отрицаний, отказаться от слов «не», «без», «кроме» и им подобных. Так что можно сказать: «Мужчина без волос». Или можно сказать: «Лысый». Как сказать другими словами, что у человека нет бороды? Каким-то позитивным утверждением выразить ту же самую идею.

Роль, поведение. Как звучит пример назначения роли? «Ты опытный программист» или «Действуй как редактор».

Итак, вот у нас уже есть контекст, у нас есть избегание отрицания, у нас есть роль. Мы на эти моменты посмотрели.

Теперь вернемся к нашему запросу: «Как писать запрос к нейросети». Уточним наш запрос. Он должен быть более конкретным. Сегодня мы в основном будем работать с вами с текстом, в ответах всплывают полезные советы про картинки. Поэтому в следующих запросах мы уже будем говорить по поводу работы с текстом, а не про запросы вообще. С учетом того, что мы узнали, попробуйте сформулировать более подробный, более детальный запрос.

Переходим к следующему инструменту, который мне очень даже нравится. Называется Perplexity. Вводим наш запрос и изучаем ответ. В процессе работы мы выясняем, что в технологии написания запросов есть расплывчатые полезные советы, а есть конкретный список элементов, из которых этот запрос будет состоять, если мы хотим генерировать текст. Какие элементы должны быть в запросе к нейросети? Сделайте более подробный запрос, чтобы получить более осмысленный список. Мы можем очень чётко сформулировать, что именно мы хотим. Из каких элементов должен состоять запрос? Или: какие полезные советы могут пригодиться при составлении запроса? Если мы говорим: «Составь список полезных советов», — мы получим именно список советов.

Мы должны объяснить нейросети, чего мы хотим. Для чего мы это хотим. Нас интересует генерация текста с помощью чат-ботов. В некоторых случаях мы получаем текст программы. А мы хотели

не текст программы, мы хотели текст запроса. Как писать запрос к нейросети, если мы генерируем текст с помощью чат-бота.

Так что указываем, что именно мы хотим из чат-бота вытащить. Это слишком общие слова, слишком расплывчатая формулировка: «как писать запрос». Конкретное задание звучит так: «Составь список их двадцати полезных советов» или «топ 10 сайтов...»

Далее появляется ещё один элемент запроса: формат вывода, формат ответа. Мы можем отдельно попросить нейросеть объяснить этот конкретный пункт нашего списка, привести примеры того, как он звучит в запросах. И мы можем даже использовать пример формата вывода в тексте нашего запроса.

И вот мы подбираемся к ещё одной формулировке запроса. В промпте должны быть роль, контекст и формат. Теперь мы можем напечатать этот список и сказать: «Дополни этот список. Объясни, какие еще элементы должны быть в запросе к нейросети». Таким образом, мы постепенно выясняем для себя что-то и дополняем наш список и уточняем наш запрос.

Далее, доступ к Гигачату можно получить не только через браузер, но и через Телеграм. А этот мессенджер доступен и как мобильное приложение, и как десктопное приложение, и как сервис в браузере. Проверьте, поэкспериментируйте с этим сервисом. Каждый раз, после каждого эксперимента не забывайте отразить результаты в отчёте.

Итак, в конечном счете нас интересует список из 5, или 10, или 20 пунктов, которые действительно должны быть в нашем запросе к нейросети. И вот с этими вещами мы дальше будем экспериментировать.

Есть ещё один полезный совет, который удалось вытащить из нейросети. Задание должно звучать как задание, как команда. Мы говорим, даём команду, что надо сделать: предложи, напиши, сформулируй, обоснуй. Команда, задание — это действие. Мы не просто говорим «список», а мы говорим «составь список». Мы не говорим «программа», мы говорим «напиши программу». Бывают

ситуации, когда мы говорим: «вот программа, исправь ошибку». Конкретная команда содержит в себе действие, и это глагол. Возможно, даже когда-то в далёком детстве, в школе что-то такое было. Теперь начинает приходить.

Следующий элемент запроса — ограничение. Мы должны указать, какой объем слов, или строк, или абзацев, или страниц хотим получить. Или: Напиши так, чтобы среднестатистический человек мог прочитать это за 5 минут. Можно даже потребовать нужное количество букв: Напиши ответ, в котором 20 букв.

Ещё один элемент запроса — это вариативность, или температура. Попробуйте уточнить у самой же нейросети: «Как нам задавать вариативность или температуру, когда мы пишем запрос к нейросети?» Как её указывать в запросе? Пускай сама нейросеть приведет пример. Мы так и говорим: Приведи 5 примеров того, как...

Что еще в запросе может быть? Оказывается, стиль. Разговорный стиль. Формальный, неформальный. Художественный. Как в такой-то книге — в качестве примера. Стиль какого-то знаменитого человека. Мы можем указать, каким языком, в каком стиле, на кого похоже.

## **ПРОГРАММИРУЕМ БЕЗ ПРОГРАММИРОВАНИЯ**

Переходим к инструменту под названием COZE, самому необычному на сегодняшний день. Здесь потребуется зарегистрироваться. Бесплатно. Например, с учетной записью Google. Нажимаем кнопку Get started справа сверху — то есть начать работу. Здесь мы сможем создать своего собственного бота, с которым будем дальше общаться.

Нажимаем кнопку Create bot слева сверху — создать бота.



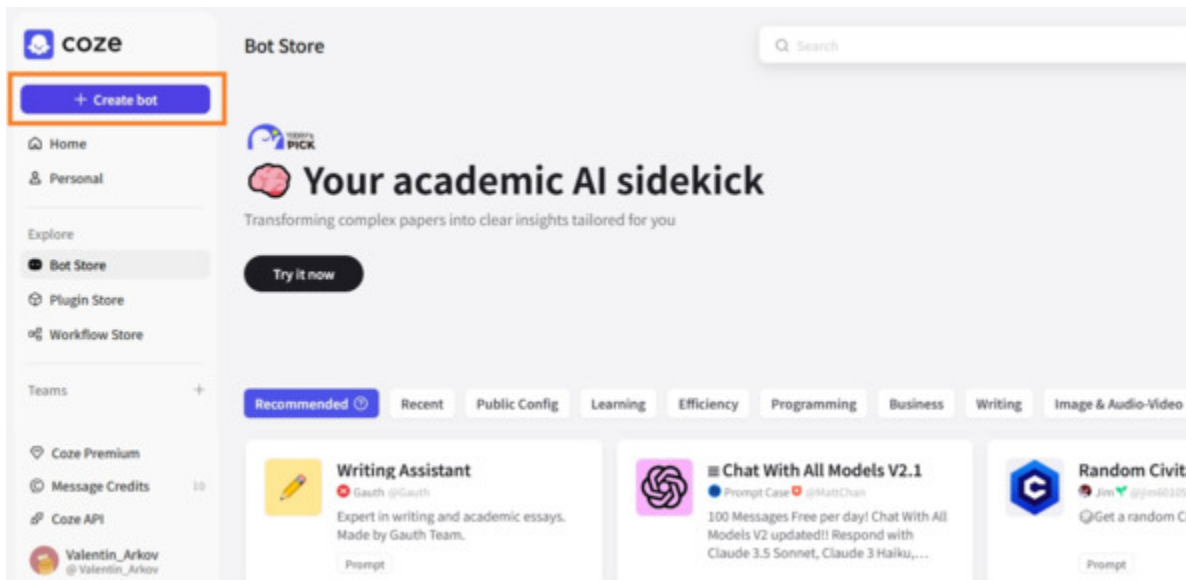


Рис. Создаём бота

Далее в диалоговом окне Create bot – Workspace – Personal. Мы придумываем ему свое собственное название Bot name. Описываем его работу Bot function description. Генерируем ему иконку – картиночку. Нажимаем кнопку Confirm – подтвердить.

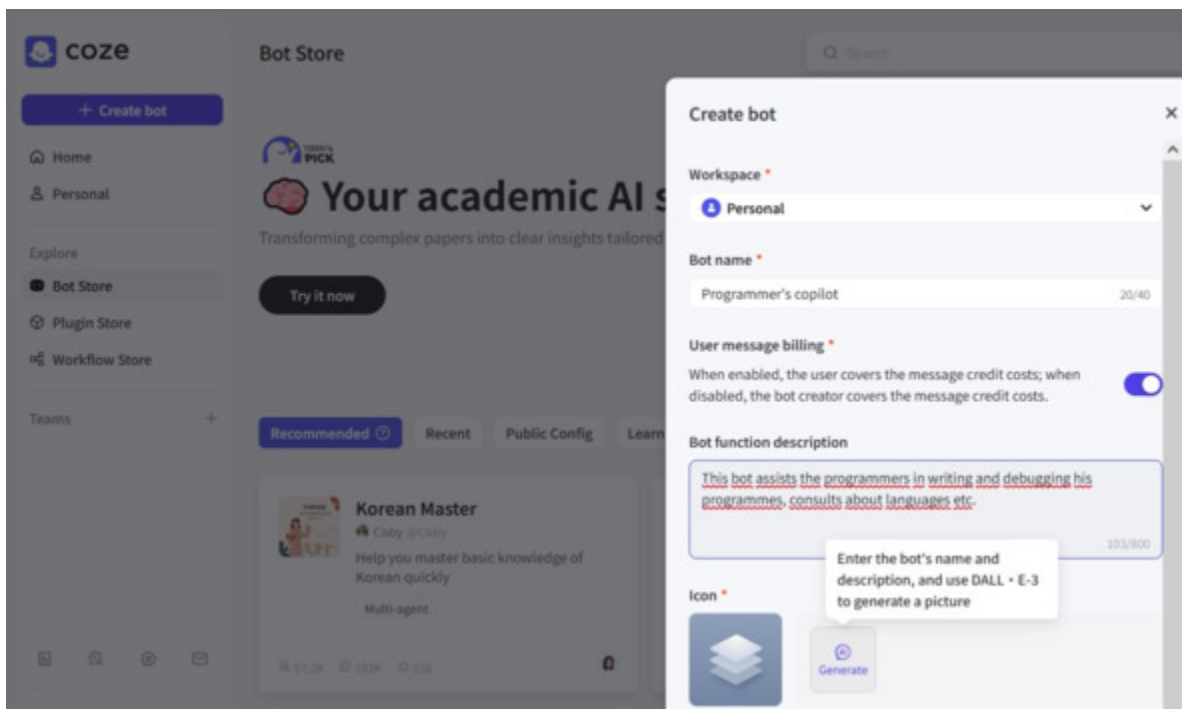


Рис. Открывающий диалог

Теперь сразу же выбираем языковую модель попроще, чтобы не потратить лимиты на запросы. На сегодняшний день этот сервис бесплатно обслуживает ограниченное число запросов. Например, для модели GPT-4o даётся только один запрос в день.

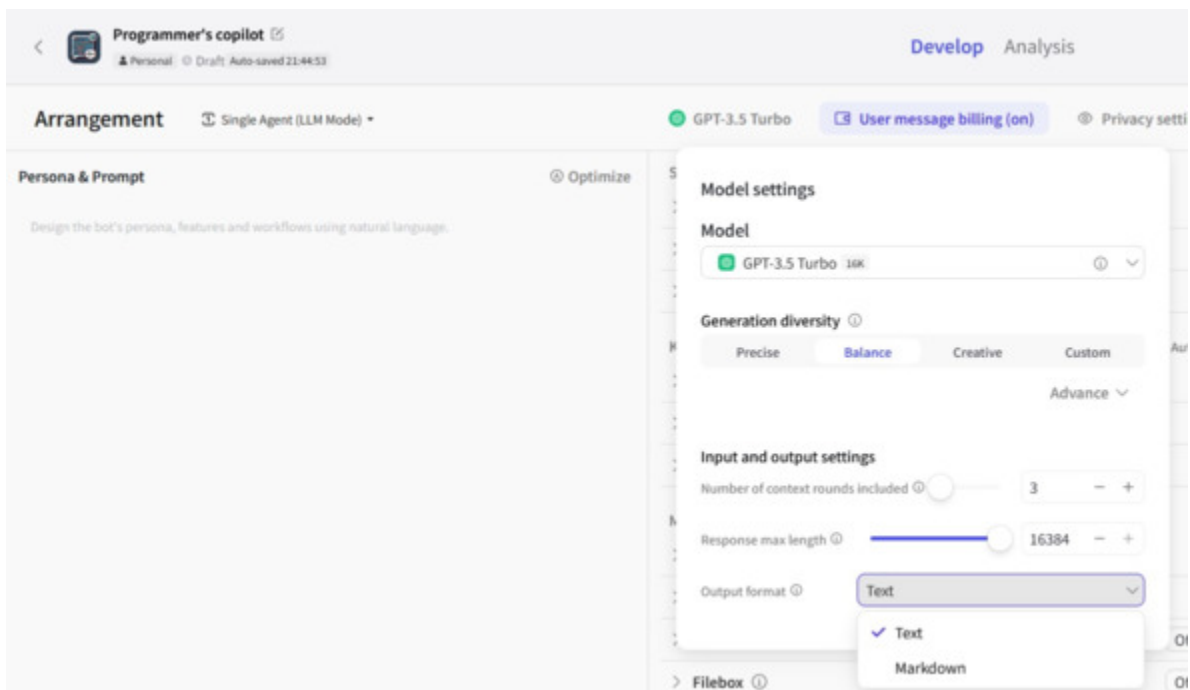


Рис. Выбор языковой модели

Далее, в разделе Skills — возможности — подключаем plugins — плагины — дополнения. Например, Google Web Search — поиск с помощью Google. Теперь у нашего бота есть выход в интернет для поиска свежей информации.

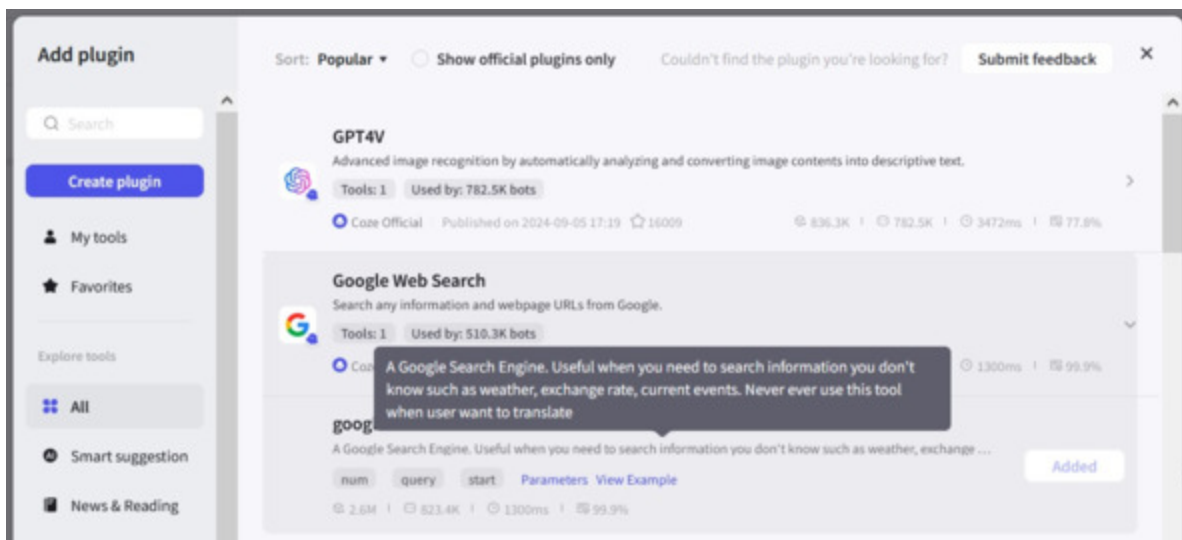


Рис. Возможности поиска

В разделе Persona & Prompt мы описываем общий запрос. И это будет System prompt. Есть запрос пользователя, а есть системный запрос. Уточните в любом чат-боте, что такое System prompt и как этим пользоваться. В двух словах, этот промпт будет всегда незаметно добавляться к запросу пользователя. И здесь мы можем описать все общие элементы наших запросов. Или можем задать любые ограничения, например: «в своих ответах говори намёками, чтобы пользователю пришлось подумать самому».

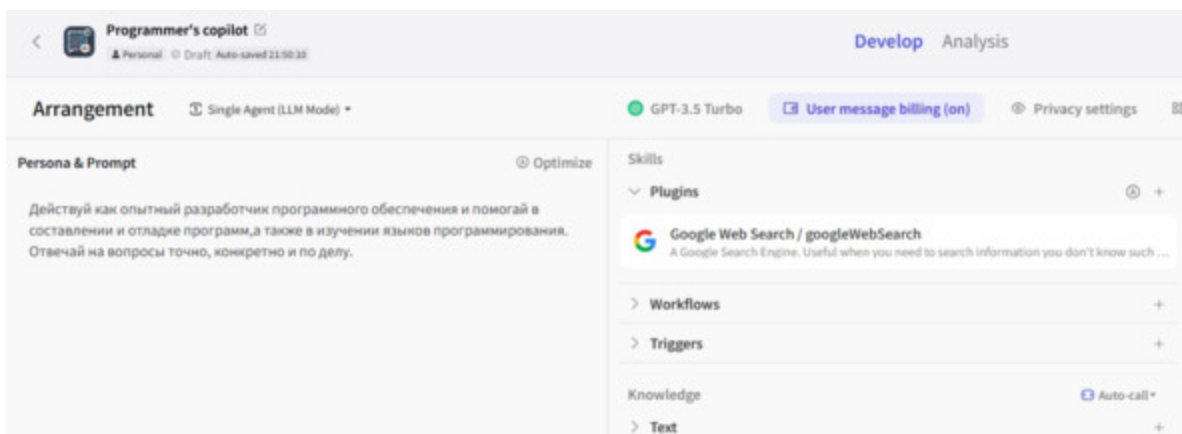


Рис. Системный промпт

В разделе Preview & Debug – предварительный просмотр и отладка – мы можем поработать с нашим ботом. Если нужно, сразу же исправляем его настройки.

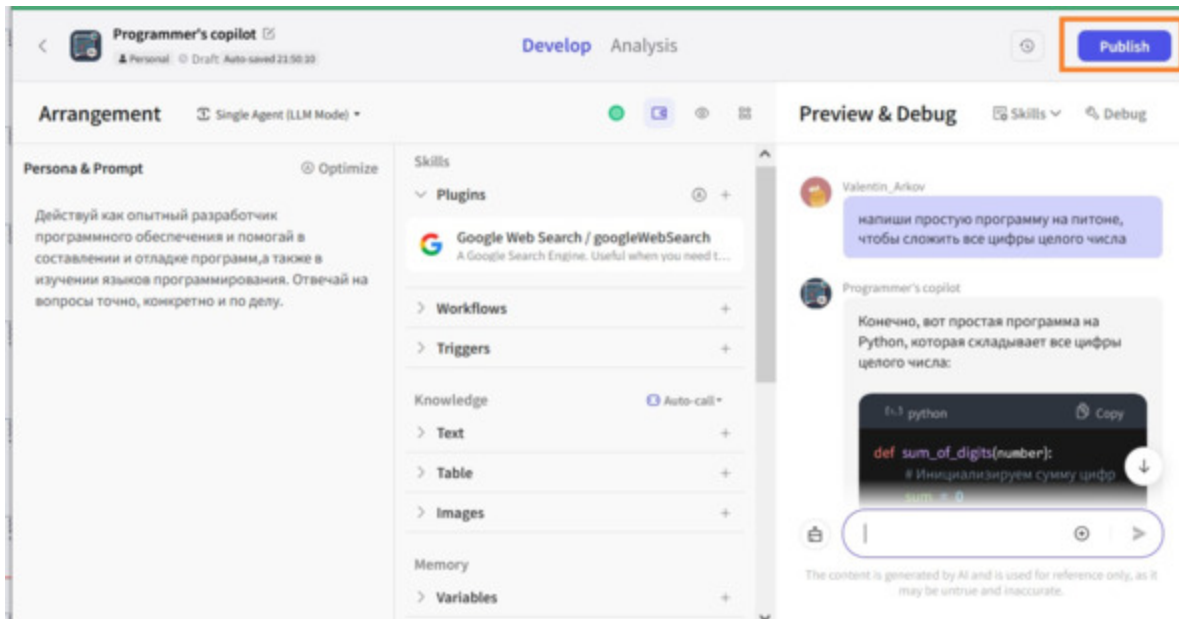


Рис. Предпросмотр

Дальше на сервисе COZE мы можем сделать бота, с которым будем общаться через Телеграм. Для этого переходим к публикации. Справа сверху нажимаем кнопку Publish. В диалоговом окне настроим Conversation opener – начало диалога и рекомендуемые вопросы (их можно удалить). Нажимаем Confirm – подтвердить.

### Complete a few more steps to make your bot stand out! ✕

#### Add a conversation opener ⓘ

A conversation opener gives users a quick understanding of your bot's functions, increasing user conversion.

Opening text

⌵ | B | I | Ⓔ | ☰ | ☷ | 66 | 🔗 | 🖼️ | {} | 👤

Привет от второго пилота! Я помогу в составлении и отладке программ.

↶ ↷

#### Preset opening questions ⓘ

⋮

Enter the opening question

🗑️

Delete question

Skip and publish

Confirm

Рис. Настройка приветствия

В диалоге Publish to указываем платформу для публикации бота снимаем галочку Coze Bot Store — официальный магазин ботов. Видим, что в разделе Telegram у нас галочка не активна. Зато справа есть кнопочка Configure — настроить.

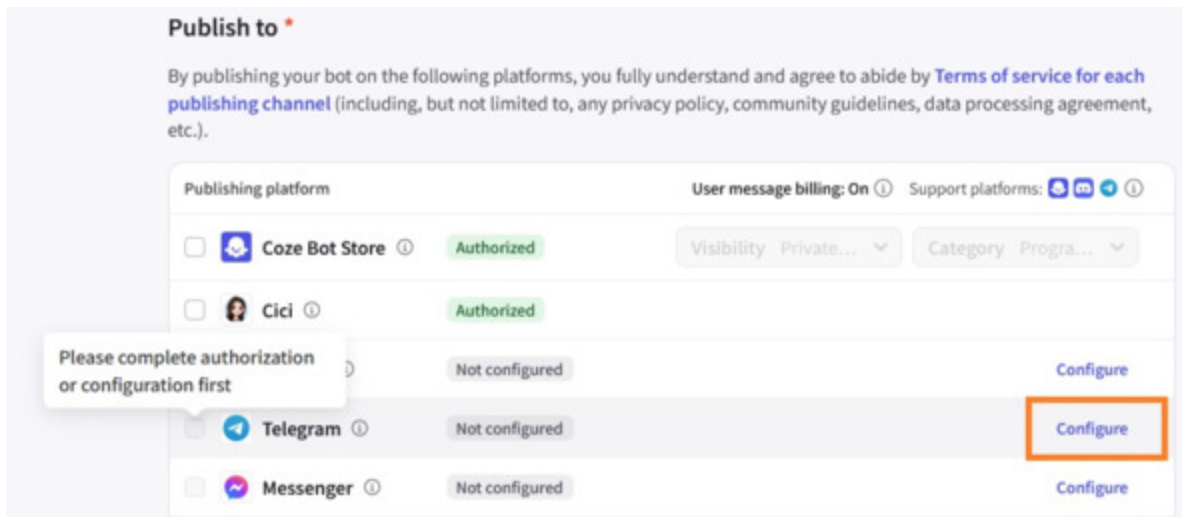


Рис. Публикуем в Телеграме

Здесь нужно ввести токен, который делается с помощью крёстного отца всех ботов – BotFather.

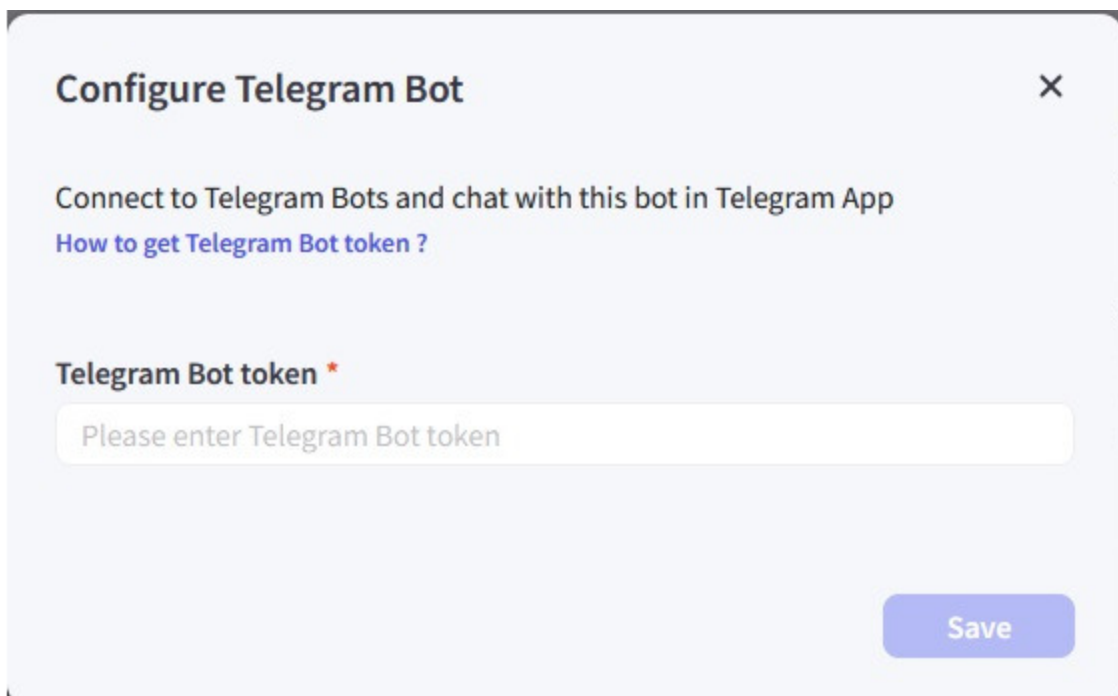


Рис. Запрос токена



В телеграмме мы находим бота под названием @BotFather. Будьте внимательны: есть много подделок с похожими названиями. Нажимаем Меню и выбираем команду /newbot. Создаём нового бота. Придумываем ему название и официальное имя для Телеграма.

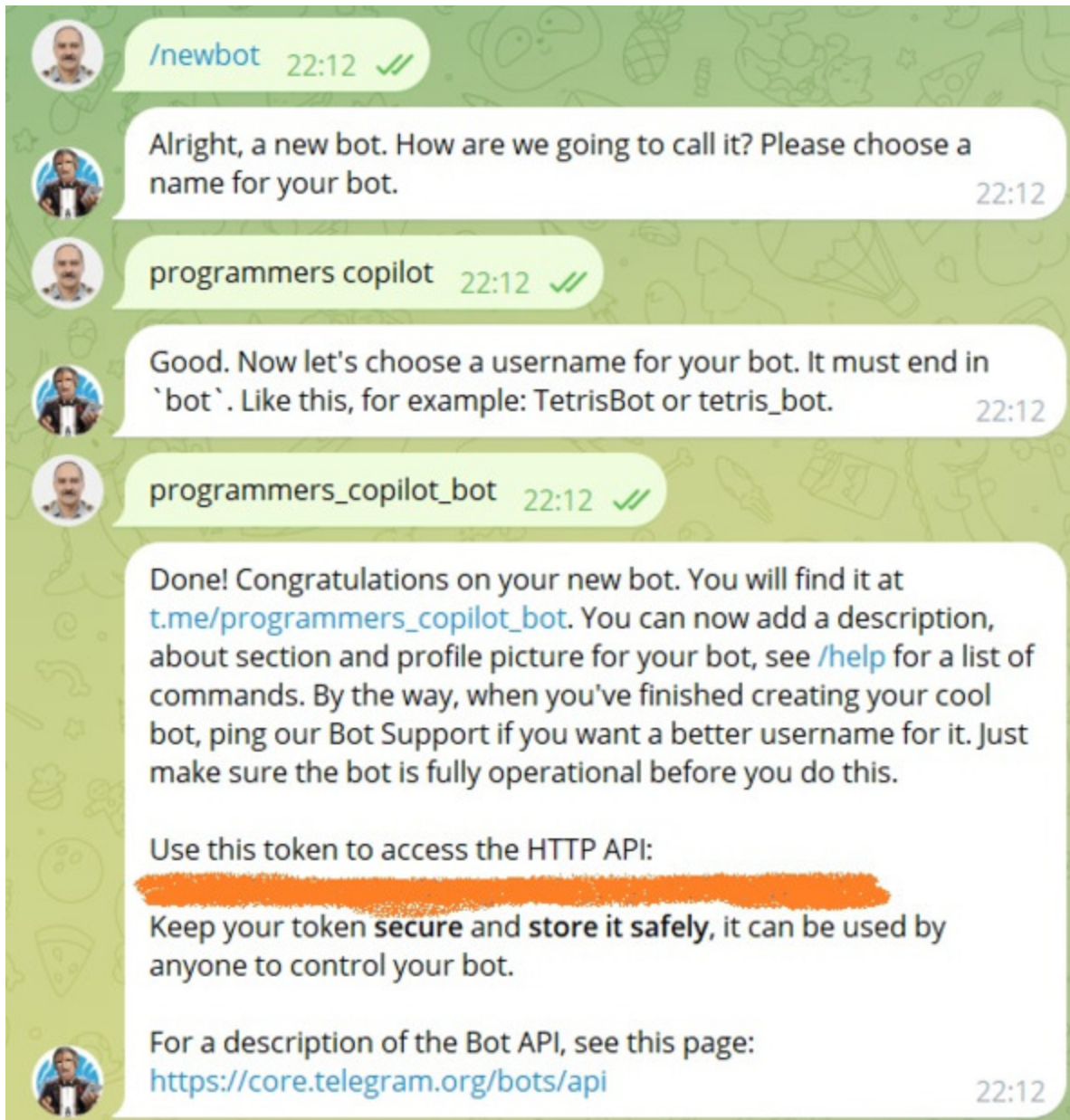


Рис. Получение токена



После создания бота нам сообщают токен. Это длинная строка символов. Копируем эту строчку в буфер обмена и вставляем в COZE. Нажимаем кнопку Save – Сохранить.

Теперь ставим галочку напротив Telegram и нажимаем кнопку Publish – Опубликовать.

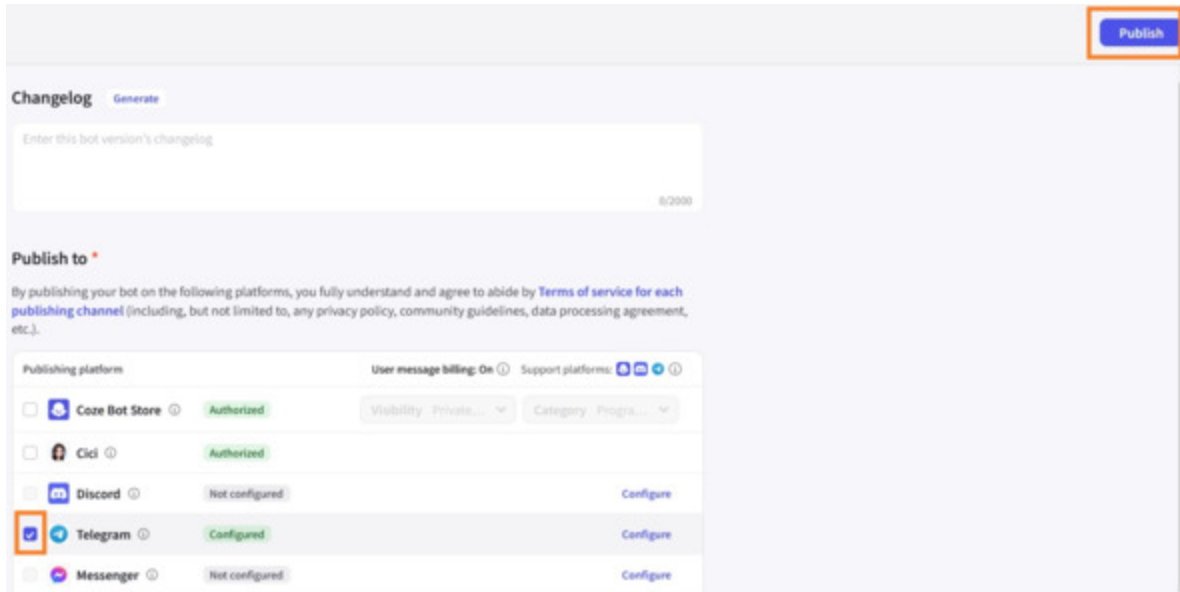


Рис. Публикация в Телеграме

Нам сообщают: Publication submitted! – Бот опубликован.

Можем перейти по ссылке Chat now – начать общаться с ботом. Либо можем просто скопировать ссылку на бота в буфер обмена.

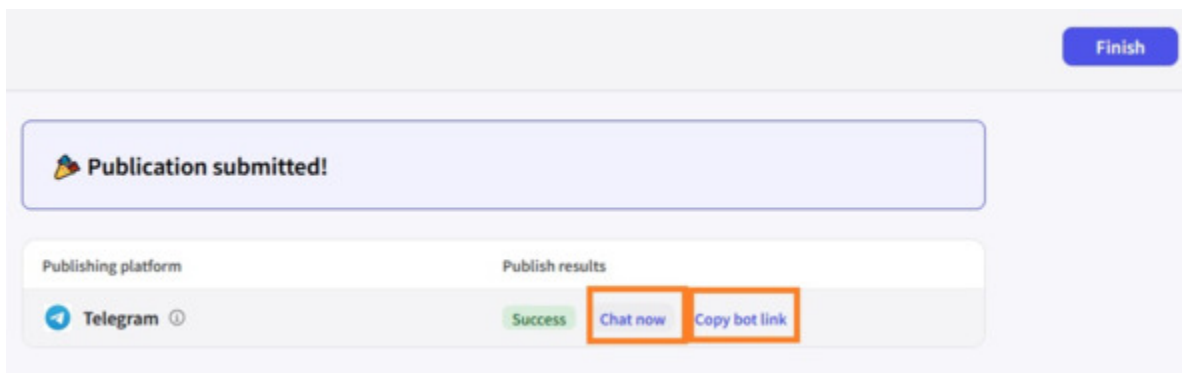


Рис. Переход к беседе

Бот поможет нам написать программу, провести анализ её вычислительной сложности и оптимизировать код — если только его об этом попросить.

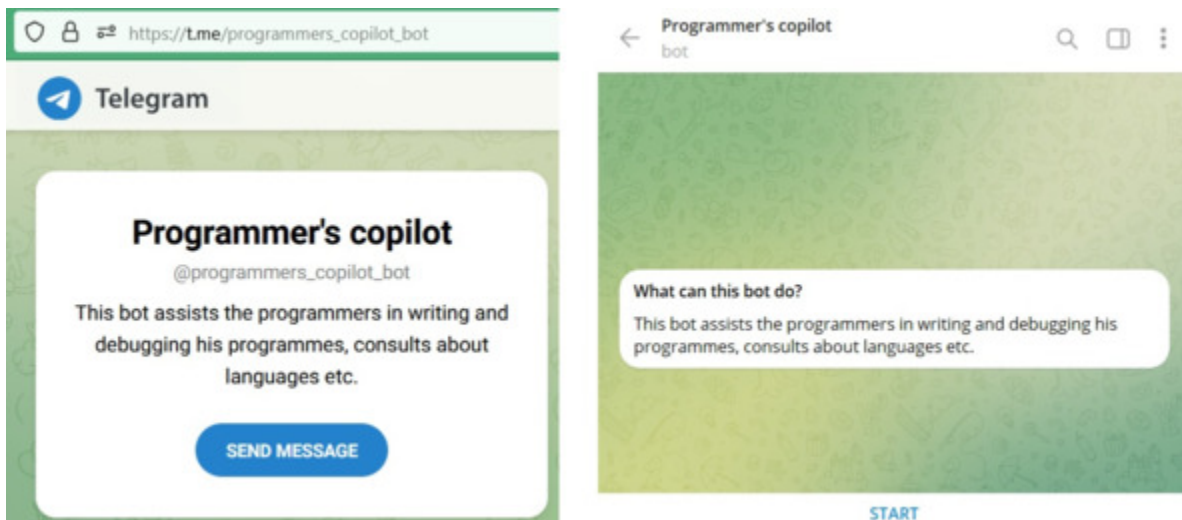


Рис. Запуск бота

Наш девиз — это классический афоризм: «Компьютер делает не то, что вы хотели, а то, что вы просили!»

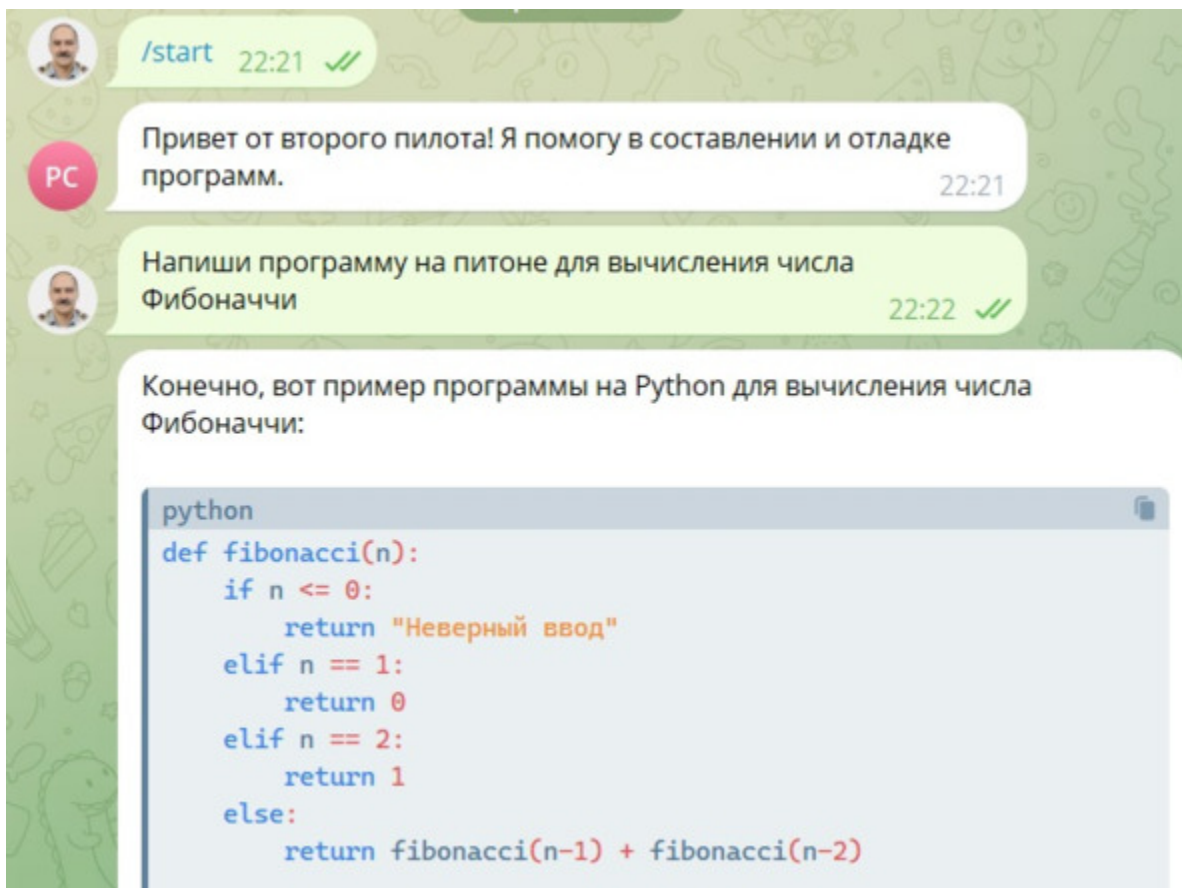


Рис. Генерируем программу

Это был пример того, как можно создавать бота без программирования. Называется такой подход No-code или Zero-code.



Рис. Оценка вычислительной сложности

Теперь пришло время поделиться своим ботом с соседом. Для этого мы с вами используем сервис сокращения ссылок, например, Яндекс. Кликер. Здесь адрес любой длины можно сократить до 5–6 символов. Можно также получить QR-код, чтобы другие могли отсканировать его телефоном и сразу перейти по нужному адресу.

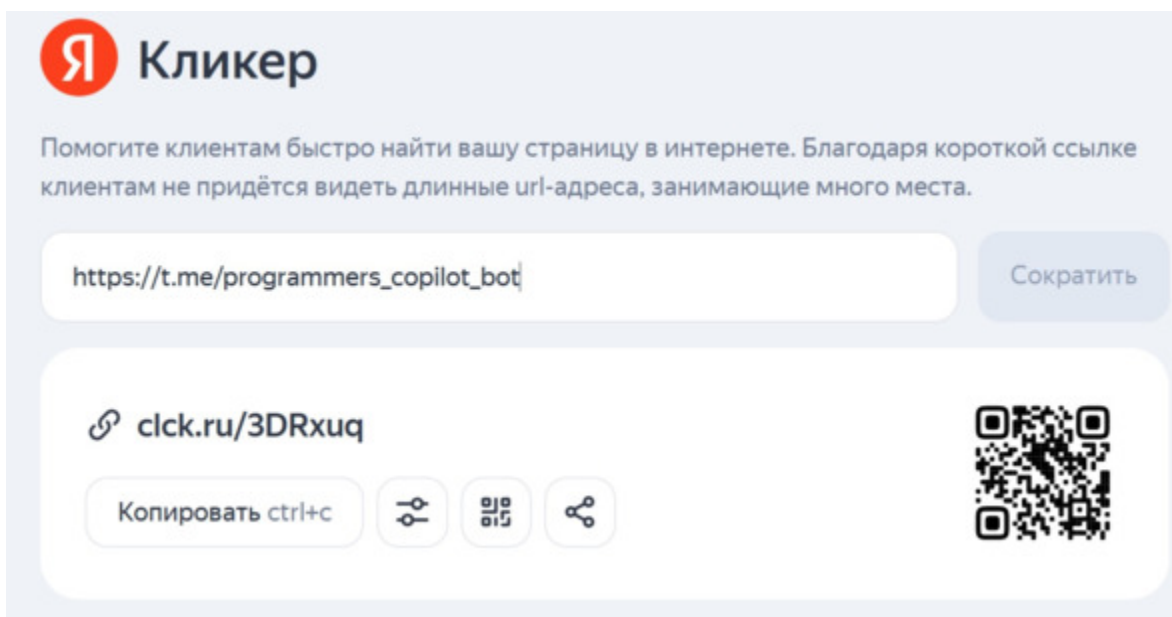


Рис. Сокращаем ссылку

## ОПЫТЫ С ЗАПРОСАМИ

Следующий наш этап — посмотреть на собранные рецепты и полезные советы и потренироваться, исследовать их в работе. Действительно применить их и посмотреть, чем это кончится.

Мы испытываем назначение роли, контекста, формата, размера, тональности и так далее. Включая температуру. Сформулируйте очень подробный запрос про то, как писать запросы к нейросетям. Используйте те самые полезные советы, которые удалось собрать. При необходимости просите его о помощи.

Добейтесь, чтобы наш интеллектуальный помощник отвечал именно именно то, что мы хотим. При этом мы фиксируем в отчете всё, что происходит.

Берем самый длинный и подробный промпт и начинаем с ним работать. Вначале мы должны были назначить роль. Попробуйте назначить другую роль. Всё остальное точно так же. Меняем роль. Посмотрите, чем это кончится, как изменится ответ. Как влияет

назначение роли на ответ нейросети? А запрос по-прежнему тот же: как писать запросы к нейросети. Основная часть та же самая, просто роли разные должны быть.

Естественно, для этого опыта мы основной текст запроса держим в файле отчета. Оттуда его копируем, чтобы не приходилось каждый раз его полностью печатать. Отмечаем результаты в отчете. Какая роль, какой ответ.

Посмотрите на температуру. Как влияет температура на результаты? Запускайте запрос несколько раз. Каждый раз обнуляем – удаляем беседу Thread и создаем новую.

Отчет по лабораторной работе отправляем в виде ссылки. Форма для загрузки расположена на странице нашего курса на гитхабе. Переходим по ссылке, вводим фамилию и ссылку на отчет. Нам достаточно использовать доступ на чтение.

Поскольку это облачный документ, мы можем открыть доступ на чтение и на запись. Нас интересует только чтение.

## **ССЫЛКИ**

Учебные материалы на GitHub

<https://github.com/Valentin-Arkov/Digital-Tech-AI>

Учебные материалы на GitVerse

<https://gitverse.ru/Valentin-Arkov/Digital-Tech-AI>

YandexGPT

<https://ya.ru/ai/gpt-3>

YandexGPT API

<https://yandex.cloud/ru/services/yandexgpt>

Документы Google

<https://docs.google.com>

Яндекс. Документы – бесплатный онлайн-редактор

<https://docs.yandex.ru>

Чат с YandexGPT

<https://console.yandex.cloud> – foundation-models – chat

Квоты и лимиты в Yandex Foundation Models

<https://yandex.cloud/ru/docs/foundation-models/concepts/limits>

YandexGPT API overview

<https://yandex.cloud/en-ru/docs/foundation-models/concepts/yandexgpt/>

GigaChat – бесплатная нейросеть без vpn на русском языке, которая общается как человек

<https://giga.chat/>

Perplexity. Where knowledge begins

<https://www.perplexity.ai/>

GigaChat. Нейросетевая модель от Сбера

[https://t.me/gigachat\\_bot](https://t.me/gigachat_bot)

GigaChat. Главный канал про AI на русском языке – про настоящее и будущее

[https://t.me/official\\_gigachat](https://t.me/official_gigachat)

Coze. Next-Gen AI Chatbot Developing Platform

<https://www.coze.com/>

BotFather. The one bot to rule them all.

<https://telegram.me/BotFather>

Яндекс Кликер. Помогите клиентам быстро найти вашу страницу в интернете.

<https://clck.ru/>



## 2. MARKDOWN

### ВВЕДЕНИЕ

В этой работе мы познакомимся с языком разметки Markdown. Это разновидность текстовых файлов. Внутри такого файла находятся только стандартные символы ASCII, то есть буквы, цифры, знаки препинания и другие символы, которые можно найти на обычной компьютерной клавиатуре. При просмотре Markdown мы получаем форматированный текст: заголовки, жирный и наклонный шрифт, таблицы и так далее.

Редактируют этот файл любым инструментом, например с помощью Блокнота — Notepad. Этот формат также понимают и более продвинутые средства редактирования, такие как Notepad++. Многие интегрированные среды программирования — Integrated Development Environment (IDE) — тоже могут работать с форматом Markdown.

Всего за один, или два, или три часа вы познакомитесь с новой технологией. К концу этого занятия вы сможете легко создавать и редактировать файлы формата Markdown и профессионально работать с ними в блокнотах Google Colab и на сервисах управления версиями типа GitHub.

### ОФОРМЛЕНИЕ ОТЧЁТА

Отчёт по этой работе оформляется в облачной среде Google Colab. Если выразиться точнее, это интерактивный блокнот — тетрадка для программирования на Питоне. Если кто-то предпочитает английское название, то «на Python».

Здесь есть возможность не только составлять программу, но и работать с ячейками форматированного текста.

Открываем страничку сервиса [Colab], ссылку смотри в конце описания работы. Входим с учётной записью Google и создаём новый блокнот — New notebook.

Создаём текстовую ячейку — Text — Add text cell. Передвигаем её в начало блокнота — Move cell up.

для редактирования содержимого текстовой ячейки — двойной щелчок мышкой — Double-click to edit.

Вводим наш текст и время от времени сохраняем файл, нажимая комбинацию клавиш [Ctrl + S] — от слова Save — Сохранить.

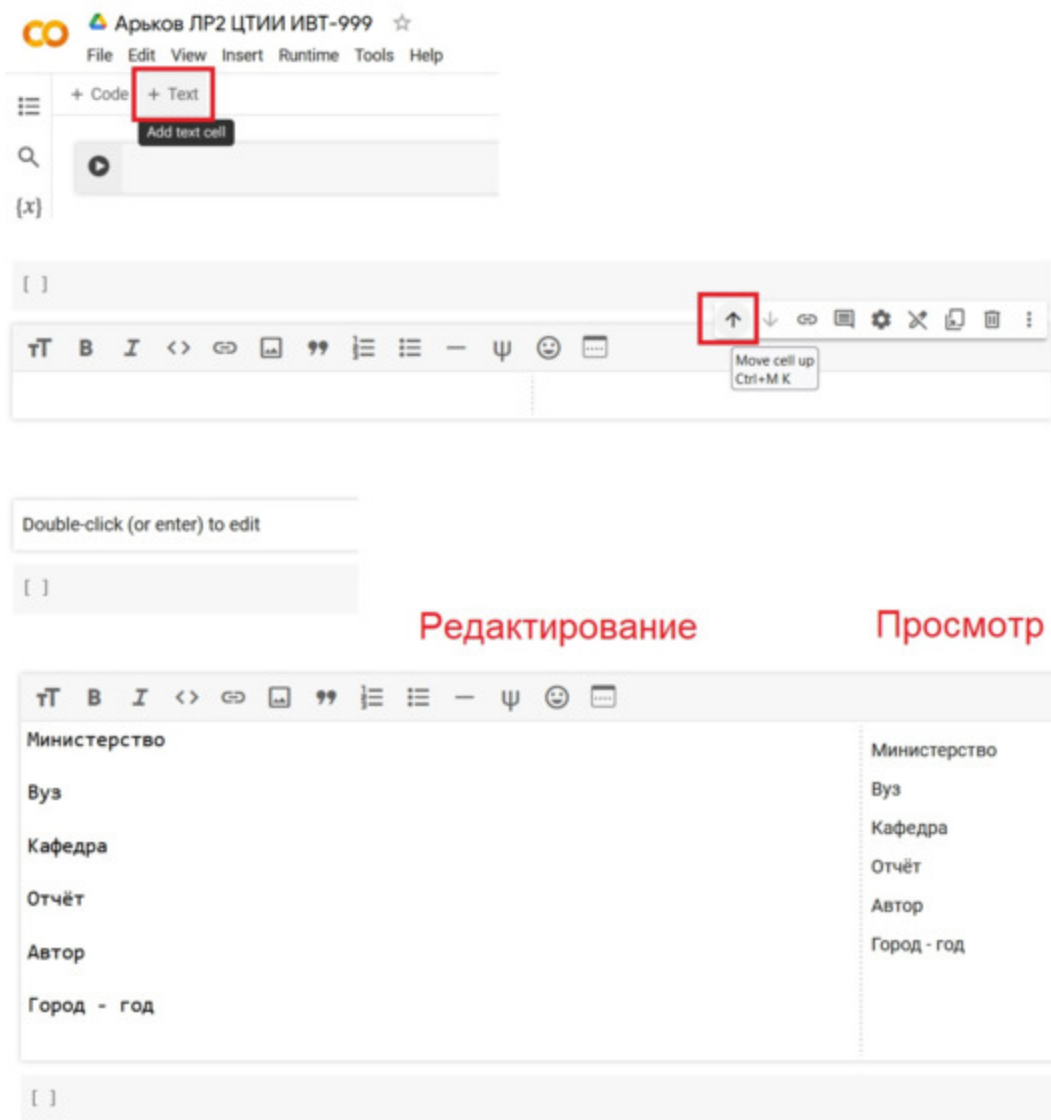


Рис. Отчёт по работе

Отчёт по традиции начинаем с титульного листа. Здесь приводим те же ключевые данные по поводу нашего документа: Министерство, вуз, кафедра, название работы, ... и даже фамилии авторов. Подробности мы уже обсуждали в предыдущей работе, повторяться не будем.

Абзацы разделяем пустой строкой.

По окончании работы нужно будет «расшарить» блокнот — Share notebook, то есть предоставить совместный доступ на чтение.

Нажимаем кнопку Share в правом верхнем углу окна. Выбираем адресатов – любой пользователь, получивший ссылку – Anyone with the link. Устанавливаем права только на чтение – Viewer – Просмотр. Копируем ссылку в буфер обмена – Copy link. Вставляем ссылку в форму для отправки отчета на странице курса на GitHub.

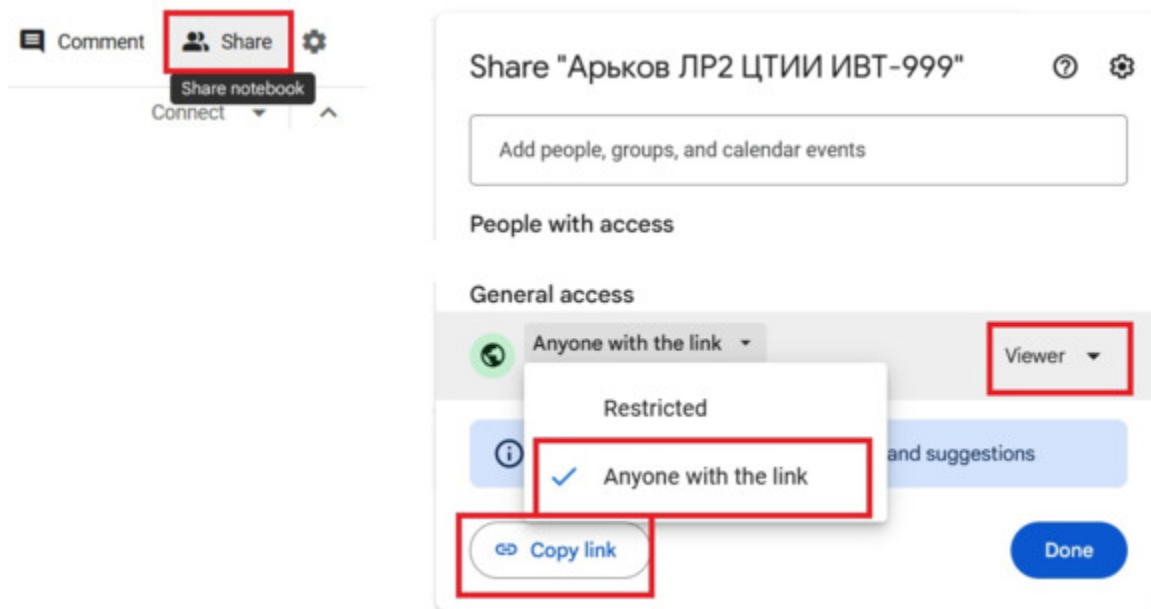


Рис. Делимся ссылкой на отчёт

## УСТАНОВКА РЕДАКТОРА

Для первого знакомства с этой технологией мы установим очень простой и популярный текстовый редактор Notepad++. Само название уже на что-то намекает. Имеется в виду стандартный блокнот Windows Notepad. Два плюсика намекают на следующий шаг, следующий уровень, на улучшение и совершенствование, на дополнительные возможности.

Первоначально два плюсика использовались в языке программирования Си для операции инкремента, то есть для увеличения значения счётчика на единицу. Затем эта символика

использовалась для обозначения языка программирования C++, чтобы подчеркнуть идею дальнейшего развития. Так вот, Notepad++ – это дальнейшее развитие программы Notepad.

Мы будем использовать такую версию этой программы, которая не требует установки. Мы просто скачиваем файл архива из интернета, распаковываем его в рабочем каталоге и запускаем на выполнение. Это можно произвести даже с правами пользователя, которому не разрешается вносить изменения в настройки операционной системы. После работы с нашей программой мы можем удалить всю папку, и на компьютере не останется никаких файлов, никаких следов, никаких записей в реестре операционной системы.

Такая программа называется «переносимый вариант», или «портируемая версия», или по-английски Portable Application. Буквально английское слово Portable означает «портативный», то есть небольшой, удобный для переноски вручную – с места на место.

Примером такого изделия служит печатная машинка. В течение многих лет люди использовали большие, стационарные печатные машинки. А для работы в поездках были придуманы небольшие, портативные машинки, которые можно было носить в небольшом чемоданчике.



Рис. Портативная печатная машинка

То же самое было сделано и с компьютерами. Первые компьютеры были очень большие и занимали отдельную комнату или даже целый дом. Затем были разработаны персональные компьютеры, которые стояли на столе или под столом — Desktop. Для использования в поездках их размеры уменьшили до ноутбуков, а затем появились и мобильные устройства — планшеты и смартфоны.

Точно так же, переносимые программы можно легко переносить с одного места на другое — то есть просто копировать всю папку с файлами и не заниматься процедурой установки.

Открываем сайт [Notepad++]. Переходим по ссылке download. Выбираем вариант Portable Zip и скачиваем себе на компьютер. Создаём новый каталог и распаковываем туда содержимое архива — со всеми его подкаталогами и файлами. Запускаем исполняемый файл — он тут в корневом каталоге один такой.

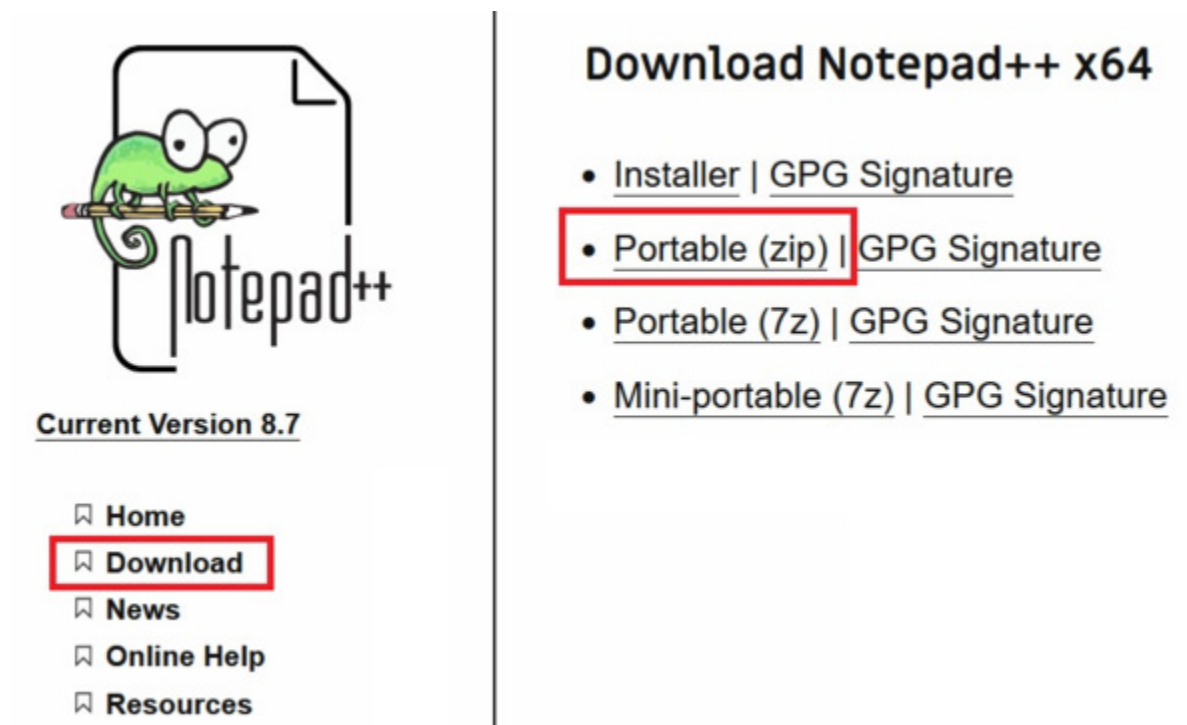


Рис. Загружаем переносимую версию

Теперь нам нужно установить дополнение для просмотра файлов markdown. в верхнем меню выбираем пункт Plugins – Plugins Admin. Находим в списке Markdown Viewer и нажимаем кнопку Install. Соглашаемся на установку дополнения и перезапуск программы.

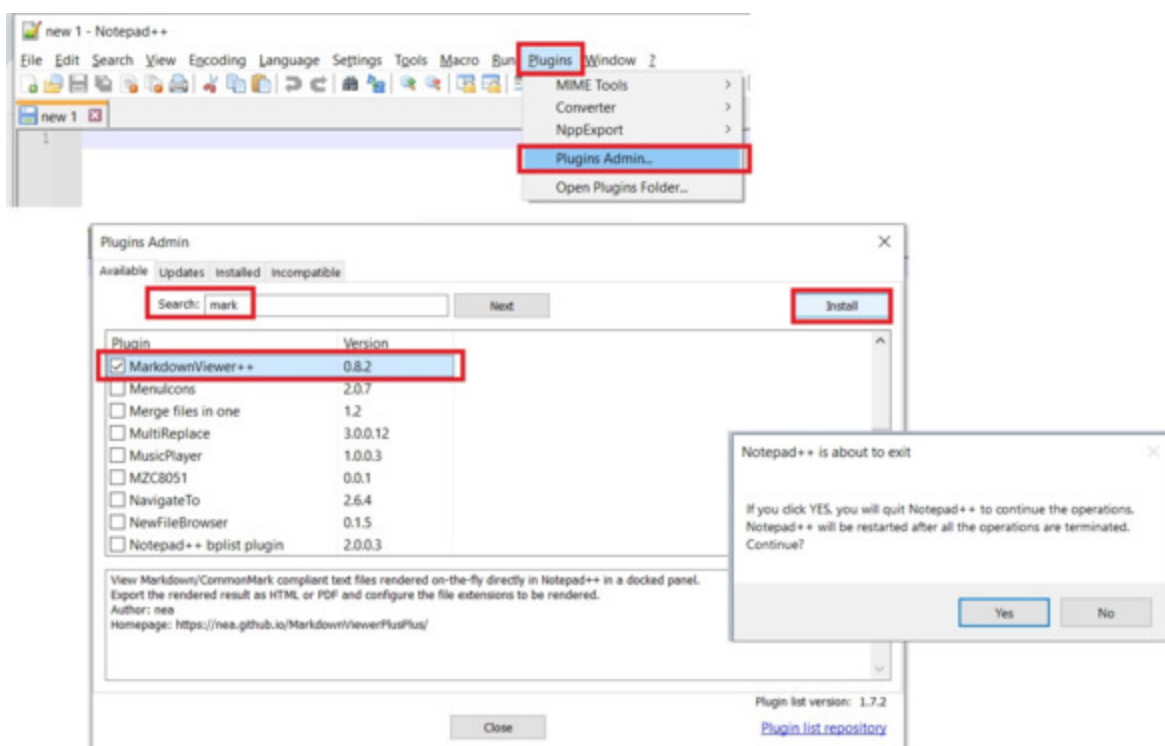


Рис. Установка дополнения для просмотра

Открываем редактор и создаём наш первый файл. Сохраняем файл с расширением \*.md. Далее в Проводнике мы можем выбрать программу для редактирования таких файлов по умолчанию.



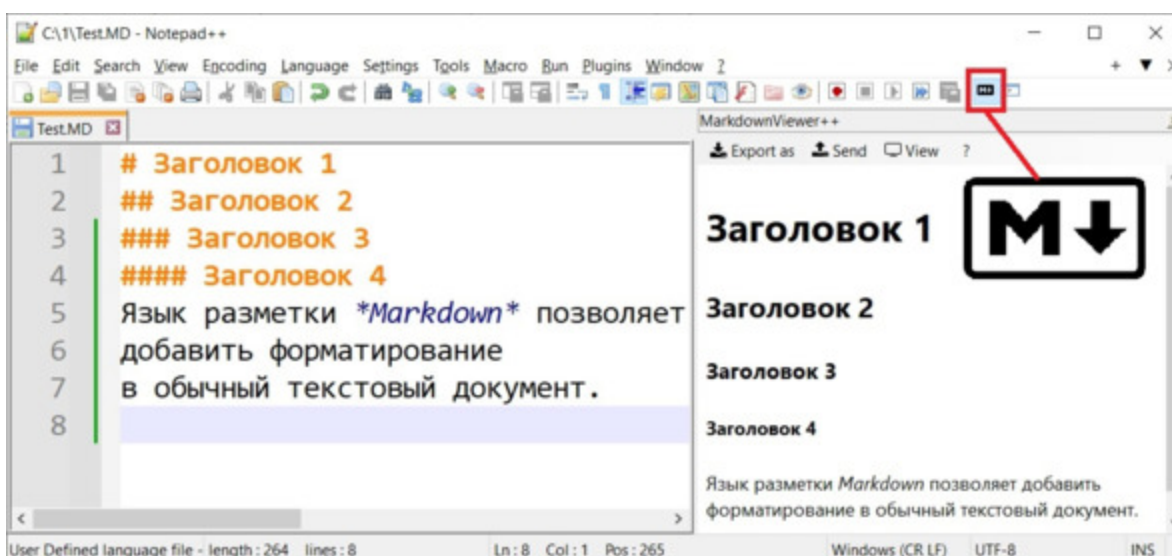


Рис. Редактирование и просмотр Markdown

На рисунке показан пример редактирования и просмотра. Можно видеть, что исходный текст легко читается. При этом он содержит дополнительные указания — разметку форматирования. Например, заголовок начинается с одного или нескольких символов решётки # и пробела. Курсив, то есть наклонный шрифт, выделяется символом звёздочки \*. При редактировании такого файла мы вводим эти символы разметки вручную, с клавиатуры.

## MARKUP И MARKDOWN

Для просмотра отформатированного текста в нашем примере служит секретная кнопка в верхней линейке инструментов. Чтобы посмотреть результаты, мы нажимаем эту кнопку, чтобы закрыть и снова открыть окно просмотра. На этой кнопке изображено условное обозначение языка Markdown — заглавная буква М и стрелка вниз. Слово «вниз» по-английски звучит как «down». Кроме движения вниз, это слово передаёт идею упрощения. То есть это упрощённый язык разметки.

Но это ещё не всё. Этот язык, упрощённый по сравнению с полноценным форматом HTML. А расшифровывается это как HyperText Markup Language — язык гипертекстовой разметки. Получается, что Markdown — это как бы противоположность Markup.

Но и это ещё не всё. Первоначально слова mark up и mark down означали повышение и понижение цены на товар. Продавец вывешивает новый ценник, то есть как бы «обозначает», или «помечает», или «отмечает» цену на товаре, который лежит на витрине. А уже потом компьютерщики использовали эти термины для языков разметки текста. Подробнее можно изучить всю эту историю в словарях и с помощью интеллектуальных чап-ботов.

Вот такая игра слов. Программисты очень любят играть словами и символами. Особенно если приходится работать с обычным текстом и командным окном, консолью, терминалом. Здесь больше играть не с чем.

Создатели этого проекта поясняют, что такой формат легко воспринимается человеком. Его легко читать и несложно редактировать. С другой стороны, этот формат автоматически преобразуется в HTML, и его можно использовать на страничках веб-сайтов [Markdown Project]. Они даже определяют Markdown как инструмент для преобразования текста в HTML для авторов веб-страниц. Что нас особенно радует, Markdown — это свободно распространяемое программное обеспечение — free software.

Общее описание формата можно найти в народной энциклопедии [ВИКИ: Markdown].

## **ОСНОВНЫЕ КОНСТРУКЦИИ MARKDOWN**

Самые простые, базовые приёмы форматирования описаны на странице проекта [Markdown Basics]. Рассмотрим эти инструменты. Если у читателя есть трудности с пониманием

английского текста, можно включить автоматический перевод. Он хорошо работает в браузере Яндекса и Google Chrome.

Абзацы разделяют пустыми строками — одной или несколькими. Это удобно при вводе текста. Мы вводим короткие строки в рамках одного небольшого окна. Даже одно предложение можно разбить на несколько строк. Пока не будет пустой строки, весь текст сливается в один длинный абзац. Затем, при просмотре, мы увидим один абзац вместо нескольких строк.

Заголовки начинаем с одного или нескольких символов решётки. После решётки ставим пробел, а затем сам текст заголовка. Так мы получаем несколько уровней заголовков. Можно сделать до 6 уровней — то есть до 6 символов решётки подряд. При просмотре они превращаются в теги от `<h1>` до `<h6>`.

Проверим, насколько это правда. Мы каждый раз должны «доверять, но проверять». Особенно при работе с компьютером. И особенно при чтении документации на программы. Мы к этой проблеме с документацией ещё вернёмся.

Итак, вводим вручную несложный текст с шестью уровнями заголовков. Затем на вкладке Markdown Viewer экспортируем его в HTML и PDF.

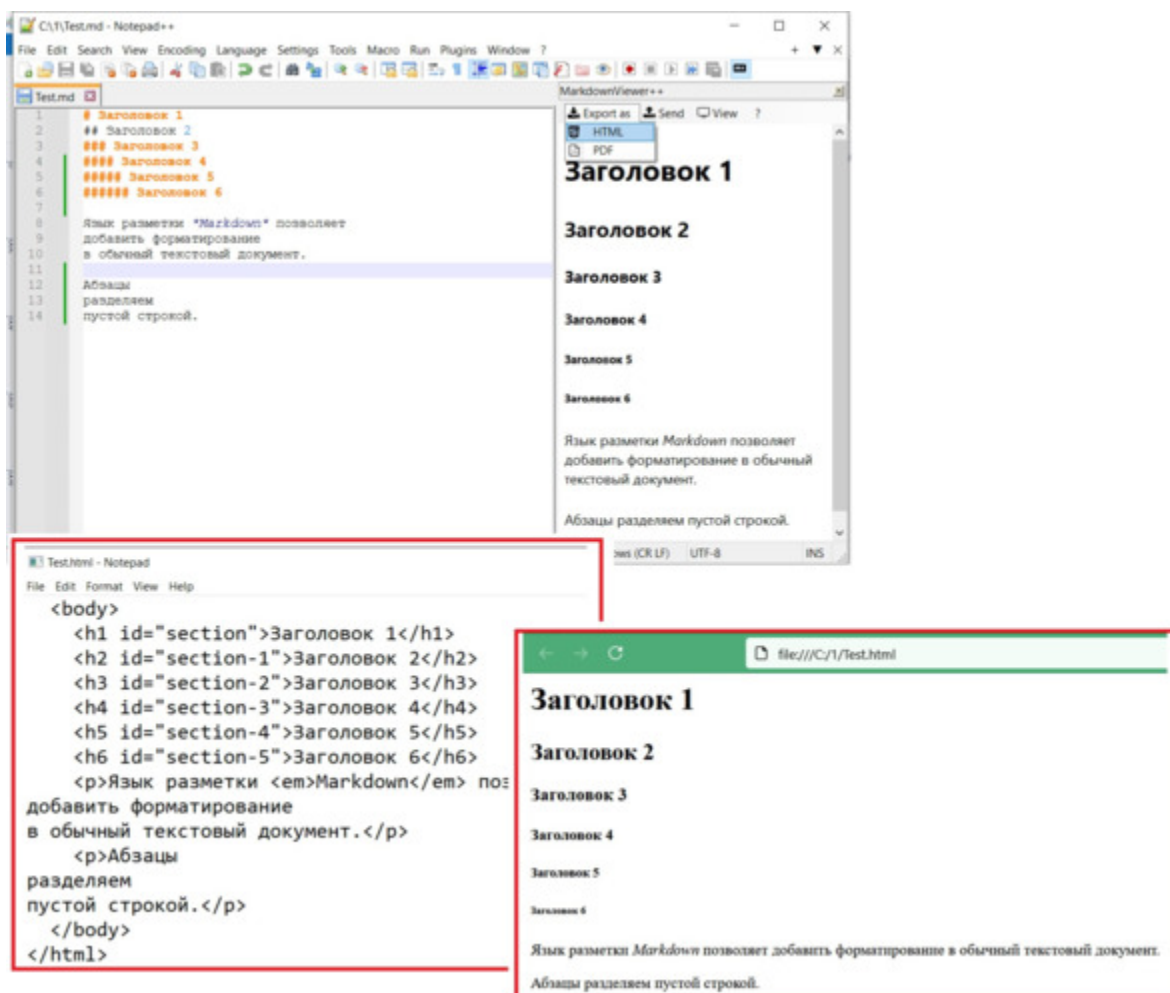


Рис. Экспорт гипертекста

Сохраняем файл в формате HTML и открываем его на просмотр любым браузером. Убеждаемся, что форматирование выполнено корректно.

Открываем тот же файл HTML на редактирование с помощью Блокнота. Изучаем теги HTML.

Как видим, появились теги заголовков h1-h6 от слова Header – заголовок.

Абзацы разделяются тегом p от слова Paragraph -параграф, абзац.

Курсив указан как *em* — от слова *emphasis* — ударение, подчёркивание, акцент — выделение текста для передачи важности.

## **ЕЩЁ О РАЗМЕТКЕ**

Заголовки можно также определять «подчёркиванием». На самом деле это строка из знаков «равно» и «минусов/чёрточек/дефисов» под самим текстом заголовка. Такое оформление читается ещё легче. Правда, третий уровень всё-таки придётся оформлять решётками.

Как мы уже говорили, курсив — это выделение одинарными звёздочками — до и после фрагмента текста. Жирный шрифт — выделение двойными звёздочками. Попробуем сообразить с одного раза, как выделить жирный курсив...

Двойная волнистая черта позволяет зачеркнуть текст. Такой приём добавляет выразительности.

На рисунке показана ещё одна особенность. Количество пустых строк между абзацами текста ни на что не влияет. И одна, и несколько пустых строк подряд — это просто разделитель двух соседних абзацев.

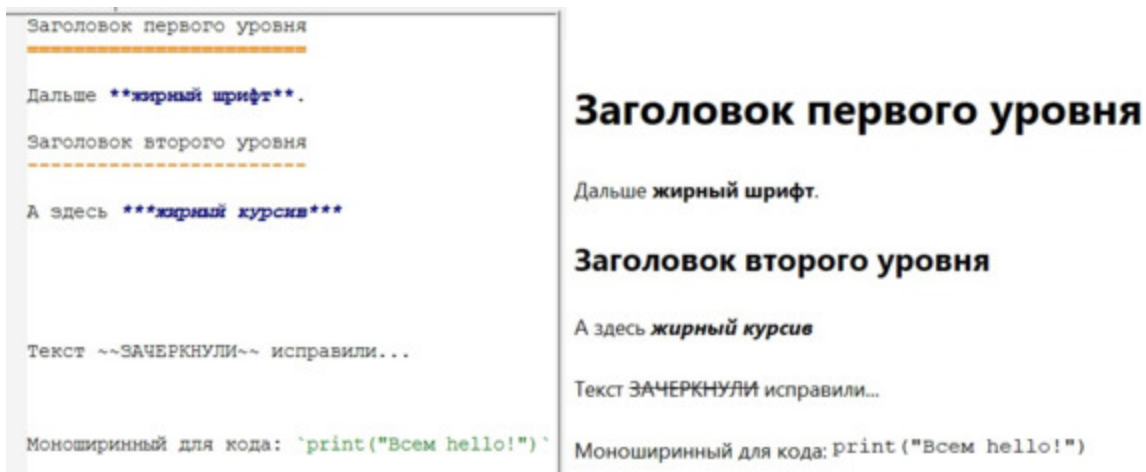


Рис. Жирный курсив и заголовки

Для использования текста программ есть свои хитрости. Во-первых, для этого удобнее использовать буквы одинаковой ширины. Так уже было в печатных машинках. И так удобнее работать с кодом. Называется это «моноширинный» шрифт — monospace. «Моно» означает «один», получается «шрифт одной ширины». Такие готовые шрифты есть в операционных системах. надо просто выбрать из списка.

**Задание.** Найдите моноширинные шрифты на своём компьютере. Посмотрите, как выглядит текст программы, напечатанный таким шрифтом.

Списки тоже делаются особым, хитрым способом.

Если список нумерованный, то можно использовать звёздочки (\*), плюсики (+) и минусы/чёрточки (-). В начале строки звёздочка, потом пробел, потом пункт списка. Но на экране будет стандартное обозначение пунктов списка: bullets — «буллеты» — буквально «пули». Это простые популярные маркеры списков. Первоначально это слово означало «шарик, пузырьёк». Теперь это кружочки, жирные точки.

Если список нумерованный, то строка должна начинаться с цифры, точки и пробела. При этом цифры могут быть любые. Могут быть одинаковые, могут быть разные, могут идти в любом порядке. При выводе на экран список будет пронумерован по возрастанию.

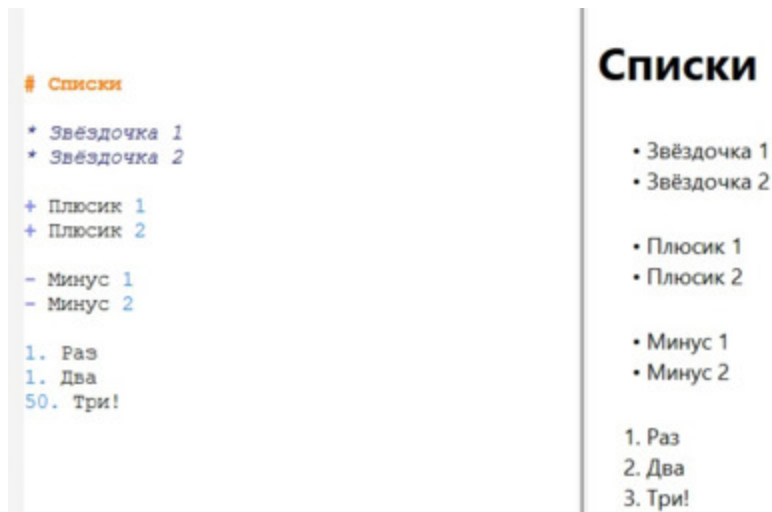


Рис. Оформление списков

Фрагмент кода выделяется пустыми строками — как абзац. Каждая строка начинается с табуляции и четырёх пробелов. Затем при выводе на экран код немного «ужимают».



Рис. Блок кода

В тексте может быть гиперссылка с указанием адреса сайта. Сюда можно добавить текст для всплывающего заголовка. Для оформления ссылки нужны квадратные и круглые скобки. Между

квадратной и круглой скобкой не должно быть пробела.



Рис. Ссылка на сайт

## MARKDOWN В ВЕБ-СЕРВИСАХ

Теперь посмотрим, как работает этот формат применительно к популярным интернет сервисам.

**GitHub.** Загружаем наш файл в репозиторий на github и gitverse. Открываем файл на просмотр и изучаем форматирование текста.



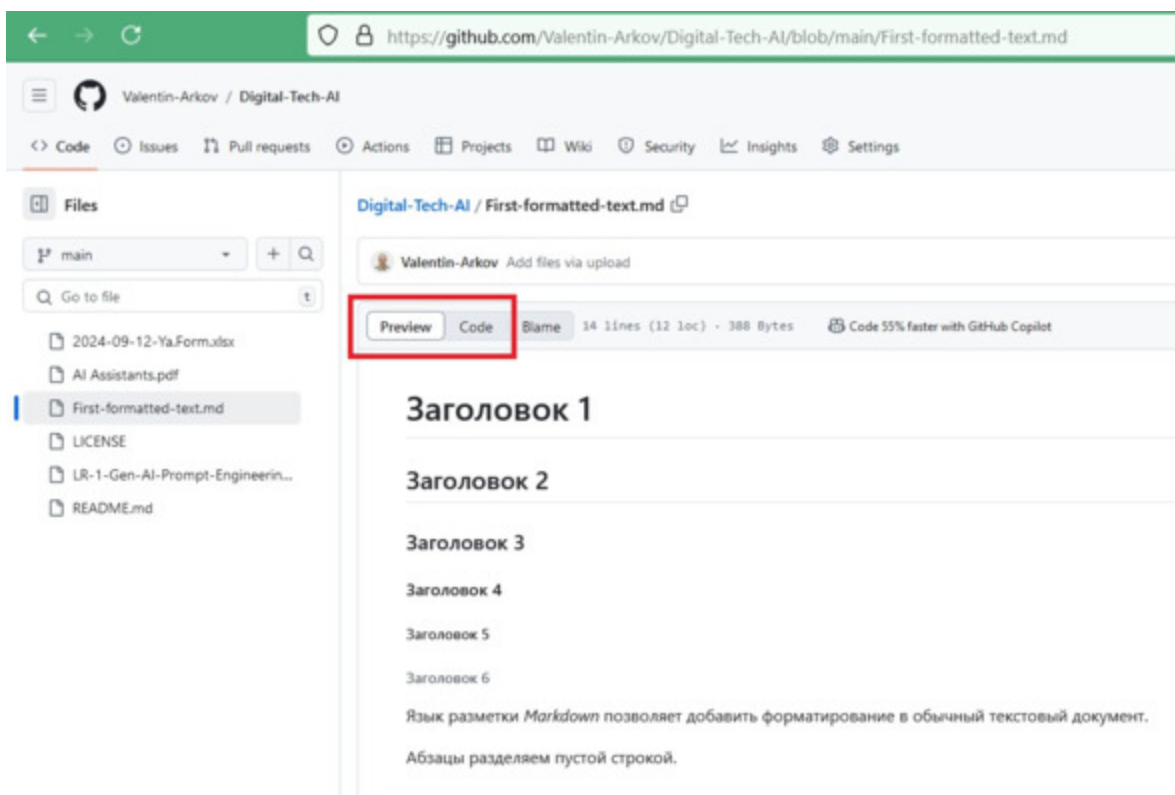


Рис. Файл в репозитории GitHub

После загрузке файла на страничке GitHub появляется возможность просматривать исходный текст и отформатированное представление – это кнопки Preview и Code.

**GitVerse.** Аналогичные кнопки на сервисе GitVerse называются Превью и Код. Кстати говоря, мы можем добавить наш файл в репозиторий разными способами.

Можно просто нажать кнопку Добавить файл над списком файлов и ввести имя файла и его содержимое. Таким образом мы добавляем исходный код, например, путём копирования и вставки через буфер обмена.

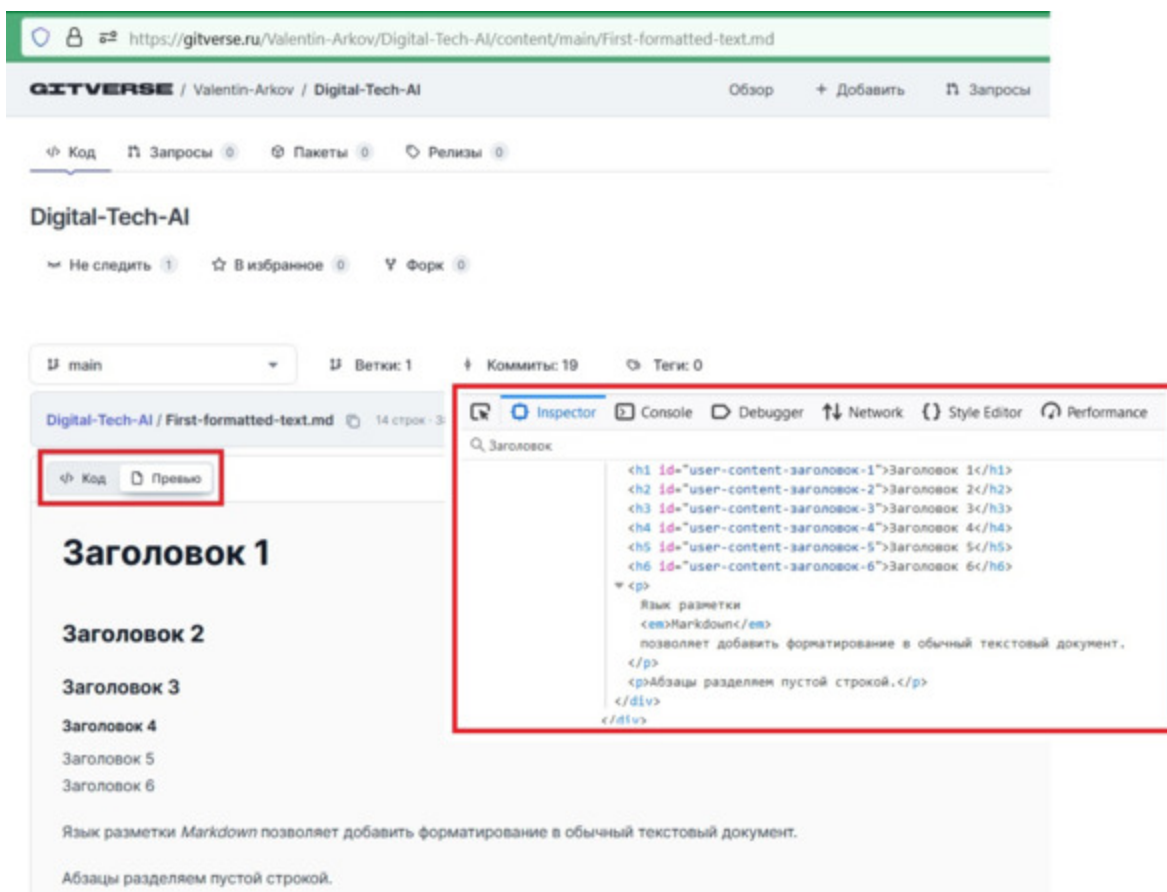


Рис. Файл в репозитории GitVerse

Конечно, есть возможность работать с репозиторием и через интерфейс командной строки CLI – Command Line Interface. Но этим мы займёмся в другой раз.

При просмотре готового текста вызываем исходный текст веб-страницы. В каждом браузере это делается по-своему. Например, в Firefox нажимаем клавишу [F12]. Убеждаемся, что HTML-код соответствует предыдущим примерам.

**Dingus.** Следующий веб-сервис – это [Dingus]. Он открывает возможности для онлайн-тестирования и демонстрации синтаксиса Markdown.

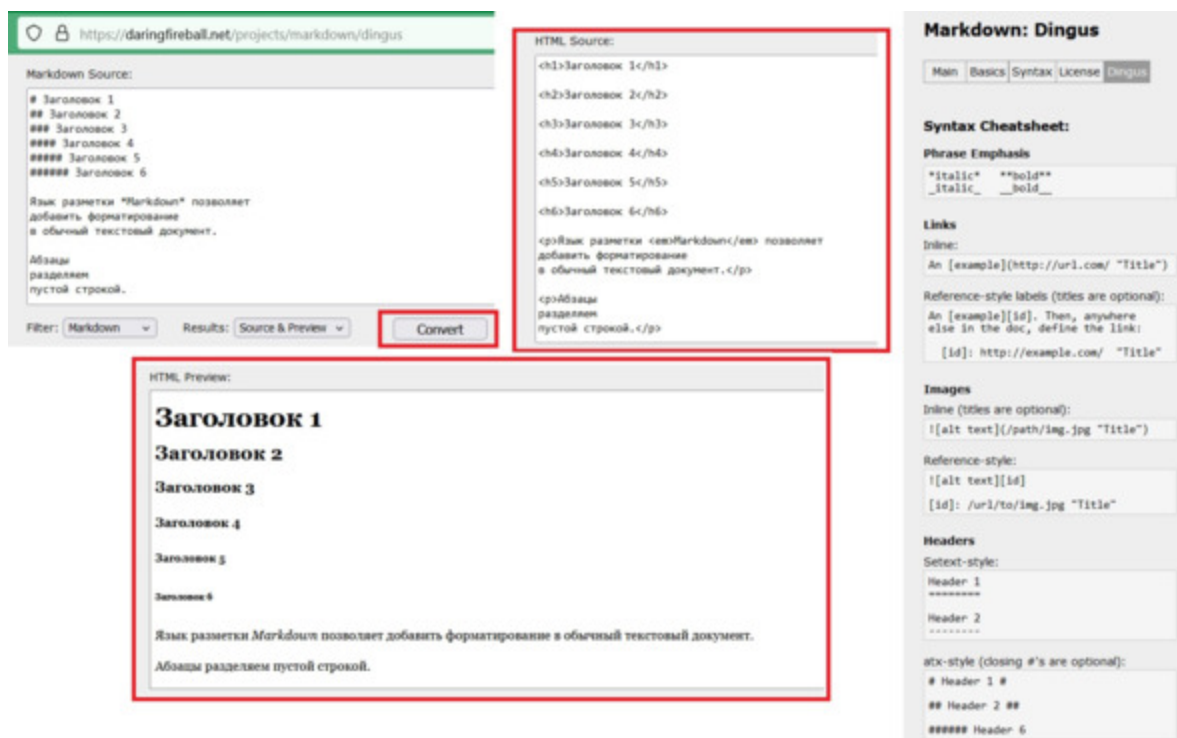


Рис. Работа с Markdown в сервисе Dingus

Мы вставляем наш код в окно Markdown Source и нажимаем кнопку Convert.

Теперь у нас появляется возможность просматривать HTML Source и HTML Preview.

Сравниваем результаты с нашими предыдущими опытами.

Здесь же в правой панели нам выводится шпаргалка по основному синтаксису языка разметки: Syntax Cheatsheet.

**Colab.** Ещё один инструмент для работы с Markdown — это Google Colab. Его основное предназначение — это интерактивная, диалоговая работа программой на Python.

С питоном можно работать двумя способами. Можно написать всю программу в рамках одного файла в Блокноте Windows или в интегрированной среде разработки, а затем вызвать транслятор/компилятор и запустить программу на выполнение — от начала до конца. Это традиционная работа с программированием.

Второй способ — это диалог. В этом случае наша программа будет состоять из отдельных фрагментов, которые оформляются как ячейки. Ячейки можно запускать на выполнение в любом порядке. При этом значение переменных сохраняются в оперативной памяти и доступны на время всего сеанса работы. Вся программа в целом оформлена в виде интерактивного блокнота, или «тетрадки» Jupyter Notebook. Можно установить соответствующий программный пакет типа Anaconda на локальный компьютер либо обратиться к облачному сервису вроде Google Colab. В обоих случаях с таким блокнотом мы работаем через браузер, через веб-интерфейс.

Название Colab — это сокращение от Colaboratory. Это облачная лаборатория для совместной работы с программным кодом. Для начала работы в бесплатном режиме достаточно использовать учётную запись Google.

Создаём новый блокнот и соединяемся с виртуальной машиной, нажав кнопку Connect.

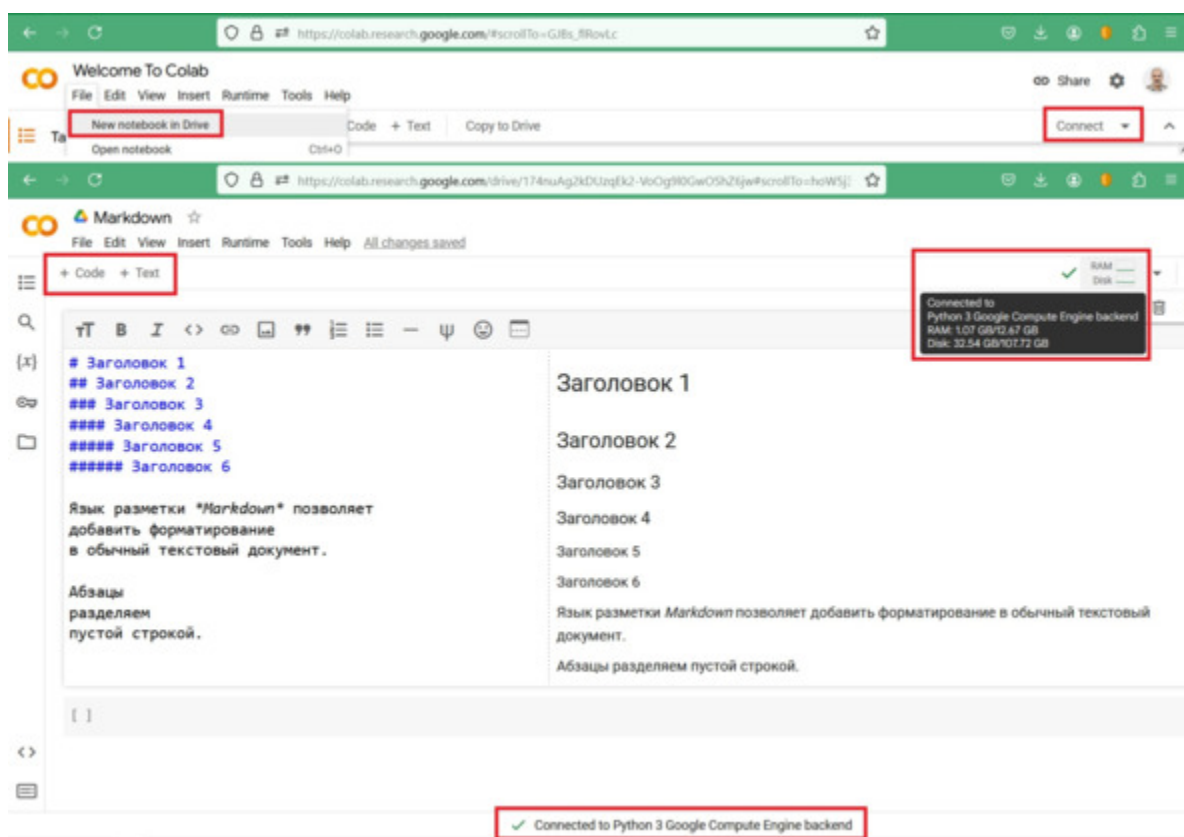


Рис. Работа с Markdown в Google Colab

В этой среде мы можем создавать ячейки с кодом и ячейки с текстом. В кодовых ячейках будут фрагменты программы на Python. В текстовых ячейках можем расположить Markdown. Для запуска конкретной ячейки на выполнение можно нажать комбинацию клавиш [Shift + Enter]. Для редактирования текстовой ячейки мы просто дважды щёлкаем по её содержимому.

Создаём текстовую ячейку и вводим наш готовый материал. Запускаем. Проверяем.

Чтобы сохранить нашу работу в файле, вызываем через меню File — Save или комбинацию клавиш [Ctrl + S]. При этом файл сохраняется на облачном диске Google Drive.

Проверяем, где именно сохраняется наш блокнот. Для этого открываем в браузере или приложении облачный диск и находим каталог Colab Notebooks.

Есть возможность скачать наш файл на локальный компьютер в виде исходного текста программы на Python \*.py либо в виде файла блокнота ipynb, в котором находятся текстовые и кодовые ячейки, а также результаты вывода на экран. для скачивания файла выбираем в меню раздел File – Download и указываем тип файла.

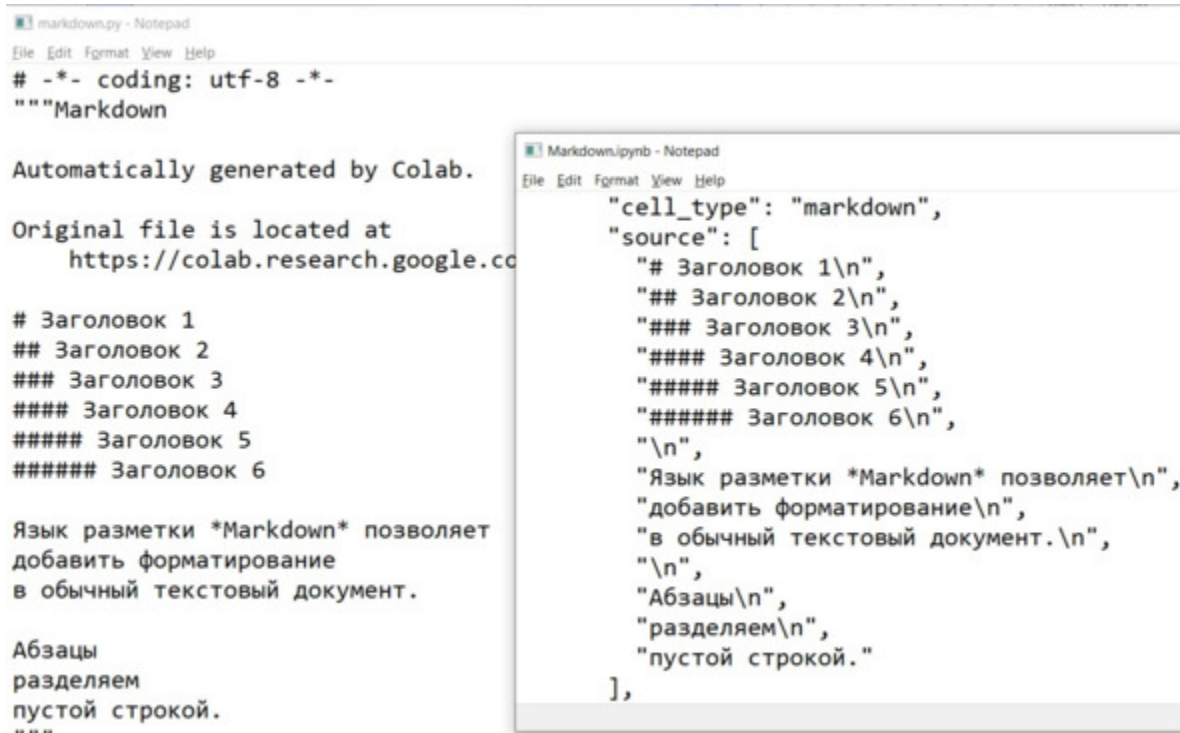


Рис. Файлы \*.py и \*.ipynb

Скачиваем оба варианта и изучаем их содержимое.

Файл типа \*.py содержит текст программы Python. Текстовые ячейки здесь представлены в виде многострочных комментариев и выделены тройными кавычками.

Файл формата \*.ipynb – это ipython (interactive python) notebook. Здесь сохраняется разбиение блокнота на ячейки. Содержимое оформлено фактически в формате json. В дополнение к нашему тексту в конце каждой строки добавлен символ перевода строки \n – от английского new line – новая строка.

## DOCKER + ANACONDA

Наконец попробуем запустить диалоговый, интерактивный блокнот на локальном компьютере. Чтобы не заниматься установкой дополнительных программ, мы будем использовать популярную технологию контейнеризации приложений Docker.

Нам понадобится приложение под названием Anaconda. Это тоже название змеи и это намёк на язык программирования Python.

Мы будем запускать этот программный продукт в виде контейнера. Для этого нам понадобится предварительно установить среду запуска контейнера docker. технические подробности зависят от конкретной операционной системы. в рамках наших лабораторных работ мы используем OS Microsoft Windows и Docker Desktop.

В дисплейных классах нашей кафедры уже установлено необходимое программное обеспечение. Если у читателя появится желание самостоятельно установить эти средства на своём домашнем компьютере или ноутбуке, можно ознакомиться с более подробным описанием технологии, которое приводится в нашем учебном пособии и видеороликах, см. [Operating-Systems Repo].

В начале мы запускаем программу Docker Desktop с правами администратора.

В нижней части окна программы запускаем командную строку. Затем в этом терминале вводим длинную, сложную команду, которую можно сформировать с помощью интеллектуального помощника. Запоминать эту команду пока не требуется.

как запустить контейнер anaconda для работы с Jupyter lab из приложения docker desktop

### 3. Sanyok konteynera c Jupyter Lab

- Выполните следующую команду в терминале, чтобы запустить контейнер и установить Jupyter Lab:

```
docker run -it -p 8888:8888 continuumio/miniconda3 /bin/bash -c "conda install jupyterlab -y && jupyter lab --ip='*' --port=8888 --no-browser --allow-root"
```

- Здесь `'-p 8888:8888'` связывает порт 8888 вашего хоста с портом 8888 контейнера, что позволяет вам получить доступ к Jupyter Lab через браузер

#### 4. Доступ к Jupyter Lab

- После запуска контейнера откройте браузер и перейдите по адресу `http://localhost:8888`. Вам может понадобиться токен для доступа, который будет выведен в терминале:

## 5. Windows-as-a-Docker Desktop

- Вы также можете управлять контейнерами через интерфейс Docker Desktop.

1. Перейдите на вкладку Images.
2. Найдите образ "cont/images/reincondal" и нажмите Run.
3. В открывшемся окне настройте параметры запуска, такие как порты, и нажмите Run.

```

Terminal
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross platform PowerShell https://aka.ms/powershell

PS C:\Users\Valentin> docker run -it -p 8888:8888 continuumio/whiconda3 /bin/bash -c "conda install jupyterlab -y && jupyter lab --ip="*" --port=8888 --no-browser --allow-root"
--
Unable to find image 'continuumio/whiconda3:latest' locally
latest: Pulling from continuumio/whiconda3

```

```

Terminal
To access the server, open this file in a browser:
file:///root/.local/share/docker/runtime/userver-5-open.html

Or copy and paste one of these URLs:
http://ocalhost:8080/?tabtoken=69f7c334c89436263358b6a555b6e713edafad3f6be3e
http://127.0.0.1:8080/?tabtoken=69f7c334c89436263358b6a555b6e713edafad3f6be3e

[i 2024-08-23 12:28:58 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-lint, java-language-server, julia-language-server, perl, python-language-server, python-lsp-server, python-lsp-server, ts-l-language-server, ts-lint, typescript-language-server, unified-language-server, vscode-css-language-server-b1a, vscode-html-language-server-b1a, vscode-javascript-language-server, yaml-language-server

```

Рис. Инструкция по запуску контейнера

При первом запуске контейнера происходит скачивание необходимых материалов с облачного сервиса Docker Hub. Это происходит автоматически, без нашего участия.

После успешного запуска контейнера в терминале выводится ссылка для подключения к блокноту.

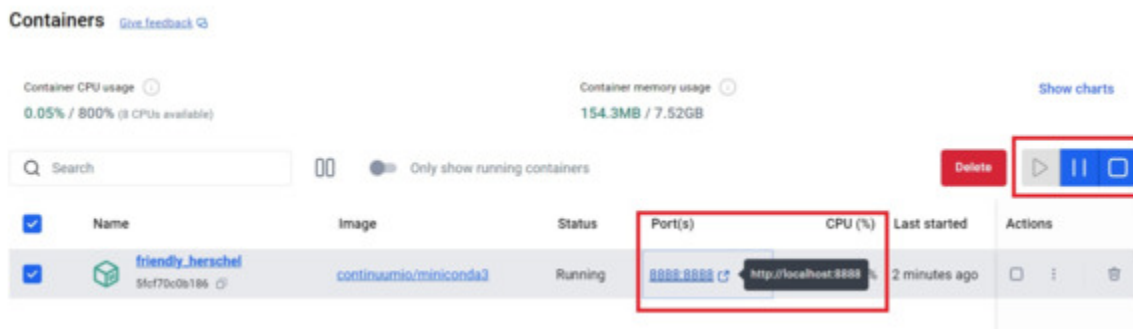


Рис. Запуск контейнера Anaconda



Щёлкаем мышкой по этой ссылке, и в браузере открывается сервис Jupyter Lab.

На вкладке Launcher запускаем диалоговый блокнот Notebook Python 3.

Появляется окно нашего блокнота. Переименуем его: File – Rename Notebook.

Здесь мы вручную можем изменять тип содержимого ячейки – кодовая (Code) или текстовая (Markdown).

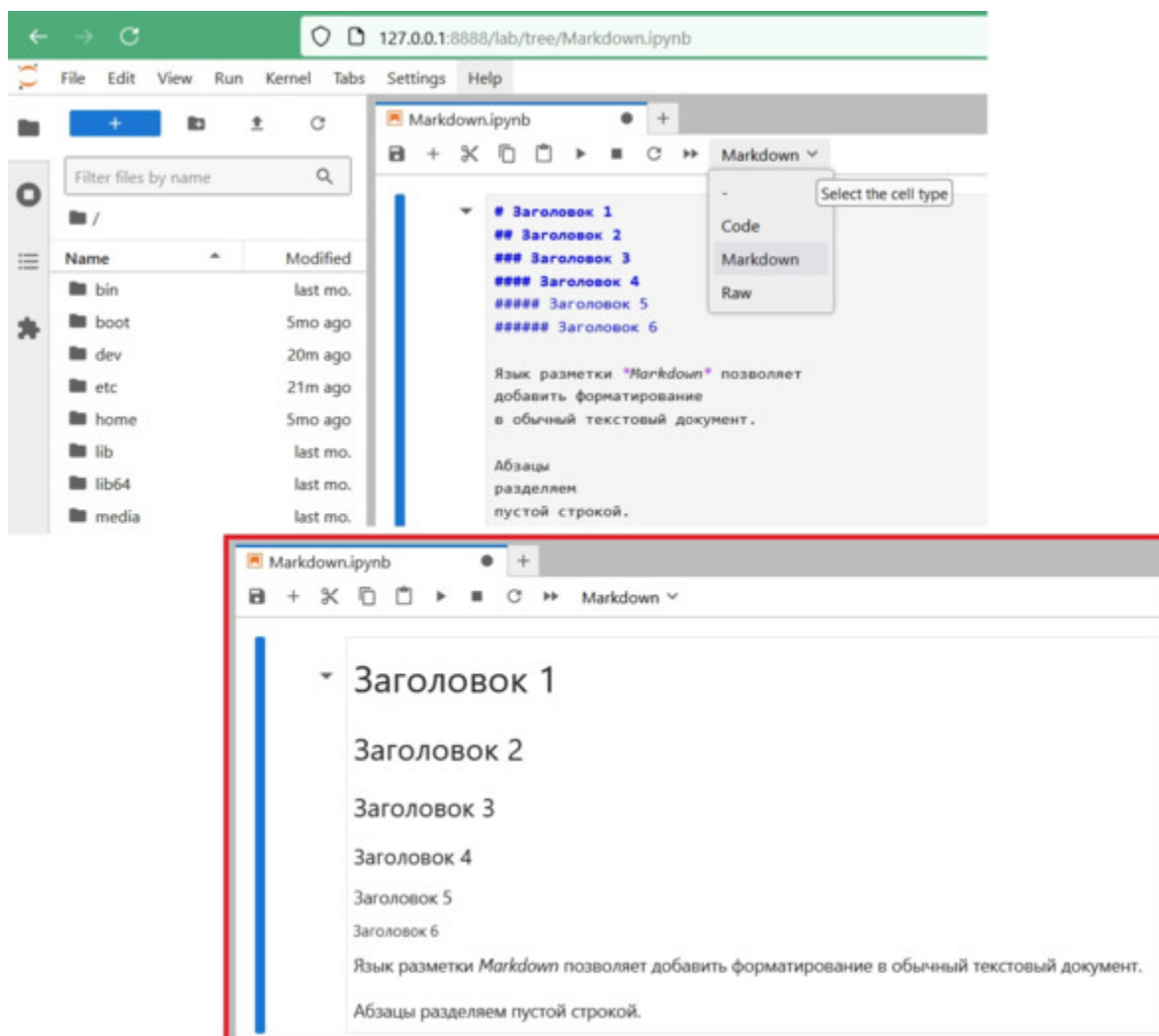


Рис. Jupyter Lab в браузере

Устанавливаем тип ячейки — текстовая. Вставляем в текстовую ячейку уже отработанное нами содержимое. Запускаем ячейку на выполнение.

Здесь тоже есть возможность скачать блокнот в формате \*.ipynb.

Мы можем завершить работу с контейнером и остановить его выполнение. Когда мы запустим его в следующий раз, нам уже не понадобится терминал/консоль. Выбираем контейнер в списке и нажимаем кнопку Пуск.

Для управления выполнением контейнера имеются традиционные кнопки Пуск, Стоп и Пауза.

Для перехода в веб-интерфейс нужно будет щёлкнуть по ссылке в графе «Порты». Кстати, в этом случае «порт» означает целое число, которое нужно указать для доступа к программному сервису.

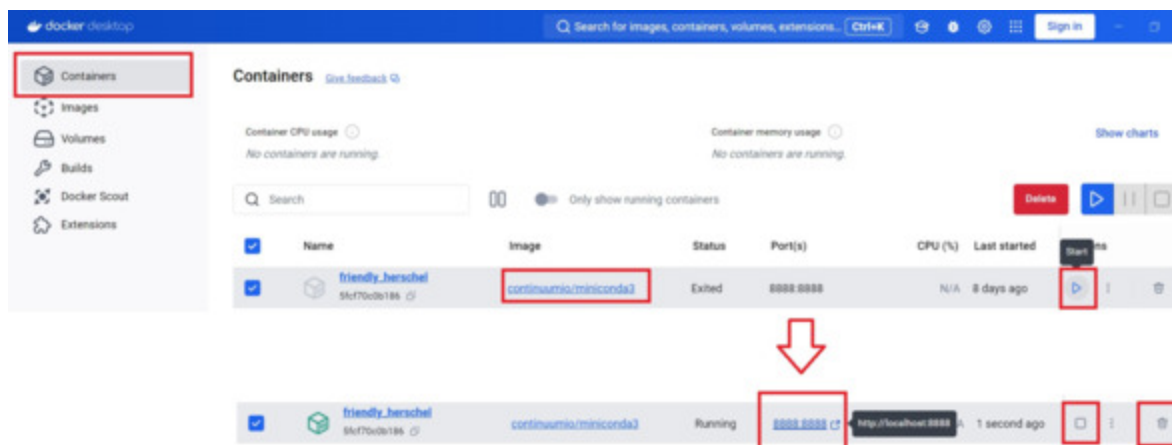


Рис. Повторный запуск контейнера

Для повторного запуска контейнера у нас всё уже готово и скачено из интернета. Выбираем вкладку Containers — Контейнеры. Затем в списке контейнеров находим тот, который нам нужен. Для этого смотрим в колонку Image — Название образа. Нас

интересует образ Conda – то есть Anaconda. Нажимаем кнопку Пуск – Start.

Запуск происходит очень быстро. Практически мгновенно. Теперь в колонке Порты – Ports становится активной ссылка на номер порта. Наводим на неё курсор и видим всплывающую подсказку – адрес сервиса – localhost. Щёлкаем по ссылке и в браузере открывается Jupyter Lab.

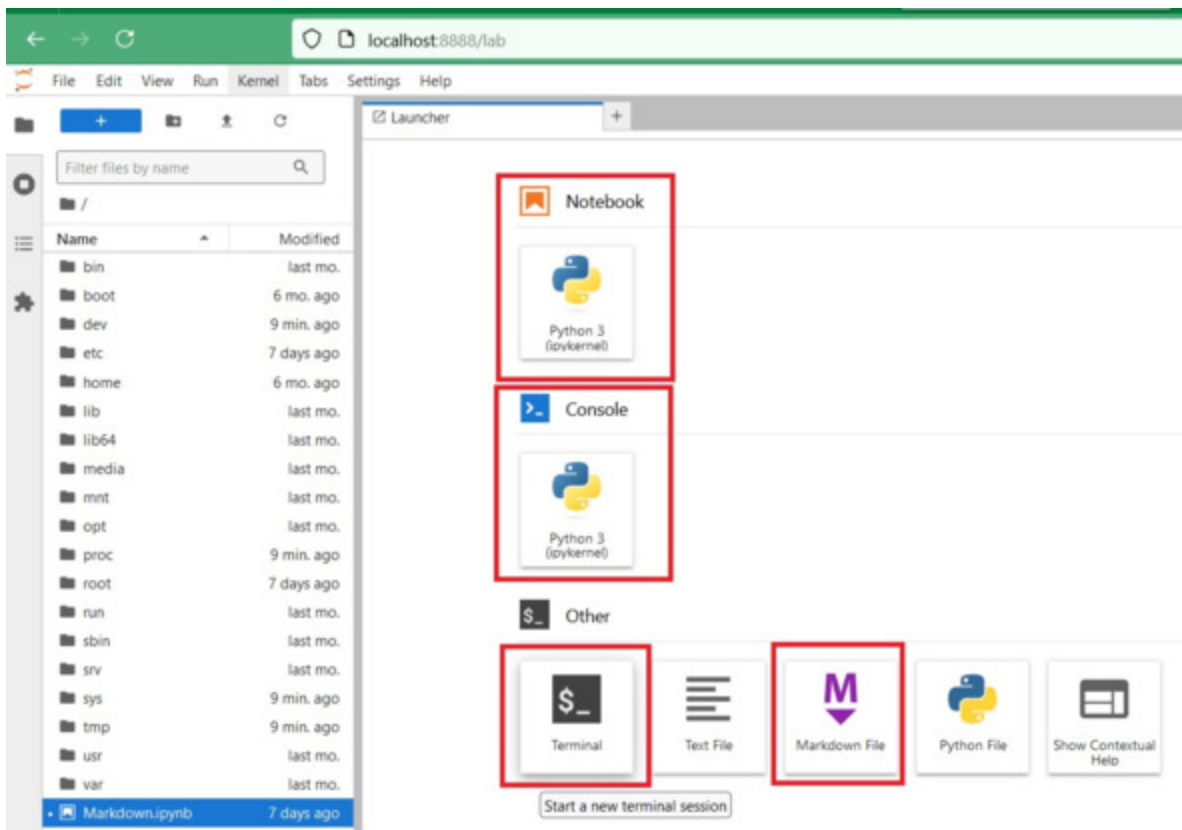
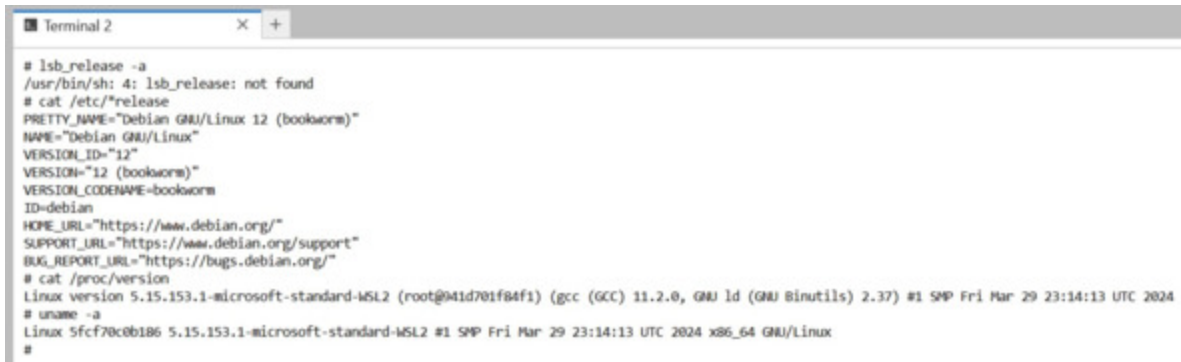


Рис. Страница запуска Jupyter Lab

В левой части окна имеется панель со списком файлов и каталогов. Здесь можно найти наш сохранённый ранее блокнот. Список каталогов явно указывает на то, что здесь развёрнут Linux.

В правой части окна имеется возможность не только поработать с Питоном, но и открыть текстовый файл, файл с разметкой и даже терминал операционной системы (ОС). Запускаем терминал:

Terminal — Start a new terminal session — Начать новый сеанс работы в терминале ОС.



```
Terminal 2
# lsb_release -a
/usr/bin/sh: 4: lsb_release: not found
# cat /etc/*release
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION_ID="12"
VERSION="12 (bookworm)"
VERSION_CODENAME=bookworm
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
# cat /proc/version
Linux version 5.15.153.1-microsoft-standard-WSL2 (root@041d701f84f1) (gcc (GCC) 11.2.0, GNU ld (GNU Binutils) 2.37) #1 SMP Fri Mar 29 23:14:13 UTC 2024
# uname -a
Linux 5fcf70c0b186 5.15.153.1-microsoft-standard-WSL2 #1 SMP Fri Mar 29 23:14:13 UTC 2024 x86_64 GNU/Linux
#
```

Рис. Выясняем версию Линукса

Вводим несколько стандартных команд — их можно запросить у любого чат-бота: «какими командами можно узнать версию linux в консоли».

Как видим, внутри нашего контейнера запущен debian Linux, а для его обслуживания работает Microsoft WSL. Про этот этот сервис можно почитать в нашем пособии [Виртуализация].

**Задание.** Выясните, где ещё используется язык разметки Markdown — в каких программах и на каких сайтах или сервисах.

При работе с Markdown, например, в среде Google Colab можно работать с текстом и одновременно видеть окончательное оформление и форматирование.

**Задание.** Проверьте, как работают рассмотренные инструменты форматирования в разных средах.

**Задание.** Выясните, какие текстовые редакторы и среды разработки поддерживают формат Markdown.

## ДОПОЛНИТЕЛЬНЫЕ ИНСТРУМЕНТЫ ФОРМАТИРОВАНИЯ

Кроме заголовков и шрифтов, у нас в распоряжении есть и более продвинутые инструменты: таблицы, рисунки, HTML теги, формулы TeX и так далее.

Таблицы оформляются очень просто. при этом текст по-прежнему легко воспринимается человеком. Мы организуем столбцы и строки с помощью следующих приёмов.

Вертикальная черта разделяет ячейки, столбцы.

Заголовок подчёркиваем дефисами-чёрточками.

Содержимое столбцов можно прижать влево, вправо или «отцентрировать». Для этого используется символ «двоеточие», см. рис.

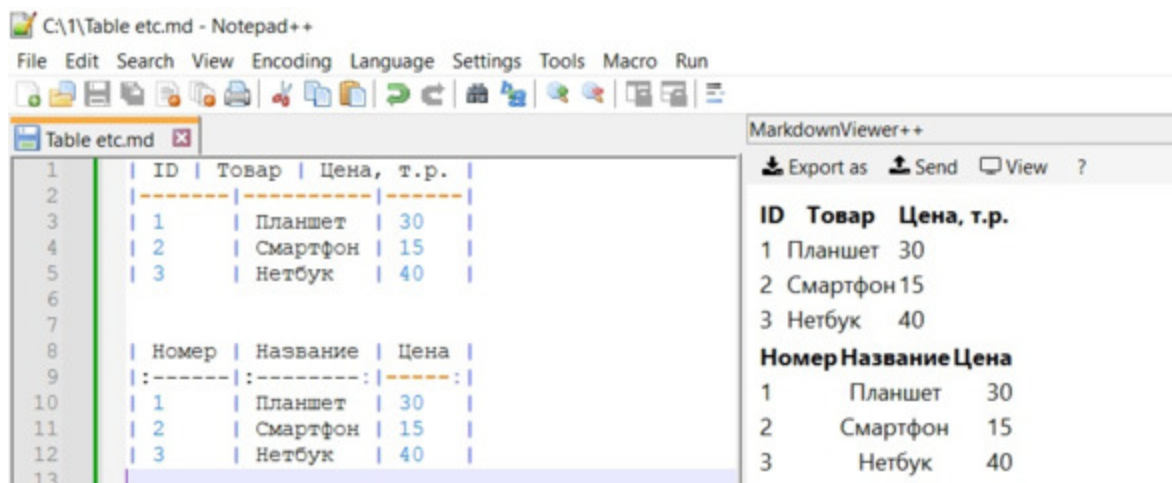


Рис. Оформляем таблицу в блокноте

Каждая среда разработки будет обрабатывать такие инструкции немного по-своему, см. рис. Поэкспериментируйте и посмотрите, как отображается одна и та же таблица разными инструментами.

ID	Товар	Цена, т.р.
1	Планшет	30
2	Смартфон	15
3	Нетбук	40

Номер	Название	Цена
1	Планшет	30
2	Смартфон	15
3	Нетбук	40

ID	Товар	Цена, т.р.
1	Планшет	30
2	Смартфон	15
3	Нетбук	40

Номер	Название	Цена
1	Планшет	30
2	Смартфон	15
3	Нетбук	40

Рис. Оформляем таблицу в Jupyter Lab

Что касается вставки рисунков, у нас есть разные варианты. Можно взять рисунок как отдельный файл и написать инструкцию по вставке этого файла.

Создадим изображение, например, в графическом редакторе Paint и сохраним в файле. Затем перетащим этот файл из проводника в окно Jupyter Lab — в панель со списком файлов.

Двойным щелчком запускаем просмотр изображения. Убеждаемся, что он нормально читается и воспринимается.

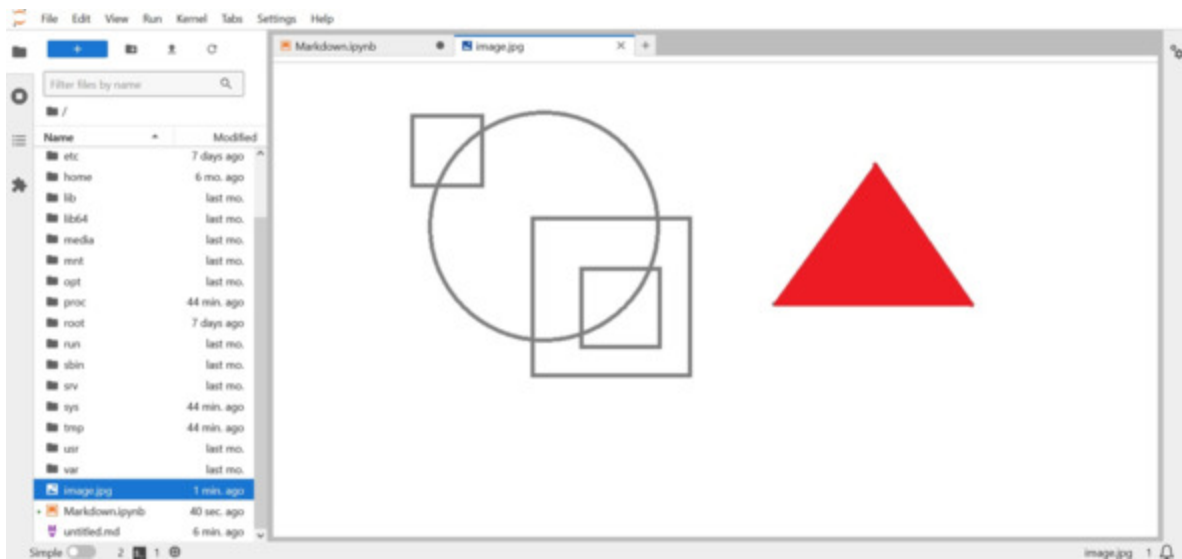


Рис. Просмотр изображения

Теперь переходим к файлу Markdown и в новой текстовой ячейке пишем инструкцию, как показано на рисунке:

- восклицательный знак
- название в квадратных скобках
- ссылка в круглых скобках
- альтернативное название для всплывающей подсказки в кавычках

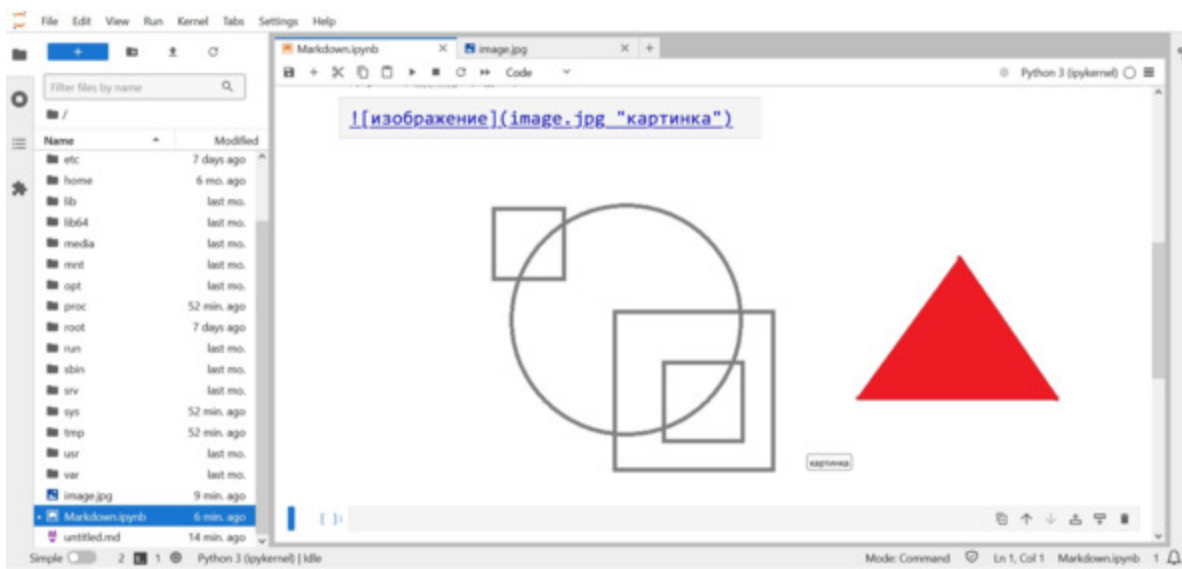


Рис. Вставка изображения

Можно также указывать ссылку на рисунок, расположенный в интернете на каком-нибудь сайте. Попробуйте это сделать.

В некоторых случаях мы можем вставить сам рисунок в наш файл.

Например, над текстовой ячейкой Google Colab при редактировании текстового содержимого появляется кнопка Вставить изображение – Insert image.

Выбираем файл на локальном компьютере, и он вставляется в виде текста в особой кодировке base64. Эта технология традиционно используется при отправке двоичных файлов через текстовые протоколы, например, по электронной почте. Здесь мы видим только обычные буквы и цифры, а за ними скрывается наша картинка.

Теперь наш документ хранится в виде одного файла, и все рисунки хранятся внутри него. Если мы «поделимся» таким документом, «расшарим» его, он отобразится на другом компьютере в своём первоначальном виде – безо всяких лишних телодвижений по копированию дополнительных файлов.



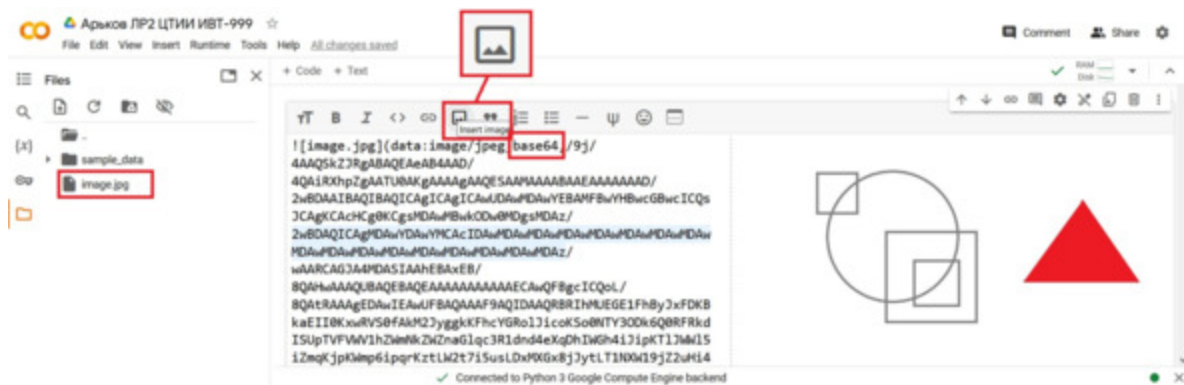


Рис. Вставка изображения в Colab

Следующая возможность — это использование HTML тегов. С одной стороны, это дополнительный функционал, с другой стороны, HTML читается гораздо тяжелее для обычного человека.

Тем не менее, попробуем этот инструмент в действии. Мы можем вставить любой html-код для того, чтобы оформить заголовки, списки, таблицы и ссылки.

## Заголовок 1 уровня

Абзац текста.

Другой абзац!

## Заголовок 2 уровня

Полный абзац...

Материалы по DT-AI



### Список

- Первый пункт
- Второй пункт

### Табличка

Столбец 1	Колонка 2
Ячейка Раз	Ячейка Два

```
<h1>Заголовок 1 уровня</h1>
<p>Абзац текста.</p>
<p>Другой абзац!</p>
<h2>Заголовок 2 уровня</h2>
<p>Полный абзац...</p>
<a href="https://github.com/Valentin-Arkov/Digital-Tech-AI" target="_blank">Материалы по DT-AI</a>

<h3>Список</h3>
<ul>
<li>Первый пункт</li>
<li>Второй пункт</li>
</ul>
<h3>Табличка</h3>
<table>
<tr>
<th>Столбец 1</th>
<th>Колонка 2</th>
</tr>
<tr>
<td>Ячейка Раз</td>
<td>Ячейка Два</td>
</tr>
</table>
```

Рис. Теги HTML в Jupyter Lab

Как видим, возможности те же, но наглядность немного пострадала. Для простейшего, базового форматирования текста обычно хватает основных возможностей Markdown.

## ФОРМУЛЫ TEX

В дополнение к привычным средствам форматирования, мы можем вставлять в документ многэтажные формулы. Здесь используется особая система построения формул, специальные условные обозначения элементов формул — нотация TeX. Компьютерных науках «нотация» — это система условных обозначений и правил построения диаграмм или формул.

Английское название TeX так и читается: «тех». Это намёк на греческое слово «техно», которое означает «искусство» или «мастерство». И пишется оно именно так: «Т» большая, «е» маленькая, «Х» большая. Это название текстового редактора, который разработал американский математик и программист Дональд Кнут. Его не устраивало то, как математические журналы публиковали его статьи с многэтажными формулами. Так что он решил разработать свой собственный инструмент.

В настоящее время чаще используют доработанный вариант программы TeX под названием LaTeX — в честь автора доработки Лесли Лампорта — так что это Ла (мпорт) + TeX. Читается «латех».

Из этого редактора мы будем использовать только формулы. Их можно вставлять в чистом виде в файлы формата Markdown.

Чтобы познакомиться с нотацией TeX, мы откроем редактор формул текстового процессора (редактора) Microsoft Word. В верхнем меню выбираем Вставка — Уравнение — Встроенные — Бином Ньютона, см. рис.

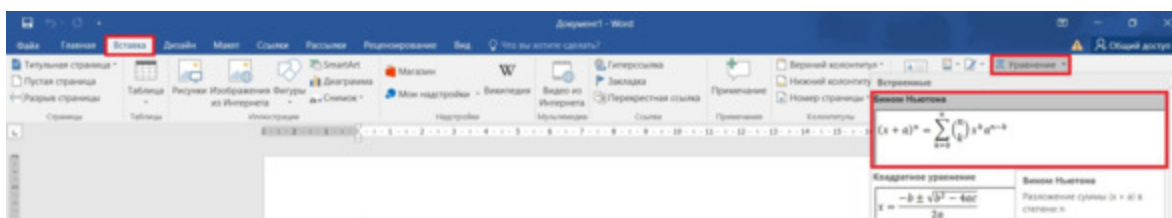


Рис. Вставляем готовую формулу

После вставки формулы автоматически открывается вкладка Конструктор – Design.

При этом уравнение формула отображается в привычном виде. мы можем работать с этим объектом визуальнo. Такой подход называется WYSIWYG – What you see is what you get – «вы получаете то, что вы видите» или «что вы видите, то и получаете». В таком режиме можно работать очень простыми формулами. Для сложных конструкций нужны продвинутые инструменты.

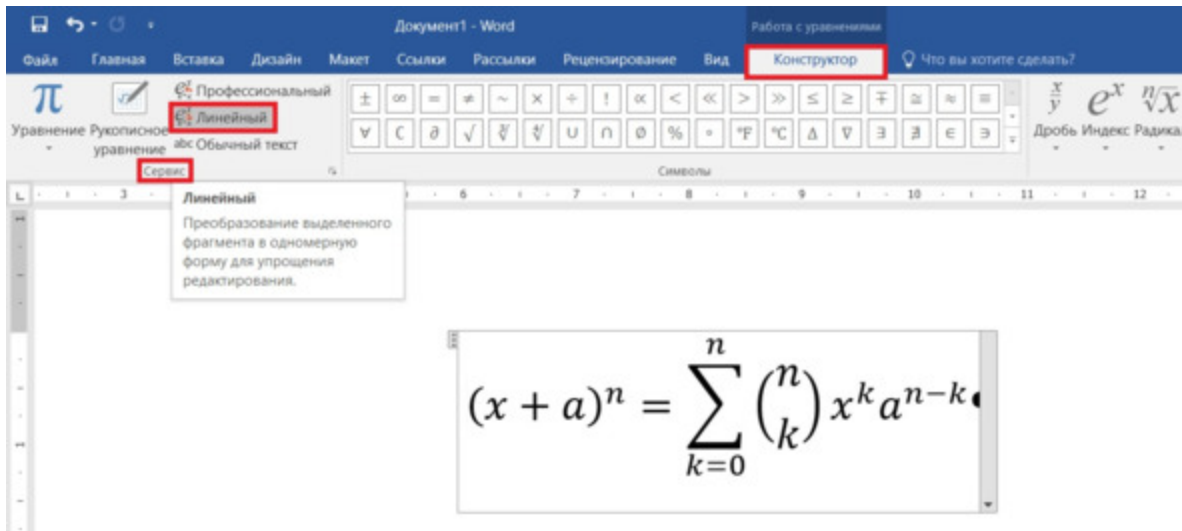


Рис. Вкладка Конструктор

Теперь, если выбрать режим отображения Линейный, мы получим другое представление нашего уравнения, см. рис. Теперь это одна строчка символов. По-английски line – это уже вообще-то

не «линия», а «строчка текста».

Мы можем переключаться между разными формами представления уравнения.

Эта функция доступна в последних версиях офисного пакета — Microsoft 365 и Office 2019.

Здесь мы видим, как вводится показатель степени через символ «крышки» ^:

$$(x + a)^n.$$

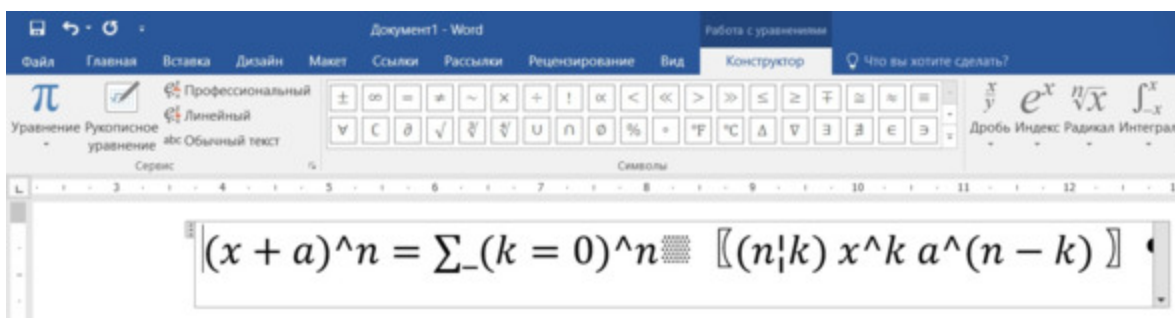


Рис. Линейный режим

Многие редакторы Markdown поддерживают нотацию TeX либо напрямую (встроенными средствами), либо после установки дополнительных модулей.

В частности, Google Colab и GitHub сразу работают с такими формулами, см. рис.

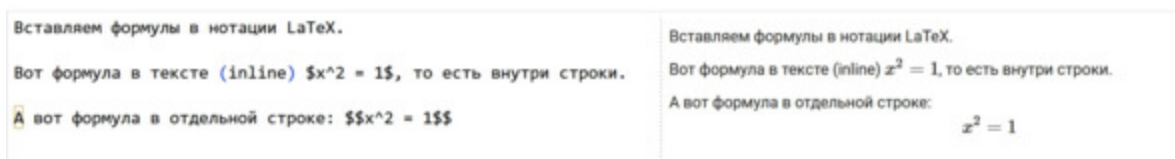


Рис. Формулы в Colab

Как видим, формула в одинарных символах доллара выводится внутри текста, как часть строки. Если же использовать двойные символы доллара, мы получаем формулу в отдельной строке, причём

расположенную по центру.

И вот здесь-то и начинается самое интересное. Мы можем использовать нотацию TeX в наших запросах к нейросети и можем просить выводить уравнение в этом формате. Например мы попросим описать вычисление суммы геометрической прогрессии и предложим вывести результат в нотации TeX. Затем мы копируем ответ, нажимая кнопку Копировать в буфер обмена. Теперь мы вставляем этот ответ из буфера обмена в текстовую ячейку Colab, см. рис.

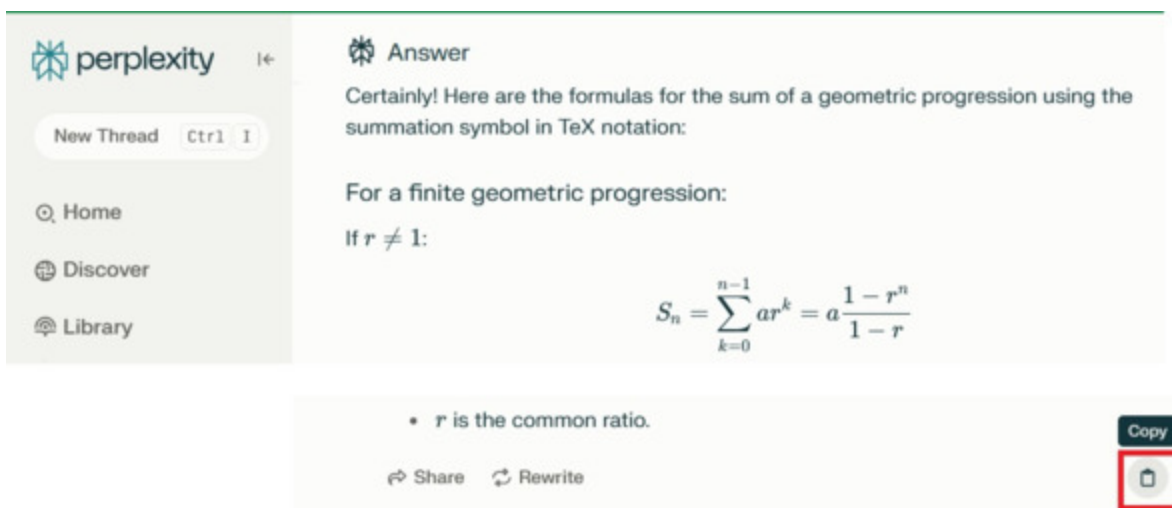


Рис. Копируем ответ нейросети в нотации TeX

Certainly! Here are the formulas for the sum of a geometric progression using the summation symbol in TeX notation:

### For a finite geometric progression:

If  $r \neq 1$ :

$$S_n = \sum_{k=0}^{n-1} ar^k = a \frac{1 - r^n}{1 - r}$$

If  $r = 1$ :

$$S_n = \sum_{k=0}^{n-1} a = na$$

Certainly! Here are the formulas for the sum of a geometric progression using the summation symbol in TeX notation:

For a finite geometric progression:

If  $r \neq 1$ :

$$S_n = \sum_{k=0}^{n-1} ar^k = a \frac{1 - r^n}{1 - r}$$

If  $r = 1$ :

$$S_n = \sum_{k=0}^{n-1} a = na$$

Рис. Вставляем ответ в нотации TeX в ячейку Colab

Теперь рассмотрим первую формулу поподробнее, см. рис.

Запись  $S_n$  означает большая буква  $S$  с нижним индексом  $n$ .

Знак суммы —  $\sum$  — заглавная греческая буква Сигма.

Начальное значение индекса, с которого начинается суммирование, это символ нижнего подчёркивания, за которым идёт выражение в фигурных скобках:  $\_ {k=0}$ .

Верхний предел значения индекса для суммирования указан через символ «крышечка»:  $^ {n-1}$ .

Показатель степени тоже указывают через «крышку».

Далее идёт результат вычислений — это дробь — по-английски fraction.

Суммы, дроби, интегралы вводятся через символ обратной наклонной черты — back slash, после которой написано название конструкции.

Для дроби указывают числитель и знаменатель.

Длинные выражения пишут в фигурных скобках.

Одиночный символ (например, показатель степени) можно вводить и без скобок. При этом фигурные скобки на экран не выводятся.

$$S_n = \sum_{k=0}^{n-1} ar^k = a \frac{1 - r^n}{1 - r}$$

Рис. Формула со знаком суммы

Мы можем сформулировать задание для нейросети с использованием нотации TeX. Например, можно попросить определить значение интеграла. Для начала мы отработаем саму формулу в Colab, см. рис.

$$I = \int_0^1 \frac{4}{1+x^2} dx$$

Рис. Формула интеграла в Colab

Затем используем это выражение в запросе к нейросети.

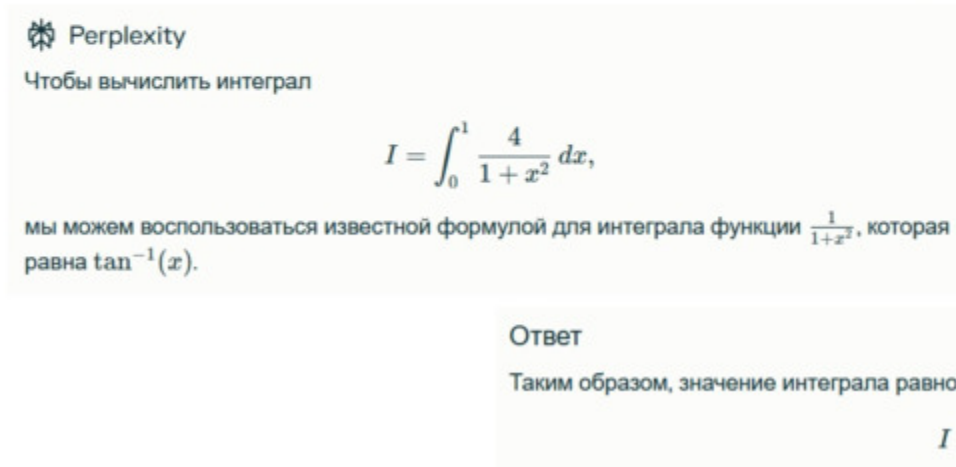


Рис. Запрос с использованием нотации TeX

**Задание.** Сформулируйте несложный запрос к нейросети с использованием нотации TeX. Это может быть возведение выражения в степень или упрощение выражения либо вычисление производной или интеграла.

Формулы в нотации TeX мы встречаем всё чаще и даже можем не обращать на них внимания. Например, «народная/свободная энциклопедия» Википедия широко использует этот формат в своих статьях.

К примеру, находим статью «Нормальное распределение» и копируем фрагмент текста с формулой/ Вставляем этот фрагмент в Блокнот и рассматриваем полученный текст, см. рис..



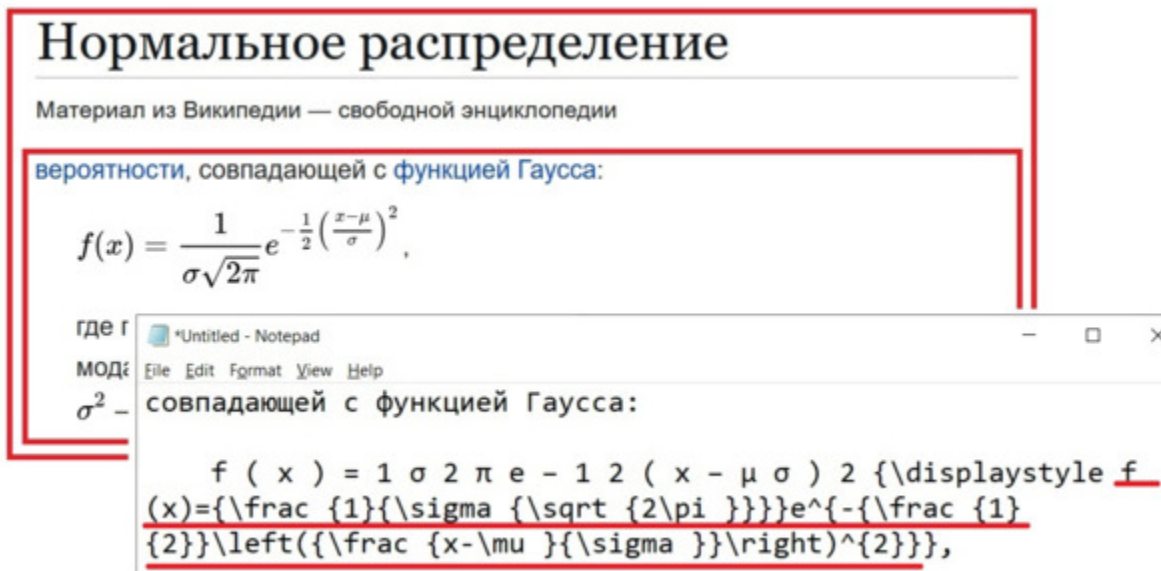


Рис. Формула из Википедии в Блокноте

Здесь скрывается нотация TeX внутри фигурных скобок  $\{\displaystyle \dots\}$  — на рисунке подчёркнуто красным.

Эту «внутренность» мы можем вставить в текстовую ячейку Colab и оформить как уравнение TeX — достаточно добавить «обрамление» знаками доллара, см. рис.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Рис. Формула для нормального распределения

На сайте Википедии можно даже задавать поиск по фрагментам формул TeX, см. рис.

Как видим, в поисковой выдаче явно показаны фрагменты формул, начинающиеся с конструкции `displaystyle`. Если же перейти

к просмотру выбранной статьи, мы увидим уже красиво оформленные математические формулы.

**Задание.** Найдите в Википедии любую статью с формулой, использующей конструкцию `\frac`, то есть дробь или отношение. Скопируйте фрагмент текста с этой формулой, изучите его в Блокноте и вставьте в текстовую ячейку Google Colab так, чтобы она превратилась в красивую математическую формулу.

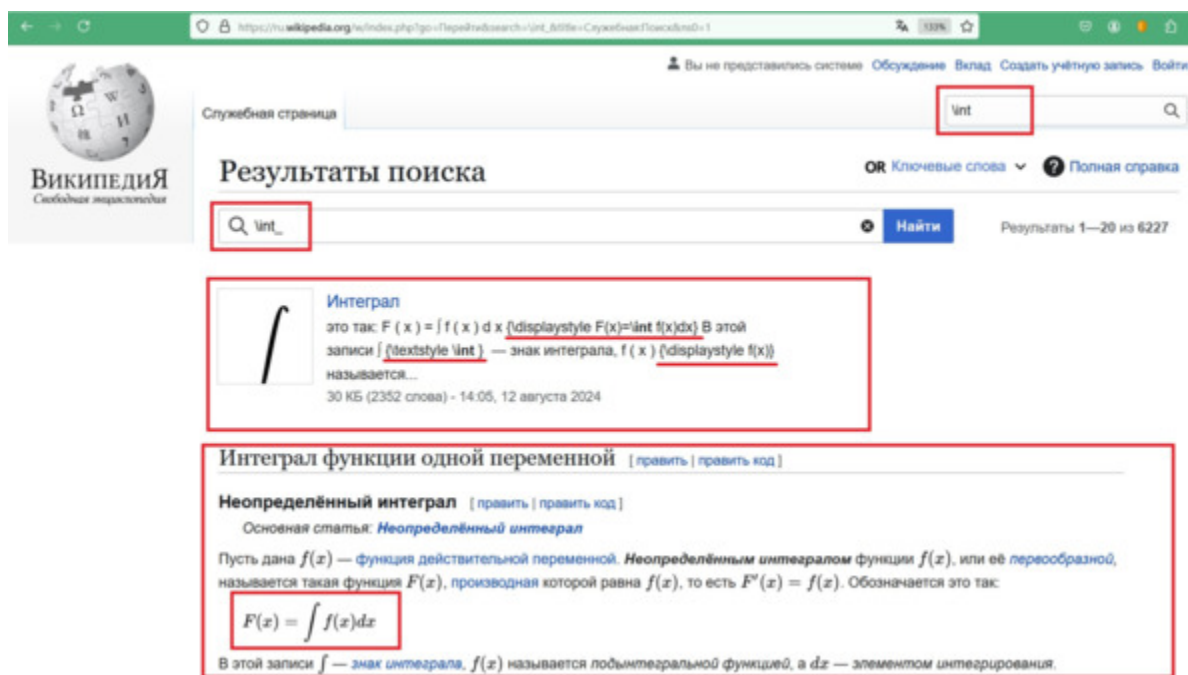


Рис. Поиск формул TeX на Википедии

**Задание.** Просмотрите статью [Википедия: Формулы]. Обратите внимание на то, как используется нотация TeX в статьях Википедии.

Отдельная проблема и технология, относящаяся к формулам, — это греческие буквы. Они часто используются формулах, потому что многие разделы математики были описаны в популярных работах древнегреческих учёных. С тех пор эти символы стали международным языком, и мы по-прежнему используем древние

названия этих букв. Хотя в современном греческом языке они уже называются по-другому. Например, буква «бета» превратилась в «виту», но в формулах это всё равно «бета»!.

Некоторые символы вообще не используются, они просто исчезли. Для некоторых букв есть так называемое альтернативное изображение (название начинается на `\var`, например, `\varkappa`). И это всё — наша грамотность — инженерная, математическая, компьютерная.

При разработке компьютерного шрифта для формул TeX были нарисованы почти все греческие буквы, но не все. Если греческая буква выглядит так же, как буква латинского алфавита, то для неё мы не найдём «TeXовского» изображения. Вместо этого придётся использовать подходящую латинскую букву. Это касается только больших, заглавных греческих букв. И это делается по-разному в разных редакторах

**Задание.** Используя раздел «Греческий алфавит» из предыдущей статьи, а также статью [Греческий алфавит], оформите таблицу греческих букв (больших и маленьких) и их русских и английских названий — как они используются в математических формулах и как они называются в современной Греции.

Ещё один пример ситуации, где мы сталкиваемся с формулами TeX — это сайт библиотеки машинного обучения [SKLearn]. Расшифровывается это название как Sci (Scientific — Научный) + Kit (набор инструментов) + Learn (Machine Learning — Машинное обучение). В результате получаем: научная библиотека с набором инструментов для машинного обучения на Питоне. У нас ещё будет отдельное занятие по этой технологии. Но пока мы посмотрим только на технологию оформления материалов.

Открываем сайт библиотеки SKLearn и переходим по ссылке в раздел Регрессия — Метод наименьших квадратов, см. рис.: scikit-

## learn – Regression – Linear Models – Ordinary Least Squares.

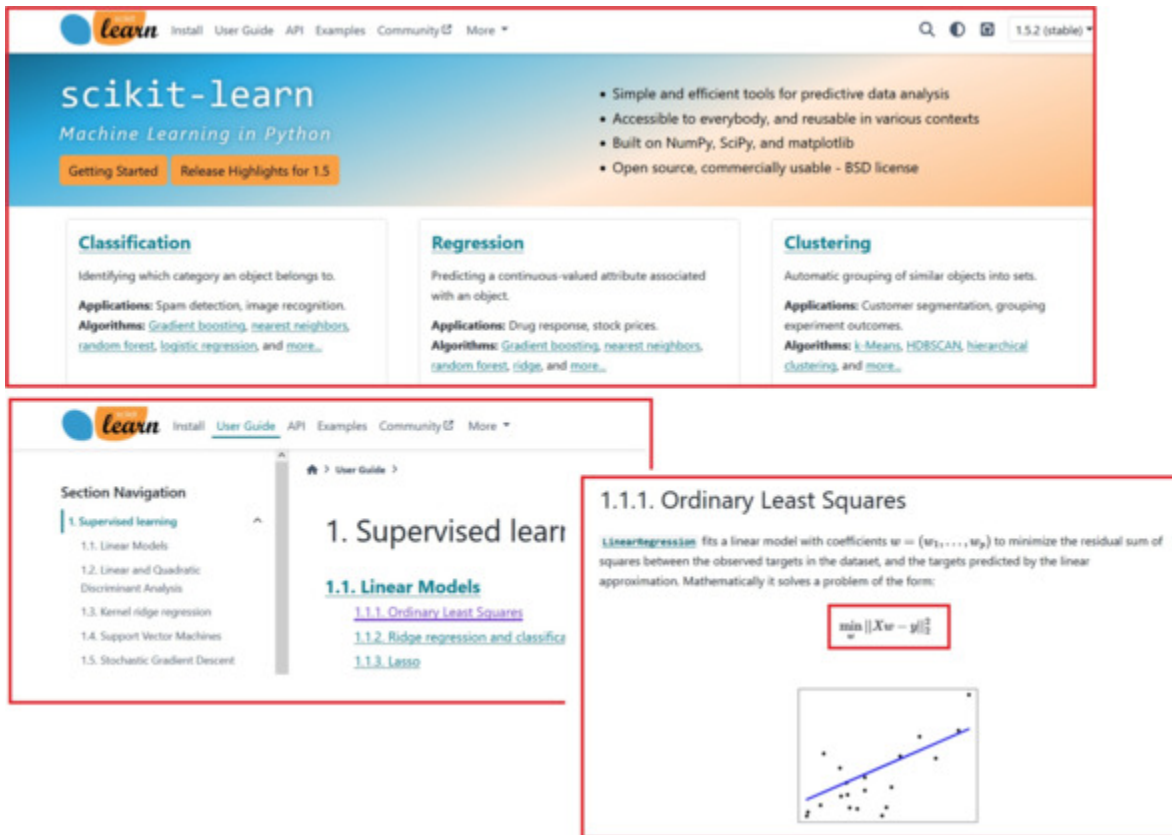


Рис. Страница метода наименьших квадратов

Для нашего удобства здесь есть возможность выводить на экран формулы в формате TeX, см. рис. Наводим курсор на нужную формулу, щёлкаем правой кнопкой мыши и выбираем пункт Show Math As – TeX Commands – Copy To Clipboard. В переводе это звучит так: отображать математические формулы в виде – Команды TeX – Скопировать формулу в буфер обмена.

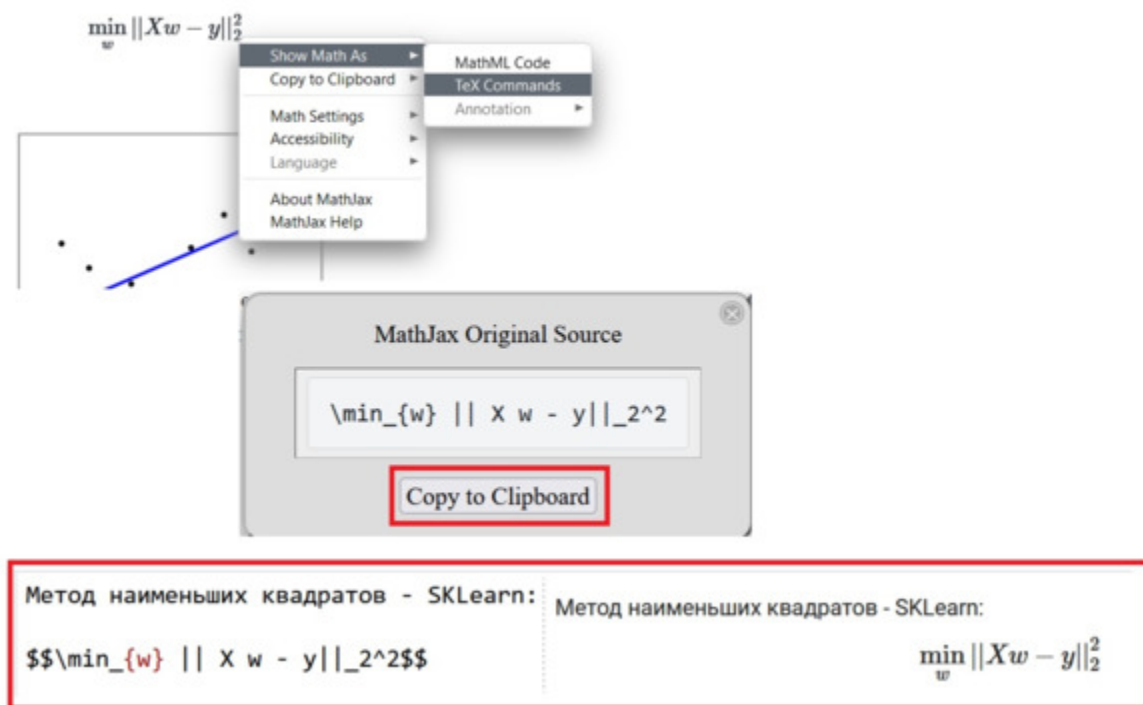


Рис. Вывод формул в нотации TeX

Затем вставляем формулу из буфера в ячейку Colab и обрамляем её символами доллара. Получаем готовую формулу. С ней можно поиграться. Её можно изучить. Это пример красивой, грамотно оформленной математической формулы.

**Задание.** Пролистайте страницу описания МНК из предыдущего примера, выберите другую формулу, скопируйте её в буфер и вставьте в Colab. Убедитесь, что она выглядит точно так же, как и в оригинале.

После такого знакомства с формулами TeX мы начинаем постепенно узнавать их — по шрифту и по компоновке уравнения. Шрифты, кстати говоря, были специально разработаны для этого программного проекта. Они включают самые разные

математические буквы и символы — такого нет ни в одном другом шрифте.

## ЗАКЛЮЧЕНИЕ

Мы познакомились в этой работе с технологией форматирования текста в формате Markdown. Мы увидели, как это работает и как это интегрируется в современные средства разработки. Мы также соприкоснулись с нотацией TeX/LaTeX для создания красивых формул.

Все эти элементы позволяют составлять документацию параллельно с процессом создания программного обеспечения, причём делать это в рамках единого документа. Эта идея соединить программу и документацию была предложена Дональдом Кнудом (Donald Knuth) и по-английски она называется Literate programming, что можно условно перевести следующим образом: «программа как литературное произведение» или «программа, предназначенная для человека, а не для компьютера». Для знакомства с этой концепцией можете задать соответствующий запрос своей нейросети.

## ССЫЛКИ

[Colab] Google Colab. Write and execute Python in your browser. <https://colab.research.google.com/>

[Notepad++] Notepad++ <https://notepad-plus-plus.org>

[ВИКИ: Markdown] Markdown — облегчённый язык разметки. <https://ru.wikipedia.org/wiki/Markdown>

[Markdown Project] The Daring Fireball Company LLC. Markdown formatting syntax. <https://daringfireball.net/projects/markdown/>

[Markdown Basics] The Daring Fireball Company LLC. Markdown Basics. <https://daringfireball.net/projects/markdown/basics>

GitHub <https://github.com/>

GitVerse <https://gitverse.ru/>

[Dingus] Dingus. Онлайн-тестирование и демонстрация синтаксиса Markdown. <https://daringfireball.net/projects/markdown/dingus>

[Operating-Systems Repo] Учебные материалы по операционным системам. <https://github.com/Valentin-Arkov/Operating-Systems>

[Digital Tech Repo] Учебные материалы по цифровым технологиям и искусственному интеллекту <https://github.com/Valentin-Arkov/Digital-Tech-AI>

[Виртуализация] Арьков В. Ю. Виртуализация и контейнеризация [https://ridero.ru/books/virtualizaciya\\_i\\_konteinerizaciya/](https://ridero.ru/books/virtualizaciya_i_konteinerizaciya/)

[Википедия: Формулы] <https://ru.wikipedia.org/wiki/Википедия:Формулы>

[Греческий алфавит] [https://ru.wikipedia.org/wiki/Греческий\\_алфавит](https://ru.wikipedia.org/wiki/Греческий_алфавит)

[SKLearn] scikit-learn. Machine Learning in Python <https://scikit-learn.org/stable/>