

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](#)

[Предисловие](#)

[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](#)

[3. Диаграмма как код](#)

[Введение](#)

[Отчёт](#)

[3.1 Алгоритм](#)

[Блок-схема или схема алгоритма?](#)

[Draw.io](#)

[Mermaid](#)

[Mermaid + Github](#)
[Mermaid + Colab](#)
[Base64 + API](#)
[Mermaid + VS Code](#)
[Mermaid + Docker](#)
[Почему русалочка?](#)
[3.2 Mind map](#)
[PlantUML](#)
[Карта профессии](#)
[Дерево растёт из корня](#)
[Должен, но не обязан](#)
[Знать или уметь?](#)
[Поверните налево...](#)
[Украсим дерево цветами](#)
[Обобщение информации из файлов](#)
[Яндекс Вики](#)
[PlantUML + VS Code](#)
[PlantUML + Docker](#)
[Заключение](#)
[Ссылки](#)
[4 Алгоритмы](#)
[Введение](#)
[Отчёт](#)
[Числа Фибоначчи](#)
[Наивный алгоритм](#)
[Время выполнения](#)
[Организация экспериментов](#)
[Загрузка результатов](#)
[Графики](#)
[Логарифм и экспонента](#)
[Сводная таблица](#)
[«Сводный» график](#)
[Красивое уравнение](#)
[Дерево рекурсии](#)
[Золотое сечение](#)

[Нотация «О большое»](#)
[Массив значений](#)
[Кэширование вызовов](#)
[Загрузка через PowerQuery.](#)
[Глубина рекурсии](#)
[Цикл вместо рекурсии](#)
[Заключение](#)
[Ссылки](#)
[Облачные отчеты](#)
[Картишка на обложке](#)
[Благодарности](#)

ПРЕДИСЛОВИЕ

Перед вами учебное пособие. Это не учебник. Эта книга не заменяет большие, толстые, серьёзные, дорогие издания в красивых обложках, а только дополняет их.

Учебное пособие помогает в освоении материала, помогает учиться. Поэтому его и называют «пособие». Это слово происходит от древнего слова «пособить» или «пособлять» — то есть «помогать, содействовать, способствовать».

В пособии подробно разбирают какой-нибудь раздел, какое-нибудь часть предмета, дисциплины. Зато здесь даётся больше деталей, нюансов, много практических примеров, упражнений и заданий. Это помогает (то есть «пособляет») в учебе.

С другой стороны, есть книги под названием «учебники». Учебник должен давать полную представление о своём предмете. Здесь должны быть описаны все разделы предмета. В нем будет много теории. Он должен позволить нам изучить предмет полностью.

А ещё бывают книги, на которых написано «учебник» или «учебное пособие». Но это совсем не означает, что внутри обязательно будет что-то полезное. Как говорится в народной пословице: «доверяй, но проверяй». Или как в рекламе: «не все йогурты одинаково полезны...»

Приходится учиться различать, какие книжки хорошие, а какие не очень. Что поможет в учёте, а что помешает.

С компьютерными технологиями есть ещё одна неприятность. Слишком быстро развиваются технологии. Учебники не успевают выпустить, а они уже устарели. Пособие выходит быстрее. Можно за неделю выпустить его в жизнь. Чем мы и занимаемся.

Какую книгу выбрать – для того, чтобы изучить какой-нибудь предмет? Это зависит от человека. Можно открыть книгу в магазине или скачать файл в интернете – и просто пролистать несколько страниц. Почувствуйте на себе, насколько вам подходит этот текст. Сможете ли вы прочитать ещё несколько страниц? Желательно с пониманием и без отвращения.

Все люди разные. У каждого свой уровень подготовки. И разным людям помогут разные материалы для изучения. Кому-то нужно побольше картинок, а кому-то подробные объяснения. А кому-то наоборот – надо покороче и побыстрее.

Возможно, это пособие поможет именно вам сделать первые шаги в изучении информационных технологий.

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.

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 текстовыми файлами и диаграммами с помощью кода
- измерять и улучшать свойства алгоритмов
- работать с электронными таблицами и текстовыми редакторами, облачными сервисами и другими полезными инструментами

В рамках этого учебного пособия мы обсуждаем самые основы предмета, самые базовые понятия и технологии. Для этого материал даётся и объясняется очень простыми словами. Эти пояснения предназначены для тех, кто только начинает знакомиться с современными технологиями. Поэтому не удивляйтесь, что объяснения будут очень приблизительные

и упрощённые. Сложные слова и запутанные материалы вы легко сможете найти в толстых учебниках, если будет такое желание.

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 практические, 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 отрава. 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 какую-нибудь поисковую машину. 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, сформируйте список основных правил составления запросов к нейросети. Это будет начало нашего отчёта. Отчёт у нас будет в электронном виде.

ОФОРМЛЯЕМ ОТЧЁТ

По каждой работе мы составляем неформальный отчёт. В этом отчёте мы описываем то, что мы сделали и что нам удалось узнать. Попутно мы с вами проходим весь цикл работы. От начала до оформления результата. Чтобы прочувствовать общую схему работы,

просмотрите раздел «Облачные отчеты» в конце пособия.

Результаты будем оформлять в облаке. Или хотя бы размещать в облаке. Как вы знаете, есть облачные офисные продукты. Есть продукты зарубежные, в том числе, и бесплатные. Есть аналогичные отечественные сервисы – в них и будем работать. 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 отечественный облачный офис. Для этого понадобится бесплатная учётная запись – такая же, как для электронной почты. Создаём новый документ. Интерфейс чем-то напоминает привычный настольный вариант. 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-ИВТ123».

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 строкой 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 Минобразование. 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 «фактчекинг» – 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 «промпт-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. 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 выразить в тексте.

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, что 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 пригождаться.

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.

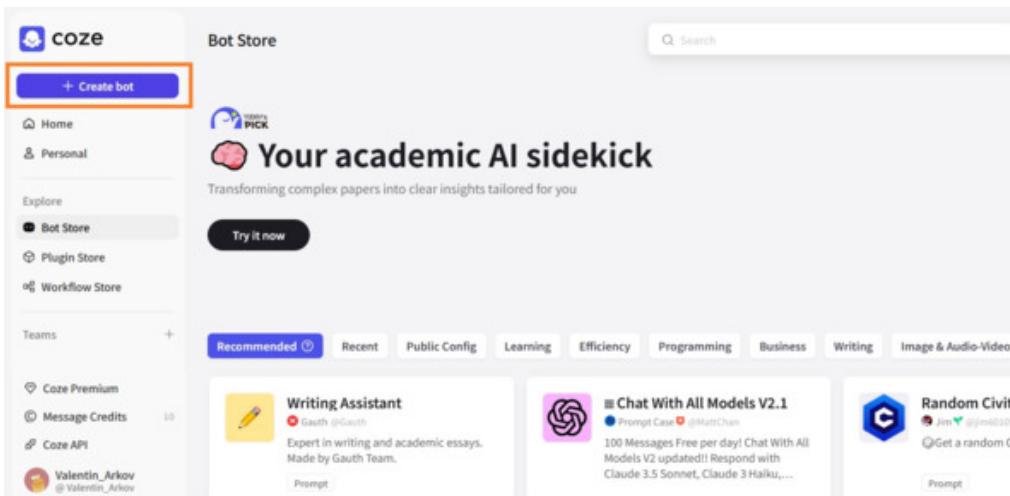


Рис. Создаём бота

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.

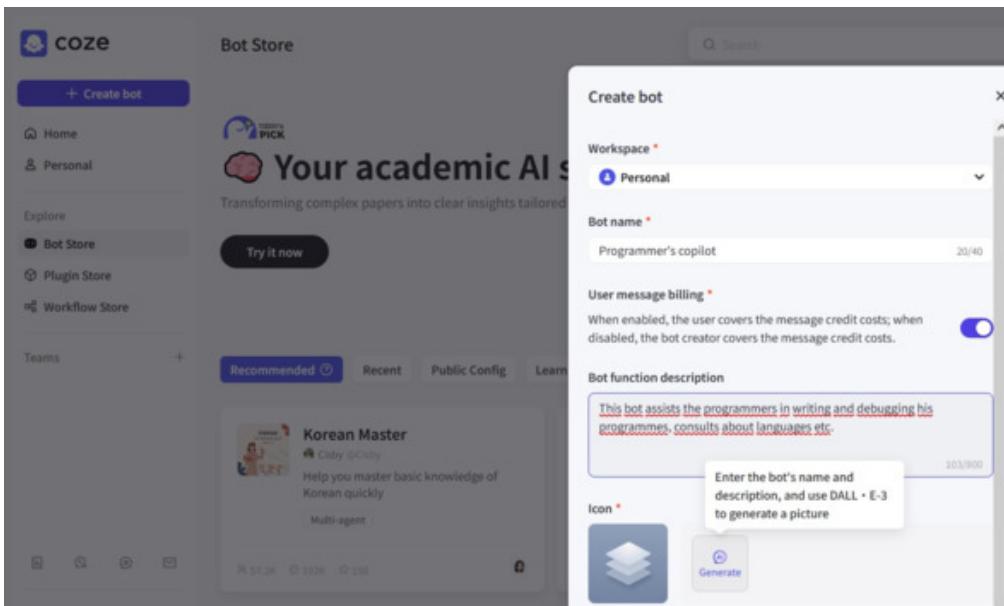


Рис. Открывающийся диалог

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-4о даётся только один запрос в день.

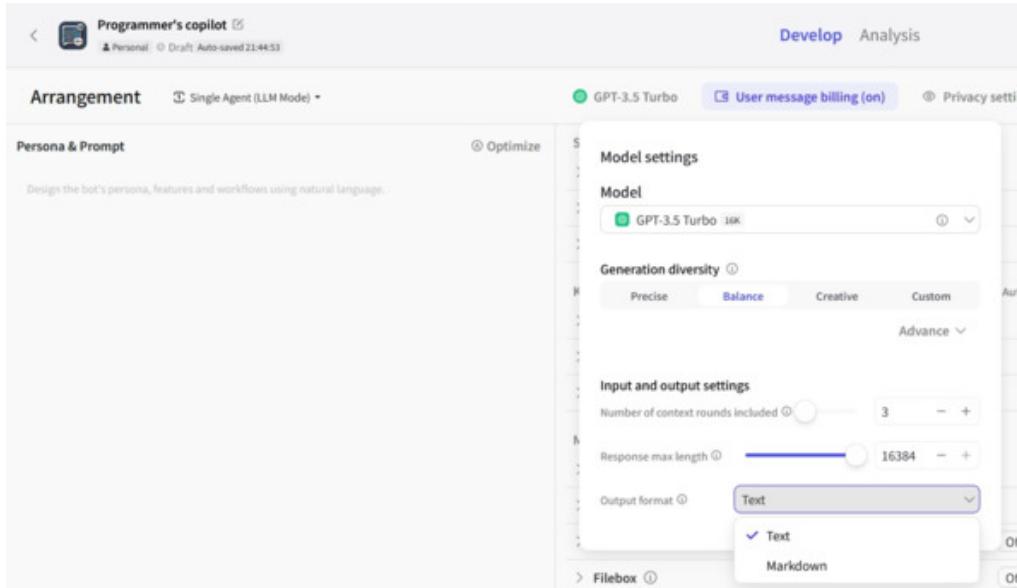


Рис. Выбор языковой модели

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.

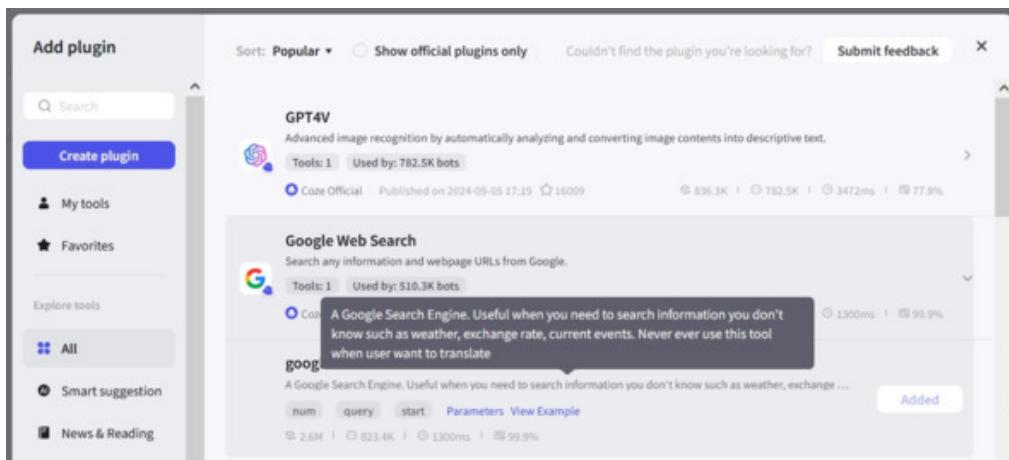


Рис. Возможности поиска

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».

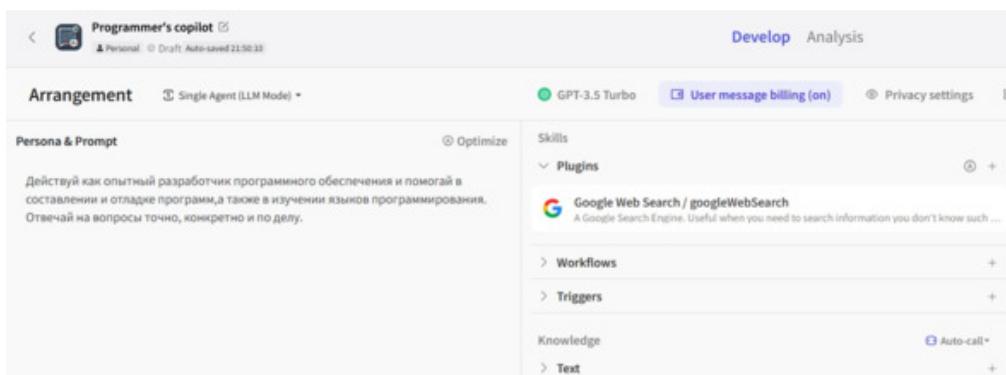


Рис. Системный промпт

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

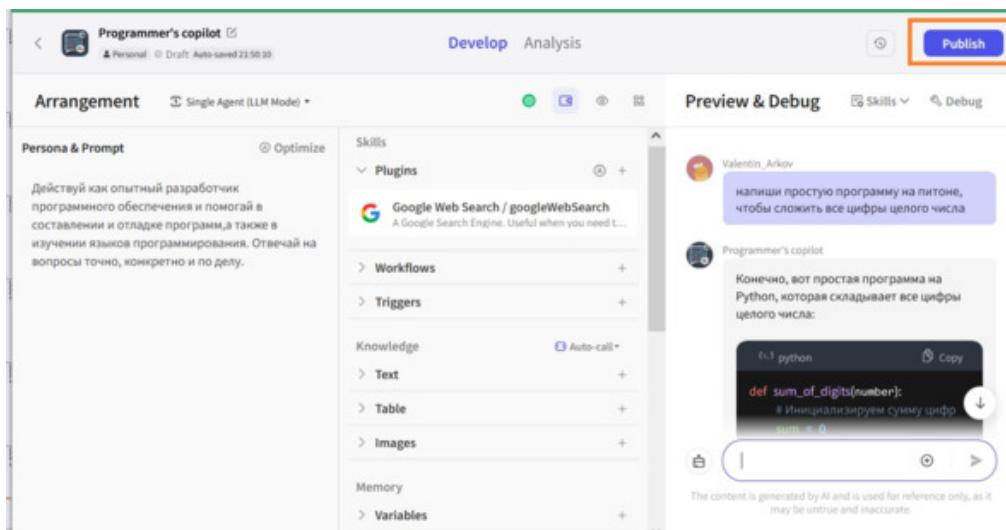


Рис. Предпросмотр

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.

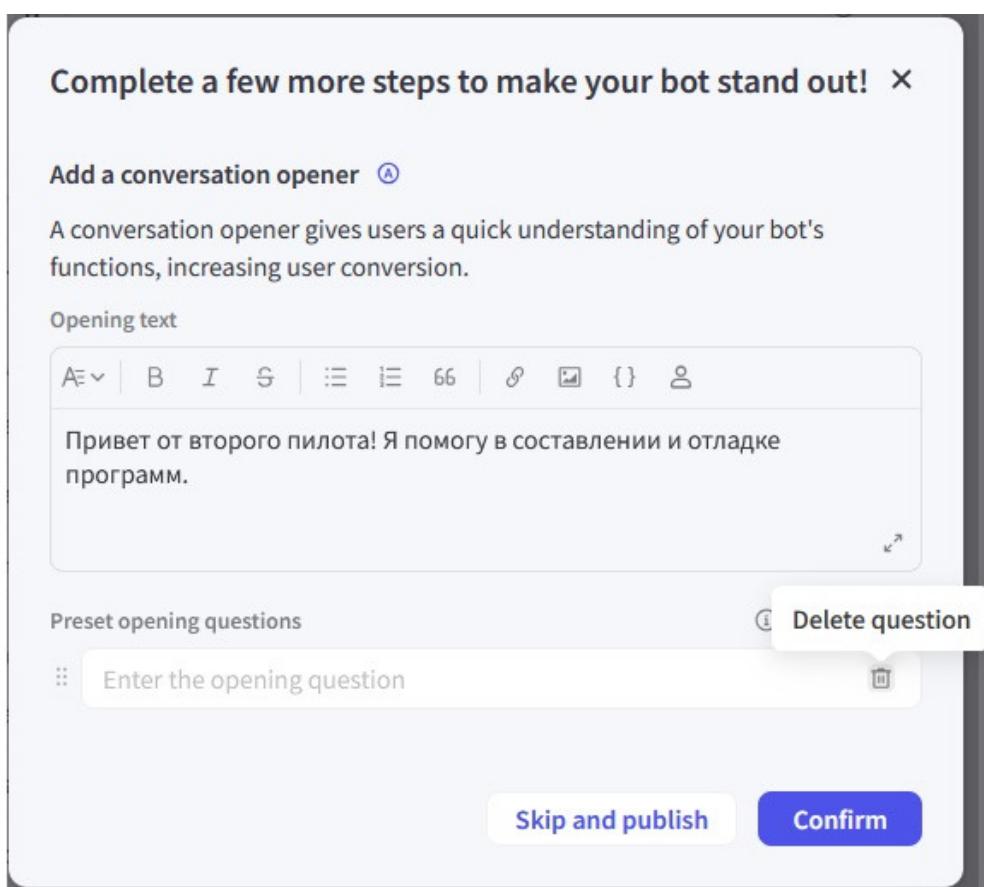


Рис. Настройка приветствия

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.

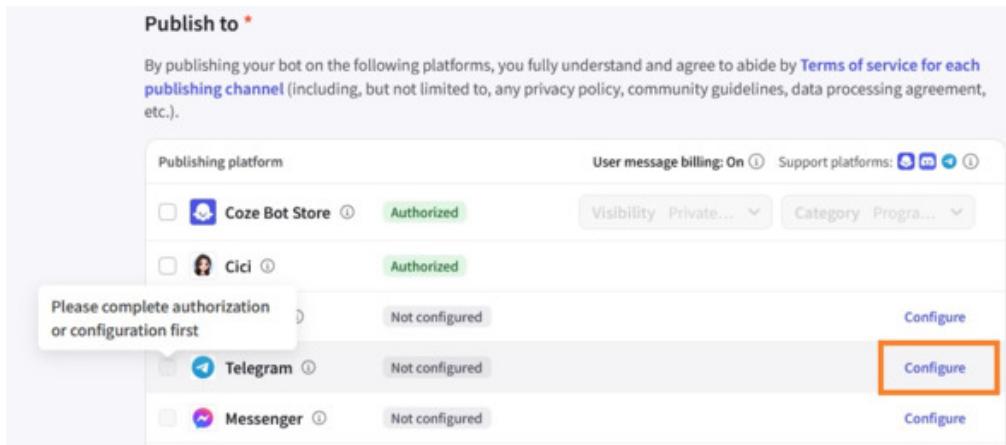


Рис. Публикуем в Телеграме

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

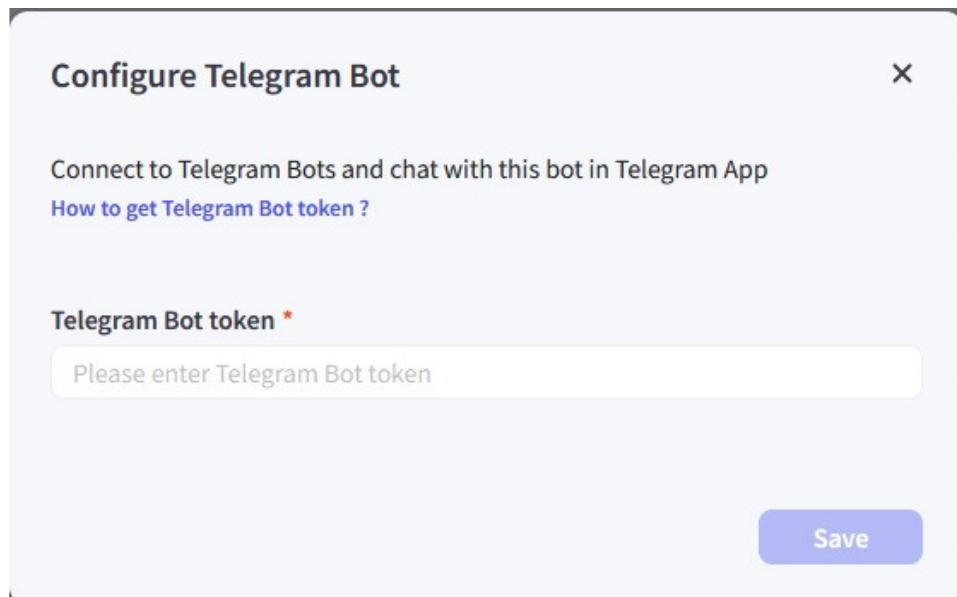


Рис. Запрос токена

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

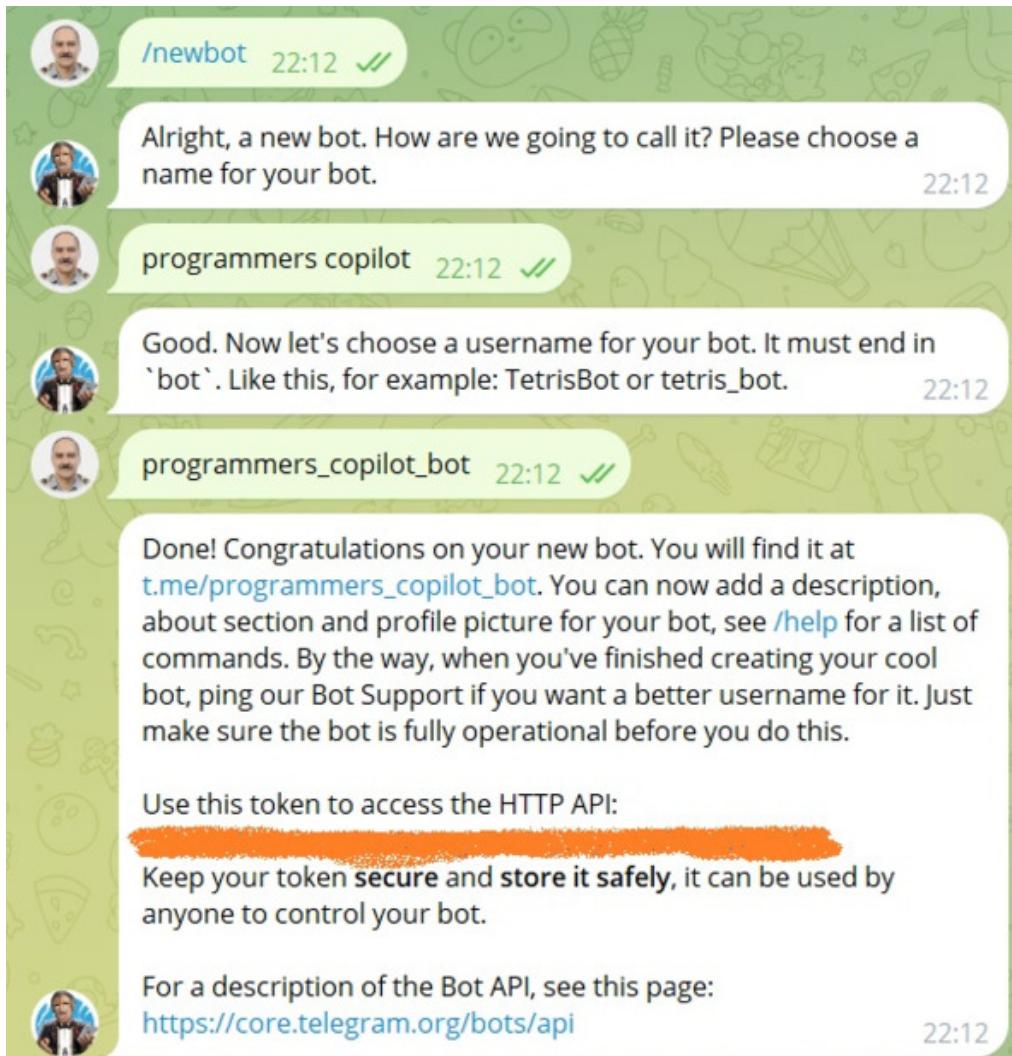


Рис. Получение токена

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.

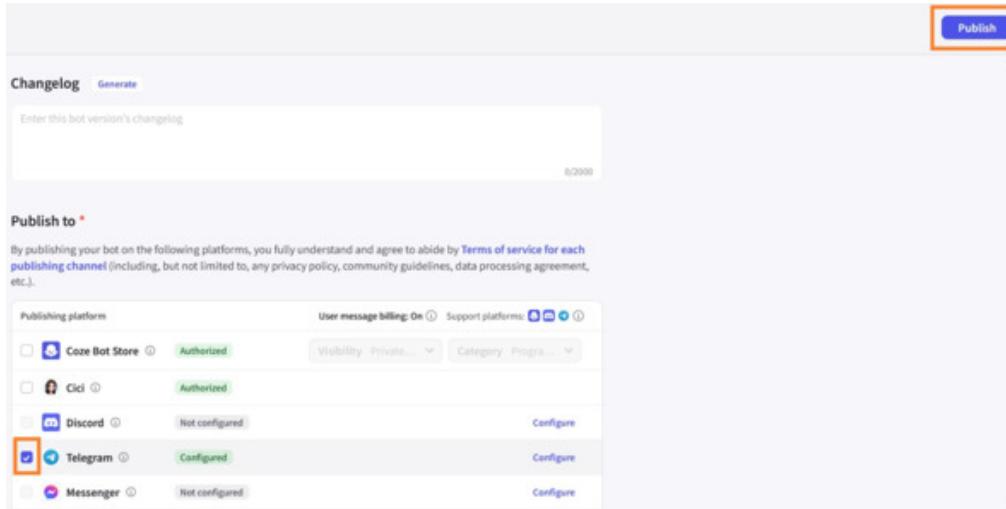


Рис. Публикация в Телеграме

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.

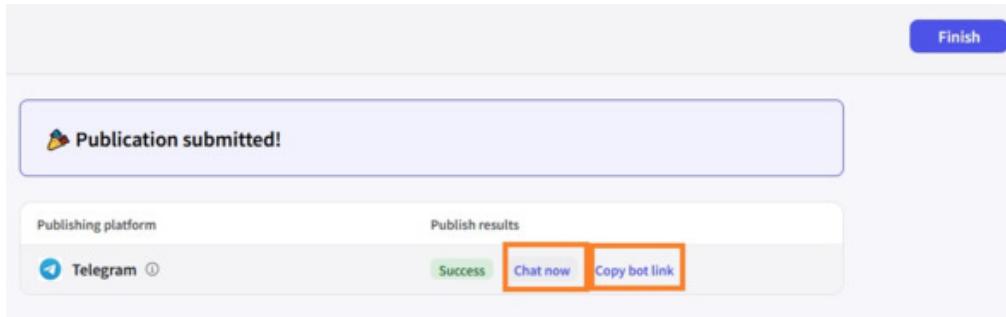


Рис. Переход к беседе

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

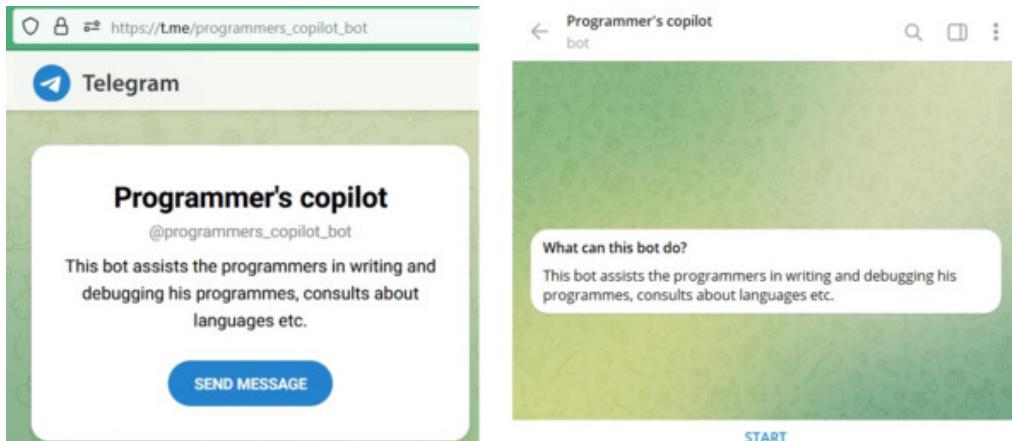
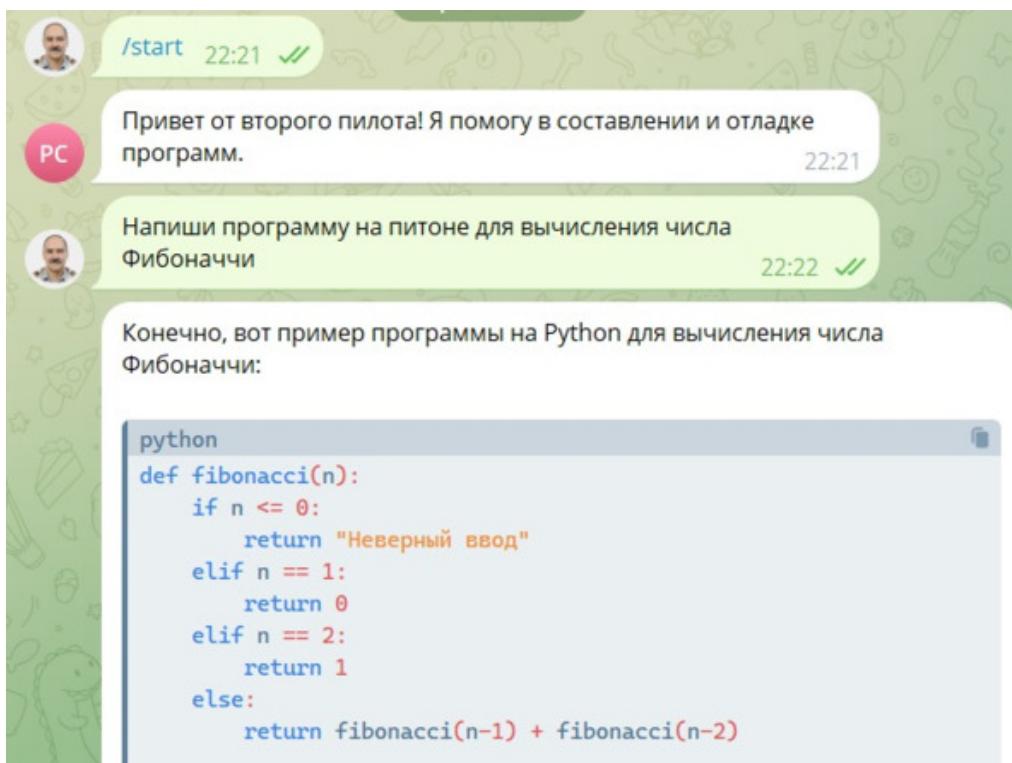


Рис. Запуск бота

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



```
python
def fibonacci(n):
    if n <= 0:
        return "Неверный ввод"
    elif n == 1:
        return 0
    elif n == 2:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)
```

Рис. Генерируем программу

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

PC Какая сложность алгоритма вычисления числа Фибоначчи с использова...

Valentin: Какая сложность алгоритма вычисления числа Фибоначчи с использованием этой программы? "

Сложность алгоритма вычисления числа Фибоначчи с использованием данной рекурсивной программы - экспоненциальная, то есть $O(2^n)$. Каждый вызов функции вызывает два дополнительных вызова функции, что приводит к повторному вычислению одних и тех же значений, делая алгоритм неэффективным для больших значений n . 22:23

Как можно улучшить данную программу для более эффективного вычисления?

Какие альтернативные подходы к вычислению чисел Фибоначчи существуют?

Можно ли использовать динамическое программирование для оптимизации?

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.

Я Кликер

Помогите клиентам быстро найти вашу страницу в интернете. Благодаря короткой ссылке клиентам не придётся видеть длинные url-адреса, занимающие много места.

https://t.me/programmers_copilot_bot Сократить

♂ clck.ru/3DRxiq

Копировать ctrl+c

QR-код

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 врп 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

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

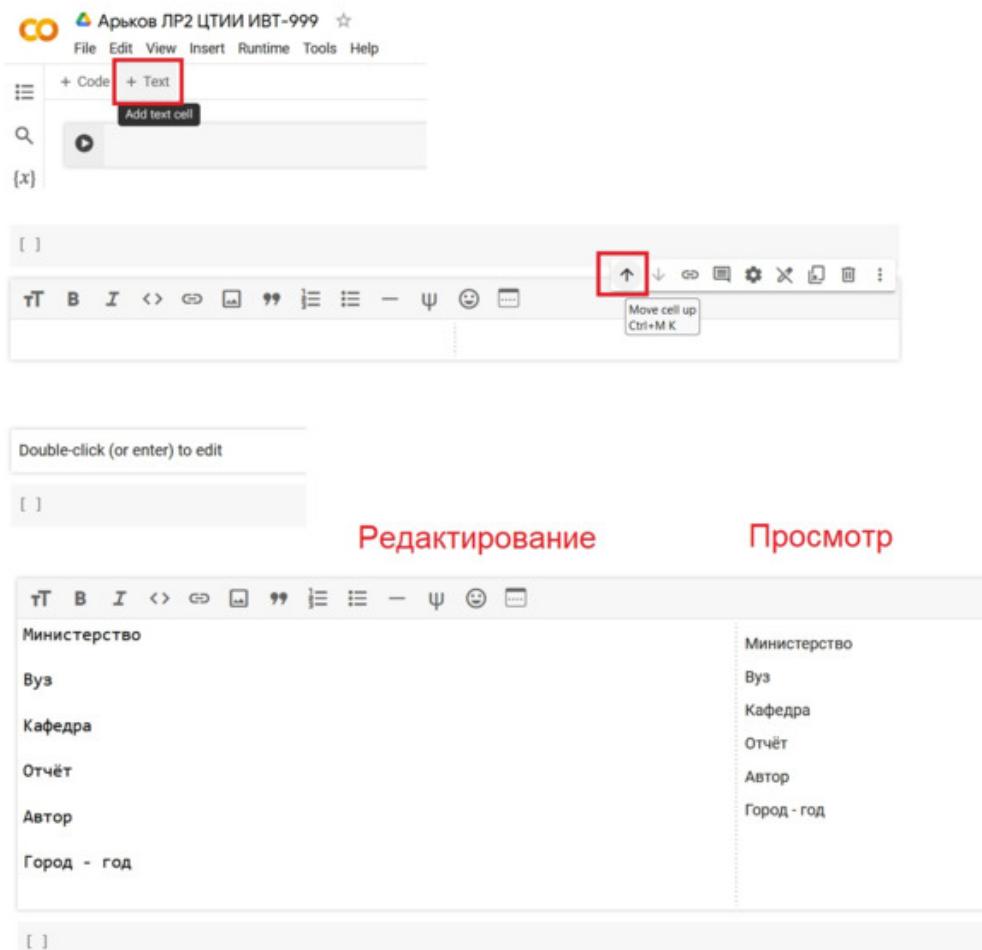


Рис. Отчёт по работе

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.

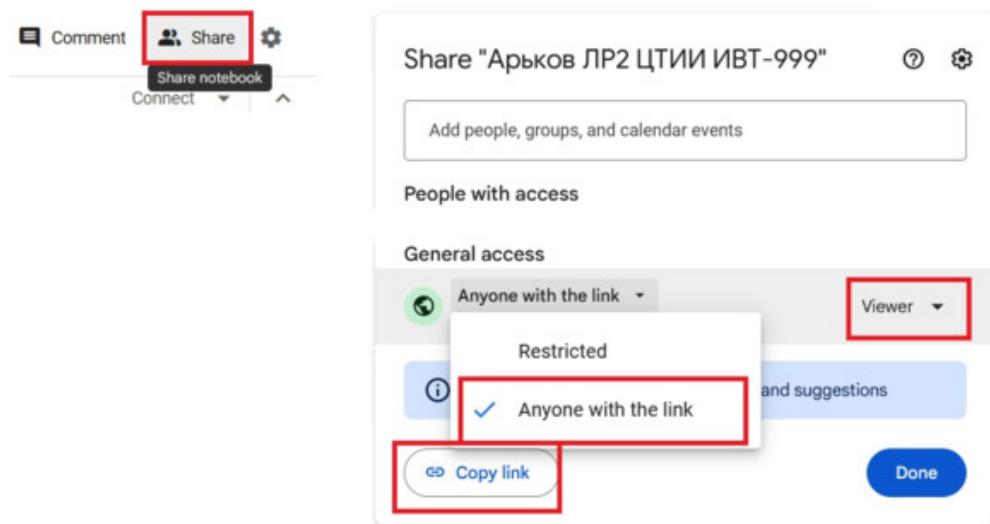


Рис. Делимся ссылкой на отчёт

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 С++, 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.

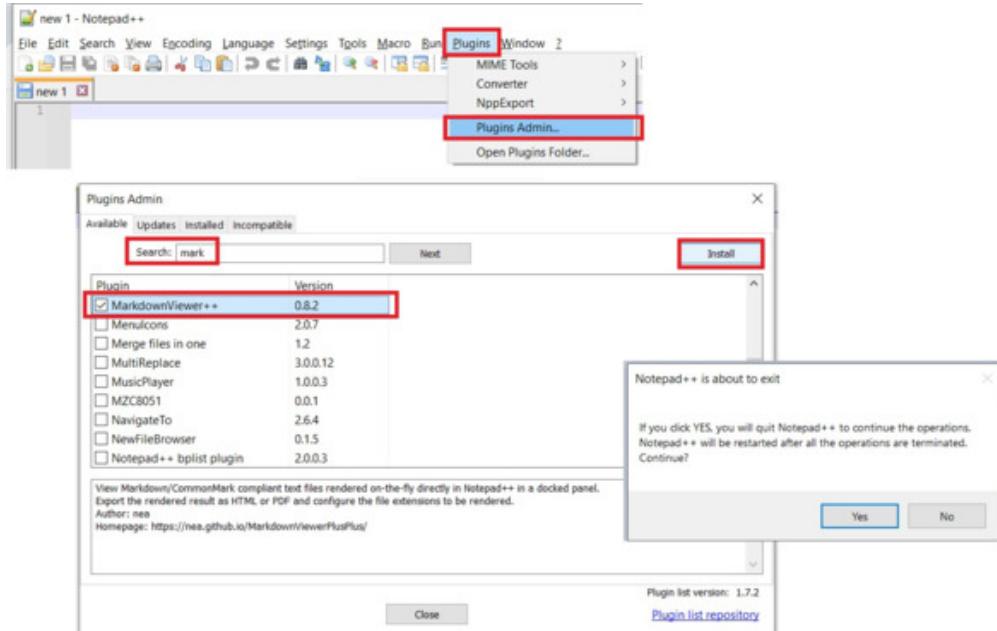


Рис. Установка дополнения для просмотра

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.

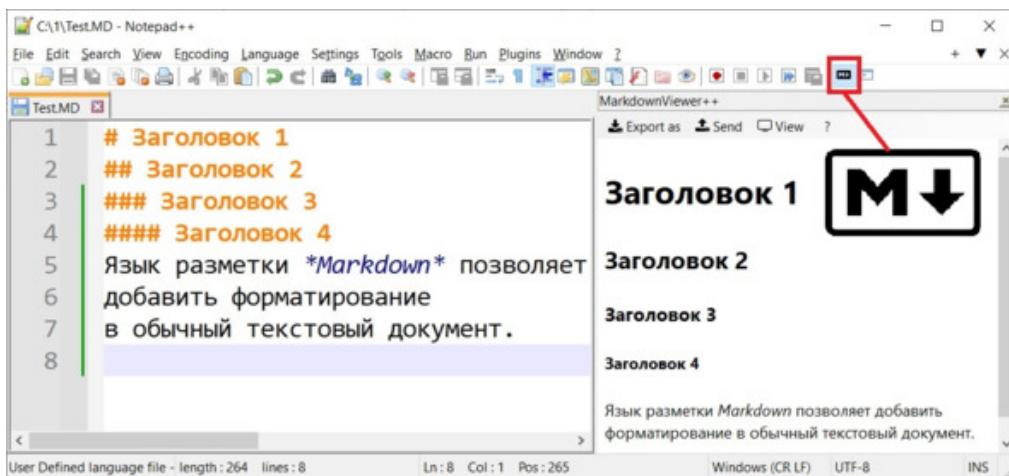


Рис. Редактирование и просмотр Markdown

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 просмотреть результаты, мы нажимаем эту кнопку, чтобы закрыть и снова открыть окно просмотра. На этой кнопке изображено условное обозначение языка Markdown — заглавная буква M 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 сливаётся 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 экспортим его в HTML и PDF.

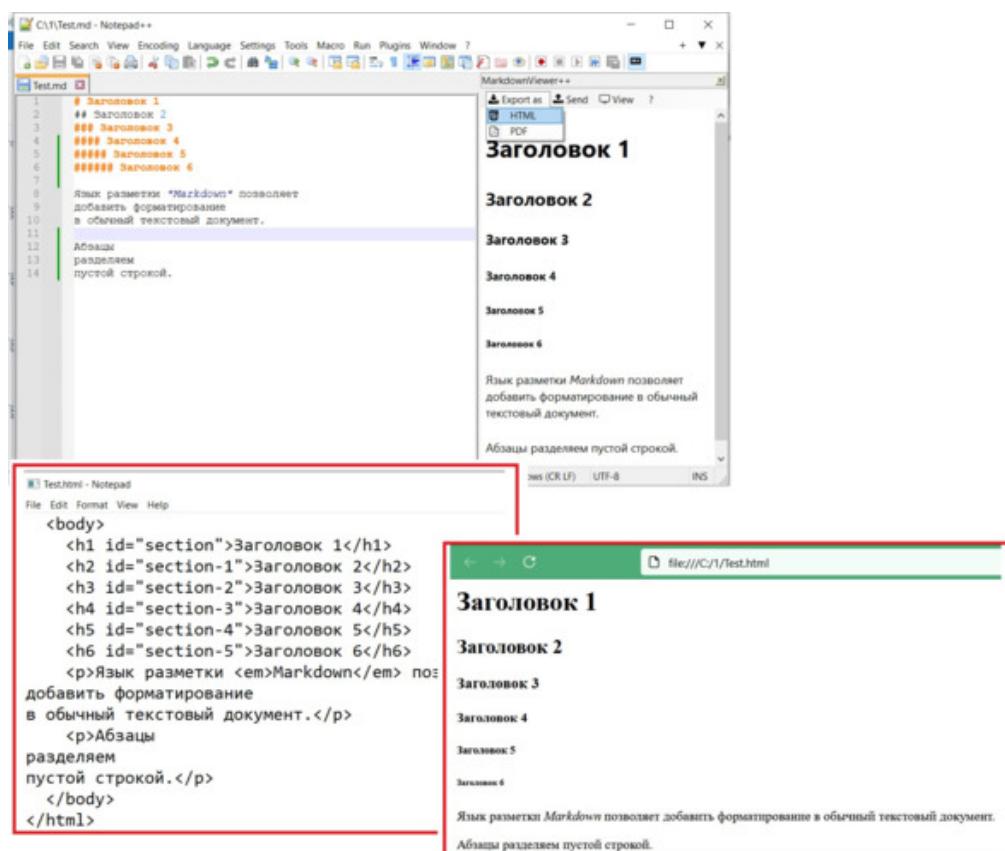


Рис. Экспорт гипертекста

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.

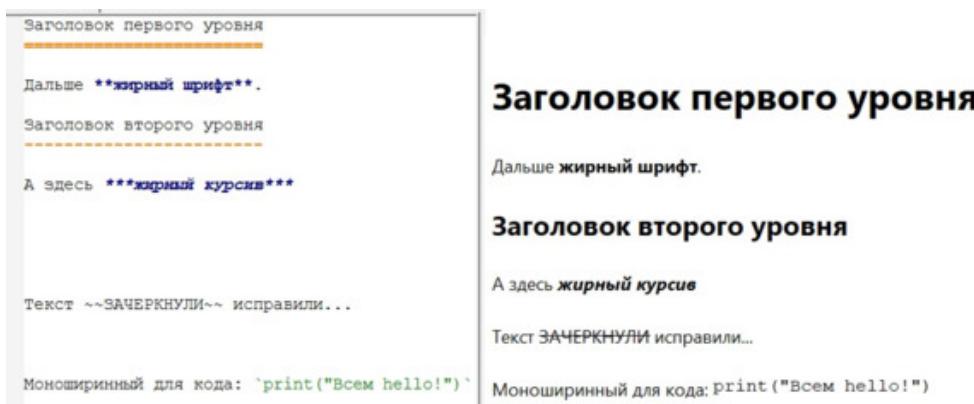


Рис. Жирный курсив и заголовки

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 ненумерованный, то можно использовать звёздочки (*), плюсики (+) и минусы/чёрточки (-). В начале строки звёздока, 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, пузырёк». Теперь это кружочки, жирные точки.

Если список нумерованный, то строчка 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 немножко «ужимают».

The image shows a code editor interface with a block of Python code. The code is enclosed in a light grey box. Above the code, there is a note: 'Блок кода – табуляция или 4 пробела'. The code itself is:

```
def pixie():
    for _ in range(7):
        print(x)
    return 0
```

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.

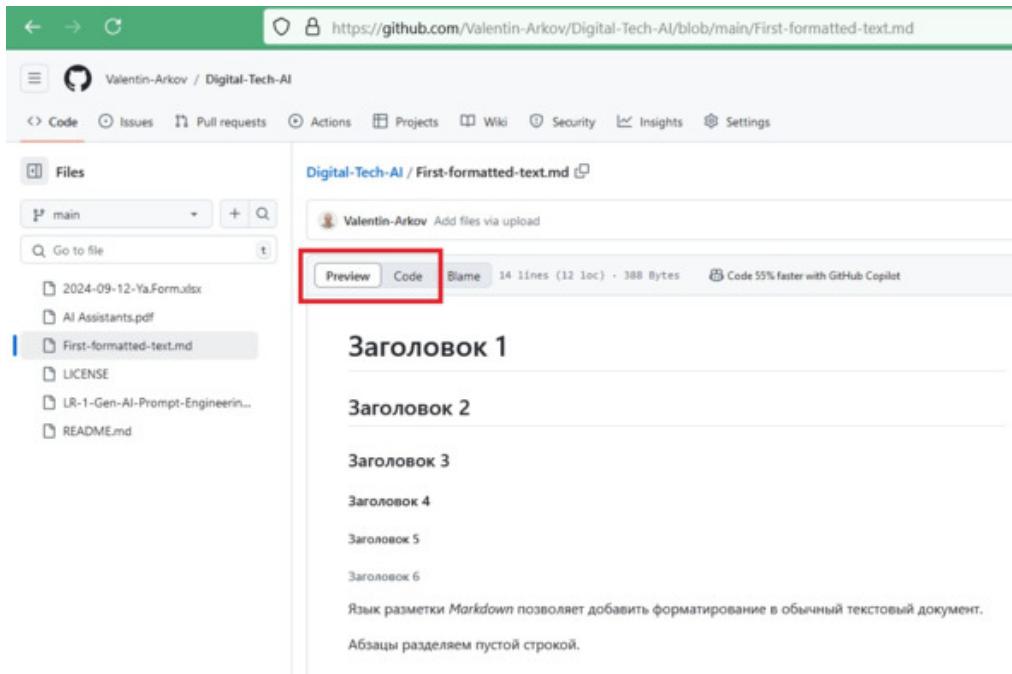


Рис. Файл в репозитории GitHub

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.

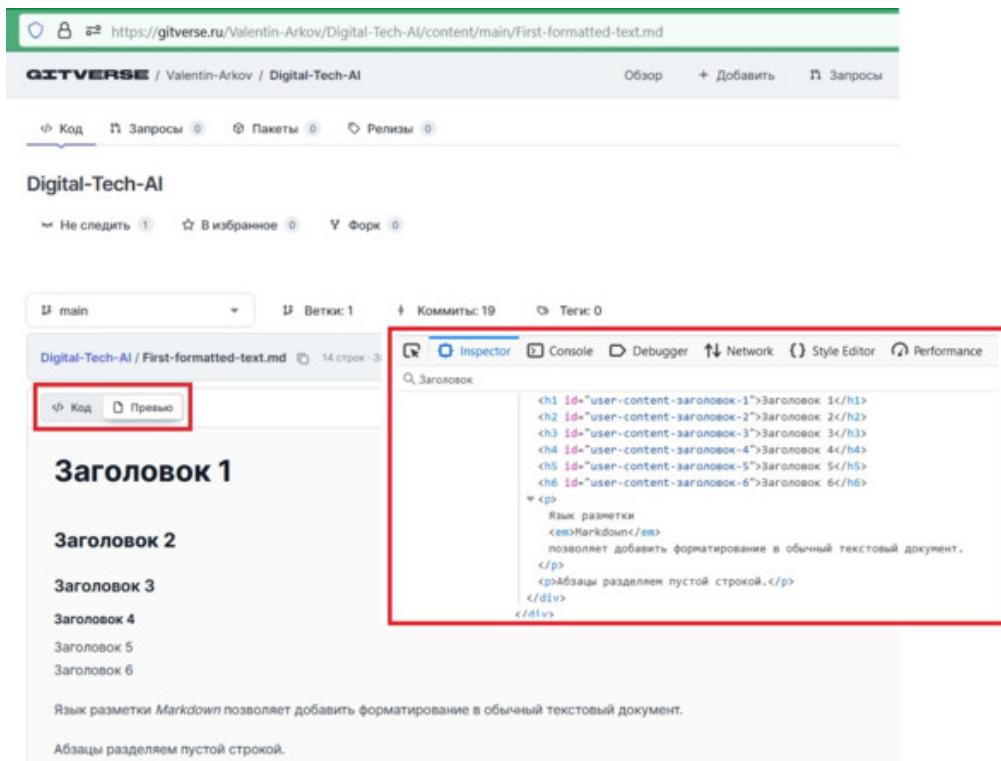


Рис. Файл в репозитории GitVerse

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.

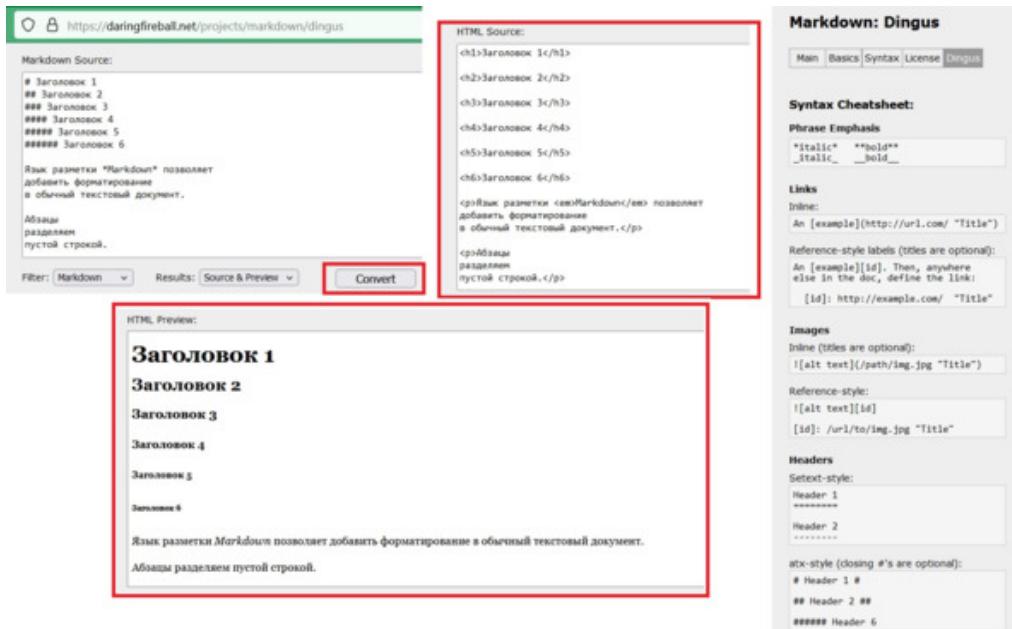


Рис. Работа с Markdown в сервисе Dingus

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.

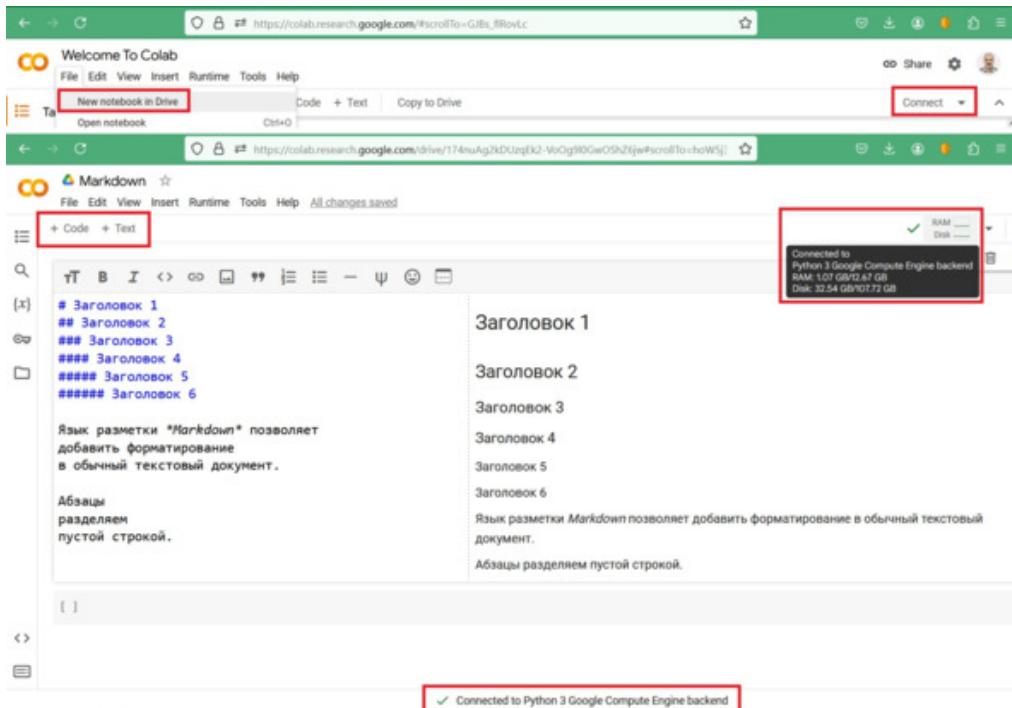


Рис. Работа с Markdown в Google Colab

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.

The image shows two side-by-side Notepad windows. The left window, titled 'markdown.py - Notepad', contains Python code for generating Markdown. It includes a multi-line string with triple quotes containing various Markdown headers and descriptions of the language's features. The right window, titled 'Markdown.ipynb - Notepad', contains JSON-formatted data for an IPython notebook cell. The 'source' field of the cell is a list of strings representing the same Markdown content as the Python script, including the headers and explanatory text.

```
markdown.py - Notepad
File Edit Format View Help
# -*- coding: utf-8 -*-
"""Markdown

Automatically generated by Colab.

Original file is located at
https://colab.research.google.com/drive/1JGzXWV9DyfjPQHgkCwvIYUOOGdLcBqA

# Заголовок 1
## Заголовок 2
### Заголовок 3
#### Заголовок 4
##### Заголовок 5
###### Заголовок 6

Язык разметки *Markdown* позволяет
добавить форматирование
в обычный текстовый документ.

Абзацы
разделяем
пустой строкой.

"""

Markdown.ipynb - Notepad
File Edit Format View Help
{
  "cell_type": "markdown",
  "source": [
    "# Заголовок 1\\n",
    "## Заголовок 2\\n",
    "### Заголовок 3\\n",
    "#### Заголовок 4\\n",
    "##### Заголовок 5\\n",
    "##### Заголовок 6\\n",
    "\\n",
    "Язык разметки *Markdown* позволяет\\n",
    "добавить форматирование\\n",
    "в обычный текстовый документ.\\n",
    "\\n",
    "Абзацы\\n",
    "разделяем\\n",
    "пустой строкой."
  ],
  "metadata": {}
}
```

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.

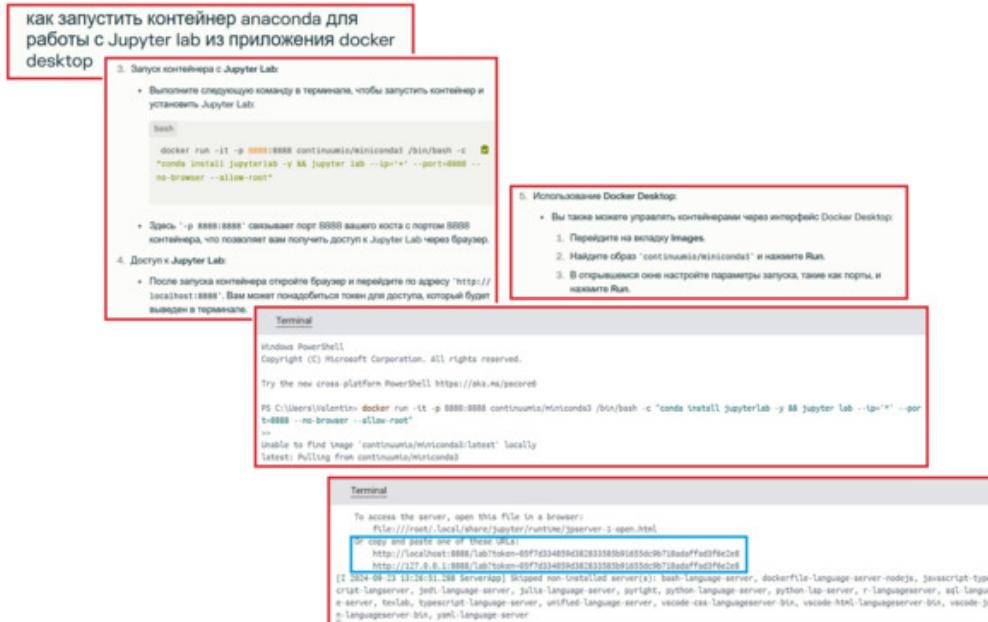


Рис. Инструкция по запуску контейнера

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.

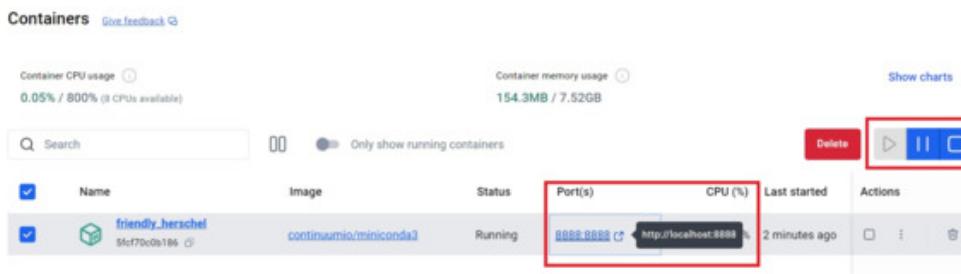


Рис. Запуск контейнера Anaconda

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).

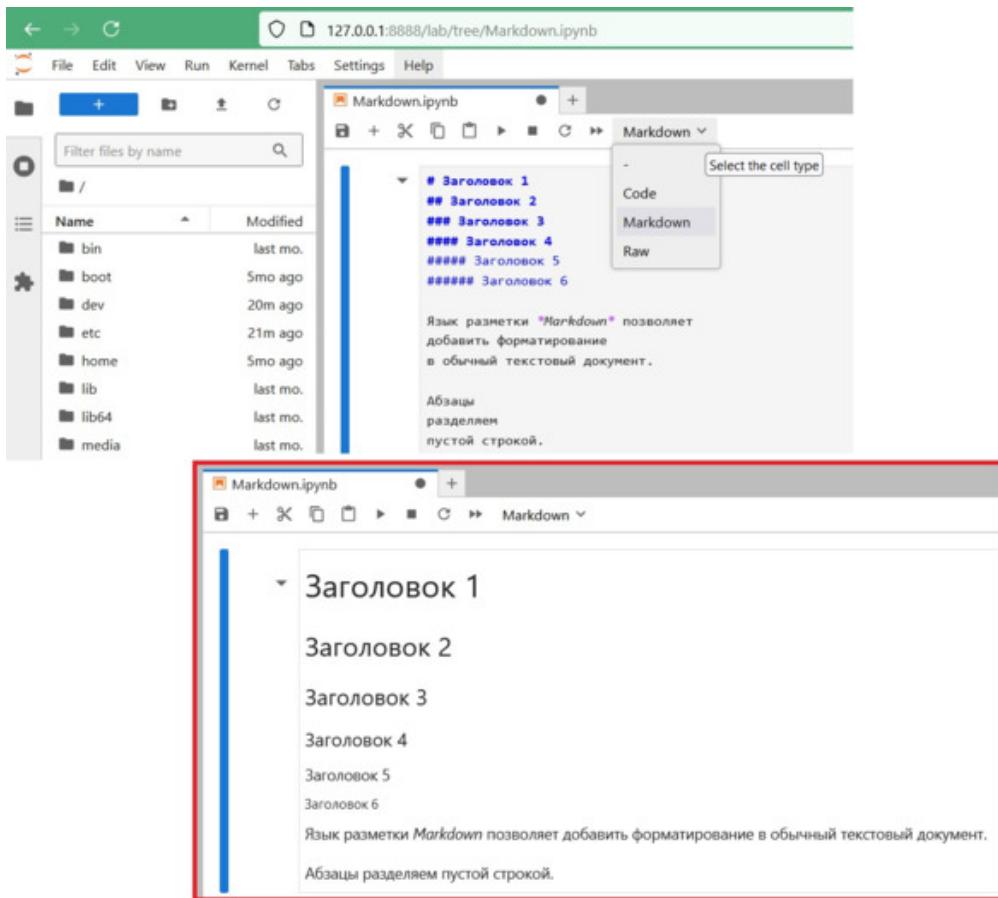


Рис. Jupyter Lab в браузере

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.

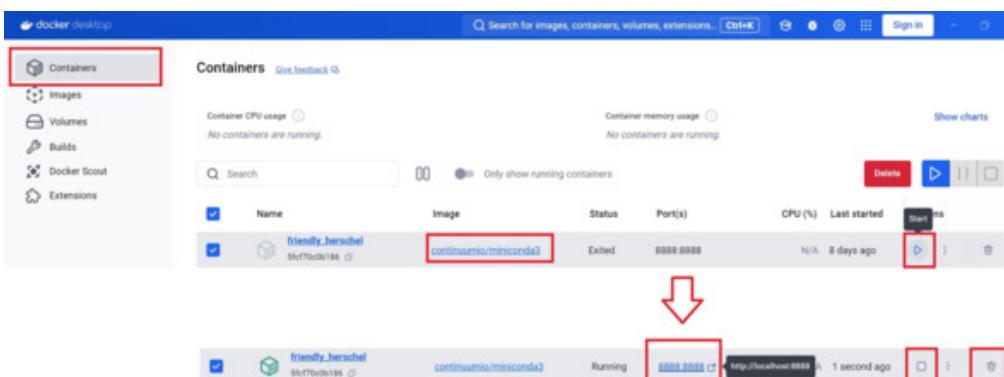


Рис. Повторный запуск контейнера

Для повторного запуска контейнера у нас всё уже готово и скачано 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.

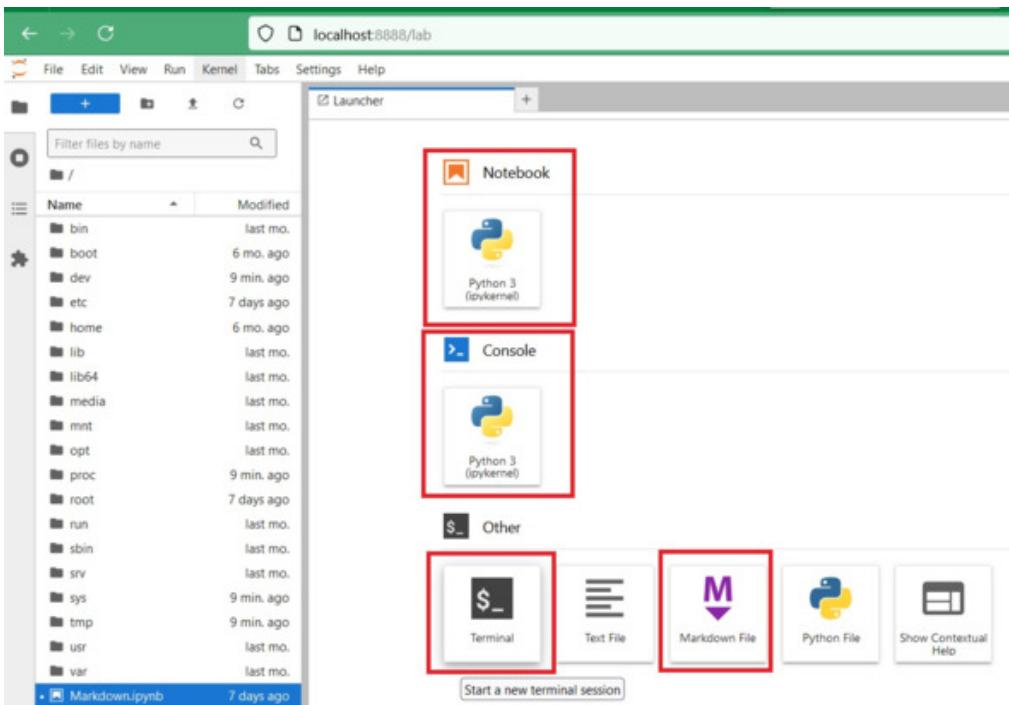


Рис. Страница запуска Jupyter Lab

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 развернут 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.

```
# 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@041d701fb84f1) (gcc (GCC) 11.2.0, GNU ld (GNU Binutils) 2.37) #1 SMP Fri Mar 29 23:14:13 UTC 2024
# uname -a
Linux 5fcf70ccb186 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.

The screenshot shows two windows side-by-side. On the left is Notepad++ with the file 'Table etc.md' open. The content is:

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

On the right is 'MarkdownViewer++' showing the rendered output:

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 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.

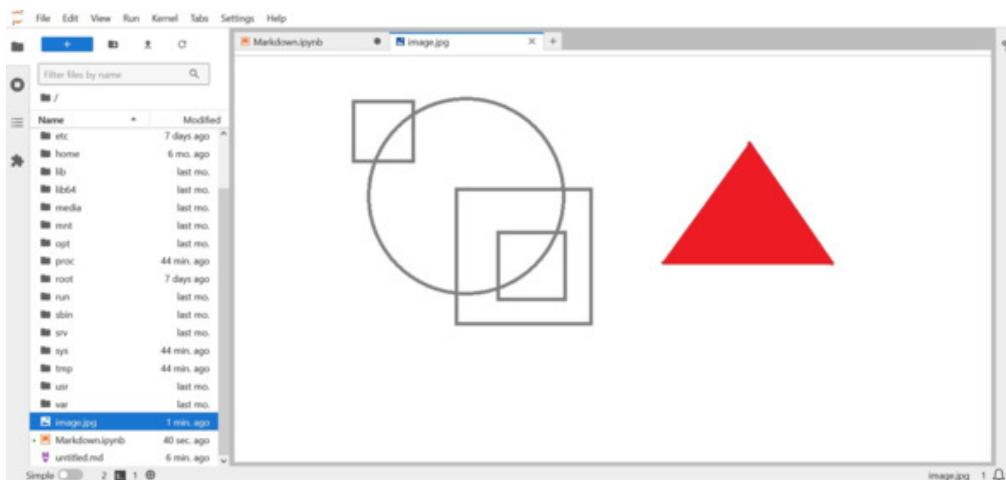


Рис. Просмотр изображения

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

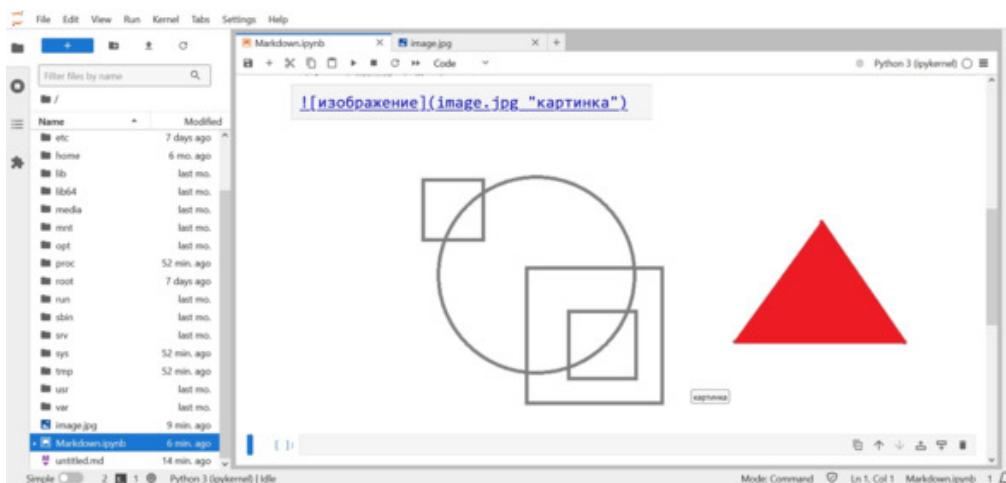


Рис. Вставка изображения

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.

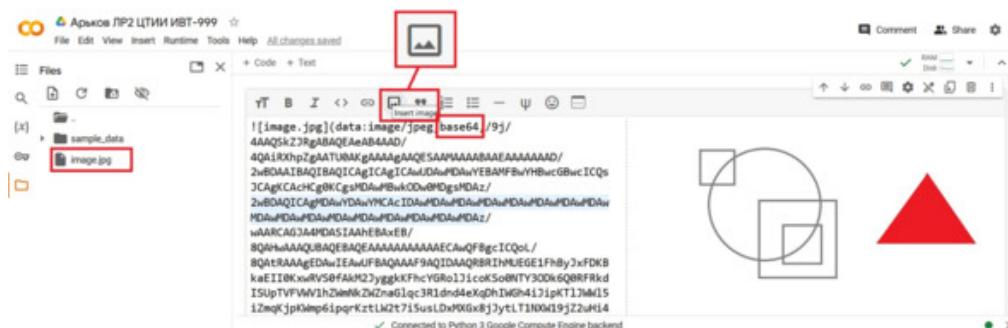


Рис. Вставка изображения в Colab

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

Таблица

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>Таблица:</h3>
<table>
    <thead>
        <tr>
            <th>Столбец 1</th>
            <th>Колонка 2</th>
        </tr>
    <tbody>
        <tr>
            <td>Ячейка Раз</td>
            <td>Ячейка Два</td>
        </tr>
    </tbody>
</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 ТЕХ

В дополнение к привычным средствам форматирования, мы можем вставлять в документ многоэтажные 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 многоэтажными 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.

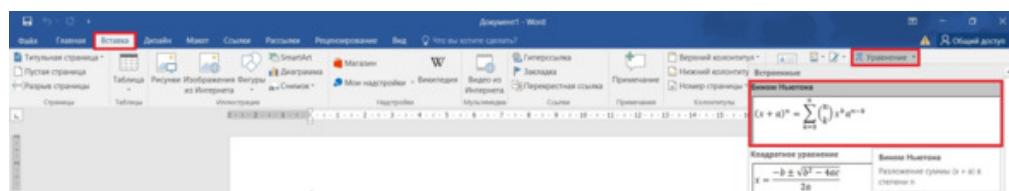


Рис. Вставляем готовую формулу

После вставки формулы автоматически открывается вкладка Конструктор – Design.

При этом уравнение формула отображается в привычном виде. мы можем работать с этим объектом визуально. 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.

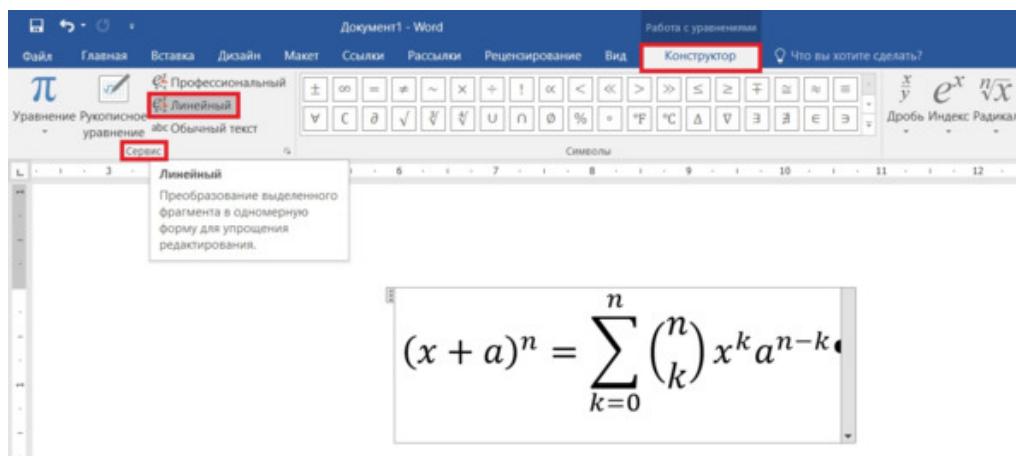


Рис. Вкладка Конструктор

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» ^:

$$(x + a)^n$$

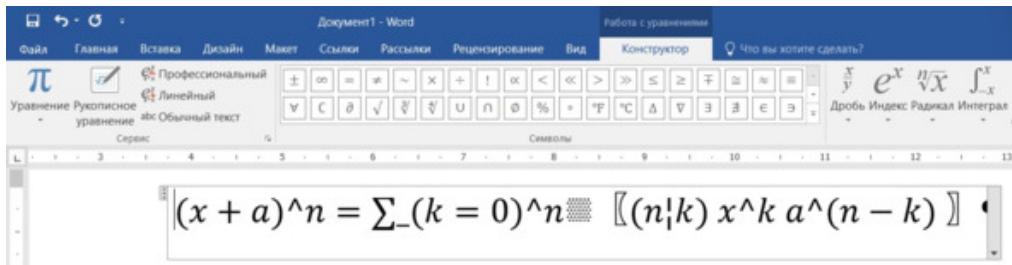


Рис. Линейный режим

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.

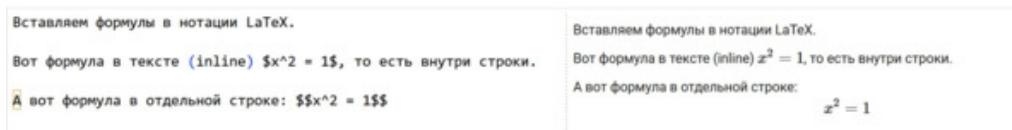


Рис. Формулы в Colab

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.

The screenshot shows the Perplexity AI interface. On the left, there's a sidebar with 'perplexity' logo, 'New Thread' button, and navigation links for 'Home', 'Discover', and 'Library'. The main area is titled 'Answer' and contains text: 'Certainly! Here are the formulas for the sum of a geometric progression using the summation symbol in TeX notation:'. Below this, it says 'For a finite geometric progression:' and 'If $r \neq 1$:'. A mathematical formula is displayed:
$$S_n = \sum_{k=0}^{n-1} ar^k = a \frac{1 - r^n}{1 - r}$$
. There's also a note: '• r is the common ratio.' At the bottom right of the main area, there are 'Copy' and 'Share' buttons, with the 'Copy' button highlighted by a red box.

Рис. Копируем ответ нейросети в нотации TeX

The screenshot shows a Jupyter Notebook cell. On the left, there's a code block with the following content:

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

On the right, there's a text block with the same formulas and notes: 'Certainly! Here are the formulas for the sum of a geometric progression using the summation symbol in TeX notation:'. It includes the formula for $r \neq 1$ and the formula for $r = 1$.

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}`

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

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.

```
$$
I = \int_{-1}^1 \frac{4}{1+x^2} dx
$$
```

Рис. Формула интеграла в Colab

Затем используем это выражение в запросе к нейросети.



Perplexity

Чтобы вычислить интеграл

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

Мы можем воспользоваться известной формулой для интеграла функции $\frac{1}{1+x^2}$, которая равна $\tan^{-1}(x)$.

Ответ

Таким образом, значение интеграла равно:

$$I = \pi.$$

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..

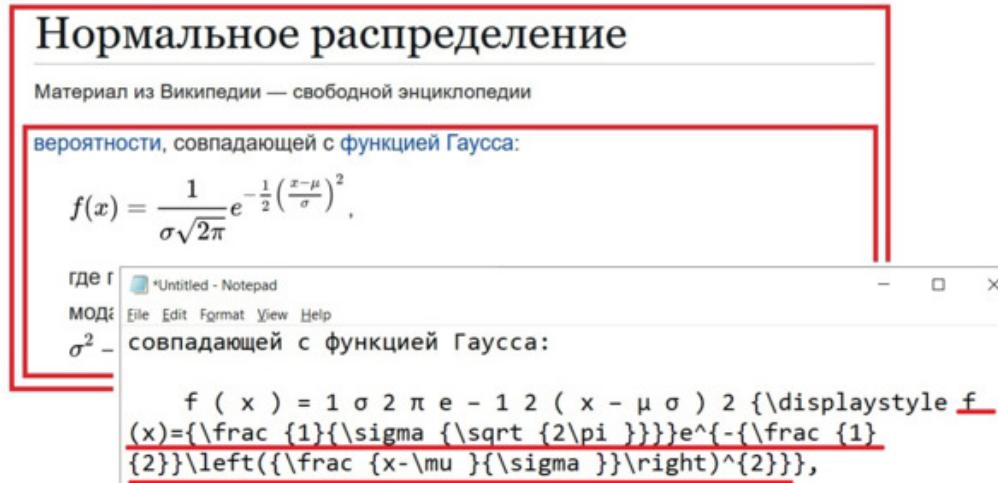


Рис. Формула из Википедии в Блокноте

Здесь скрывается нотация TeX внутри фигурных скобок $\{\text{\displaystyle }...\}$ – 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 $\textit{displaystyle}$. Если же перейти к просмотру выбранной статьи, мы увидим уже красиво оформленные математические формулы.

Задание. Найдите в Википедии любую статью с формулой, использующей конструкцию \textit{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.

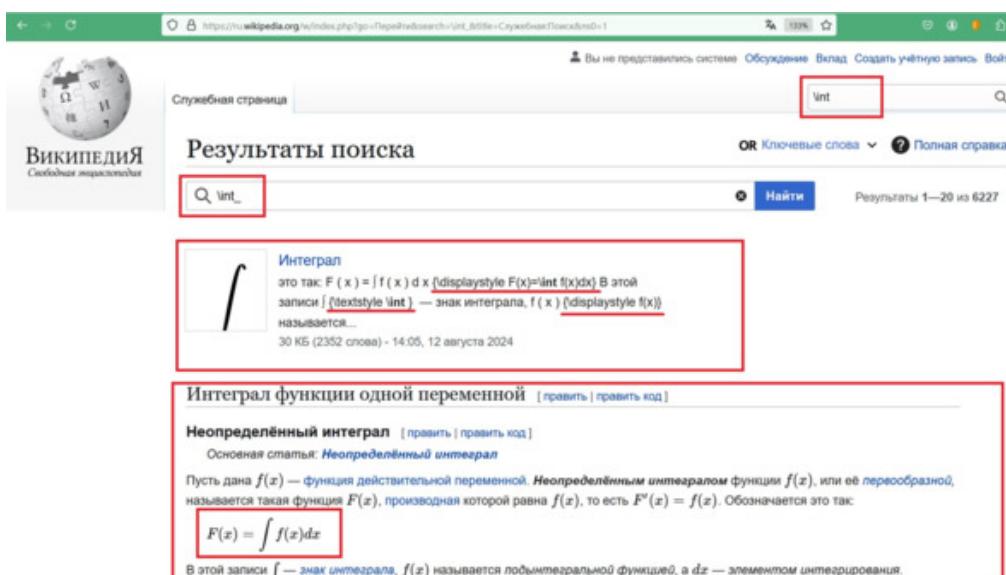


Рис. Поиск формул TeX на Википедии

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

Отдельная проблема и технология, относящаяся к формулам, – это греческие буквы. Они часто используются в 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 оформление 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.

The screenshot shows two screenshots of the scikit-learn User Guide. The top screenshot displays the main homepage with sections for Classification, Regression, and Clustering. The Regression section is highlighted with a red box, showing its description and applications. The bottom screenshot shows a detailed view of the Regression section, specifically the Ordinary Least Squares subsection. It includes a mathematical formula for minimizing residual sum of squares, a plot illustrating the linear fit to data points, and a brief description of the process.

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.

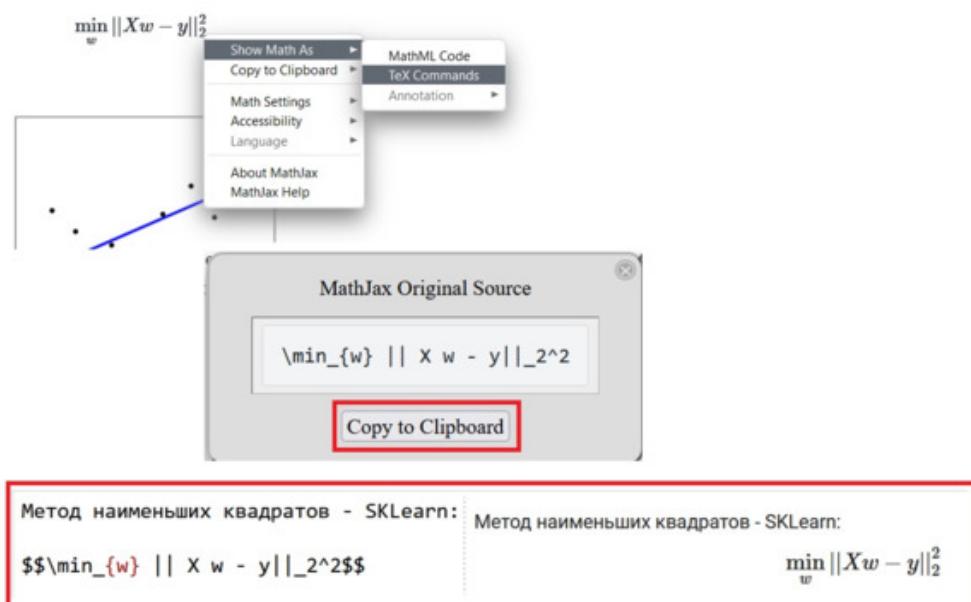


Рис. Вывод формул в нотации TeX

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 Кнутом (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 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/

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

[Греческий алфавит] https://ru.wikipedia.org/wiki/Греческий_алфавит

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

3. ДИАГРАММА КАК КОД

ВВЕДЕНИЕ

В этой работе мы познакомимся с рисованием схем и диаграмм. Это оборотная сторона программирования. Дело в том, что одновременно с написанием текста программы нам предстоит её документировать. Другими словами, мы будем описывать то, что мы делаем. А иногда описывать программу нужно заранее, до начала её составления. Начинающие программисты пренебрегают составлением описания, руководства или даже комментированием своего кода. Понимание приходит со временем.

Написание программы с одновременным составлением документации – это технология Literate Programming. Непереводимая игра английских слов означает, что компьютерная программа – это разновидность литературного произведения. Вроде бы человек пишет программу для компьютера. Но программа должна быть написана так, чтобы её мог легко прочитать и понять человек, а не только компьютер. Через некоторое время после написания программы на неё посмотрит сам автор (который уже всё забыл) или другой человек (который её в глаза не видел), и для этого читателя она должна быть понятной. Так что в программировании важны и комментарии, и описания – в том числе, картинки, схемы, диаграммы.

В работе будут задания, когда вам предлагается почитать, просмотреть, выяснить и так далее. Имеется

в виду, что после прочтения пары страниц текста (иногда с картинками) вы должны быть готовы объяснить основную идею своими словами. Желательно очень простыми словами. Это не то, что нужно зубрить. Это то, что вам предлагается понять.

ОТЧЁТ

В этой работе мы будем оформлять отчёт в виде странички на github. Для этого нам понадобится бесплатно зарегистрироваться на этом сайте и создать новый репозиторий. В этом репозитории мы создаём новый файл в формате Markdown, указав расширение файла *.MD. Внутри этого файла мы используем стандартные средства форматирования, а также вставляем свои результаты по ходу выполнения работы.

Здесь мы время от времени используем знания и навыки из предыдущих работ. Будем считать, что вы эти предыдущие работы уже успешно выполнили. Поэтому здесь мы не будем второй раз объяснять, что такое Markdown, как в него вставляют картинки и как писать запросы к нейросети. А ещё мы не будем во второй раз объяснять, что такое «титульный лист» и что такое «информационное название файла».

Внутри отчёта нужно будет фиксировать те действия, которые вы выполняли, и те результаты, которые Вы получили. Если вы используете внешние источники информации, указываем название источника и даём ссылку на него. Если это обращение к какому-нибудь чат-боту, указываем свой промпт и его ответ. Будем считать, что это упражнение по «документированию».

По окончании работы вам нужно вставить ссылку на файл своего отчёта (который лежит на гитхабе) в соответствующую форму.

Задание. Создайте новый репозиторий на GitHub. Создайте новый файл *.MD. В начале файла предоставьте информацию, как в титульном листе отчёта.

3.1 АЛГОРИТМ

Программирование начинается с алгоритма. Перед тем, как начать писать программу и быстро щёлкать клавишами, желательно понять смысл решаемой задачи и хотя бы приблизительно наметить алгоритм её решения. Лучше сначала подумать, а потом уже поработать. Это лучше, чем сначала сделать, а потом десять раз переделывать. Поэтому и появилась народная пословица: «Семь раз отмерь, один раз отрежь». Так что лучше начать с алгоритма.

Что же такое алгоритм? Одно из объяснений звучит так. Алгоритм – это последовательность шагов по решению какой-то задачи или по достижению какой-то поставленной цели. Стало быть, сначала кто-то (заказчик) ставит нам задачу, объясняет конечную цель работы. А уже потом мы делаем шаги, выполняем действия в определённом порядке и приходим к тому результату, который интересует нас (или заказчика). Если мы сами себе заказчики, то всё равно лучше вначале определиться с целью. А уже потом составлять алгоритм (путь к решению) и писать программу. Так вот, само слово «алгоритм» – это целая история. Первоначально оно означало «Человек из города Хорезм». Это было прозвище одного известного математика.

Задание. Просмотрите статью алгоритм на Википедии. Найдите определение слова алгоритм. Изучите историю происхождения этого слова. Будьте готовы объяснить своими словами, что такое алгоритм и откуда это странное слово к нам пришло.

БЛОК-СХЕМА ИЛИ СХЕМА АЛГОРИТМА?

Алгоритм можно описывать словами, или картинками (комиксами), или схемами. Схема алгоритма – это «кубики» и «стрелочки». А ещё здесь могут быть «ромбики» и «шарики».

Раньше это изображение, эту весёлую картинку называли «блок-схема» (то есть схема, состоящая из блоков). В последнее время решили называть просто «схема алгоритма». Но, как говорится, дело не в названии.

Задание. Просмотрите статью Блок-схема на Википедии. Обратите внимание на стандартные обозначения блоков на этой схеме.

Задание. Выясните с помощью любого интеллектуального помощника, когда «блок-схемы алгоритмов» стали называть «схемы алгоритмов» и почему.

Вообще-то существуют международные соглашения, стандарты – по поводу того, как эти схемы рисовать. У нас это называется ГОСТ – ГО (сударственный) + СТ

(андарт). В этом сокращении взяли по две первых буквы из каждого слова, чтобы лучше звучало. Конечно, ГОСТ приятнее звучит, и его легче произнести, чем ГС.

Задание. Найдите в интернете отечественный стандарт по рисованию схемы алгоритма. Номер и название стандарта можно найти в конце статьи про блок-схему из предыдущего задания. Просмотрите текст стандарта и обратите внимание на примеры оформления схем в конце текста.

Если открыть русскую статью «Блок-схема» на Википедии, а потом перейти на соответствующую английскую страничку, мы увидим совсем другое название. Кстати, с помощью такого приёма можно найти эквивалент или общепринятый перевод терминов с одного языка на другой. Иногда это помогает и работает как дополнение к словарям. А иногда и не помогает.

Мы попадаем на статью под названием Flowchart – это график Chart, изображающий какие-то потоки Flow. Эта статья упоминает другой термин – Workflow – это поток Flow работ Work. Более того, в последнее время часто упоминается похожее название: Workflow Diagram (WFD) – диаграмма потока работ, или «потоковая диаграмма». Хорошо бы нам с этим разобраться и определиться.

Задание. Выясните с помощью интеллектуального помощника, что такое Workflow Diagram и есть ли здесь какая-то связь со схемой алгоритма Flowchart.

DRAW.IO

И так, в настоящее время общепринятое название звучит как «схема алгоритма». Вот её мы и будем рисовать. Есть разные способы и инструменты для рисования схемы алгоритма.

Раньше схемы вообще рисовали от руки и на бумаге. Для этого нужно было знать стандарты и уметь их использовать.

Можно использовать графический редактор типа Paint. Здесь есть кое-какие готовые блоки – и кубики, и стрелочки. Вроде бы всё хорошо, но редактировать и исправлять такую схему не слишком удобно.

Есть и более специализированные средства. Вот, например, бесплатный онлайн-сервис [Draw.io]. При запуске сервиса мы видим сообщение по поводу рисования Flowchart.

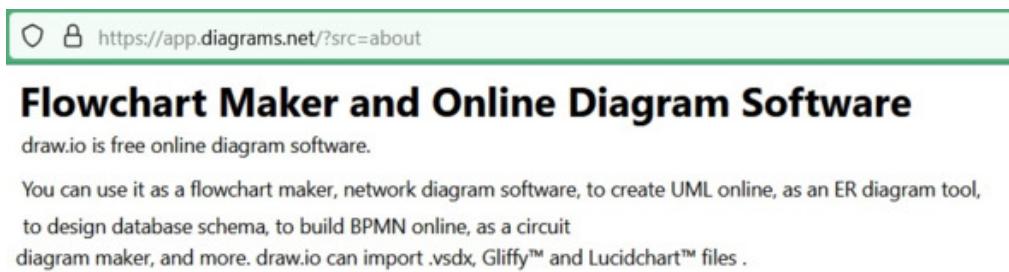


Рис. Запуск draw.io

Затем нас перенаправляют на другой адрес [diagrams.net](https://app.diagrams.net/). Это первоначальный адрес сервиса. Нажимаем кнопку Start – Начать. Начинаем работу и создаём новую диаграмму – Create New Diagram. Можно выбрать шаблон Flowchart, а можно начать с чистого листа.

Далее выбираем создание пустой диаграммы: Blank Diagram – Create.

Задание. Запустите сервис Draw.io. Изучите шаблоны Flowchart. Создайте пустую диаграмму.

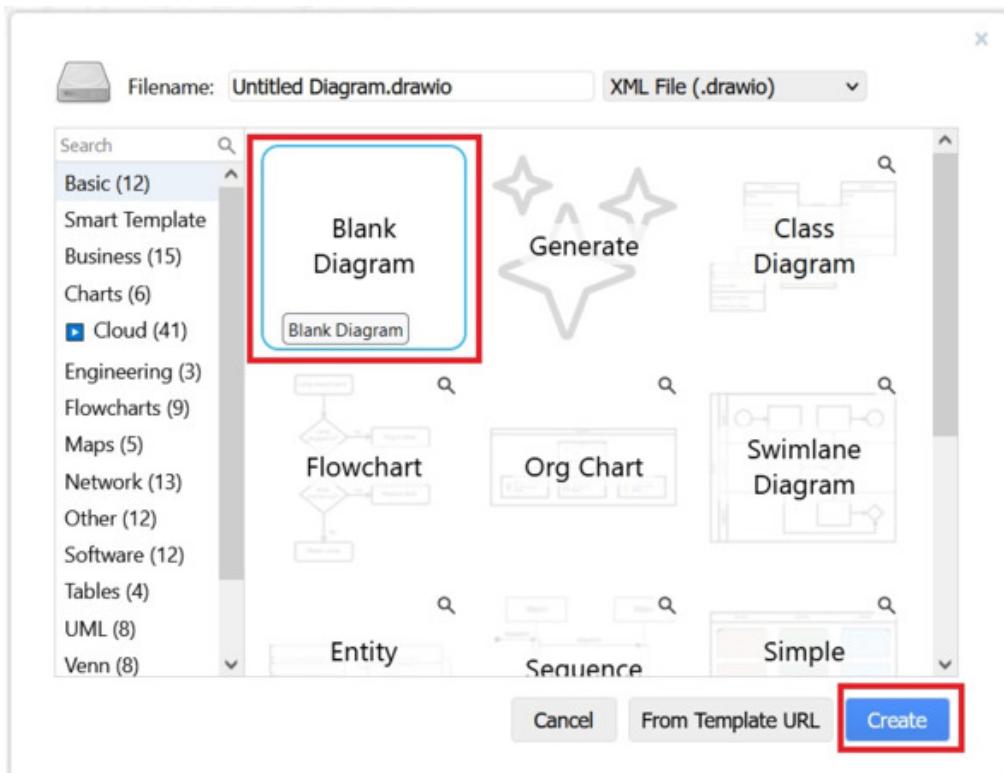


Рис. Создаём пустую диаграмму

Теперь мы можем выбирать нужные формы блоков из палитры инструментов в левой панели экрана. Нас будут интересовать два списка: блоки общего назначения General и блоки для рисования схемы алгоритма Flowchart.

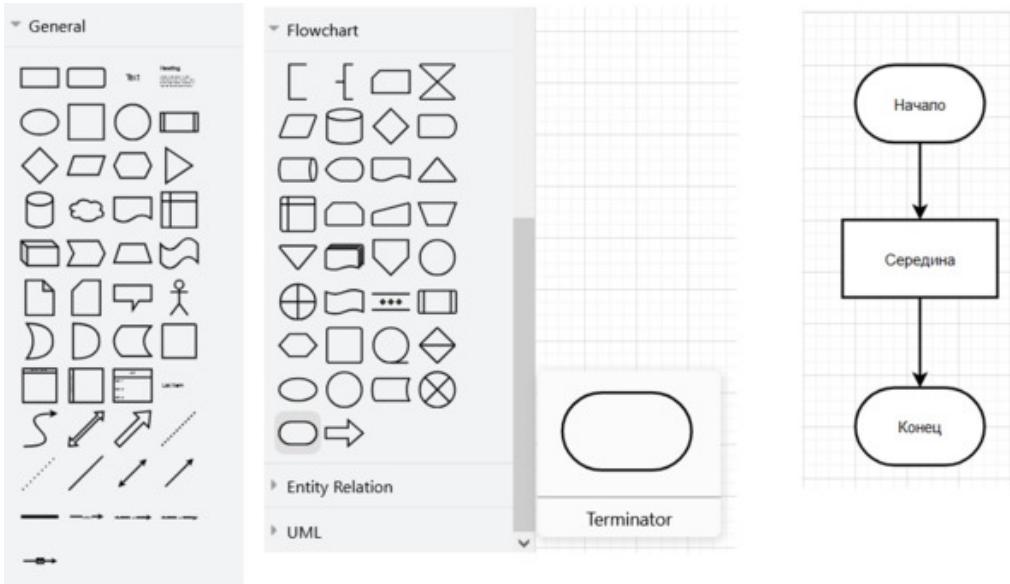


Рис. Палитры блоков и схема алгоритма

Некоторые студенты ограничиваются первым списком. В результате мы получаем распространённую ошибку, когда начало и конец программы обозначается овальчиком из списка General. Правильный блок начала и конца программы находится во второй палитре инструментов для рисования схемы алгоритма. Если навести на него курсор, появится всплывающая подсказка с названием блока. Очень интересное название.

Задание. Выясните, что означает английское слово Terminator (Терминатор), его происхождение, и какое отношение всё это имеет к рисованию схемы алгоритма и к фильмам с участием Шварценеггера.

Чтобы построить нашу первую схему, открываем нужную палитру инструментов и перетаскиваем выбранный блок на свободное место нашего холста. Затем щёлкаем мышкой внутри блока и пишем – что

выполняется внутри этого блока. Чтобы соединить блоки между собой, подводим курсор к нижней или верхней стороне блока и, нажав левую кнопку мыши, «вытягиваем» стрелку из одного блока и «втыкаем» её в другой блок. Здесь есть масса дополнительных настроек, таких как толщина линий и так далее. Но это всё мелкие несущественные подробности. Для начала просто нарисуем такую схему, как показано на рис.

Задание. Запустите сервис Draw.io и постройте простой алгоритм из трёх блоков, как показано на рисунке. Сохраните схему в файле на локальном компьютере.

Есть возможность установить эту программу на своём локальном компьютере, но для первого знакомства нас устроит вариант работы через браузер.

Попробуем сохранить нашу схему. Выбираем в верхнем меню File – Save as... и изучаем доступные варианты: XML, PNG, SVG, HTML.

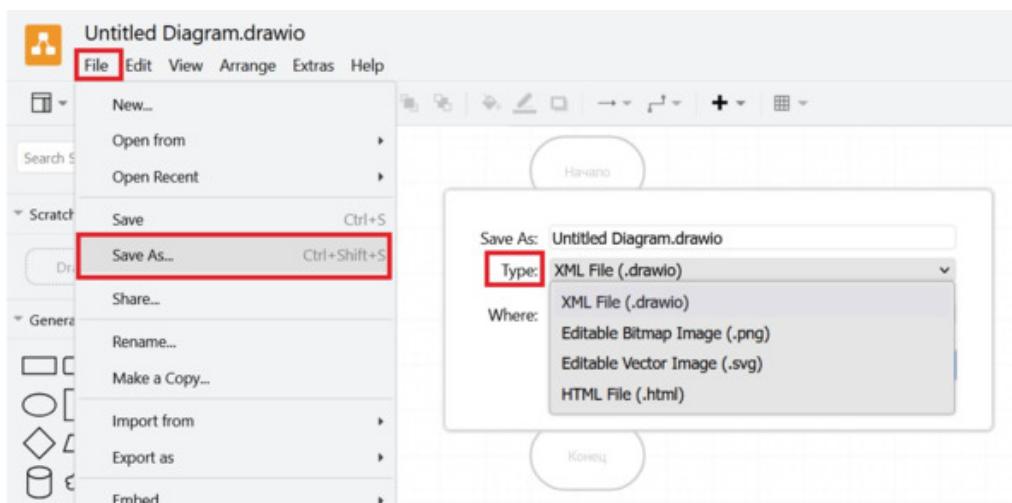


Рис. Варианты сохранения диаграммы

Задание. Сохраните вашу диаграмму во всех четырёх форматах. Ознакомьтесь с содержимым файлов. Изучите, как эти файлы отображаются на экране при просмотре. Проверьте, как меняется качество изображения при увеличении.

Задание. Разберитесь, как вставить эти изображения в документы типа DOCX и MD.

Одна из проблем с использованием таких схем-это редактирование диаграммы. чтобы внести изменения в нашу схему, нужно открыть её в соответствующем редакторе точка затем сохранить этот файл и снова вставить его в документ. В некоторых случаях есть возможность полной интеграции редактора текста и редактора диаграммы. Но всё это вызывает много дополнительных сложностей и лишних телодвижений.

MERMAID

Новый, модный, популярный подход к рисованию схем называется Diagram as Code – «Диаграмма как код». Такая технология постепенно набирает популярность среди разработчиков программного обеспечения, и её стараются включить в свои продукты многие ИТ-компании. Здесь мы не рисуем диаграмму, а только описываем её с помощью сценария, скрипта. Потом программа сама генерирует нужную картинку. Это рисование с помощью текста, текстового описания.

Один из примеров бесплатных, популярных онлайн сервисов такого рода – это [Mermaid]. Открываем сайт <http://mermaid.js.org/> и переходим по ссылке Live editor.

Вся программа рисования схемы занимает пару строку кода. В первой строке мы объявляем тип диаграммы: Flowchart. Во второй строке мы описываем блоки и стрелки между ними.

Скобки задают форму блока. Как видим, квадратные скобки позволяют создать прямоугольник. Круглые скобки плюс квадратные формируют блок начала-конца программы (терминатор). Есть и другие варианты, но об этом попозже.

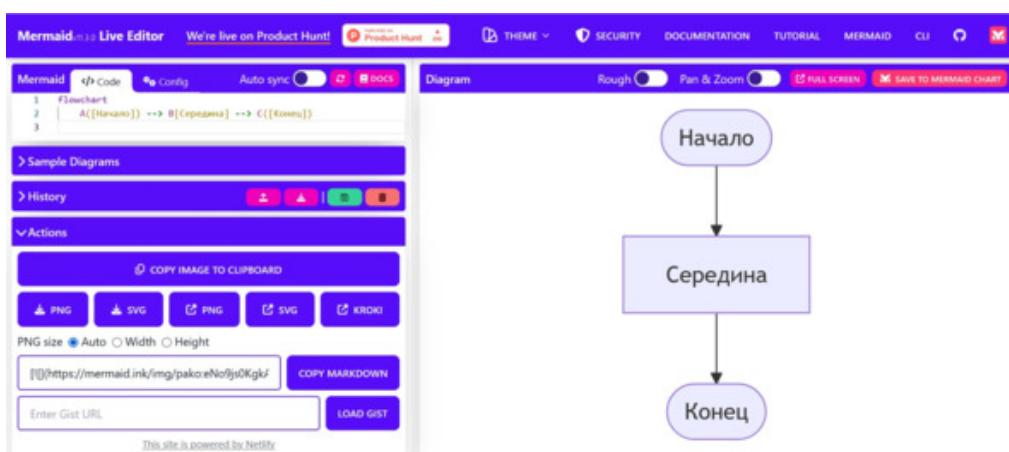


Рис. Схема простого алгоритма

Обратим внимание на форматы для сохранения нашей схемы на локальном компьютере: PNG и SVG. Эти возможности открываются в разделе Actions – Действия.

Задание. Сохраните свою диаграмму в разных форматах и изучите качество изображения.

Можно сразу скопировать схему в буфер обмена: Copy image to clipboard. Что при этом происходит, пока неизвестно. Но с этим можно разобраться на опыте. Давайте проведём несложный эксперимент.

Задание. Скопируйте схему в буфер и вставьте её в текстовый редактор типа Word и в графический редактор типа Paint. Сделайте вывод о том, в каком формате наше изображение копируется в буфер.

Кроме скачивания файлов, нам дают возможность поделиться ссылкой на веб-страничку с нашей диаграммой – в тех же форматах PNG и SVG. На самом деле это не адрес «нашей странички», а возможность оперативно нарисовать нашу схему онлайн. При этом в строке адреса мы передаем длинную строчку символов, где зашифрован сценарий рисования нашей диаграммы.

Сформируйте ссылки, чтобы поделиться своей схемой онлайн. Обратите внимание на строчку адреса, в которой ваша схема передаётся на сервер для отрисовки.

Самое интересное – это кнопка копирования в формате Markdown. Здесь тоже передаётся ссылка для форматирования диаграммы онлайн. Это уже интересно, но для редактирования диаграммы нам снова придётся вернуться в редактор, а потом заменить ссылку на диаграмму.

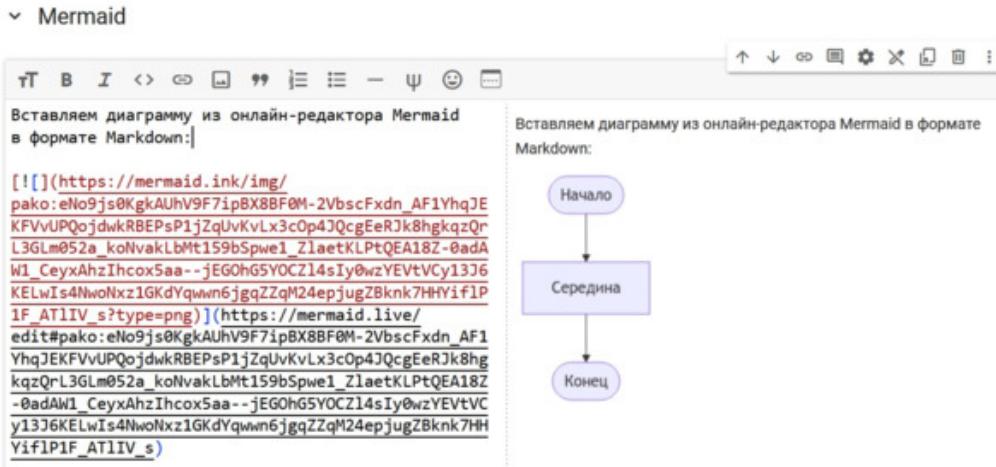


Рис. Вставка диаграммы в виде ссылки

Задание. Скопируйте диаграмму в буфер обмена в формате Markdown и вставьте её в текстовую ячейку Google Colab. Затем запустите текстовую ячейку на выполнение и перейдите по ссылке, щёлкнув мышкой по диаграмме. Откроется страница редактирования. Внесите небольшое изменение в диаграмму. Скопируйте ссылку на код Markdown и вставьте его в эту же текстовую ячейку. Обратите внимание на изменение текстовой строки.

MERMAID + GITHUB

Совсем недавно, буквально пару лет назад на сервисе Github добавили возможность вставки диаграмм формата Mermaid в текст Markdown. Для этого нужно написать блок кода, описывающего диаграмму, и окружить его тройными наклонными апострофами, указав название формата, см. рис.

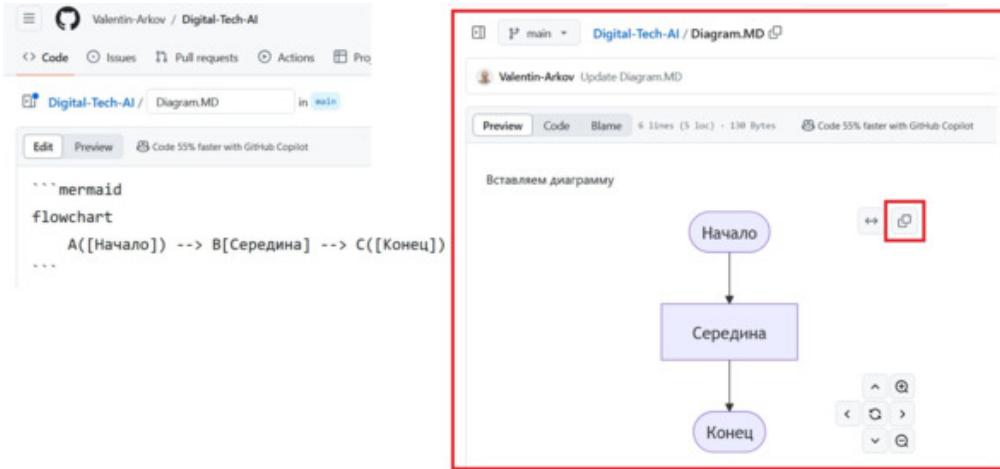


Рис. Вставка диаграммы в файл MD

После сохранения изменений мы можем перейти к просмотру этой страницы и увидеть нашу диаграмму. справа от неё появляется кнопка копирование в буфер. интересно, В каком формате будет скопирована наша картинка.

Задание. Создайте в своём репозитории на GitHub новый файл с расширением MD и вставьте в него свою схему. Убедитесь, что схема отображается корректно. Скопируйте свою схему в буфер обмена и вставьте её в текстовый и графический редактор. Сделайте вывод о том, в каком формате копируется изображение.

Для рисования схемы алгоритма в нашем распоряжении имеются самые разные блоки и стрелки. Эти средства подробно описаны в онлайн документации на странице проекта [Mermaid: Flowcharts].

Задание. Постройте более сложную схему алгоритма, в которой будут условный переход и вывод на печать. Вставьте схему в свою страничку на github.

MERMAID + COLAB

Чтобы вставить диаграмму Mermaid в кодовую ячейку Google Colab, придётся выполнить более сложные действия. Весь текст программы можно будет сгенерировать любым более-менее интеллектуальным чат-ботом. Имеется также общее описание на странице [Mermaid in Jupyter].

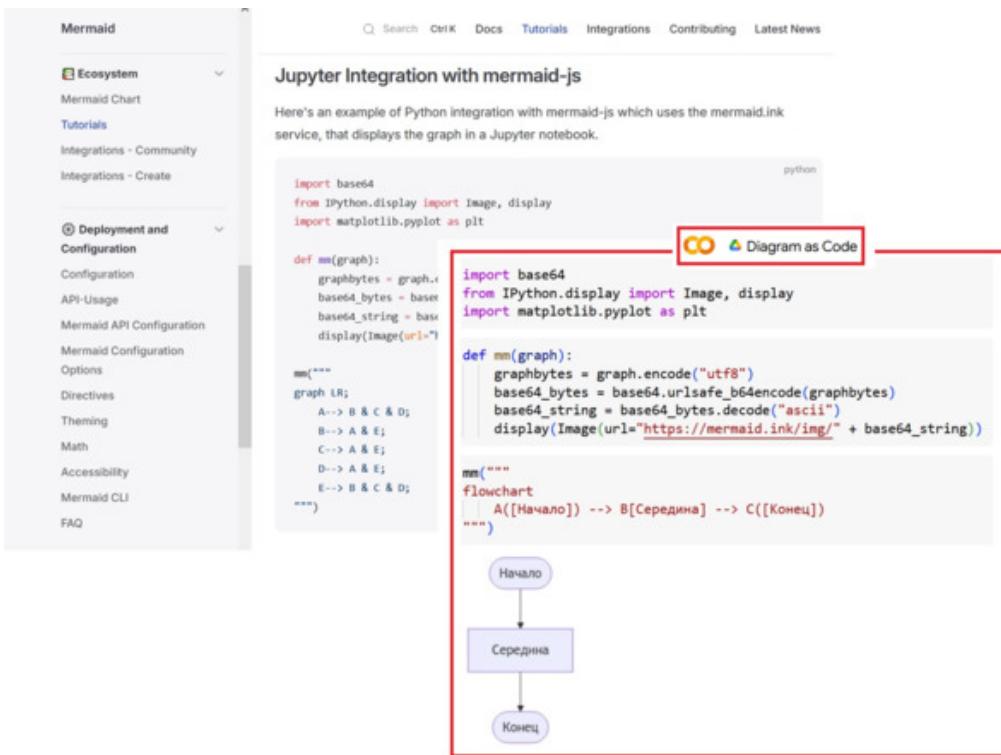


Рис. Диаграмма в кодовой ячейке

В этом примере можно наблюдать всю цепочку действий по созданию закодированного скрипта для онлайн-отрисовки нашей диаграммы сервисом [Mermaid Ink].

Начнём разбираться на следующем рисунке. Мы приводим весь текст этой программы с указанием номеров строк. Дальше мы обсудим каждую строчку.

```
1 import base64
2 from IPython.display import Image, display
3 import matplotlib.pyplot as plt
4
5 def mm(graph):
6     graphbytes = graph.encode("utf8")
7     base64_bytes = base64.urlsafe_b64encode(graphbytes)
8     base64_string = base64_bytes.decode("ascii")
9     display(Image(url="https://mermaid.ink/img/" + base64_string))
10
11 mm("""
12     graph TD
13     A([Начало]) --> B([Середина])
14     B --> C([Конец])
15 """)
```

Рис. Программа с номерами строк

Вот номера строк программы и наши комментарии.

(1) Программа начинается с импорта библиотек. При этом необходимый модуль автоматически загружается в память машины, чтобы мы могли вызывать из него готовые функции. Библиотека `base64` позволяет закодировать двоичные (бинарные) данные в текстовый формат Base64. Кодировка `base64` позволяет записать любые данные и файлы с помощью латинских букв и цифр.

(2) Конструкция `from` загружает две функции для работы с изображениями. Мы берём их в модуле `display` из библиотеки `IPython`. Эти функции нужны для работы с изображениями в интерактивном блокноте `Jupyter Notebook`. Функция `Image` создаёт объект, в котором находится наше изображение. Функция `display` позволяет вывести это изображение на экран.

(3) Загружаем модуль `Pyplot` из библиотеки `Matplotlib`. Библиотека занимается визуализацией данных, а вот что означает её название – выясните сами. Модуль `Pyplot` облегчает создание графиков. Наконец, с помощью конструкции `as plt` мы назначаем общепринятый псевдоним `plt` для модуля `Pyplot`. Это немного упростит дальнейшую программу.

(4) Пустая строка отделяет загрузку библиотек и объявление функции. Таким способом мы делаем программу более удобной для чтения человеком.

(5) Определяем функцию под названием `mm`. У этой функции будет один аргумент под названием `graph` – это строка символов, в которой и будет закодирована наша схема. Ключевое слово `def` – это начало английского слова `define` – определять, задавать.

(6) Берём строку `graph`, полученную как аргумент функции, и записываем её в кодировке UTF-8. Это промежуточный, подготовительный этап для дальнейшей работы. Попутно проясните для себя, что такое UTF-8.

(7) Используем функцию `urlsafe_b64encode`, чтобы закодировать график в формат Base64. Название `urlsafe` говорит о том, что полученный код будет «безопасным» (`safe`) в строке адреса URL. Другими словами, в нём не будут использоваться нежелательные символы типа наклонной черты, для которых в веб-адресах есть своё предназначение.

(8) Полученные символы перекодируем в формат ASCII. Полученная строка символов лучше работает как часть адреса URL.

(9) Вызываем функцию `display`, чтобы вывести нашу схему на экран. Диаграмма передаётся для отображения с помощью `Image`. Собственно изображение формируется на сайте `mermaid.ink`. Мы отправляем свою диаграмму в виде строки текста `base64_string`, которую мы только что сформировали. Передача параметров под видом адреса URL – это пример API в действии. Скоро мы обсудим эту технологию.

Таким образом, наша функция `mm (graph)` получает текстовое описание диаграммы и выводит на экран

готовое изображение. При этом функция никаких значений не возвращает. Она просто рисует диаграмму.

(10) Ещё одна пустая строка помогает нам отделить определение функции от дальнейшей программы. Такую программу легче будет читать.

(11) Вызываем нашу функцию `mm` и передаём ей в качестве аргумента текстовое описание диаграммы. Этот текст состоит из нескольких строк (`line`). Но при этом с точки зрения Питона это переменная типа «строка» – `string`. Таким образом, это особый объект под названием `multiline string` – «многострочная строка». Здесь разные английские слова `line` и `string` переводятся одинаково – как «строка», и это может по началу сбивать с толку. Такие объекты обрамляют тройными кавычками.

Задание. Ознакомьтесь со следующими инструментами и технологиями. Можете использовать Википедию и интеллектуальный чат-бот либо просто поиск в интернет.

`base64`, `IPython`, `Jupyter Notebook`, `Pyplot`, `Matplotlib`, `UTF-8`, `URL`, `ASCII`, параметры и аргументы функции, `string`, `multiline string`.

Задание. Перейдите на сайт <https://mermaid.ink/> и изучите пояснение Getting Started. Проверьте, как поведёт себя на вашем компьютере приведённый в статье пример.

Итак, для вывода графика параметры передают на сайт в виде URL. Как этот «адрес» выглядит? Попробуем вывести сформированный URL на экран. Для

этого добавим команду print в нашу функцию, см. рис. Получаем длинную строку букв и цифр.

```
def mm(graph):
    graphbytes = graph.encode("utf8")
    base64_bytes = base64.urlsafe_b64encode(graphbytes)
    base64_string = base64_bytes.decode("ascii")
    url="https://mermaid.ink/img/" + base64_string
    print(url)
    display(Image(url=url))

mm("""
graph TD
    A([Начало]) --> B[Середина] --> C([Конец])
""")
```

The screenshot shows a Jupyter Notebook cell containing Python code. The code defines a function `mm` that takes a `graph` object, encodes it to `utf8`, converts it to `base64`, decodes it to `ascii`, and then constructs a URL using `https://mermaid.ink/img/`. The resulting URL is printed and displayed as an image. Below the code, a Mermaid diagram is shown with three nodes: `Начало`, `Середина`, and `Конец`, connected by arrows indicating a linear flow from start to middle to end.

Рис. Генерация адреса для Mermaid Ink

Теперь мы можем скопировать эту ссылку в буфер и оформить текстовую ячейку, как показано на рис.

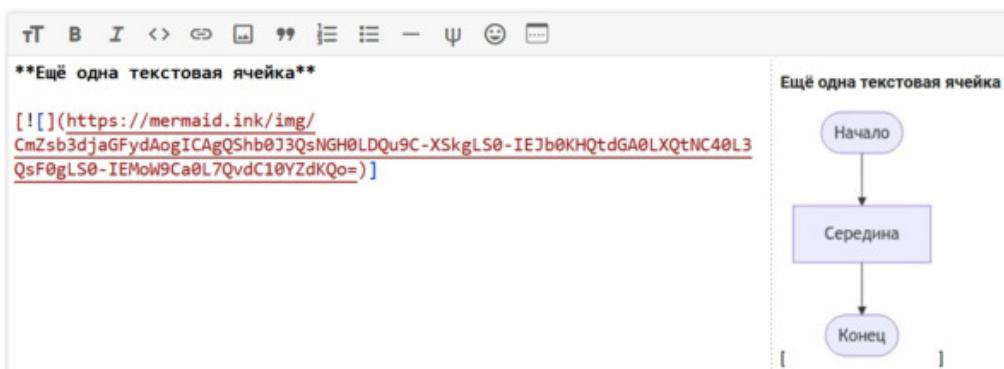


Рис. Вставка изображения диаграммы в текстовую ячейку

Задание. Сформируйте ссылку для вывода вашей диаграммы и скопируйте её в буфер. Используя эту ссылку, напишите код для текстовой ячейки. Убедитесь, что диаграмма корректно отображается на экране.

BASE64 + API

Теперь осталось разобраться с кодировкой. Что же такое base64?

Задание. Просмотрите статью [Wiki: Base64] на Википедии. Выясните, в чём заключается основная идея такого кодирования и что означает число 64 в этом обозначении. Обратите внимание на то, какие символы входят в «алфавит» этой кодировки – base64 alphabet. Это поможет понять, почему их набралось именно 64 штуки. Конечно, это еще и степень двойки – для удобства передачи байтами.

То, что мы видим в строке адреса, на самом деле очень популярный способ общения с сервером. Внешне это выглядит как обращение к какой-то страничке сайта. Мы видим адрес, наклонной черту и что-то вроде каталога и название – то ли файла, то ли каталога. Это такой способ передать запрос на сервер. Мы сообщаем, какую услугу мы хотим получить, а также указываем дополнительные параметры. В нашем примере мы говорим серверу, что хотим получить изображения нашей схемы виде графического файла. Затем в этой же строке мы в зашифрованном виде передаем описание нашей схемы на языке Mermaid. В ответ сервер передает нам изображение схемы.

Вся эта технология называется Rest API. В переводе на человеческий язык это красивое название означает, что сервер не запоминает наши предыдущие запросы. Вся необходимая информация содержится в этой строчке «веб адреса». Для использования этого сервиса

мы вызываем готовые команды и формируем запрос из нашего скрипта для рисования схемы. Вот и всё.

MERMAID + VS CODE

Мы с вами немного познакомились с изображением схемы алгоритма и интеграции с форматом Markdown. Конечно, здесь есть и другие возможности. Например, с этим форматом можно работать в среде программирования типа Visual Studio Code. Для этого потребуется установить специальный «плагин» – дополнение для просмотра диаграмм – Markdown Preview Mermaid Plugin.

Для опытов во многих случаях можно использовать переносимую версию по-английски называется Portable application. Мы про это уже упоминали в предыдущей главе, когда показывали возможность поработать с текстовым редактором Notepad++. Так вот, для VSCode тоже есть вариант Portable. Подробности описаны на странице Portable Mode:

<https://code.visualstudio.com/docs/editor/portable>

здесь говорится, что переносимый режим имеется для Windows, Linux и macOS.

Для операционной системы Windows скачиваем архив ZIP и распаковываем. Затем запускаем на выполнение файл Code.exe.

Открываем вкладку Extensions – Расширения и дополнения. В строке поиска пишем Markdown. Выбираем Markdown Preview Mermaid Support by Matt Bierner. Нажимаем Install – Установить.

По окончании установки можно знакомиться с кратким описанием. Здесь говорится: Adds Mermaid diagram and flowchart support to VS Code's builtin markdown preview – Это расширение добавляет поддержку диаграмм и схем Mermaid в инструмент предварительного просмотра Markdown, встроенный в Visual Studio Code.

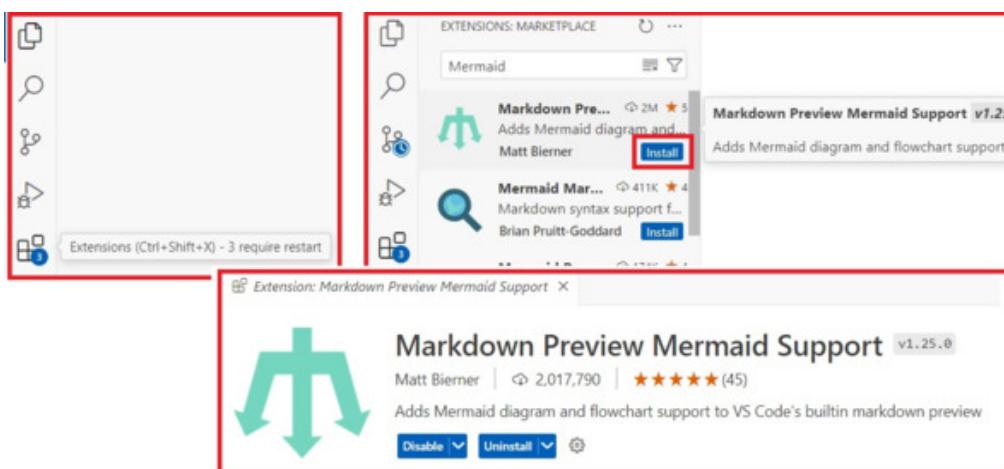


Рис. Установка расширения для просмотра Mermaid

Далее создаём новый файл в формате Markdown. Для этого добавляем новый каталог: File – Add folder to workspace. Пусть это будет новая папка diagrams. Создаём новый файл: File – New file. Придумаем ему простое название и указываем расширение *.md.

Вставляем наш код для диаграммы Mermaid, оформленный точно так же, как и на GitHub. Вызываем предварительный просмотр через контекстное меню: Open preview.

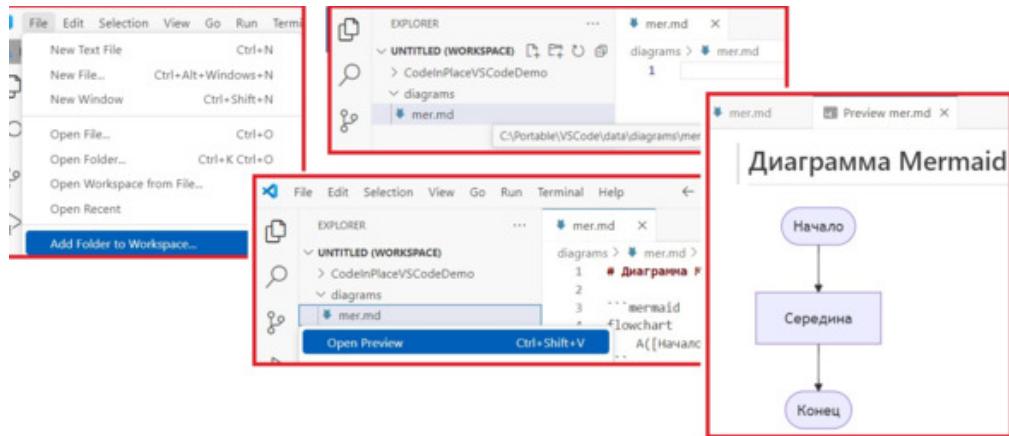


Рис. Просмотр диаграммы Mermaid

MERMAID + DOCKER

Наконец попробуем ещё одну комбинацию инструментов. Нам предстоит запустить онлайн редактор Mermaid из контейнера docker. Мы будем обращаться к нему через веб-браузер. Внешне это будет напоминать работу в интернете, но все программы будут работать на локальном компьютере.

Для запуска контейнеров необходимо установить платформу контейнеризации докер. Частично мы уже рассматривали эту технологию в предыдущих разделах. Подробно технология контейнеризации описана в нашем пособии, см. [Арьков Контейнеризация].

Для работы в операционной системе Microsoft Windows потребуется установить программу Docker Desktop. Первоначально мы запускаем на выполнение эту программу, причём с правами администратора. Затем из командной строки (тоже с правами администратора) запускаем контейнер со всеми настройками и параметрами.

Основная идея технологии контейнеры состоит в том, чтобы собрать все необходимые программы

и даже кусочек операционной системы в один файл. Его называют «образ». Обычно эти файлы лежат на каком-нибудь сервере. На сегодняшний день самый популярный бесплатный сервер для хранения образов называется Docker Hub.

Когда мы запускаем на выполнение такой образ, он скачивается с сервера на локальный компьютер и разворачивается для запуска. В процессе выполнения создаётся ещё один объект, который называется контейнер.

Можно сравнить эту технологию с привычным запуском программы. Пока наша программа лежит на жёстком диске, её обычно называют словом «файл». Когда мы запустили файл на выполнение, он копируется, разворачивается в оперативной памяти компьютера. Теперь его можно увидеть в Диспетчере задач в списке запущенных программ. Их ещё называют словом «процесс». По своему поведению образ напоминает файл на диске, а контейнер похож на вычислительный процесс, то есть на запущенную программу. На самом деле всё гораздо сложнее, но для первого знакомства такого понимания уже будет достаточно. Нам нужно просто запустить эту «штуку» и увидеть, что она работает.

Чтобы найти нужный образ контейнера, открываем сайт Docker Hub:

<https://hub.docker.com/>

Для наших задач достаточно будет бесплатной регистрации на сайте.

В строке поиска сайта пишем название инструмента, который нас интересует:

mermaid-live-editor.

Это онлайн редактор Mermaid. Мы уже использовали его в начале данной работы. Здесь слово *live* означает «онлайн» или «работа в реальном времени». Между прочим, это не глагол, а прилагательное. И читается оно так: «лайв». Очень популярный термин. На телевидении означает передачу в прямом эфире, а не в записи. В компьютерных технологиях может означать «онлайн» или работу без установки на компьютер.

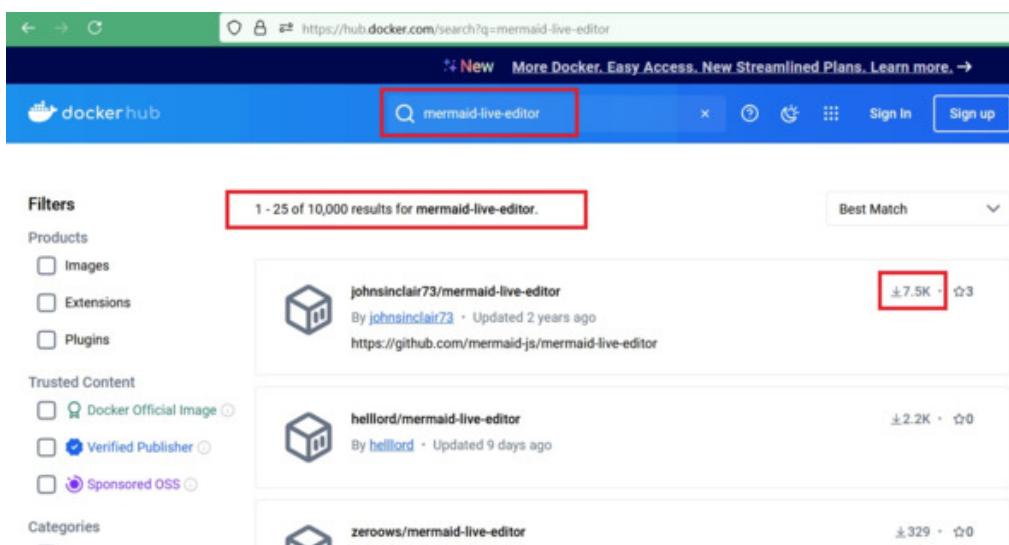


Рис. Поиск образа

Нам сообщают, что найдено 10.000 ссылок точка обычно Они отсортированы по убыванию популярности. мы можем видеть количество загрузок образа и количество положительных отзывов.

Возьмём для примера самый популярный образ. Здесь говорится что его скачали 7,5 тысяч раз. Напомним, что буква «К» для программистов означает «кило», то есть «примерно тысяча». Переходим по ссылке и изучаем страничку образа.

Здесь будет много технических подробностей, но нам достаточно будет найти всего одну строчку. Это

команда для запуска контейнера. И находится она в разделе Run with Docker.



Рис. Команда для запуска

Команда будет длинная, и запоминать её пока не потребуется. Подводим курсор к этой команде и нажимаем кнопку Copy – копировать в буфер обмена. Затем запускаем командное окно, или командную строку, или терминал, или консоль, или CLI (command line interface). Это Разные названия одного и того же инструмента. это Чёрное окно с белыми буквами, где мы вводим команды. так вот, нам нужно запустить это командное окно тоже с правами администратора точка Мы вставляем из буфера нашу длинную команду и запускаем её на выполнение. В этом примере она будет выполняться, и сама она свою работу не остановит. Мы не будем ей мешать и не будем закрывать это командное окно.

В этой команде есть один самый полезный параметр. Это число 8080. И называется оно «номер порта». В данном случае «порт» – это число, которое говорит серверу о том, какой сервис, какую услугу мы хотим от него получить. Номер порта указывают в строке в адреса через двоеточие после адреса сайта. Таким

способом мы получаем доступ к web-интерфейсу нашего контейнера через браузер.

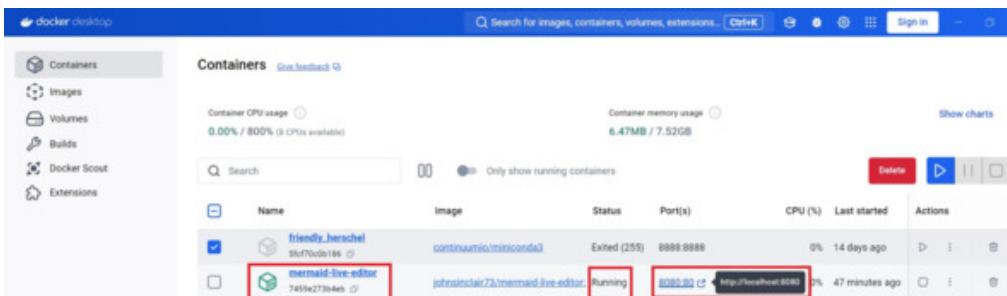


Рис. Доступ к контейнеру

Переходим в графический интерфейс Docker Desktop. Находим в списке контейнеров знакомое название. В колонке Status – состояние – мы видим что он выполняется – Running. Колонки Force порты появляется работающая ссылка подводим точка подводим к ней курсор и во всплывающей подсказке видим полную ссылку. В ней указан протокол работы с этим сервисом http, название нашего компьютера localhost и через двоеточие адрес порта 8080.

Щёлкаем по этой ссылке и наблюдаем, как в браузере открывается этот сервис. Внешне он выглядит точно так же, как и онлайн редактор. Только в строке адреса указан локальный компьютер. Вставляем в раздел кода текстовое описание своей диаграммы и нажимаем кнопку обновить – Sync diagram. Можно также переключиться в режим автоматического обновления – Auto sync – синхронизация изменений кода и диаграммы.

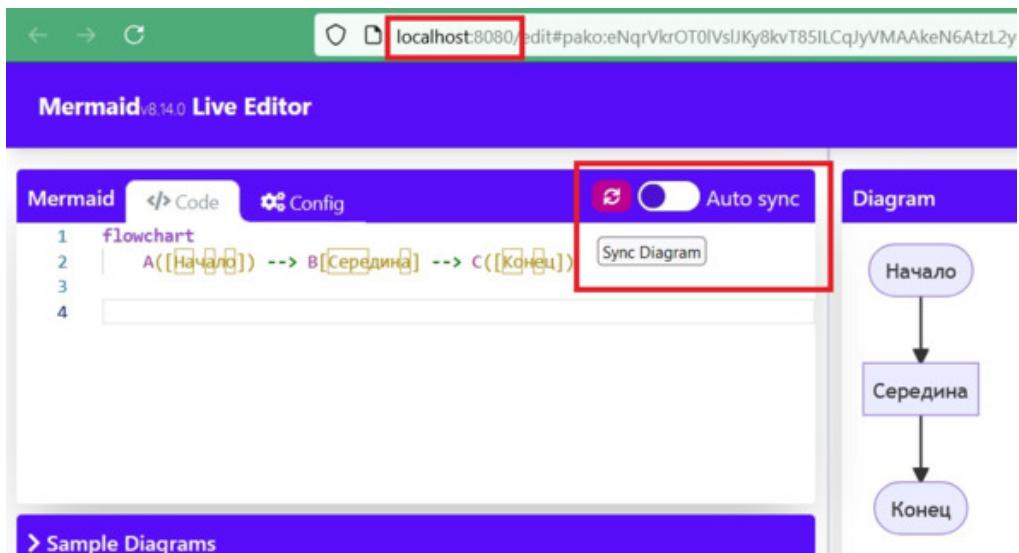


Рис. Синхронизация изменений

Задание. Выполните описанные выше действия. Запустите контейнер с онлайн редактором. Вставьте свой код и сгенерируйте диаграмму. Вставьте копии экрана в отчёт.

Технология контейнеризации позволяет выполнять программу, не занимаясь её установкой и настройкой. На локальном компьютере не остаётся никаких следов от нашей работы. Мы не засоряем системный диск и реестр операционной системы. Мы также можем удалить все следы, если в Docker Desktop удалить контейнеры, а затем удалить их образы.

Задание. После завершения работы с контейнерами удалите сначала контейнеры, а затем образы.

ПОЧЕМУ РУСАЛОЧКА?

Итак, мы познакомились с разнообразными вариантами работы в продукте под необычным названием Mermaid. Для полноты картины хорошо бы выяснить, что хотели нам сообщить разработчики таким названием. У программных продуктов бывают названия «говорящие», бывают названия информативные, а ещё бывают названия безо всякого смысла.

Mermaid переводится с английского как «русалочка» = *mer* (море, морская) + *maid* (девушка). Возможно, этот сказочный персонаж намекает на полёт фантазии, эстетику и изящество форм. Конечно, речь идёт про изящество и визуальную привлекательность диаграмм и схем. В общем, про красоту и гибкость во всех отношениях. Но пока это предположение.

Задание. Используйте интеллектуальных помощников и выясните значение названия и возможный смысл Mermaid, который вложили в это название авторы продукта.

3.2 MIND MAP

Второй вид диаграмм, который мы будем рассматривать, называется «ментальная карта» или «интеллект-карта». Это упрощённое изображение взаимосвязи элементов в виде дерева. Такая схема может визуально представить структуру документа или доклада, а также какие-либо требования, организованные в виде иерархии. Есть много программных средств для построения ментальной карты. А раньше такие схемы вообще рисовали от руки.



Рис. Простая карта

Разные программные средства стараются отличаться от других хоть чем-нибудь. Иногда выглядит необычно. Конечно, это может привлечь внимание. Можно даже рассматривать её как произведение искусства – в стиле поп-арт. Но, в конечном счёте схема должна передать идею от составителя к читателю.

Схема на рисунке выполнена в Mermaid. На наш взгляд, авторы немного перестарались. Эксперты рекомендуют использовать на схемах два-три основных цвета, а при большом желании добавлять их оттенки.

Сравните схему на рисунке ниже с предыдущей. Какая из них более информативна?



Рис. Оформление карты в стиле поп-арт

Задание. Просмотрите статью Mind map на Википедии и обратите внимание на примеры таких диаграмм.

PLANTUML

Для рисования ментальной карты мы будем использовать второй популярный инструмент под названием PlantUML. Он тоже строит диаграммы по текстовому описанию. Это тоже «диаграмма как код».

Задание. Просмотрите статью PlantUML на Википедии. Обратите внимание на программы, которые поддерживают эту технологию.

Название данной программы состоит из двух частей. Слово Plant может означать «растение», а также «сажать, выращивать» — если это глагол. Вторая часть названия UML — unified modeling language — Универсальный, унифицированный, стандартизованный язык моделирования, построение моделей. Это ряд

стандартов по изображению различных схем. Скорее всего, авторы программы хотели нам сказать или намекнуть, что эта программа позволяет как бы «выращивать» диаграмму по словесному описанию. Примерно также, как мы можем вырастить целое дерево из семечка. Кстати говоря, именно дерево мы с вами и будем рисовать.

Задание. Откройте страницу проекта PlantUML на GitHub. Обратите внимание на изображение зеленого листочка рядом с названием.

На странице <https://plantuml.com/mindmap-diagram> подробно описаны основные приемы построения такой диаграммы и приводятся простые понятные примеры.

Задание. Просмотрите начало описания Mind map и обратите внимание на области применения данного инструмента.

КАРТА ПРОФЕССИИ

Мы будем строить ментальную карту для того, чтобы систематизировать, упорядочить основные идеи, касающиеся какой-нибудь компьютерной специальности. В наших примерах в данном разделе будет рассматриваться специальность, профессия под названием системный аналитик. Это специалист, который изучает требования и пожелания по автоматизации и информатизации, то есть требования к будущей информационной системе.

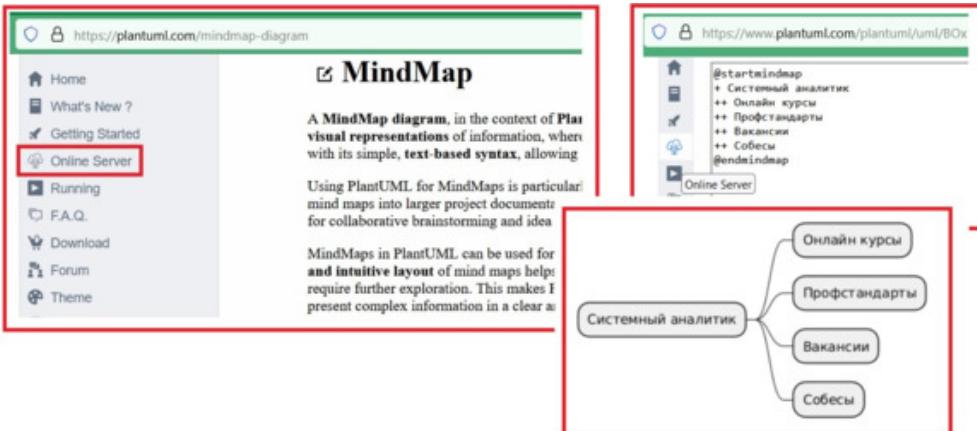


Рис. Карта знаний системного аналитика

На этой схеме мы изобразили основные источники сведений о профессии системного аналитика. И это личное мнение автора. Другие источники могут сыграть более важную роль – для другого человека.

Есть популярные онлайн-курсы, на которых за некоторое вознаграждение всех желающих обещают сделать профессионалом в какой-нибудь ИТ-специальности. А ещё почти гарантируют трудоустройство, конечно, с некоторыми оговорками. Например, нужно будет разослать сотню-другую резюме и посетить с десяток собеседований в неделю. При этом можно ознакомиться с программой курса и определиться с основными разделами подготовки.

Далее, для многих профессий в нашей стране разработан ряд профессиональных стандартов – то есть это «стандарты с описанием профессий». Это фактически пожелания ведущих работодателей. В таких документах описаны «трудовые функции» – это действия специалиста на рабочем месте. Здесь также перечисляются основные разделы знаний, необходимые для этой работы. Окончательный вариант профстандарта выпускается Министерством труда.

Конечно, университеты учитывают эти пожелания при составлении своих программ обучения. Но в университетах действуют образовательные стандарты, которые выпускает Министерство высшего образования. А уже потом к ним добавляются профстандарты.

С другой стороны, можно просто зайти на сайты с объявлениями о вакансиях. Например на HeadHunter. Здесь мы найдём объявления о текущих открытых вакансиях, где перечислены основные требования к соискателям.

Наконец, есть интересные мероприятия, которые обычно проводят при приёме на работу. И называется оно «собеседование». Жаргонное, простонародное название: «собес». Крупные ИТ-компании часто проводят так называемые «открытые собеседования» или «экспресс-собеседования», а также выставляют видеоролики с разбором собеседований. Есть масса видеороликов по подготовке к таким собеседованиям. Всё это находится в открытом доступе.

Исходя из перечисленных материалов, можно составить список необходимых знаний и навыков для конкретной профессии.

Задание. Выберите какую-нибудь компьютерную специальность, которая лично вам кажется интересной, привлекательный и многообещающей. Если у вас появляются затруднения с выбором специальности, возьмите любую, первую попавшуюся. Это упражнение, которое вас ни к чему не обязывает.

Чтобы определиться с будущей профессией, нам понадобится специалист. Вообще говоря, специалист

нужен в любом деле. Мы даже можем привлечь карьерного консультанта из любой обучающей платформы. Но и это ещё не всё. У нас есть интеллектуальные помощники, которым мы можем назначить любую роль и дать любое задание. Конечно, в рамках приличий и в пределах здравого смысла. Проверим, как наш чат-бот справится с такой задачей.

Задание. Сформулируйте подробный промпт для карьерной консультации. Назначьте чат-боту роль карьерного консультанта и дайте ему детальное описание задачи. Если появились какие-то сложности, попросите самого чат-бота сформулировать этот самый промпт и посмотрите, как он на него же и отреагирует – в рамках новой беседы. Пройдите эту «карьерную консультацию». В отчёте приведите только свой промпт и название профессии, которую вам порекомендовал чат-бот. Все подробности и ваше общение с этим консультантом оставим за кадром. Это ваше личное дело. Нас будет интересовать только название профессии, которую мы далее будем рассматривать и моделировать.

ДЕРЕВО РАСТЁТ ИЗ КОРНЯ

Предположим, что мы выбрали профессию системного аналитика – по своему собственному решению, или по рекомендации специалиста, или по совету чат-бота, или просто так – чтобы потренироваться. Это просто пример.

У вас будет свой вариант, своё название подходящей интересной профессии из мира информационных технологий (ИТ). Не относитесь к этому упражнению слишком серьёзно. Считайте, что это игра. Сделайте вид,

что вы действительно заинтересовались этой профессией. Потом всегда можно передумать.

Название профессии у нас теперь есть! И это будет начало нашего «дерева». Применительно к «деревьям» эта точка называется «корень дерева». И вот из этого корня у нас будет постепенно «растить» вся наша схема.

Итак, мы уже начали составлять нашу карту – наш первый Mind map.

Приступаем к рисованию. Схему будем строить тоже в духе «диаграммы как код» – как и в предыдущем разделе.

Переходим на сайт PlantUML:

<https://plantuml.com/>

Открываем страничку MindMap diagram:

<https://plantuml.com/mindmap-diagram>

В другой вкладке браузера открываем онлайн редактор – Online Server.

Вводим в окне редактора три строчки, как показано на рисунке и нажимаем кнопку Submit – Отправить.

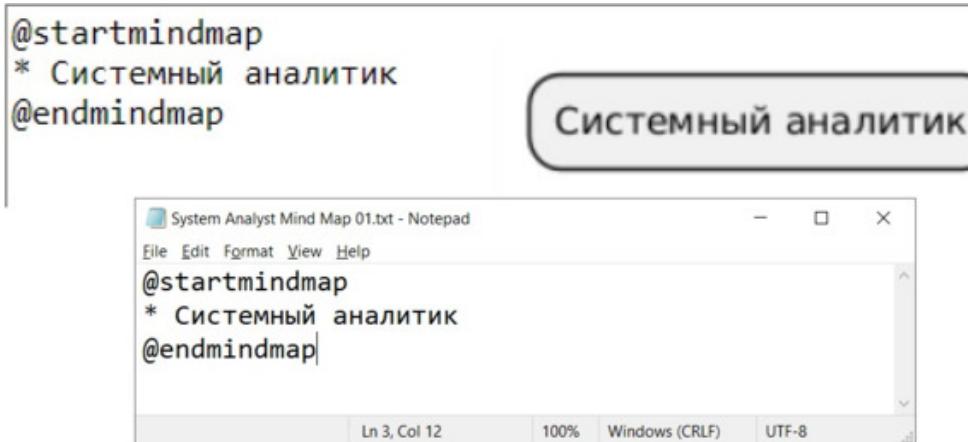


Рис. Начинаем строить дерево

Наша программа состоит из трёх строк. Символ @ в PlantUML – это начало команды. В нашем примере с помощью специальных команд мы обозначили начало и конец диаграммы, а именно ментальной карты. В новой строке мы вводим символ звёздочки, пробел и слова, которые будут внутри первого элемента схемы. Пока что всё достаточно просто и очевидно. Очень напоминает язык Markdown – в том смысле, что такой текст может легко прочитать человек.

Схема наша постепенно будет усложняться. Хотелось бы сохранить всю историю нашей работы. Поэтому мы начнём с простого действия. Выделяем текст программы [Ctrl + A], копируем нашу программу в буфер обмена [Ctrl + C], вставляем в новый текстовый файл в Блокноте и сохраняем на диске под простым названием. В названии файла будет название диаграммы и указание номера версии этого произведения. По номеру версии мы сможем отслеживать наши изменения. Конечно, это простое, наивное решение проблемы. Но на этом примере мы начинаем знакомиться с основными идеями и технологиями управления версиями – Version control. Подробно мы будем обсуждать эту технологию

в следующих работах, а пока нам нужно почувствовать на своем собственном опыте масштабы проблемы.

Задание. Просмотрите статью Version control на Википедии. Обратите внимание на графическое представление истории версий программного проекта (history graph) – просто в качестве примера.

ДОЛЖЕН, НО НЕ ОБЯЗАН

И так, мы выбрали профессию. Сделаем вид, что мы ничего об этой работе не знаем. Попробуем хотя бы что-нибудь об этом деле узнать. Поскольку это работа, профессия, занятие, деятельность... этот человек должен что-то ДЕЛАТЬ. Попробуем выяснить, чем же занимается наш специалист.

Спросим какого-нибудь более или менее интеллектуального чат-бота об этом. В ответ получим список основных действий нашего специалиста. Внесём вручную эти действия в нашу схему, в нашу ментальную карту. Новые пункты будут начинаться с двойных звёздочек. Это второй уровень, это «подразделы».

Сохраним текст программы в файл с указанием номера версии в названии. Теперь появляется вторая проблема. У нас будет много однотипных файлов с номерами по порядку. Но что именно мы в них вносили или меняли? Хорошо бы также как-то фиксировать наши изменения в виде комментариев. И мы можем вставлять комментарии прямо в текст программы, начиная их с одиночной кавычки (апострофа), см. рис.



Рис. Основные функции аналитика

Задание. С помощью интеллектуального помощника выясните, чем занимается специалист по выбранной вами профессии. Сформулируйте запрос так, чтобы получить список основных обязанностей в рамках этой профессии. Постройте вторую версию карты и сохраните программу в новом файле.

Некоторые строчки на нашей схеме получились слишком длинными. Хотелось бы сделать схему более компактной. И у нас есть возможность перенести остаток слов на новую строчку. Для этого используется комбинация символов: обратная наклонная черта (back slash) и латинская буква n – от выражения new line – «новая строка», или «переход на новую строчку», или «перевод строки». Такая комбинация символов ещё называется «управляющий символ». Этот инструмент используется во многих языках программирования при выводе текста на экран.



Рис. Улучшаем схему

В нашей схеме есть два блока, которые можно было бы объединить. На первом этапе системный аналитик собирает требования и проводит их анализ. Затем он описывает их в форме документа, то есть «документирует» требования. В принципе, вся эта деятельность может идти под девизом «управление требованиями» – Requirements Management. Так что мы объединяем два блока в один. Теперь схема стала немного проще и понятнее.

Чем проще такая схема, тем её легче воспринимать. То, что мы сделали – это объединили однотипные, похожие, родственные элементы. Можно сказать, что мы «укрупнили» близкие по смыслу пункты нашего списка. Теперь разные пункты относятся к совершенно разным действиям.

Конечно, нет предела совершенству. Здесь есть ещё два блока, которые можно было бы объединить. Дело в том, что разработкой и внедрением может заниматься одна и та же группа людей. Их обычно называют «разработчики» программного обеспечения или информационной системы. Поэтому общение с «разрабами» в процессе создания и внедрения информационной системы можно было бы считать

одним направлением деятельности. В больших программах проектах это действительно могут быть Две разных группы специалистов, Поэтому будет две разных обязанности системного аналитика.

Задание. Рассмотрите на свою схему и подумайте, как её можно упростить. Можно даже проконсультироваться с чат-ботом по этому поводу. Проверьте, как у вас будет работать перенос слов. Сохраните новую версию схемы с соответствующей нумерацией и комментариями.

Теперь попросим нашего чат-бота самого сформировать программу для рисования ментальной карты по его собственному предыдущему ответу. Тут мы результаты не гарантируем, но надежда есть. В крайнем случае, можно будет вручную внести мелкие поправки. У нас получилось пара уровней иерархии, см. рис.

Новые уровни начинаются с трёх звездочек. Пока что всё достаточно очевидно. Здесь под каждым пунктом с двумя звёздочками идёт список из нескольких подпунктов с тремя звёздочками.

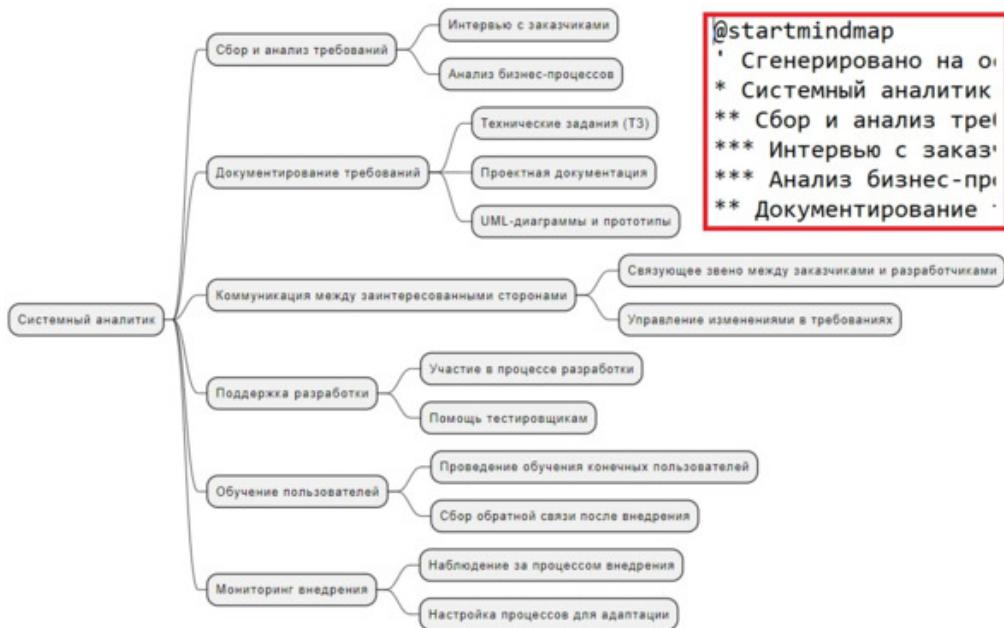


Рис. Карта, созданная чат-ботом

Задание. Попросите чат-бота сгенерировать код на основе его предыдущего ответа. Проверьте, как этот сценарий превратится в схему. При необходимости исправьте ошибки вручную. Добавьте комментарий и сохраните программу в новый файл.

ЗНАТЬ ИЛИ УМЕТЬ?

Теперь пришло время обратиться к более надёжным источникам информации. Как мы уже говорили, существуют более или менее объективные списки требований к ИТ специальностям — то есть к профессиям в мире информационных технологий. Это профессиональные стандарты, вопросы к собеседованию, объявление о вакансиях и программы онлайн курсов. Для многих профессий можно найти «Руководство к своду знаний» — Body of Knowledge Guide. Например, для программистов есть руководство к своду знаний по программной инженерии — SWEBOK Guide. В области информационных технологий есть

примерно десяток таких руководящих документов. На основе всех этих сведений мы можем составить список знаний и умений, которыми нужно овладеть чтобы стать хорошим специалистом в своей области. Ещё может встретиться иностранное слово «компетенции». Оно как раз и означает «знания, умения и навыки, необходимые для выполнения работы в рамках выбранной профессии». Или, как ещё говорят, «скиллы» – skills. Здесь речь идёт и о теоретических знаниях, и о практических навыках применения этих самых знаний и технологий.

Мы можем начать действовать с любой точки, с любого источника. Можно идти от общего к частному, или от частного к общему, или вообще собирать сведения хаотично, случайным образом, действовать «на ощупь». В любом случае, мы пытаемся построить систему из этих обрывочных сведений, то есть мы систематизируем информацию.

Возьмём, к примеру, профессиональный стандарт. В нашем примере для профессии системного аналитика можно найти такой стандарт. Он утверждён Министерством труда, и в этом документе перечисляются основные трудовые функции. Фактически, это основные функциональные обязанности такого специалиста – что ему придётся делать на рабочем месте. Для каждой такой функции в стандарте описаны выполняемые действия, а также перечислены необходимые знания и умения.

Построим очень упрощённую схему компетенций, то есть знаний и умений. Перечислим основные дисциплины, материалы и технологии, на которые нужно

будет обратить внимание при подготовке. Эти знания понадобятся для выполнения тех самых трудовых функций из профстандарта. В рамках стандарта для каждой функции указаны необходимые компетенции, поэтому они будут повторяться в тексте этого документа. Нам нужно просто обобщить эти сведения и хоть как-то их «структурировать», то есть представить в виде структуры. В нашем случае эта структура называется дерево, или иерархия. Формально говоря, такая модель называется ментальная карта – Mind Map.

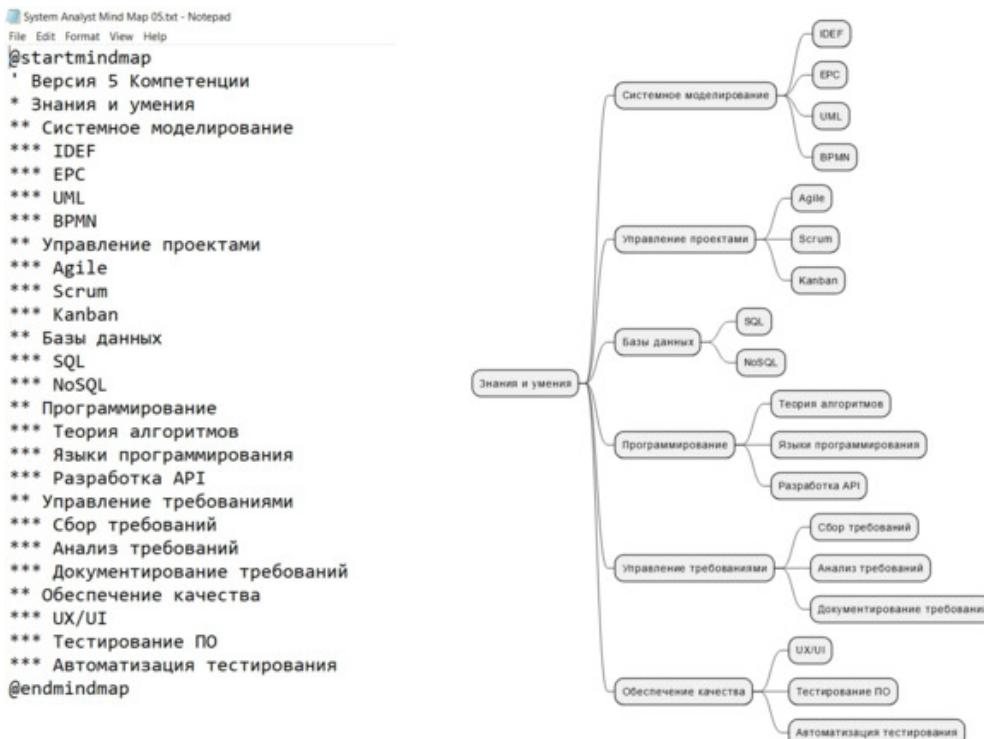


Рис. Схема знаний и умений

Задание. Найдите подходящий профессиональный стандарт для выбранной специальности. На основе этого документа составьте схему знаний и умений такого специалиста в форме списка учебных предметов (дисциплин) – что нужно будет изучить и освоить, чтобы подготовиться к такой работе. Можете использовать интеллектуального помощника, но потом такую схему

нужно будет вручную подчистить, подкорректировать, «допилить».

Наша схема постепенно увеличивается и обрастает подробностями и деталями. Нам предстоит добавить сюда сведения из других источников – это онлайн-курсы ведущих учебных платформ, программы собеседований при приёме на работу и требования к соискателям на вакантные места. Многие пункты будут повторяться. Это лишнее подтверждение того, что мы движемся в правильном направлении.

Задание. Найдите дополнительные сведения в перечисленных выше источниках для выбранной специальности. Добавьте ключевые знания и умения в свою схему. Сохраните новую версию.

ПОВЕРНИТЕ НАЛЕВО...

Сейчас у нас в руках две разных схемы. В начале мы построили дерево трудовых функций (должностных обязанностей). Затем мы построили второе дерево для знаний и умений, необходимых для выполнения этих самых функций.

Теперь попробуем объединить эти два дерева. Для этого в PlantUML есть так называемая «арифметическая нотация» – Arithmetic notation. Напомним, что нотация – это очередное английское слово notation, которое относится к системе условных обозначений. Минусы указывают на ветки, идущие влево. Плюсы дают нам ветки, идущие вправо. Пример объединённой схемы приводится на рисунке.

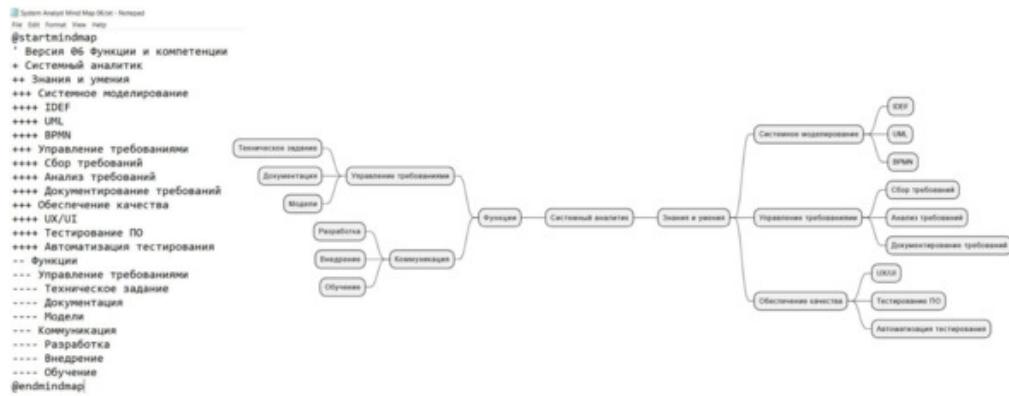


Рис. Общая схема

Задание. Объедините две части схемы в одну ментальную карту. Пусть в левой части будут показаны трудовые функции, а в правой необходимые знания и умения. Сохраните новую версию.

УКРАСИМ ДЕРЕВО ЦВЕТАМИ

На диаграммах PlantUML можно использовать разные цвета, шрифты и прочие украшательства. Здесь главное – не переборщить. Цвет позволяет выделить ключевые моменты и подчеркнуть общие свойства. Если блоки одного цвета, значит они чем-то похожи, у них должно быть что-то общее.

Попробуем упростить нашу схему. Это для тренировки. Оставим самые ключевые сведения и сопоставим рабочие функции с теми знаниями и умениями, которые нужны для выполнения этих функций. На рисунке мы оставил по три элемента слева и справа и раскрасили их разными цветами. Так мы подчеркиваем связь компетенций с функциями.

```

@startmindmap
' Версия 07 Цвета
+[#cyan] Системный\паналитик
++ Компетенции
+++[#lightpink] Моделирование
+++[#lightyellow] Требования
+++[#lightgreen] Качество
-- Функции
---[#lightpink] Коммуникация
---[#lightyellow] Тех. задание
---[#lightgreen] Сопровождение
@endmindmap

```



Рис. Цвета со смыслом

Теперь объясним смысл цветовой «кодировки». Использование схемы «красный-жёлтый-зелёный» – это только шуточный пример. Вы можете выбрать более приятные для глаза цвета и оттенки. Кстати, дальнейшие объяснения тоже служат только для демонстрации. Они очень приблизительные и нужны для иллюстрации того, как развивается наше понимание профессии. Настоящий системный аналитик увидит этот текст и, скорее всего, прослезится.

Красный цвет: коммуникация и моделирование. Коммуникация аналитика с заинтересованными сторонами, общение со стейкхолдерами требует систематизации и документирования полученных сведений. Для этого аналитик использует инструменты построения системных моделей. Таким образом, владение средствами моделирования поможет ему в эффективной коммуникации с заказчиком и членами команды разработки.

Жёлтый цвет: Техническое задание (ТЗ) и системные требования. ТЗ – это формальный документ. Иногда его

называют спецификация требований или задание на разработку. В любом случае, ТЗ содержит требования с будущей информационной системе (ИС). Сначала это требования предприятия, потом это технические требования для разработчиков ИС. Для составления ТЗ нужно знать, как работать с требованиями.

Зелёный цвет: Сопровождение процесса разработки и внедрения позволяет обеспечить качество программного продукта (информационной системы). Системный аналитик организует тестирование отдельных программных модулей и всего продукта в целом. Он участвует в проведении приёмо-сдаточных испытаний (ПСИ). В конечном счёте, это нужно для обеспечения желаемого уровня качества ИС. Будем считать, что сопровождение связано с обеспечением качества – Quality Assurance (QA).

Теперь по поводу цветов на схеме. Название цвета – это стандартные настройки, которые используются в самых разных программных продуктах.

Общее описание технологии названия и кодирования цветов можно почитать на странице проекта:

<https://plantuml.com/color>

Нам пригодится табличка из раздела View colors in PlantUML.

Задание. Упростите свою схему до нескольких ключевых функций и необходимых знаний. Используйте несколько цветов для того, чтобы указать на соответствие между функциями и компетенциями. Сохраните новую версию схемы.

Готовых цветов много, но это ещё не всё. При желании можно использовать стандартное, шестнадцатеричное представление цвета. Такое представление работает и в графических редакторах (типа GIMP или Photoshop), и в HTML коде веб-страниц. Любой цвет можно разложить по яркости трёх цветовых каналов RGB: Red-Green-Blue, то есть красный-зелёный-синий. Яркость кодируется двумя шестнадцатеричными разрядами, то есть числами от 00 до FF.

На рисунке приводится пример формирования цвета в бесплатном графическом редакторе GIMP.

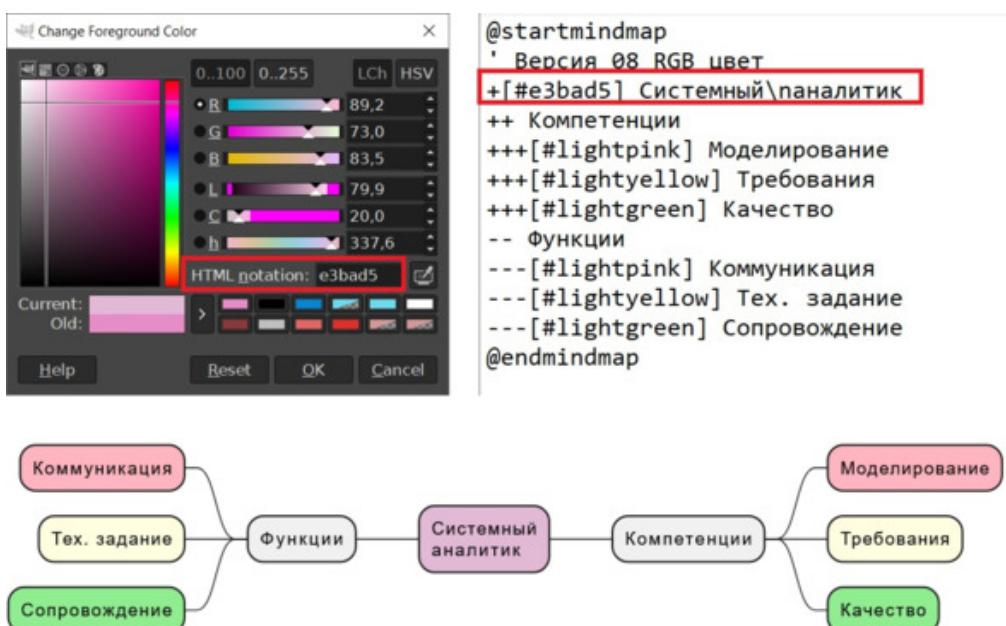


Рис. Нестандартный цвет из GIMP

Задание. Используйте нестандартные цвета на своей схеме для того, чтобы организовать не слишком яркие оттенки и неброские цвета. «Раскрашенная» схема не должна отвлекать внимание читателя от своего содержания, а помогать ему увидеть главное. Сохраните новую версию.

ОБОБЩЕНИЕ ИНФОРМАЦИИ ИЗ ФАЙЛОВ

До этого момента мы с вами вводили запрос к нейросети в виде текста – в окошке для запроса. В этом случае мы можем столкнуться с ограничением на количество слов в запросе (промпте). Многие чат-боты позволяют прикрепить файлы с дополнительной информацией к этому короткому запросу. в этом случае тогда нейросеть может обобщить сведения из достаточно длинного объёмного файла. Можно также задавать конкретные вопросы по содержанию документа. Обычно здесь тоже действует ограничение на количество и объём прилагаемых файлов. Тут всё зависит от конкретного инструмента и тарифного плана.

Возьмём для упражнения Профессиональный стандарт Минтруда России по специальности Системный аналитик. Вам предстоит использовать документы, описывающие вашу выбранную профессию.

В качестве первого примера рассмотрим сервис Perplexity. Сформулируем очень подробный запрос по поводу содержания прилагаемого файла. Используем правила составления / конструирования запросов (Prompt Engineering), с которыми мы познакомились в рамках первой работы. Укажем роль, контекст, требования и пожелания, оформление результатов. Чтобы прикрепить к запросу наш файл, нажимаем кнопку Attach – Прикрепить.

Во многих случаях мы можем использовать текстовые файлы формата ASCII / TXT и PDF.

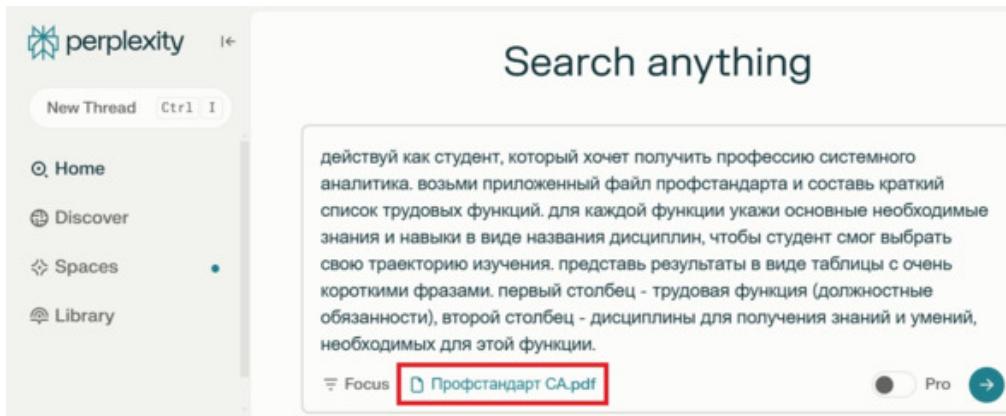


Рис. Прикрепляем файл к запросу

В результате обработки нашего запроса мы получаем табличку – в соответствии с нашими пожеланиями по оформлению. Обратим внимание, что наш файл указан в качестве единственного источника информации для этой нейросети.

Мы провели демонстрацию способности нейросети обобщать большие объёмы информации. Это очень полезная функция, и она может пригодиться при составлении обзоров и анализе больших документов.

Как и ранее, мы не можем просто слепо копировать эти ответы. Всегда требуется проверка человеком – «факт-чекинг». Всегда требуется здравый смысл, способность рассуждать и отличать важное от неважного, второстепенного.

За результаты в любом случае будет отвечать человек, пользователь, а не компьютер и не программы. Вычислительная техника и искусственный интеллект – это только инструменты, которые выполняют наши запросы и задания.

The screenshot shows the Perplexity platform interface. On the left is a sidebar with navigation links: New Thread, Home, Discover, Spaces, and Library. The main area displays a search result from user valentinarkov posted 2 minutes ago. The result is titled 'Sources' and lists a single file: 'Profstandart-SA.pdf'. Below this, there is a table titled 'Perplexity' with two rows of results. The first row has columns 'Трудовая функция' (Work function) and 'Дисциплины для получения знаний и умений' (Disciplines for obtaining knowledge and skills). The second row has columns 'Техническое сопровождение проектирования Системы' (Technical support for system design) and 'Информатика, Базы данных, Программирование' (Informatics, Databases, Programming). The third row has columns 'Сбор исходных данных для проектирования' (Collection of primary data for design) and 'Управление данными, Структурное программирование' (Data management, Structural programming).

Рис. Табличный формат вывода

Задание. Используйте большие документы, имеющие отношение к выбранной вами специальности (профстандарты, программы курсов, описания вакансий и т.п.). Прикрепите их к своим запросам – по отдельности и вместе. Опишите в отчёте полученные результаты. Используйте эти результаты при уточнении своих диаграмм.

В качестве второго примера прикрепления документов к запросу мы рассмотрим бесплатный отечественный продукт: ГигаЧат от Сбера. При анонимном использовании сервиса без регистрации мы получаем ограниченный функционал. После регистрации мы можем прикреплять файлы к нашим запросам.

На первом, стартовом экране сервиса упоминается возможность голосового ввода запросов. Доверяем вам самостоятельно ознакомиться с этой функцией.

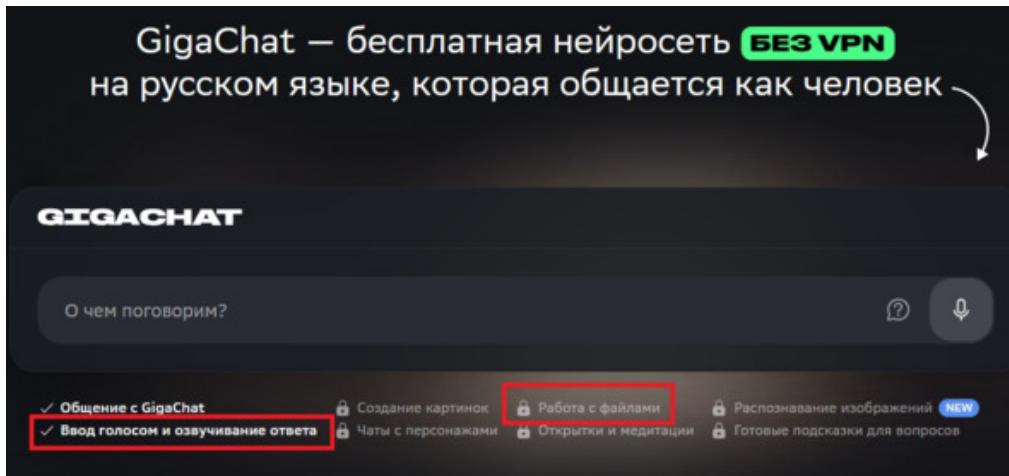


Рис. Возможности зарегистрированного пользователя

Мы будем использовать тот же самый исходный файл и ту же самую формулировку запроса к нейросети. Нажимаем кнопочку [+] Прикрепить и соглашаемся с условиями использования данного сервиса. В этот момент нас предупреждают о необходимости соблюдать действующее законодательство в отношении персональных данных и сведений, составляющих тайну.

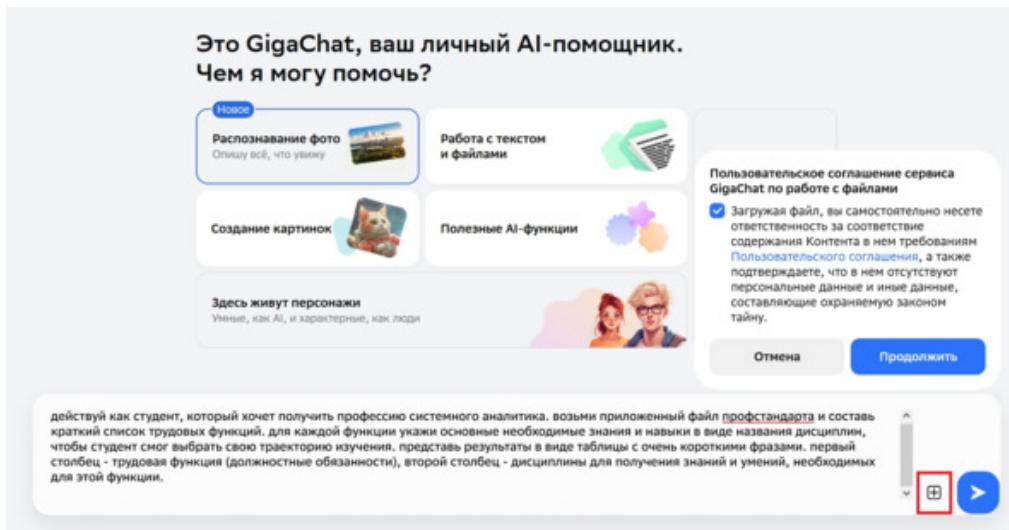


Рис. Прикрепляем файл к запросу

На рисунке ниже приводится фрагмент ответа нейросети на наш запрос. Здесь можно видеть основную таблицу, построенную по нашим требованиям. Далее идёт список основных предметов и направлений, по которым рекомендуется развиваться будущему специалисту.

действуй как студент, который хочет получить профессию системного аналитика. возьми приложенный файл профстандарта и составь краткий список трудовых функций. для каждой функции укажи основные необходимые знания и навыки в виде названия дисциплин, чтобы студент смог выбрать свою траекторию изучения. представь результаты в виде таблицы с очень короткими фразами. первый столбец - трудовая функция (должностные обязанности), второй столбец - дисциплины для получения знаний и умений, необходимых для этой функции.

Профстандарт СА.pdf

Трудовая функция (должностные обязанности)	Дисциплины для получения знаний и умений
Сбор исходных данных для проектирования Системы из установленных при обследовании текущей ситуации источников информации	Электронная почта, системы управления задачами, мессенджеры, работа с данными, SQL, основы программирования, основы работы с базами данных, формальная логика, русский язык, риторика, процесс коммуникации, методы совмещения контекста отправителя и получателя информации.
Разработка эксплуатационной документации по Системе в рамках надзора за реализацией требований к элементу поставки	Формальная логика, процесс коммуникации и контекст, русский язык и риторика, работа с текстовыми документами, системы управления знаниями, методы отладки и поиска ошибок, методы причинно-следственного анализа, управление инцидентами и проблемами, тестирование, написание технической документации.}
Эти функции требуют знаний и навыков в таких областях, как:	<ul style="list-style-type: none">◦ Электронная почта и системы управления задачами;◦ Работа с базами данных и SQL;◦ Основы программирования и структурного/объектно-ориентированного программирования;◦ Текстовое, графическое и табличное описание деятельности;

Рис. Ответ в виде таблицы и списка

Задание. Используйте те же документы и тот же запрос при обращении к ГигаЧату. Опишите результаты в отчёте. Дополните и уточните свои диаграммы – при необходимости.

ЯНДЕКС ВИКИ

Настало время познакомиться с ещё одной платформой, которая поддерживает Markdown, Mermaid и PlantUML. Этот инструмент называется Яндекс Вики. Здесь реализована технология Википедии в том смысле, что информационное наполнение сайта производится силами участников проекта. Это своеобразная «народная энциклопедия». В данном случае речь идёт об участниках проекта или о работниках предприятия. Здесь совместными усилиями можно создать «базу знаний» для сотрудников компании.

Задание. Откройте главную страницу проекта <https://wiki.yandex.ru/>. И знакомьтесь с основными возможностями этой платформы. Выясните, какова стоимость этой услуги.

Задание. Просмотрите статью под названием Вики на Википедии. Выясните происхождение этого странного названия. Это поможет понять основную идею данного проекта.

Для того, чтобы войти в сервис Яндекс Вики, нам понадобится учётная запись Яндекса. Например, нам подойдут логин и пароль от почты Яндекса. Мы входим в этот сервис и создаём страничку к предприятию. дальнейшей работе смогут подключиться сотрудники этого предприятия. Для начала мы поработаем с этим ресурсом самостоятельно.

На главной странице нажимаем кнопку Редактировать и переходим к редактированию страницы. В настройках изменяем автора на себя и удаляем из авторов робота.

Удаляем примеры оформления. Добавляем диаграмму Mermaid. Вставляем текстовое описание диаграммы (код) и нажимаем кнопку Save – Сохранить.

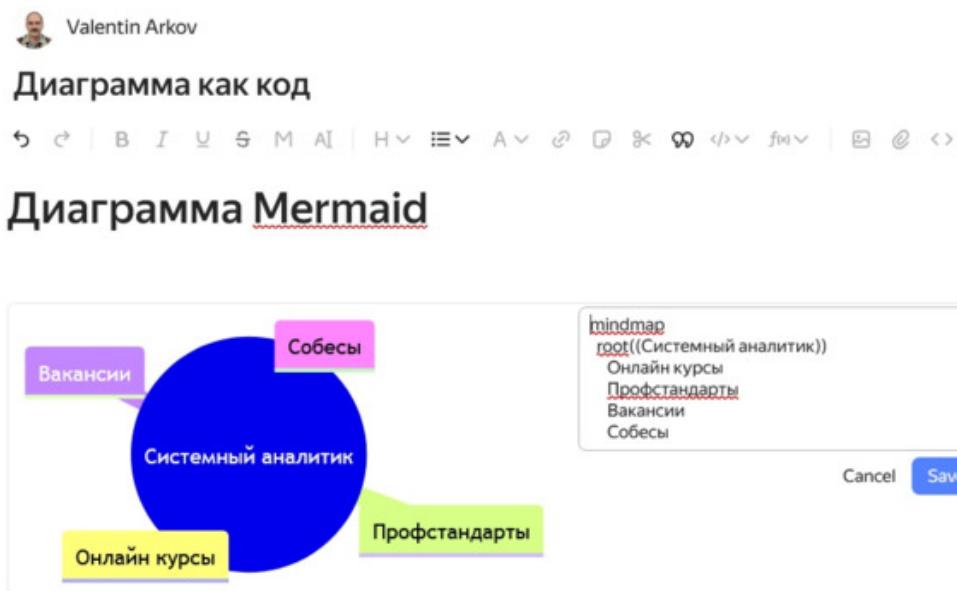


Рис. Первая диаграмма для Я. Вики

Задание. На первой странице своего проекта Яндекс Вики вставьте свою диаграмму из предыдущего раздела. Для выбора команды из меню можно использовать наклонную черту /.

Мы можем использовать два режима редактирования: визуальный редактор и разметка Markdown. В начале мы получили доступ к визуальному редактору. Далее в настройках мы можем выбрать режим редактирования, как показано на рисунке. Наша страничка будет состоять из блоков, или ячеек разного типа и содержания.

При выборе визуального режима мы видим загадочные буквы WYSIWYG.

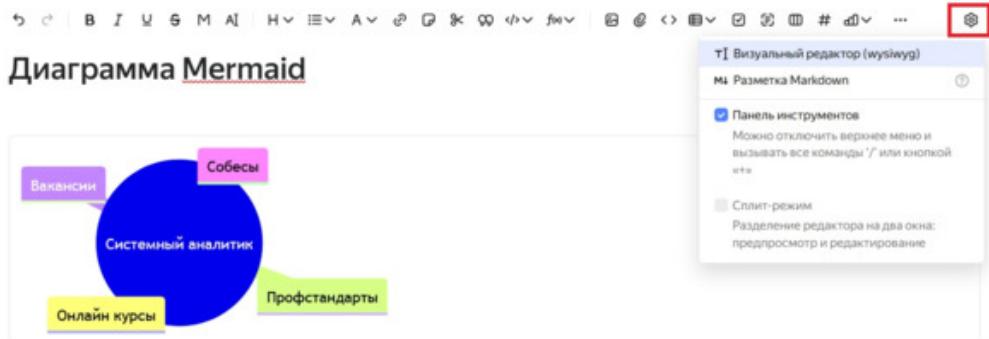


Рис. Режим редактирования

Задание. Выясните, что такое WYSIWYG, как это сокращение расшифровывается и что оно означает.

Задание. Переключите режим редактирования на разметку Markdown и ознакомьтесь с содержимым ячейки.

Совершенно аналогично мы можем вставить ячейку с диаграммой PlantUML. Для украшения документа нам доступны средства разметки Markdown. Как минимум, мы можем использовать заголовки разных уровней. Напомним, что заголовки начинаются с одного или нескольких символов решётки #.

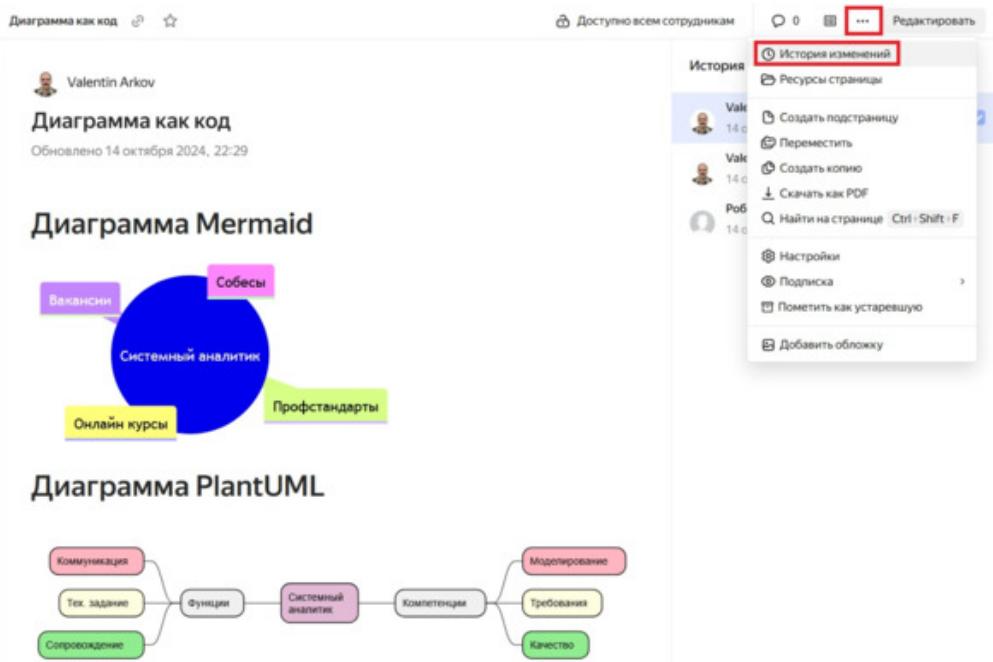


Рис. Вторая диаграмма для Я. Вики

Задание. Создайте новый заголовок. Вставьте последнюю версию диаграммы PlantUML. Сохраните изменения ячейки. Сохраните изменения страницы (кнопка справа вверху).

Как выяснилось при ближайшем рассмотрении, Яндекс Вики поддерживает даже вставку диаграмм Draw.io. Здесь в рамках новой ячейки документа мы получаем возможность работать со всеми инструментами а рисование диаграмм. Щёлкаем по свободному месту документа, нажимаем клавишу /. В выпадающем меню выбираем соответствующий пункт, см. рис.

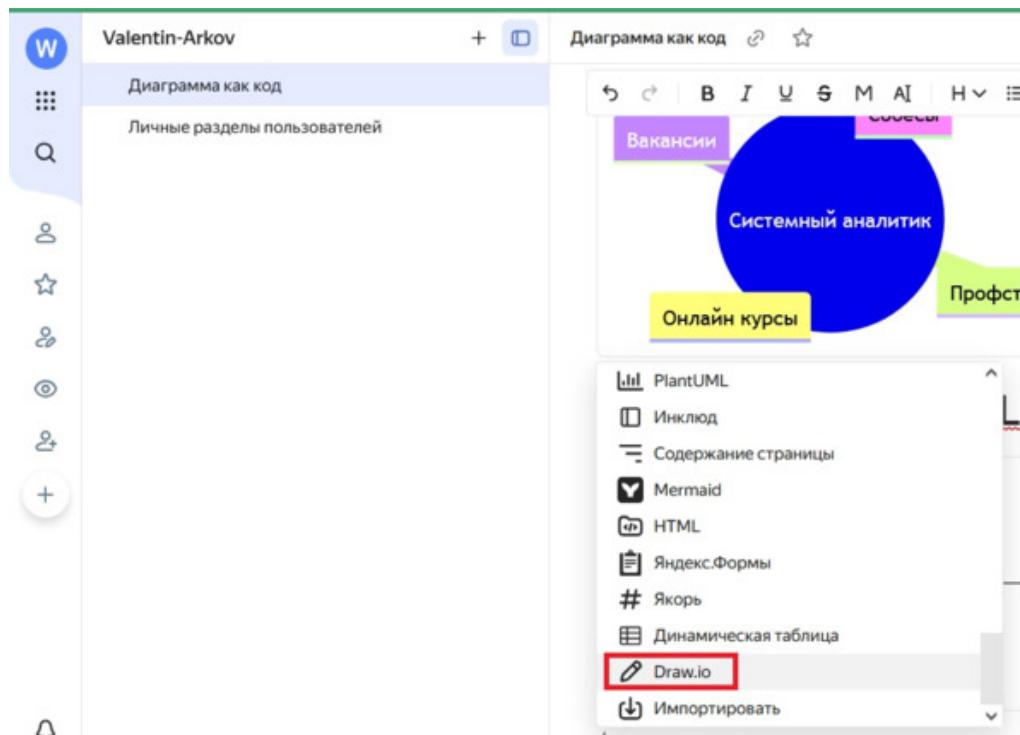


Рис. Выбираем тип новой ячейки

В левом верхнем углу новой ячейки мы можем вызвать меню настройки. Нас будет интересовать панель инструментов для выбора стандартных блоков Shapes, см. рис.

Ну а среди этих групп блоков нам понадобятся блоки общего назначения General и средства создания схемы алгоритма Flowchart.

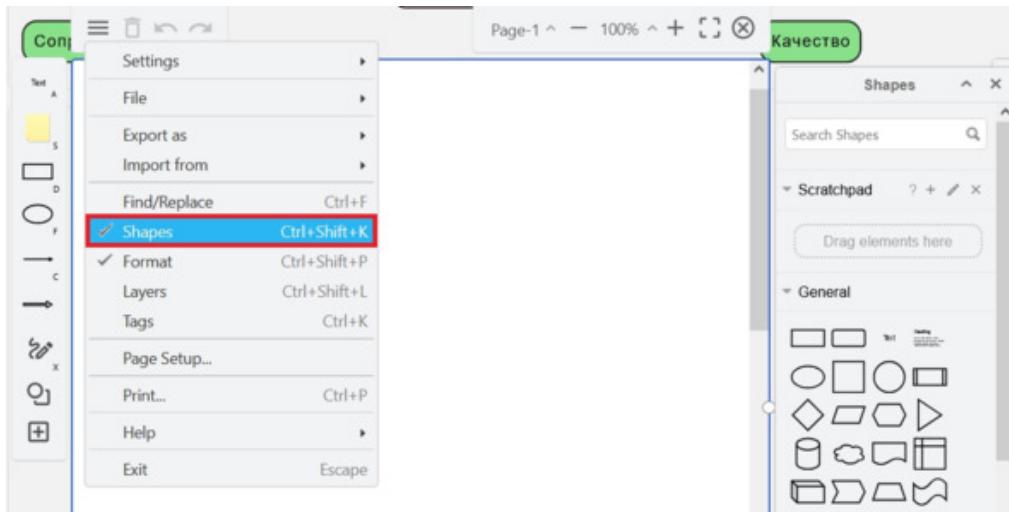


Рис. Включаем панель инструментов

Общая схема работы должна быть уже знакома нашим участникам. Мы выбираем блоки из палитры панели инструментов, перетаскиваем блоки, вводим текст на три блока. Соединяем блоки стрелками.

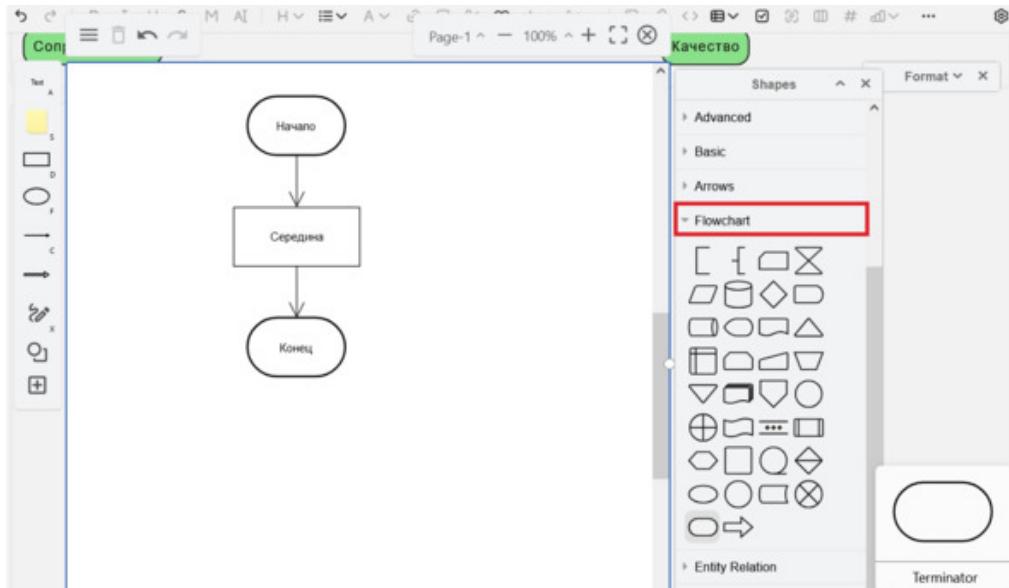


Рис. Рисуем схему алгоритма

Завершив создание схемы алгоритма, мы нажимаем кнопку сохранить точка. После этого мы переходим в режим просмотра документа и можем насладиться

результатами нашей работы.

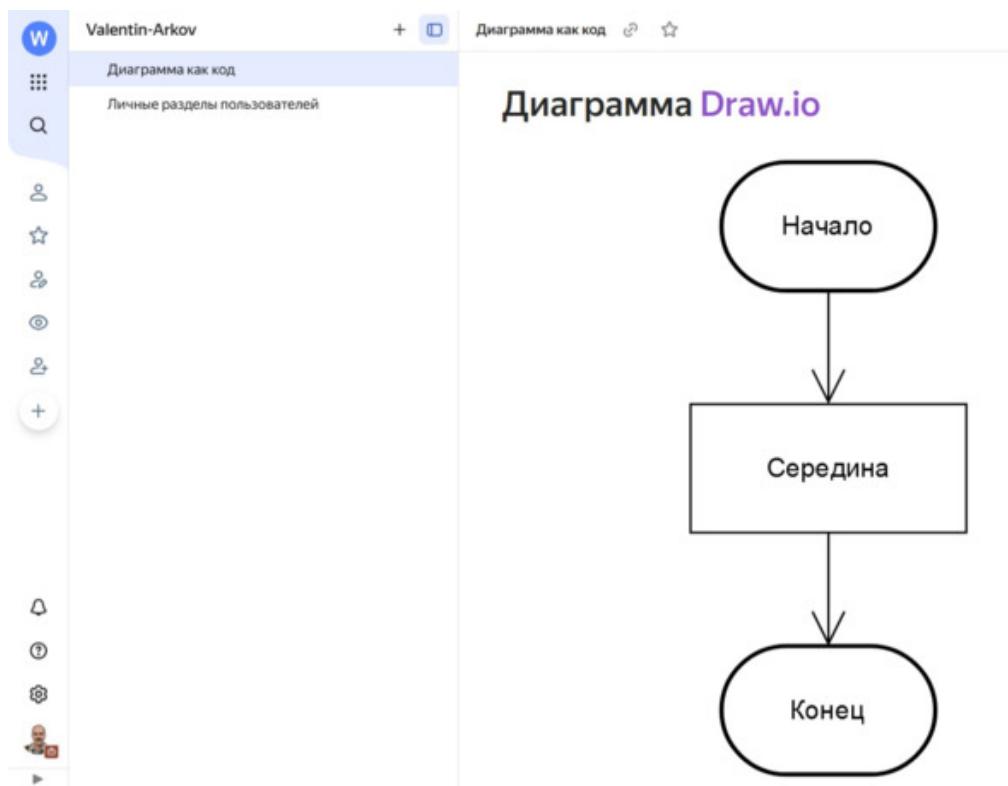


Рис. Диаграмма в составе документа

Задание. Создайте новую ячейку в Яндекс Вики и выберите тип содержимого как диаграмму Draw.io. Нарисуйте несложный алгоритм и сохраните результаты. Ознакомьтесь в режиме просмотра с представлением диаграммы на экране.

После сохранения изменений страницы мы можем просмотреть историю изменений. Это уже элемент технологии управления версиями. В правом верхнем углу окна находим кнопку с многоточием [...]. Выбираем в меню раздел История изменений.

Задание. Просмотрите историю изменений своей страницы. Попробуйте переключиться на предыдущую

сохранённую версию страницы. Вернитесь к последней версии страницы.

Напомним, что весь ход работы должен найти своё отражение в отчёте, который нужно вести на гитхабе. Всё, что мы делаем на других платформах, тоже нужно отразить в отчёте. Для этого можно использовать снимки экрана (скриншоты). Желательно давать комментарии и пояснения.

PLANTUML + VS CODE

Средства разработки программного обеспечения также поддерживают нотацию PlantUML. Для Visual Studio Code можно установить расширение PlantUML от автора qjebbs.

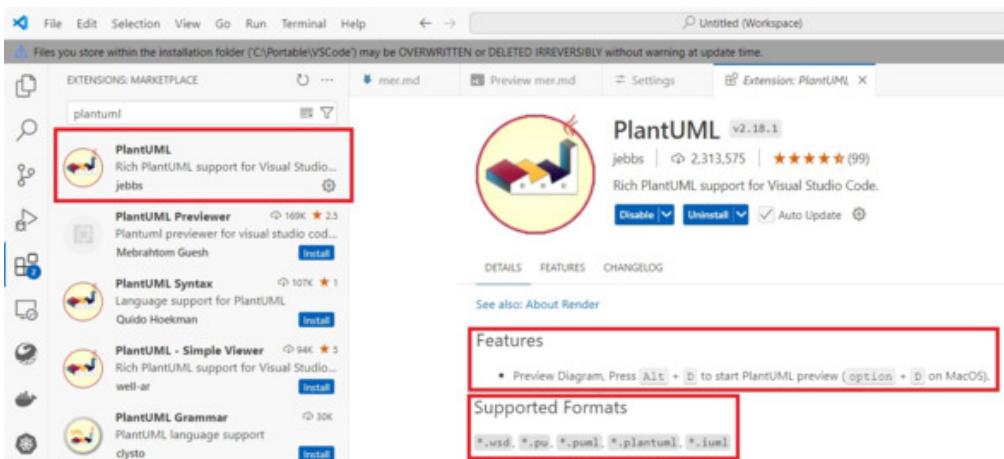


Рис. Установка расширения PlantUML

Задание. Установите расширение VS Code для работы с диаграммами plantUML. Просмотрите описание этого инструмента. Обратите внимание на поддерживаемые форматы файлов, а также на возможность просмотра диаграмм.

Переходим в Explorer и создаем новый файл. Назовём его map.ru.

Вставляем текстовое описание нашей диаграммы и вызываем просмотр PlantUML Preview, нажав комбинацию клавиш [Alt + D].

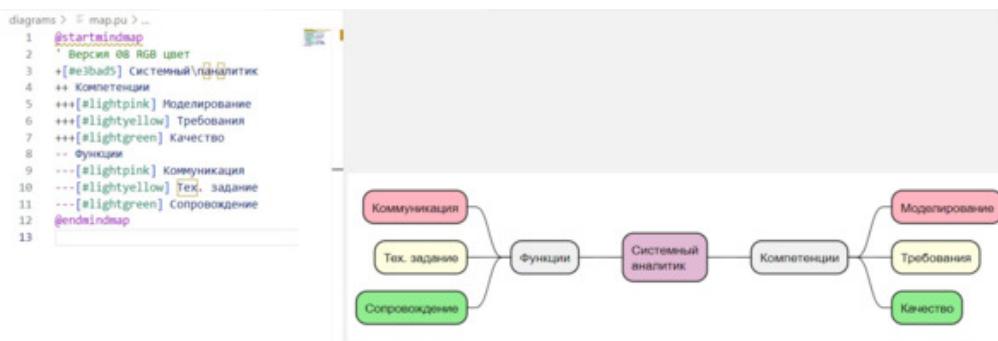


Рис. Предварительный просмотр

Задание. Создайте новый файл. Вставьте в него свою ментальную карту. Запустите предварительный просмотр диаграммы.

PLANTUML + DOCKER

В качестве прощального упражнения запустим онлайн редактор диаграмма с помощью контейнера. Технология работы очень похожа на то, как мы запускали Mermaid. Так что повторяться не будем.

Переходим на Docker Hub и в строке поиска вводим строчку: plantuml-server.

Выбираем самый популярный образ. Как видим его загрузили более 10 миллионов раз. Напомним, что в компьютерных технологиях большая, заглавная буква «M» намекает на приставку «Мега» и означает «примерно 1 млн». Значок плюсик после числа может

означать «более, чем». Вместе получается, что запись 10M+ означает «больше 10 млн».

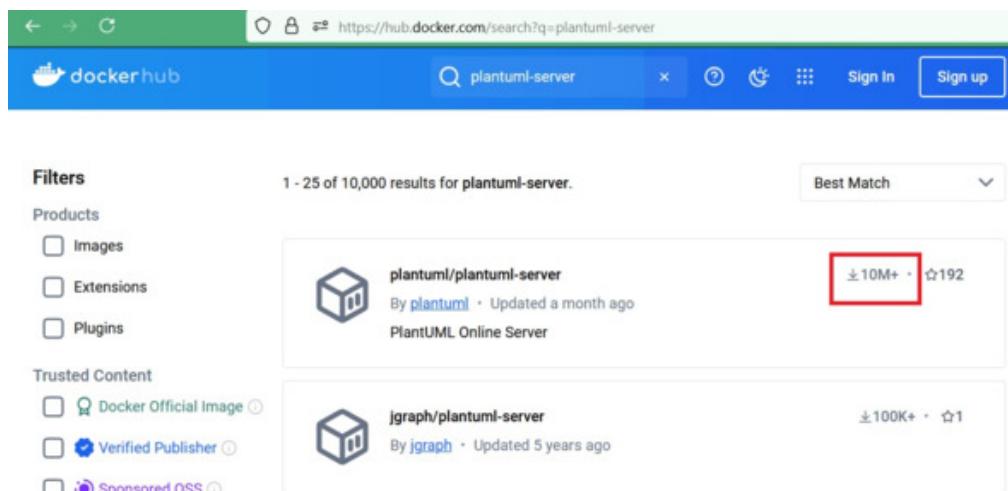


Рис. Находим образ сервера

ОбратиM внимание, что в данном случае автором образа выступает пользователь под ником plantuml. Переходим по этой ссылке и изучаем страничку с описанием данного образа. Описание здесь довольно скромное, без особых подробностей. Нас будет интересовать команда для запуска в разделе How to run the server with Docker. В данном примере нам предлагают сразу две команды. Копируем первую строку в буфер обмена.

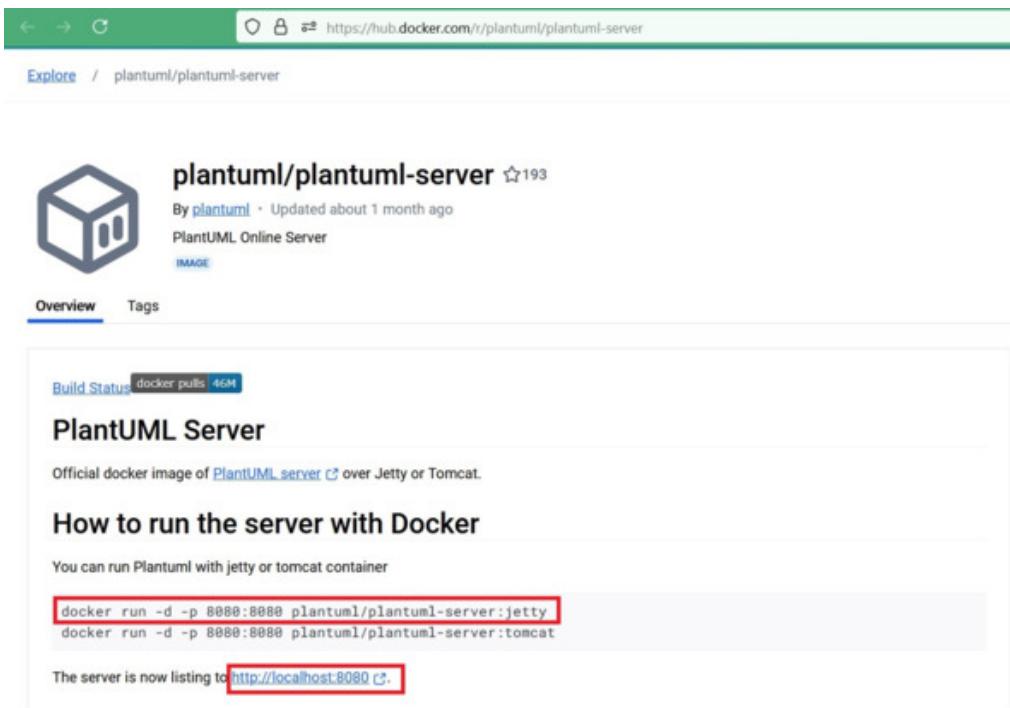


Рис. Команда запуска

Эти команды отличаются последним параметром: `jetty` и `tomcat`. Вам предстоит выяснить, что это такое. И вообще, пора разобрать эти длинные команды с кучей параметров «по косточкам». В этом нам поможет искусственный интеллект.

Задание. Попросите интеллектуального чат-бота объяснить каждый параметр обеих команд. При необходимости задавайте ему дополнительные вопросы. Это очень полезная функция современных чат-ботов. Мы можем вставить в запрос интересующую нас команду или даже целую программу. Затем мы просим его подробно объяснить каждый параметр и даже каждый символ. Раньше приходилось использовать поисковые системы и искать ответы на форумах. Сейчас всё гораздо проще. Но, к сожалению, к этим инструментам требуется добавить мотивацию и желание учиться. А этого чат-бот обеспечить пока не в состоянии.

На странице с описанием образа также даётся пояснение, как обратиться к серверу после запуска контейнера. Здесь говорится, что после выполнения команды будет запущен сервер, и он будет «слушать» наши обращения по адресу:

<http://localhost:8080>.

Чтобы наши серверы не мешали друг другу, мы изменим номер порта. При запуске сервера PlantUML мы укажем порт 8088. Это нужно сделать и в команде `docker run`, и в адресе URL для браузера.

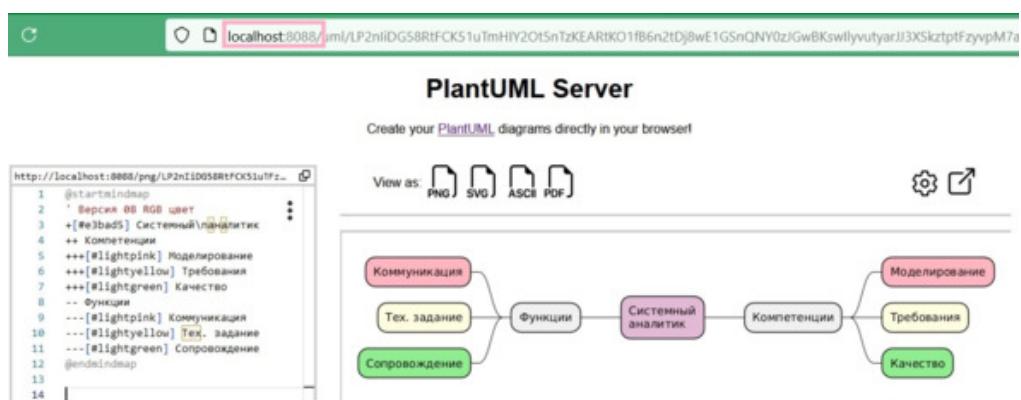


Рис. Локальный сервер PlantUML

Задание. Запустите контейнер с сервером PlantUML. Вставьте код своей диаграммы. ознакомьтесь с тем, как ваш локальный сервер генерирует вашу схему. Внесите в схему небольшое изменение, например, измените цвет одного из блоков. Обратите внимание на изменение диаграммы. Поэкспериментируйте с вариантами просмотра диаграммы `View as` и настройками сервера `Settings`.

ЗАКЛЮЧЕНИЕ

В этой работе мы познакомились с одной из популярных нынче технологий работы с диаграммами под названием Diagram as code – «диаграмма как код». Можно сказать, что это развитие технологии облегчённой, упрощённой разметки текста Markdown.

Оба инструмента – и Mermaid, и PlantUML поддерживаются и интегрируются самыми разными платформами разработки и документирования.

Попутно мы увидели самые разные подходы и инструменты для запуска программ, в том числе переносимые приложения Portable Applications и контейнеры Docker.

Кроме того, мы на своём опыте почувствовали необходимость организации хранения разных версий нашей диаграммы. У нас ещё будет отдельное занятие на тему управления версиями.

ССЫЛКИ

<https://en.wikipedia.org/wiki/Org-mode>

https://en.wikipedia.org/wiki/Literate_programming

Computational notebook

https://en.wikipedia.org/wiki/Notebook_interface

https://en.wikipedia.org/wiki/Reproducibility#Reproducible_research

https://en.wikipedia.org/wiki/Mind_map

<https://plantuml.com/ru-dark/mindmap-diagram>

[Mermaid]

<http://mermaid.js.org/>

Yandex Wiki. Бесплатная база знаний для сотрудников компании.

<https://wiki.yandex.ru/>

[Wiki: Version control] Version control

https://en.wikipedia.org/wiki/Version_control

[Вики: Система управления версиями] Система управления версиями

https://ru.wikipedia.org/wiki/Система_управления_версиями

[Профстандарт Системный аналитик]
Профессиональный стандарт 06.022 «Системный аналитик». Утвержден приказом Министерства труда и социальной защиты Российской Федерации № 367н от 27 апреля 2023 года.

<https://fgosvo.ru/uploadfiles/profstandart/06.022.pdf>

[Colab: Markdown] Markdown Guide

https://colab.research.google.com/notebooks/markdown_guide.ipynb

[Арьков Контейнеризация]

Арьков В. Ю. Виртуализация и контейнеризация

https://ridero.ru/books/virtualizaciya_i_konteinerizaciya

a/

4 АЛГОРИТМЫ

ВВЕДЕНИЕ

Программисту хочется писать программу и не хочется делать что-то ещё. Ведь конечный результат его работы – это именно программа. Но тут появляется одна проблема: сначала надо подумать, а уже потом действовать. Когда мы решаем, что и в каком порядке нужно делать, мы сталкиваемся с алгоритмом. Любая программа – это реализация алгоритма.

В этом разделе мы познакомимся с примерами того, как не надо строить алгоритмы и как не надо писать программу. Потом мы разберём примеры того, как можно улучшить программу. Оказывается, для этого есть много готовых инструментов, потому что программисты наступали на эти «грабли» очень много раз – задолго до нас.

А ещё мы узнаем, что есть целый предмет под названием «Алгоритмы и структуры данных». Другое название: «Теория алгоритмов». Эта область знания находится где-то между программированием и математикой. В программировании действуют законы природы, математика изучает и описывает эти законы природы, а программистам приходится изучать математику. Причём изучать её приходится на своём опыте и на своих собственных ошибках. Конечно, лучше и дешевле было бы учиться на чужих ошибках, но это мало кому подходит.

ОТЧЁТ

В этой работе мы будем оформлять отчёт в виде электронной таблицы с несколькими страницами. В программах типа электронных таблиц (Spreadsheet) весь файл целиком называется «Рабочая книга» – Workbook. Отдельные страницы, которые открываются на разных вкладках, – это «Рабочие листы» – Worksheets.

Электронные таблицы можно создавать и редактировать на локальном компьютере. Это привычный, традиционный подход. Кроме этого, у нас есть возможность работать в облаке. Мы будем создавать облачный отчёт и отправлять ссылку на него в конце работы. Попутно мы с вами знакомимся с ещё одной современной технологией, когда вся работа происходит в облаке.

Задание. Откройте облачный диск Яндекс. Для этого понадобится учётная запись Яндекс. Создайте новую таблицу. При создании таблицы сразу же укажите информативное название своего отчёта. Ваш документ должен будет отличаться от нескольких десятков таких же отчетов.

Как и ранее, отчёт начинается титульного листа. На нём должны быть представлены все необходимые сведения. В предыдущих разделах мы уже обсуждали требования к оформлению титульного листа.

Задание. Выясните, что означает слово «титульный» в выражении «титульный лист», а также для чего вообще служит титульный лист в книгах, отчётах и документах.

Мы с вами будем работать в облачном офисном пакете Яндекса. Здесь есть одна особенность. Если мы внесли в документ изменения, их нужно будет сохранить вручную. В других программных продуктах сохранение может выполняться автоматически. Но здесь — только вручную! Для этого есть кнопка «Сохранить» и горячие клавиши, как показано на рисунке. После успешной записи в файл (на сервере) внизу окна браузера, в строке состояния появляется сообщение об успешном сохранении изменений.

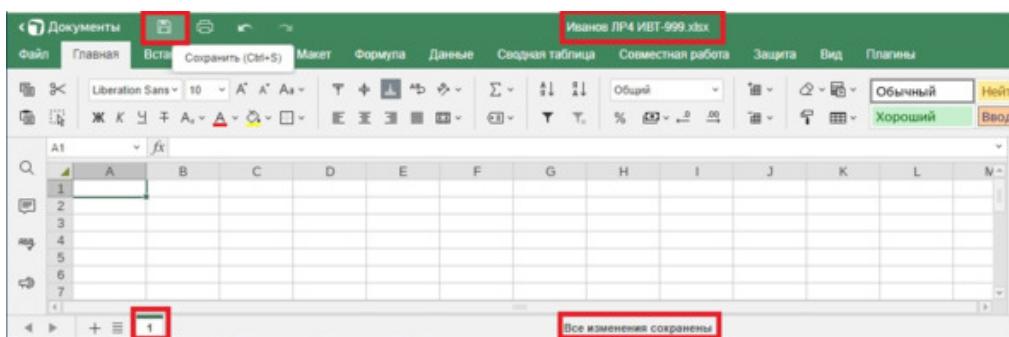


Рис. Титульный лист

Задание. Оформите титульный лист и укажите в нём все необходимые сведения о своём отчёте. Переименуйте вкладку так, чтобы указать только номер страницы. Такой приём поможет нам, когда в отчёте наберётся с десяток страниц. Сохраните изменения в файле. Для этого нужно нажать кнопку Сохранить либо комбинацию клавиш [Ctrl + S].

Вторая страница отчёта, особенно если в нём много страниц, — это содержание. Его также называют оглавление. В интернете можно даже встретить обсуждение: в чём разница и как грамотно назвать эту

страницу отчёта. Хорошо бы нам сразу же и разобраться с этими названиями.

Задание. Выясните, что такое «содержание» и что такое «оглавление». Есть ли между ними разница? Можно ли найти документ типа стандарта, в котором это разъясняется?

Задание. Создайте третью вкладку и озаглавьте её «Основные термины и определения». Переименуйте название вкладки так, чтобы это был номер страницы. По мере выполнения работы вносите новые термины и значения на эту страницу. Кстати говоря, мы уже познакомились с несколькими терминами в предыдущих заданиях. Внесите эти сведения в свой отчёт.

Итак, на второй странице отчёта будут названия разделов и номера страниц. В электронных документах в содержании можно разместить не только названия, но и ссылки. Внешне это напоминает гиперссылки на веб-страницах.

Мы наводим курсор на нужную ячейку и нажимаем правую кнопку мыши. В выпадающем, контекстном меню выбираем Гиперссылка – Параметры гиперссылки – Внутренний диапазон данных. Затем указываем нужную страницу нашего отчета – Связать с... и нажимаем ОК.

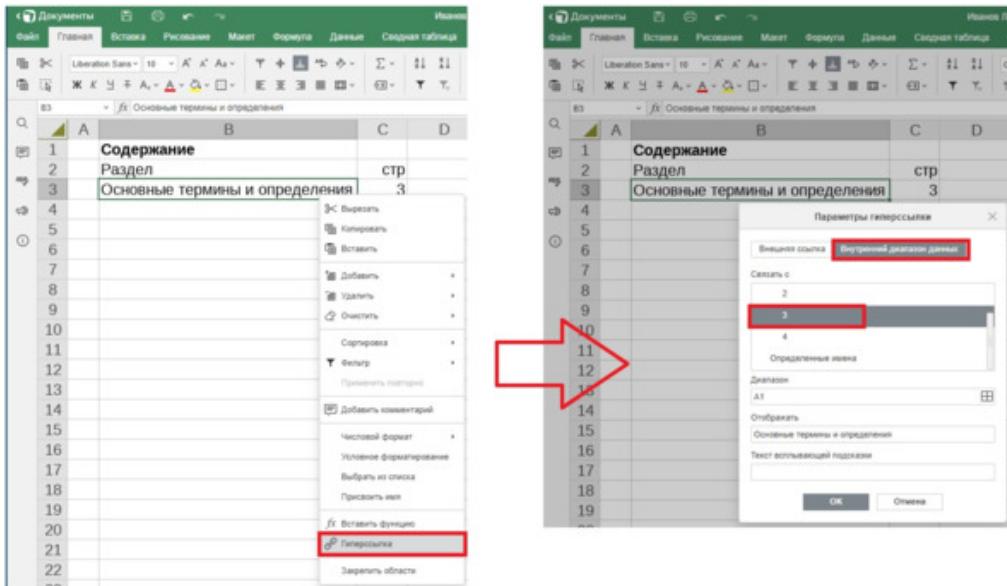


Рис. Ссылка на страницу

Задание. Настройте ссылку в содержании отчета на третью страницу. Убедитесь, что ссылка работает.

Для удобства навигации по тексту мы можем добавить ссылку на содержание в начале каждой страницы. Теперь мы можем от содержания перейти к нужной странице и можем вернуться к содержанию для выбора любой другой страницы.

Задание. Вставьте ссылку для возврата к содержанию отчёта в начале третьей страницы. Убедитесь, что ссылка работает. Сохраните изменения.

Теперь Мы готовы продолжать оформлять отчёт по мере выполнения работы. в конце работы нужно будет отправить ссылку на этот документ с возможностью чтения другим человеком.

ЧИСЛА ФИБОНАЧЧИ

В этом разделе мы будем заниматься нахождением числа Фибоначчи $F(n)$ под номером n . Эти числа используют как вспомогательный инструмент при решении самых разных задач.

Задание. Выясните, как строится последовательность чисел Фибоначчи, а также узнайте, что такое Фибоначчи – это имя, фамилия или прозвище.

Задание. Узнайте, для решения каких задач используют числа Фибоначчи – в каких областях деятельности, с какой целью.

НАИВНЫЙ АЛГОРИТМ

В статье Википедии мы находим основное соотношение для нахождения числа Фибоначчи: сумма двух предыдущих чисел ряда.

$$F(n) = F(n - 1) + F(n - 2)$$

В самом начале этой последовательности задана два числа: нулевое равно нулю, а первое равно единице.

$$F(0) = 0$$

$$F(1) = 1$$

Здесь нумерация элементов ряда начинается с нуля. Такой подход реализован во многих языках программирования, в том числе, в Python.

Программы можно запускать и на локальном компьютере, и в облаке. Результаты будут очень похожи. Мы имеем в виду не результат вычислений $F(n)$, а внутренние свойства каждого алгоритма. В процессе

выполнения работы вы это сможете прочувствовать и ощутить на собственном опыте. А также измерить объективно.

Наша первая программа будет реализовывать наивный алгоритм вычисления. Мы берём известные соотношения. Напишем простую программу, которая выполняет именно эти вычисления.

Оформим нашу программу виде функции. Затем мы сможем вызывать эту функцию и проведём с ней разные опыты, эксперименты.

Начнём работать в облачной среде Google Colab. Нам понадобится учётная запись Google. Создаем новый блокнот jupyter Notebook. Указываем краткое информативное название и сохраняем его. В дальнейшем нам нужно будет вставить в наш отчёт ссылку на этот блокнот с разрешением на чтение.

В нашем блокноте будет общее название, а также названия разделов – разных алгоритмов. Создаем подраздел для наивного алгоритма и вставляем туда расчётные формулы. Используем нотацию TeX и позаимствуем формулы со страницы Википедии. Напомним, что в рамках наших работ мы практикуем одновременное, параллельное составление программы и оформление документации.

+ Code + Text

Comment Share

All changes saved

RAM Disk

Числа Фибоначчи

Наивный алгоритм

Расчёты соотношения

$n = 0 : F_0 = 0$

$n = 1 : F_1 = 1$

$n > 1 : F_n = F_{n-1} + F_{n-2}$

Рис. Оформление программы

Как и ранее, наши комментарии и формулы мы оформляем в рамках текстовых ячеек, формате Markdown.

Задание. Создайте новый блокнот и оформите заголовки. В новой ячейке введите формулы для расчётов в нотации TeX.

Теперь переходим к составлению программы. Сама программа очень простая, и она выполняет именно то, что говорится в начале статьи Википедии про последовательность чисел Фибоначчи.

```
def f(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return f(n - 1) + f(n - 2)
```

f(0), f(1), f(2), f(3), f(4), f(5)💡

(0, 1, 1, 2, 3, 5)

Рис. Реализация наивного алгоритма

Обсудим текст нашей программы. Мы начинаем описание нашей функции $f(n)$ и сообщаем, что наша функция принимает для расчётов один аргумент. Мы предполагаем, что это целое неотрицательное число, то есть 0, 1, 2 и так далее.

Для простоты эксперимента мы не вводим дополнительные проверки. При написании настоящей программы нам понадобится проверить, что n – это целое и что это число неотрицательное. При нарушении этих условий нужно будет выдать соответствующее сообщение на экран и вернуть логическое значение `False` либо пустое значение `None` – в соответствии с общепринятой практикой конкретного предприятия.

Внутри функции мы проверяем значение аргумента. При n , равном нулю, мы возвращаем ноль. При n , равном единице, мы возвращаем единицу. Во всех остальных случаях мы возвращаем новое значение как сумму двух предыдущих чисел ряда.

Это означает, что наша функция использует рекурсию. Слово «рекурсия» в программировании означает, что функция вызывает сама себя. Такой прием позволяет получать достаточно краткие и легко читаемые тексты программ.

Однако, если программу легко читать человеку, это вовсе не означает, что эта программа будет хорошо работать на компьютере. И в этом мы скоро убедимся.

После составления программы нам нужно убедиться, что она правильно работает. Для этого у программистов выполняется этап тестирования. В простейшем случае мы подаем на вход нашей функции несколько значений n и проверяем, что полученные на выходе числа соответствуют нашим ожиданиям. Первые несколько чисел ряда мы можем легко найти вручную. В нашем примере мы не используем функцию `print()`, а просто вызываем функцию `f(n)` через запятую с разными значениями аргумента n . Мы можем произвести такие опыты в Colab, но при запуске программы на локальном компьютере желательно будет использовать команду `print()`.

Итак, мы протестировали нашу программу, и она работает правильно. Она выдает правильные значения на выходе. Теперь проведём вычисления и найдём несколько первых чисел ряда с помощью цикла `for`. Мы находим первые 10 чисел, первые 20 чисел, первые 30 чисел, первые 40 чисел. В каждом случае цикл начинается с нуля.

```
[4]: for n in range(10):
    print(n, f(n))
0 0
1 1
2 1
3 2
4 3

[5]: for n in range(20):
    print(n, f(n))
0 0
1 1
2 1
3 2
4 3
5 5

[6]: for n in range(30):
    print(n, f(n))
0 0
1 1

[7]: for n in range(40):
    print(n, f(n))
... 0 0
1 1
2 1
3 2
4 3
5 5
6 8
7 13
8 21
9 34
10 55
```

Рис. Время выполнения

Каждый опыт мы проводим в отдельной кодовой ячейке. После выполнения ячейки слева от неё выводится отметка о выполнении в виде зелёной галочки, а под ней указано время расчётов в секундах.

Первые три опыта проходят быстро, причём 10 и 20 чисел определяются менее, чем за секунду. Первые 30 чисел потребовали больше 1 секунды. А вот расчёт 40 чисел занял гораздо больше времени. Мы можем даже и не дождаться результата. Больше одной минуты уже нет смысла ждать. Обратим внимание, что во время выполнения последнего опыта в нижней части экрана выводится общее время вычислений Executing (... s), а также показана цепочка вызовов функции $f(n)$. Можно видеть длинную цепочку рекурсивных вызовов. К этой ситуации мы ещё вернёмся.

Задание. Составьте программу в соответствии с наивным алгоритмом и запустите её для 10, 20, 30 и 40 первых чисел Фибоначчи. Обратите внимание на длительность работы цикла и сообщения внизу окна

браузера, в строке состояния. Подберите величину n , при которой вычисления $f(n)$ будут занимать несколько секунд.

ВРЕМЯ ВЫПОЛНЕНИЯ

Мы с вами наблюдали подозрительно долгое время выполнения нашей программы. Нахождение первых 40 чисел в нашем примере было слишком долгим.

Попробуем измерить время вычислений. Для этого нам понадобится функция `time()` из библиотеки `time`. Это стандартная, встроенная библиотека Питона, нам потребуется её подключить командой `import`.

Затем мы указываем название библиотеки и через точку пишем название функции: `time.time()`. Таким способом мы можем получить текущее значение времени в секундах. Чтобы измерить время вычислений, мы зафиксируем текущее время до и после вызова нашей функции. Разность этих значений даст нам время выполнения программы в секундах. Теперь мы можем подобрать размер нашей задачи n так, чтобы время работы программы $f(n)$ было в пределах десяти-двадцати секунд.

<pre>import time n = 30 t1 = time.time() x = f(n) t2 = time.time() t2 - t1</pre>	<pre>import time n = 35 t1 = time.time() x = f(n) t2 = time.time() t2 - t1</pre>	<pre>import time n = 36 t1 = time.time() x = f(n) t2 = time.time() t2 - t1</pre>
0.39966511726379395	4.4078428745269775	8.20757508277893

Рис. Подбираем размер задачи

Задание. Подберите размер задачи так, чтобы время вычислений было в пределах от 10 до 20 секунд. Запустите программу с этим параметром несколько раз и обратите внимание на изменение времени выполнения.

Задание. Выясните, что принято за начало отсчета в библиотеке time. От какого момента отсчитывается количество секунд текущего времени и был выбран именно этот момент?

Можно убедиться, что время выполнения программы меняется случайным образом. Это случайные колебания, но они происходят вокруг некоторого среднего значения. Чтобы его определить, нам нужно запустить программу хотя бы 10 раз. Для этого используем цикл for.

```
n = 30
for _ in range(10):
    t1 = time.time()
    x = f(n)
    t2 = time.time()
    print(n, t2 - t1)

30 0.4076223373413086
30 0.4251720905303955
30 0.408022403717041
30 0.3991823196411133
30 0.4186282157897949
30 0.39321637115362549
30 0.4040796756744385
30 0.41590452194213867
30 0.3972351551055908
30 0.39862966537475586
```

```
n = 30
for _ in range(10):
    t1 = time.time()
    x = f(n)
    t2 = time.time()
    print(n, t2 - t1)

30 0.4174296595265137
30 0.3985307216644287
30 0.3925313949584961
30 0.4291994571685791
30 0.3930017948150635
30 0.41635560989379883
30 0.3959782123565674
30 0.39893102645874023
30 0.4115746021270752
30 0.40154170989990234
```

```
n = 30
for _ in range(10):
    t1 = time.time()
    x = f(n)
    t2 = time.time()
    print(n, t2 - t1)

30 0.5052793025970459
30 0.41782283782958984
30 0.3973841667175293
30 0.40341639518737793
30 0.4101145267486572
30 0.4003896713256836
30 0.42081522941589355
30 0.3940255641937256
30 0.4028308391571045
30 0.41011548042297363
```

Рис. Цикл для определения среднего

В нашей программе вместо счётчика цикла мы используем символ подчёркивания: `_`. Такой прием в Питоне применяют в ситуации, когда значение этой переменной не потребуется внутри цикла для расчётов.

Мы просто сообщаем компьютеру, что нужно повторить команды в теле цикла указанное количество раз.

Можно запускать программу и гораздо больше раз. Однако, как мы видим, наши цифры в среднем разбросаны случайным образом вокруг одного и того же, более или менее стабильного значения. Так что для дальнейших опытов мы ограничимся 10 запусками для каждого значения n .

Задание. Составьте программу для вычисления $f(n)$ с измерением времени десять раз при заданном n . Запустите ячейку несколько раз и обратите внимание на случайные колебания времени вокруг среднего значения.

Итак, мы провели подготовительные мероприятия. Теперь нам предстоит сделать по 10 прогонов нашей программы $f(n)$ для каждого значения n в диапазоне от нуля до максимального. Мы уже определили, какое максимальное значение n потребуется для разумного времени выполнения программы. В результате в нашей программе будет два вложенных цикла. Внешний цикл – это изменение n , внутренний цикл – это десятикратное повторение вызова $f(n)$ при одном и том же n . Программа выводит на экран значение n и значение t в каждой новой строке.

```
print("n,t")
for n in range(36):
    for _ in range(10):
        t1 = time.time()
        x = f(n)
        t2 = time.time()
        print(f"{n},{t2 - t1}")

Show/hide output
Clear selected outputs
View output fullscreen
```

n	t
0	3.57627
1	1.90734
2	1.43051
3	1.9073486328125e-06
4	1.6689300537109375e-06
5	1.430511474609375e-06
6	1.6689300537109375e-06
7	1.430511474609375e-06
8	4.76837158203125e-07
9	4.76837158203125e-07
10	4.76837158203125e-07
11	4.76837158203125e-07
12	4.76837158203125e-07
13	4.76837158203125e-07
14	4.76837158203125e-07
15	4.76837158203125e-07
16	4.76837158203125e-07
17	4.76837158203125e-07
18	4.76837158203125e-07
19	4.76837158203125e-07
20	4.76837158203125e-07
21	4.76837158203125e-07
22	4.76837158203125e-07
23	4.76837158203125e-07
24	4.76837158203125e-07
25	4.76837158203125e-07
26	4.76837158203125e-07
27	4.76837158203125e-07
28	4.76837158203125e-07
29	4.76837158203125e-07
30	4.76837158203125e-07
31	4.76837158203125e-07
32	4.76837158203125e-07
33	4.76837158203125e-07
34	4.76837158203125e-07
35	4.76837158203125e-07
36	4.76837158203125e-07

To exit full screen, press Esc

Рис. Два вложенных цикла

ОРГАНИЗАЦИЯ ЭКСПЕРИМЕНТОВ

Полученные результаты мы будем обрабатывать с помощью электронной таблицы. Чтобы загрузить наши цифры в электронную таблицу, лучше всего подходит формат CSV – Comma Separated Values – Значения, разделённые запятыми. Это текстовый формат файла ASCII. Каждая строка будущей таблицы состоит из значений отдельных ячеек (полей), а между ними стоят разделители полей – запятые.

Задание. Посмотрите на Википедии статьи ASCII и CSV. Обратите внимание на примеры и таблицы в этих статьях. Зафиксируйте в отчёте расшифровку и перевод этих названий.

Мы организуем вывод данных в файл CSV. Поэтому в нашей программе дополнительные моменты.

Во-первых, вначале мы выводим заголовок для будущих столбцов. В первом столбце у нас будет номер числа n . Во втором столбце мы выводим время расчёта t для числа $f(n)$ в секундах.

Во-вторых, в конце нашего примера мы используем не просто команду `print()`, а вывод форматированный строки: `f-string`. Перед строкой мы ставим букву `f` – от слова `formatted`. Затем в кавычках идёт содержимое строки. Внутри строки в фигурных скобках мы вводим название переменных. Между ними мы ставим разделитель – запятую. При выполнении такой команды на экран выводятся значения переменных. Таким образом, мы выводим значения `n` и `t`, разделённые запятой.

Если запустить такую ячейку на выполнение, на экран выводится две колонки чисел, разделённые запятыми. Мы можем рассмотреть результаты выполнения такой ячейки, нажав кнопку `View output fullscreen` – Вывести результаты для просмотра на полный экран. Пока что это числа на экране. И это уже хорошо.

Задание. Составьте программу по последнему образцу. Запустите её на выполнение и убедитесь, что на экран выводится две колонки чисел, разделённые запятыми. Переключитесь в полноэкранный режим для просмотра результатов. Выходите из просмотра, нажав кнопку `Esc`.

Наш следующий шаг – отправить эти результаты в файл. Желательно это сделать автоматически, чтобы не заниматься ручным копированием через буфер обмена. Любая ручная операция замедляет работу и открывает возможности для дополнительных ошибок.

Если всё делать очень серьёзно, нужно открыть файл на чтение, записать туда информацию и закрыть файл. Но гораздо проще будет решить эту задачу с помощью

перенаправления вывода в файл. Для этого мы соберём нашу программу в одну кодовую ячейку. Затем мы создадим файл с этой программой на диске сервера. Далее мы запустим эту программу на выполнение и отправим результаты вместо экрана в файл. При этом мы отправляем в файл именно то, что мы наблюдали на экране. Так что здесь не будет двойной работы, когда мы одной командой выводим данные на экран, а другой командой отправляем их в файл.

Соберём нашу программу в новую кодовую ячейку. Скопируем необходимые фрагменты из предыдущих ячеек. Вставим дополнительную команду в начале этой ячейки: два символа процента и специальная команда `writefile` – записать ячейку в файл. Далее через пробел мы указали имя файла. Запись будет выполняться в текущем каталоге.

`%%writefile` – это пример инструмента, который называется *Cell magic* – Магические команды для кодовой ячейки Jupyter Notebook. Такие команды изменяют поведение целой ячейки – всех строк кода.

После выполнения такой кодовой ячейки мы получим сообщение об успешной записи текста программы в файл – `Writing fib.py`.

Задание. Выясните, как работают магические команды, которые начинаются с одного и с двух символов процента. Поэкспериментируйте с ними в новых кодовых ячейках своего блокнота.

```
✓ 0s  %writefile fib.py

def f(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return f(n - 1) + f(n - 2)

import time

print("n,t")
for n in range(36):
    for _ in range(10):
        t1 = time.time()
        x = f(n)
        t2 = time.time()
        print(f"{n},{t2 - t1}")
```

→ Writing fib.py

Рис. Запись текста в файл

Мы записали текст программы в файл. Этот файл должен оказаться в текущем каталоге. Мы можем открыть панель проводника виртуальной машины, нажав на кнопку Files – файлы – в левой части окна.

Мы также можем вывести на экран список файлов, находящихся в текущем каталоге. Для этого мы используем команду LS – от слова LIST – список. Чтобы выполнить команду операционной системы, в начале строки нужно добавить восклицательный знак. Перед

нами операционная система Linux Ubuntu. Поэтому здесь работают только команды Linux.

Теперь мы можем запустить эту программу на выполнение. В начале строки мы опять используем восклицательный знак, чтобы ввести команду операционной системы. Мы вызываем интерпретатор Python. Через пробел сообщаем ему имя файла, котором находится наша программа: fib.py. Результаты работы программы выводятся на экран компьютера.

Чтобы перенаправить вывод вместо экрана в файл, мы используем «стрелку вправо», то есть знак «больше», и указываем имя текстового файла. В нашем случае это файл `fib.csv`.

Затем мы снова можем вывести на экран список файлов командой LS и убедиться, что у нас появился новый объект.

Вся эта манипуляция обычно называется «перенаправление стандартного потока вывода в файл». Подробно эти действия изучаются в курсе «Операционные системы».

```
+ Code + Text  
[25] !ls  
fib.py sample_data  
[48] !python fib.py  
[27] !python fib.py > fib.csv  
[28] !ls
```

Рис. Перенаправление вывода в файл

Задание. Запустите программу в командной строке. Перенаправьте вывод в файл. Убедитесь, что этот файл появился в текущем каталоге.

ЗАГРУЗКА РЕЗУЛЬТАТОВ

Мы получили файл с данными. Пока что он находится в виртуальной машине, в облаке. Для дальнейшей работы мы скачаем его на локальный компьютер. Подводим курсор к файлу и нажимаем правую кнопку мыши. В выпадающем, контекстном меню выбираем Download – Скачать. Загружаем файл на локальный компьютер и не забываем обратить внимание на тот каталог, в котором этот файл будет сохранён.

Переходим в наш отчёт. Создаём новый лист нашего отчета. Загружаем наш файл на эту страницу. Для этого переходим на вкладку Данные. Нажимаем кнопку Получить данные. Выбираем пункт Из локального файла.

Открывается диалоговое окно мастер импорта текста. Здесь нам нужно будет сделать некоторые настройки, исходя из формата нашего файла данных *.CSV – как показано на рисунке. Разделитель полей – запятая. Дополнительные параметры: Десятичный разделитель целой и дробной частей – точка.

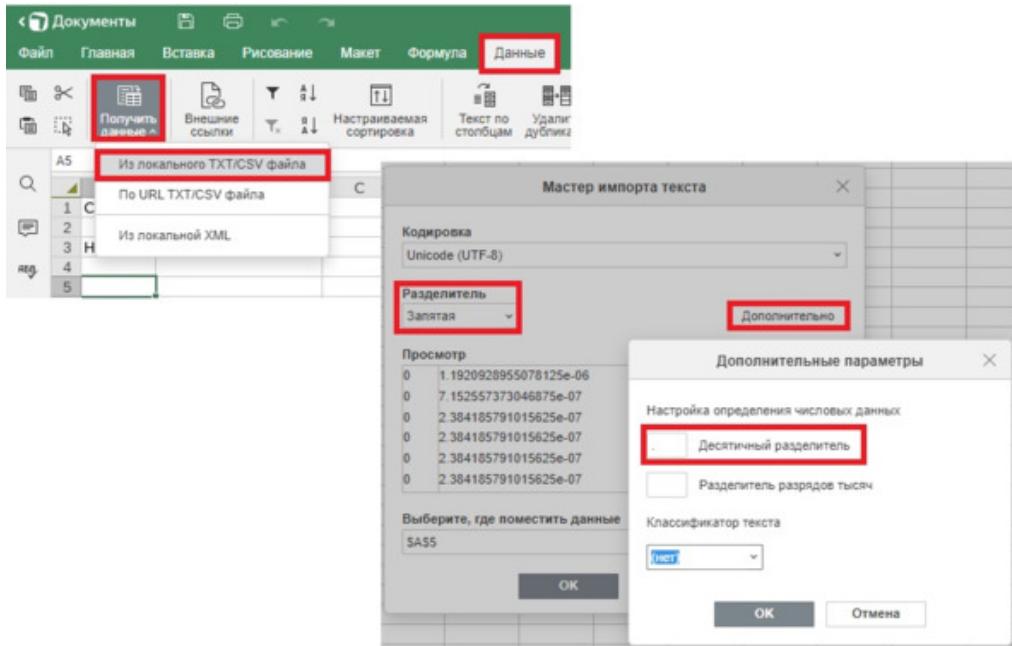


Рис. Загрузка файла CSV

Если загрузка файла прошла удачно, мы получим два столбца чисел. По умолчанию числовые значения будут прижаты к правому краю ячейки. Если же загружается текст — текстовая строка — строка символов, это содержимое будет прижато к левому краю ячейки.

Числовые значения выводятся на экран в подходящем формате — «подходящем» по мнению электронной таблицы. В нашем примере — это представление с плавающей точкой (или запятой). Щёлкаем по числовой ячейке и наблюдаем полное, точное значение в строке формул.

Задание. Загрузите свой файл с результатами эксперимента в отчёт — на новую страницу электронной таблицы. Сделайте соответствующие ссылки и заголовки.

Мы можем настроить формат вывода чисел. Для этого щёлкаем по первой ячейке с числовыми данными.

Затем нажимаем комбинацию клавиш [Ctrl + Shift + Down]. Имеется в виду «стрелка вниз». Когда мы нажимаем и удерживаем клавишу Shift, а затем перемещаем курсор, это позволяет выделить диапазон ячеек в электронной таблице. Чтобы быстро переместиться в начало или конец заполненного столбца, мы нажимаем комбинацию клавиш: Control и соответствующую стрелку. Таким образом, комбинация трёх клавиш позволяет быстро выделить столбец, который заполнен какими-нибудь значениями.

После того, как мы выделили диапазон ячеек, переходим в раздел меню Главная. В выпадающем списке выбираем числовой формат. Теперь мы можем увеличить и уменьшить количество разрядов, которые выводятся на экран.

Архив ЛР4 ИВТ-999.xlsx	
Файл	Главная
B5	fx 0,0000011920928955078
	Научный
	Общий
	Числовой
	Научный
	Финансовый
	Денежный
	Краткий формат даты
	Длинный формат даты
	Время

Архив ЛР4 ИВТ-999.xlsx	
Файл	Главная
B5	fx 0,0000011920928955078
	Числовой
	Увеличить разрядность
	Уменьшить разрядность
	0,00

Рис. Настраиваем формат вывода

Задание. Настройте формат вывода на экран – так, чтобы в наших числах выводилось достаточное количество разрядов: не слишком мало и не слишком много.

Данные загружены. Единственная проблема – пропала первая строка с заголовками столбцов. Проверим, не случайно ли это так произошло. Данный облачный продукт предоставляется уже не первый день, и даже не первый год. Возможно, здесь мы имеем ситуацию, которую программисты объясняют так: «Это не баг, это фича».

Задание. Выясните, что означает выражение «Это не баг, это фича».

Проверка наша будет очень простой. Мы сами сформируем небольшой файл формата CSV с помощью Блокнота и загрузим его в таблицу. Если эффект повторится, то это уже не просто случайность.

Пусть в нашем файле будет первая строка с заголовками столбцов и две-три строчки с данными. Будем использовать запятую как разделитель полей и точку как десятичный разделитель целой и дробной частей. Загружаем файл в таблицу и удивляемся результатам.

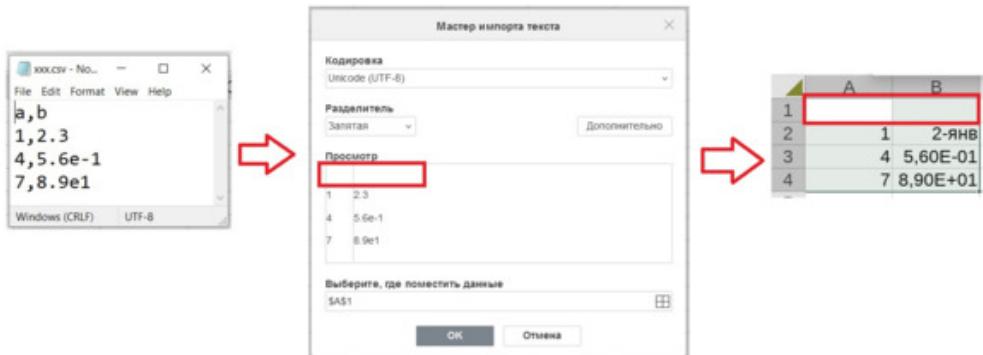


Рис. «Особенности» загрузки CSV

Задание. Проведите своё собственное исследование по проблеме загрузки CSV и зафиксируйте результаты на новой странице отчёта. Сформируйте небольшой файл формата CSV и проверьте, что происходит при импорте файла в таблицу Яндекс. Проведите несколько опытов, например, данные с заголовками, данные без заголовков, текстовые и числовые значения в первой строке, пустая первая строка.

На момент написания данного учебного пособия неприятный эффект подтвердился. По отзывам пользователей, облачные продукты могут уступать по своей функциональности настольным, полноценным версиям – Desktop. Это может касаться сложных графиков и других продвинутых возможностей электронных таблиц. Но для простых задач и для просмотра готовой таблицы облачный «офис» обычно работает хорошо.

Поэтому мы просто скачаем наш файл на компьютер и основную часть работы будем выполнять в настольном варианте. Дальнейшее описание работы будет касаться Excel, хотя вы можете использовать и любые другие аналогичные инструменты.

Выбираем в верхнем меню Файл – Скачать как – XLSX. Сохраняем на локальном компьютере и открываем в Excel.

Теперь займёмся загрузкой файла CSV в Excel. Выбираем в верхнем меню вкладку Данные, нажимаем кнопку Получение внешних данных и выбираем вариант Из текста. Всплывающая подсказка подтверждает, что мы будем заниматься импортом данных из текстового файла.

Когда говорят «текст», обычно имеется в виду файл формата ASCII, который можно открыть в обычном Блокноте. Такой файл содержит только цифры, буквы и привычные символы типа точки или запятой.

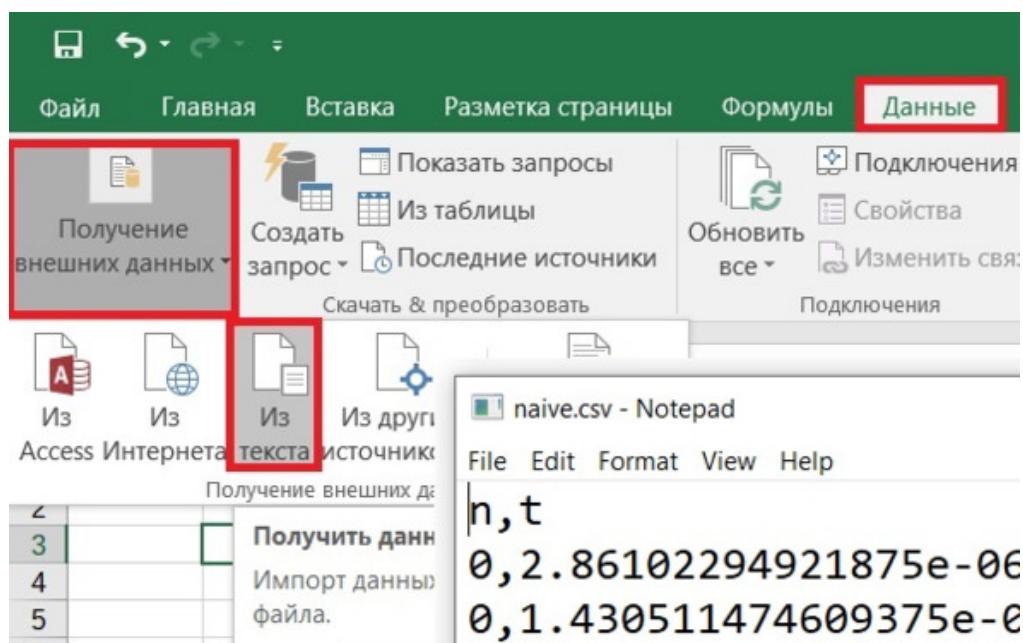


Рис. Импорт данных из файла CSV

Мы указываем файл, который нужно загрузить. Далее мы пройдём через несколько диалоговых окон для настройки Мастера импорта данных из текста.

На Первом шаге импорта мы указываем, что наш исходный файл содержит разделители полей, то есть столбцов с данными.

Здесь же мы говорим, что наш файл содержит заголовки. Другими словами, в первой строке файла будут заголовки и скорее всего, это будут строчки текста. Содержимое столбцов может быть совсем другого типа – в отличие от заголовков. Типы данных – это важный момент в аналитике.

Здесь же можно выбрать номер строки, начиная с которой будут загружаться данные. Нас устроит загрузка, начиная с первой строки.

В нижней части диалогового окна мы можем наблюдать результаты наших настроек. Такой предварительный просмотр помогает нам следить за правильностью основных настроек. В нашем примере все данные представлены на экране просмотра, включая первую строку с заголовками.

Нажимаем кнопку Далее.

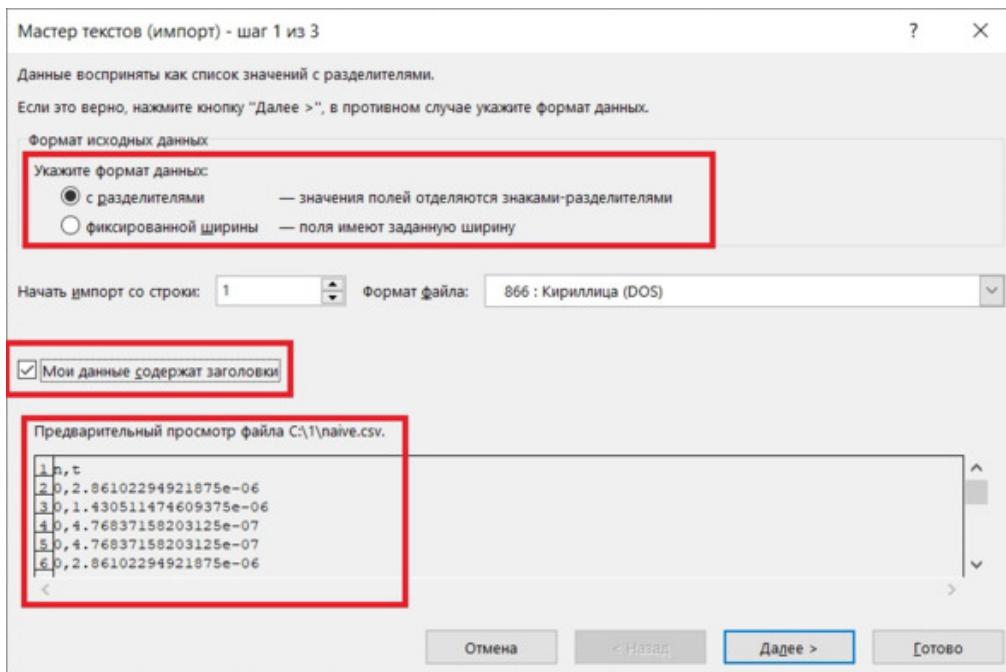


Рис. Настройка первого шага

На втором шаге настройки импорта мы выбираем разделитель полей. В нашем примере это запятая. Предварительный просмотр вначале может показать данные в виде одного столбца. После выбора правильного символа-разделителя мы видим разбивку файла на столбцы.

Нажимаем кнопку Далее.

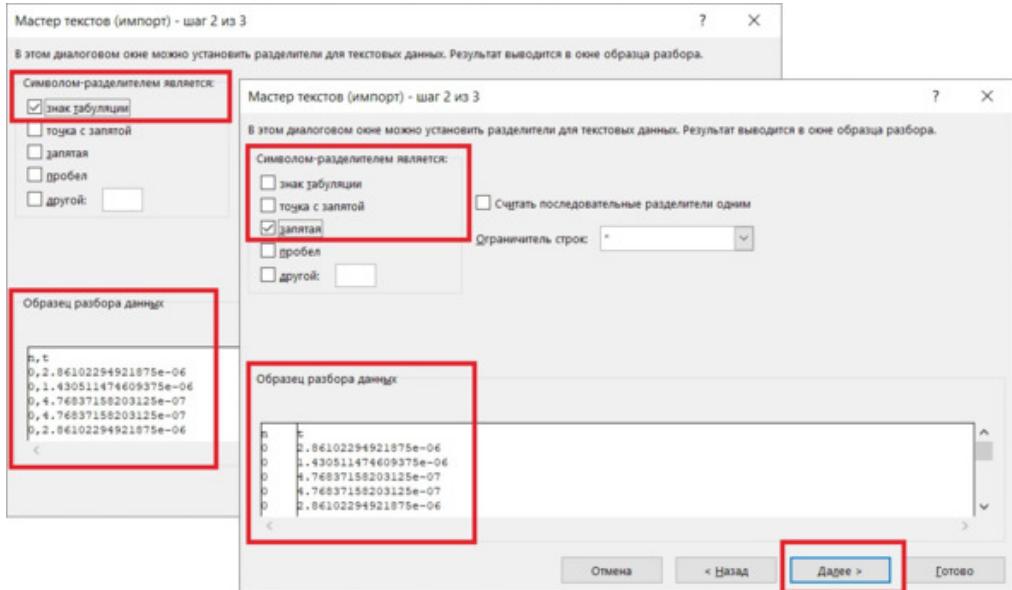


Рис. Настройка второго шага

На третьем шаге мы выбираем десятичный разделитель целой и дробной части. В зависимости от языка интерфейса Excel будет различаться настройка по умолчанию. За этим нужно проследить и подправить вручную.

Нажимаем кнопку Подробнее. Затем в дополнительных настройках указываем точку как десятичный разделитель. Нажимаем OK и Готово.

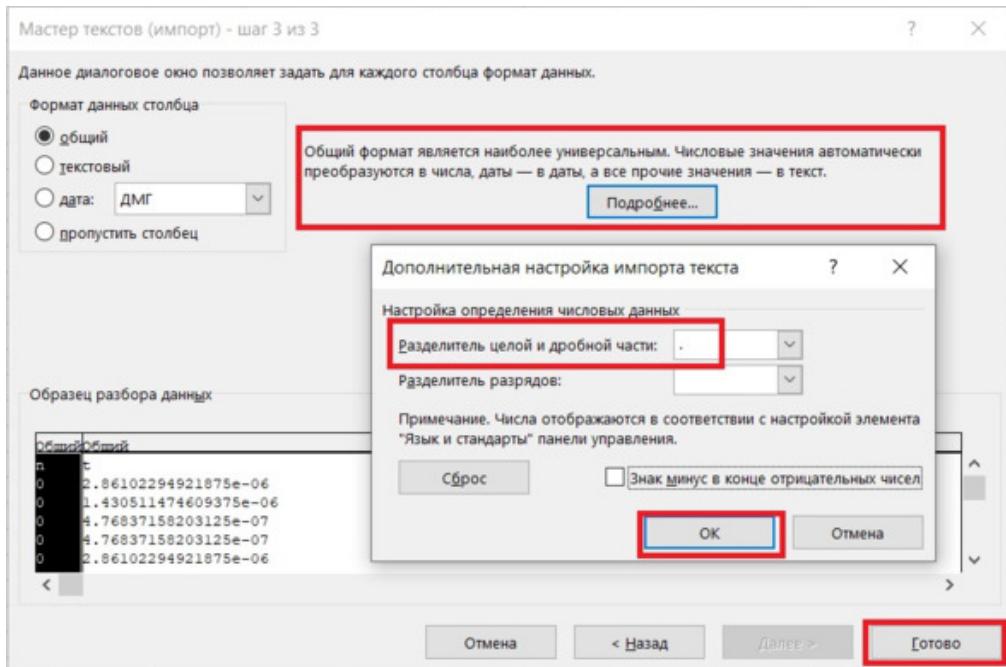


Рис. Настройка третьего шага

Выбираем ячейку текущего листа, начиная с которой будут загружены данные из текстового файла.

Обратим внимание на форматирование содержимого ячеек.

Текстовое содержимое прижато к левой стороне ячейки.

Числовое содержимое прижато вправо.

В нашем примере всё распознано правильно.

The dialog box shows options for importing data into a table, with 'Table' selected. The destination is set to 'Existing sheet' with the range '\$B\$3'. The spreadsheet shows a table with columns A, B, C, D, and E, containing 10 rows of data starting from row 3.

	A	B	C	D	E
1					
2					
3	n				
4	0	2,86E-06			
5	0	1,43E-06			
6	0	4,77E-07			
7	0	4,77E-07			
8	0	2,86E-06			
9	0	2,86E-06			
10	0	9,54E-07			
11	0	4,77E-07			
12	0	4,77E-07			
13	0	9,54E-07			

Рис. Результат импорта данных

Задание. Загрузите свой CSV файл в Excel и убедитесь, что формат представления данных распознан правильно.

ГРАФИКИ

Данные загружены. Пора построить график и посмотреть на него.

Самый простой график, который мы будем использовать, это диаграмма разброса. Другие названия: корреляционное поле, поле корреляции, диаграмма рассеяния – Scatter Plot.

Выделяем диапазон ячеек вместе с заголовками. для выделения диапазона можно использовать комбинации клавиш: [Shift + Right], [Ctrl + Shift + Down].

Далее через верхнее меню проходим следующие шаги: Вставка – Диаграммы – Вставить точечную... – Точечная – Точечная.

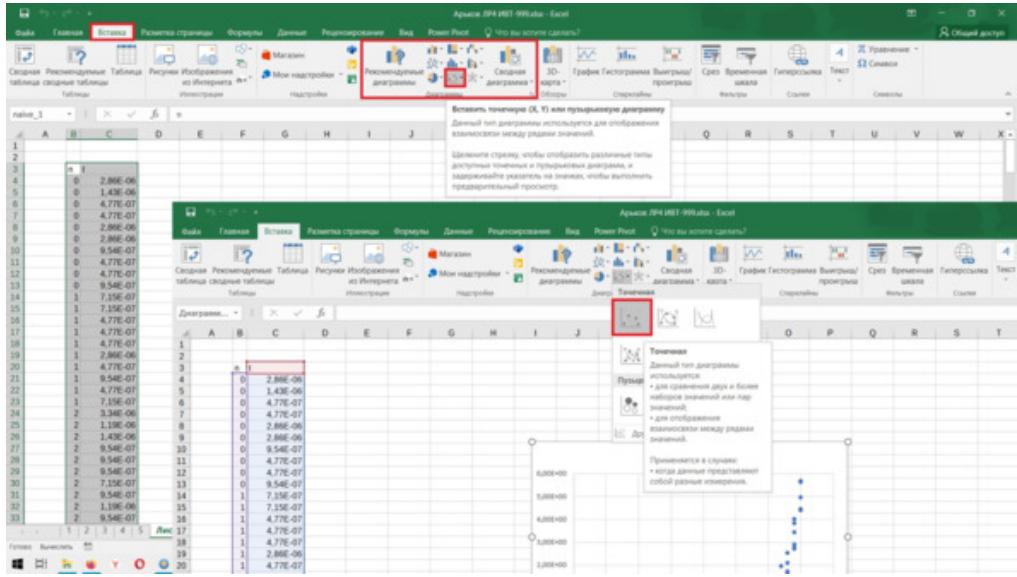


Рис. Вставка диаграммы разброса

На этом графике даже без дополнительной обработки можно обнаружить общую закономерность. При увеличении размера задачи время вычислений растет экспоненциально. И есть небольшой разброс значений вокруг этой общей тенденции.

В плане оформления графика можно отметить несколько моментов.

Данные из первого столбца работают по оси «иксов», второй столбец – по оси «игреков».

Заголовок второго столбца стал заголовком всего графика.

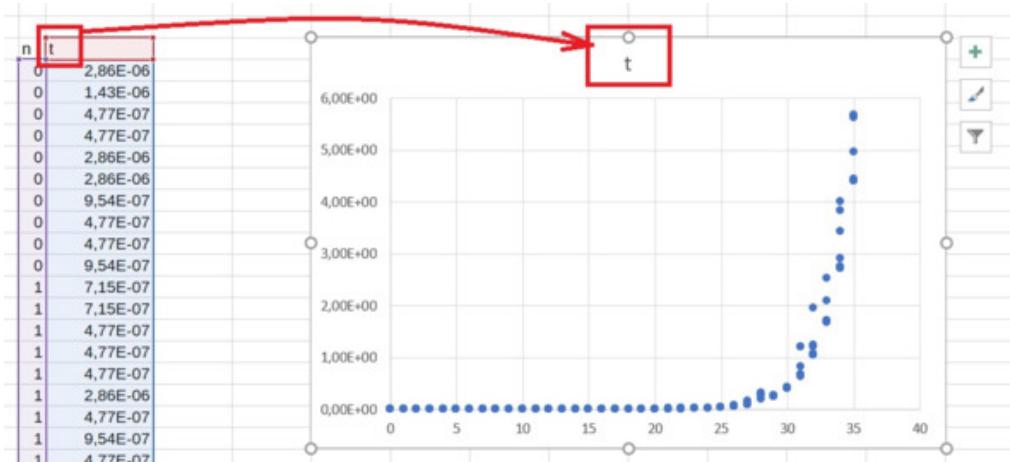


Рис. Диаграмма разброса без настройки

Задание. Выделите диапазон ячеек с загруженные данными и постройте диаграмму разброса. Опишите в отчете общий характер зависимости, который можно наблюдать на графике.

ЛОГАРИФМ И ЭКСПОНЕНТА

Чтобы более подробно исследовать эту зависимость, попробуем взять логарифм от времени выполнения программы. Тогда наши выводы будут более обоснованными. Мы даже сможем количественно оценить некоторые параметры зависимости.

Напишем пару формул, и сделаем мы это в блокноте Colab. Заодно попрактикуемся в работе с Markdown и LaTeX.

Запишем простейшую формулу для экспоненциальной зависимости. У нас будет постоянный коэффициент С и основание степени А.

Если взять логарифм от левой и правой частей уравнения, мы получим линейное уравнение. Здесь икс участвует в первой степени. График такой функции должен выглядеть как прямая линия.

<p>Экспоненциальная зависимость y от x:</p> $y(x) = C \cdot A^x$ <p>Логарифмируем левую и правую части уравнения:</p> $\lg y = \lg C + x \lg A$ <p>Логарифм y линейно зависит от x.</p>	<p>Экспоненциальная зависимость y от x:</p> $y(x) = C \cdot A^x$ <p>Логарифмируем левую и правую части уравнения:</p> $\lg y = \lg C + x \lg A$ <p>Логарифм y линейно зависит от x.</p>
---	---

Рис. Логарифмируем уравнение связи

Задание. Создайте новую текстовую ячейку в Colab. Введите текст, приведённый на рисунке. Измените обозначения так, чтобы в формуле участвовало время выполнения программы t и номер числа фибоначчи n . Укажите эти условные обозначения и названия переменных.

Добавим новый столбец к нашим данным. Найдём десятичный логарифм времени выполнения расчётов. Это встроенная функция `LOG10()`.

Выделим первый и третий столбцы таблицы. Для этого выделяем первый столбец, затем нажимаем кнопку `[Ctrl]`, удерживаем её нажатой и выделяем мышкой третий столбец.

Вставляем новую диаграмму разброса по этим данным. Здесь мы надеемся увидеть линейную зависимость.

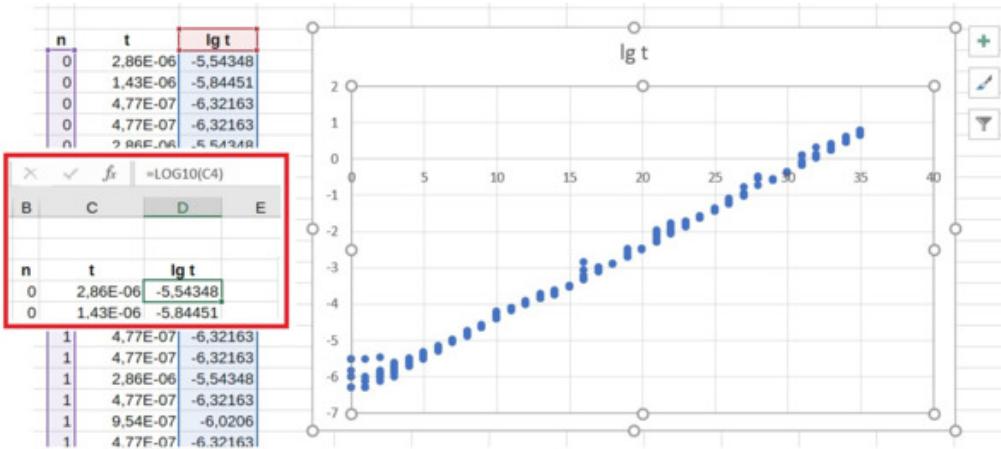


Рис. Линейная зависимость

Задание. Найдите логарифм времени расчётов и постройте новую диаграмму разброса. Обратите внимание на линейный характер зависимости.

Если внимательно рассмотреть последний график, можно обнаружить, что линейная зависимость просматривается на уровне минимальных значений времени для каждого размера задачи. Через эти нижние точки и можно будет провести прямую линию.

Случайные отклонения в нашем примере просматриваются в сторону увеличения продолжительности расчётов. Это может объясняться тем, что мы проводили вычисления в облаке. В этом случае время расчётов скорее всего не гарантируется. Если в момент выполнения нашей программы облачный сервер (датацентр) будет решать и другие вычислительные задачи, то нам может достаться меньше ресурсов и время работы увеличится.

СВОДНАЯ ТАБЛИЦА

Попробуем построить линию зависимости по нижним точкам графика. Для этого нужно будет найти минимальное значение времени t для каждого размера задачи n . В этом нам поможет сводная таблица – Pivot Table. Это популярный инструмент для обработки и анализа данных. Он уже встроен во многие табличные процессоры, то есть электронные таблицы.

Выделяем диапазон ячеек, в котором расположены наши данные. Это три столбца, которые включают и рассчитанный нами логарифм времени.

Выбираем в верхнем меню Вставка – Сводная таблица.

В диалоговом окне Создание сводной таблицы настроим некоторые параметры. Убедимся в том, что у нас указан нужный диапазон ячеек в качестве исходных данных. Выберем вариант вывода отчёта на текущий, существующий лист и укажем ячейку, начиная с которой начнётся вывод. Это будет левый верхний угол будущего отчёта.

Всплывающая подсказка говорит что-то невнятное про «сведение данных». На самом деле, речь идёт об агрегировании данных, то есть о вычислении итоговых цифр типа среднего значения. Видимо, первоначально нам хотели что-то сказать про «сводку и группировку данных». Это термин из области статистики и анализа данных.

Но для первого знакомства проще будет показать, как это работает, чем долго объяснять теорию. После нашей демонстрации можно будет осмысленно полистать учебник и достичь большего понимания материала.

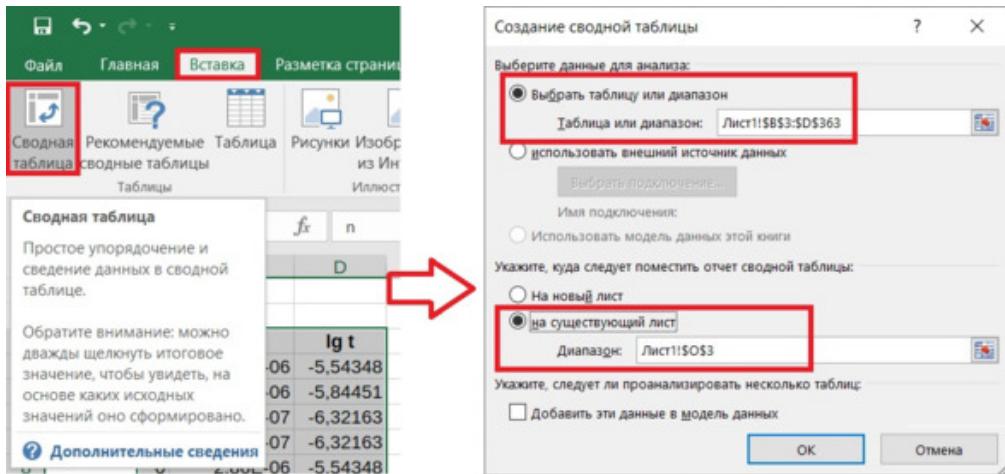


Рис. Выбор данных для сводной таблицы

Нажимаем кнопку ОК, и на экране появляются очертания будущей таблицы – шаблон под названием «отчет». Слово «отчёт» в данном случае появилось не случайно. Это отголоски технологии под названием «аналитические отчеты». Их обычно строят по информации из базы данных. В простейшем случае в роли отчета будет выступать небольшая табличка, в которой подсчитываются некоторые итоги. Например, выручка магазина за каждый месяц.

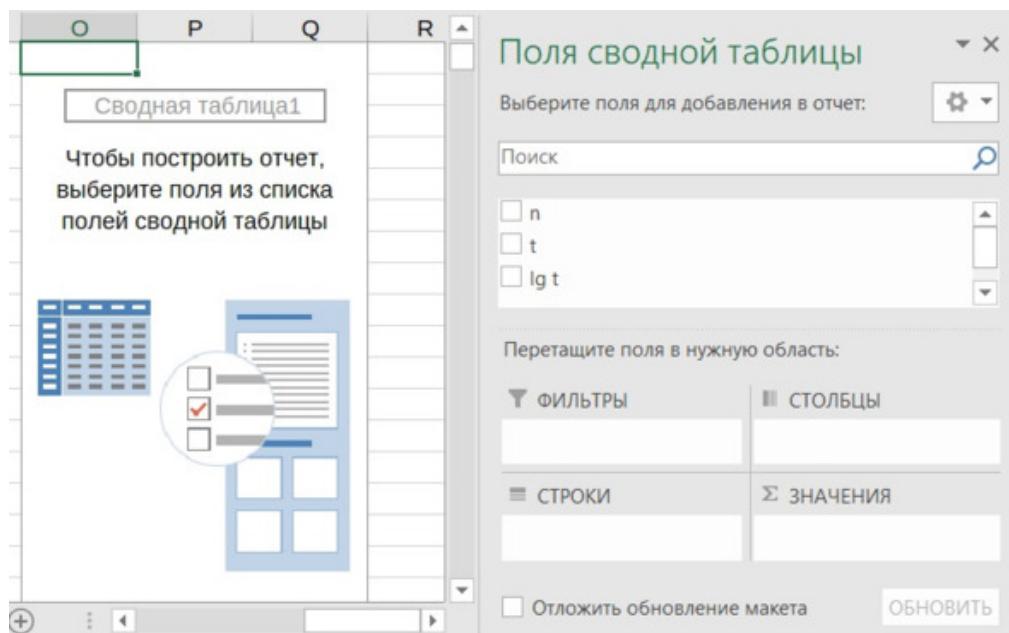


Рис. Шаблон и панель настройки сводной таблицы

Здесь нам предлагают выбрать поля из списка. Также здесь имеется очень схематичное изображение процесса настройки сводной таблицы. После того, как мы с вами построим сводную таблицу, станет понятной вся эта схема.

В правой части экрана появляется панель под названием Поля сводной таблицы. В нашем примере в списке полей мы видим название, заголовки трёх столбцов нашей исходной таблички: n , t и $\lg t$.

В нижней части этой панели имеется четыре окошечка. Вся настройка сводной таблицы сводится к перетаскиванию «названий переменных или столбцов» из списка полей в соответствующее окошечко. Таким способом мы можем указать, какие столбцы будут использованы для формирования строк и столбцов, а также что нужно будет подсчитать в качестве итоговых показателей.

Мы перетаскиваем поле n в окошко Строки.

Затем мы перетаскиваем логарифм времени $\lg t$ в окно Значения.

После этого на экране появляется сводная таблица, в которой автоматически подсчитаны некоторые итоги. По умолчанию здесь подсчитывается сумма по указанному столбцу с разбивкой по категориям, указанным в строках. В окошке Значения появляется указание, что здесь вычисляется сумма по полю $\lg t$.

Почему именно сумма? В экономике чаще всего итоги выражаются суммой. Например, руководителя будет интересовать сумма всех поступлений за месяц или сумма всех расходов. Поэтому в окошке Значения можно видеть знак суммы – большую, заглавную греческую букву Сигма. Это символ суммы во многих математических формулах.

The screenshot shows a Microsoft Excel interface. On the left is a pivot table with rows labeled from 0 to 15. The first column is 'Названия строк' (Row Labels) and the second column is 'Сумма по полю lg t'. The values in the second column are numerical, decreasing from -59,80266409 at row 0 to -35,29048571 at row 15. To the right of the pivot table is the 'Поля сводной таблицы' (Pivot Table Fields) pane. It has a search bar and three sections: 'ФИЛЬТРЫ' (Filters), 'СТОЛБЦЫ' (Columns), 'СТРОКИ' (Rows), and 'ЗНАЧЕНИЯ' (Values). Under 'ЗНАЧЕНИЯ', there is a dropdown menu set to 'Сумма по полю lg t'. Below the pane, the status bar displays 'Рис. Сумма как итоговое значение'.

Названия строк	Сумма по полю lg t
0	-59,80266409
1	-61,60884407
2	-59,41695854
3	-58,16168603
4	-56,22805515
5	-54,49598781
6	-52,4818338
7	-50,30896427
8	-48,03087897
9	-46,44967698
10	-43,6116885
11	-41,38094526
12	-40,00034347
13	-38,2543422
14	-37,09507866
15	-35,29048571

Рис. Сумма как итоговое значение

Задание. Выделите диапазон исходных данных и вставьте сводную таблицу. Настройте таблицу так, чтобы по строкам был номер числа n , а в качестве итоговых значений обрабатывался логарифм времени $\lg t$. Обратите внимание на то, что по умолчанию

производится вычисление суммы в качестве итогового, сводного значения.

Теперь перейдём к более тонкой настройке сводной таблицы. Нас будет интересовать вовсе не сумма, а минимальное, наименьшее значение времени для каждого размера задачи n .

В окошке Значения щёлкаем левой кнопкой мыши по нашему полю $\lg t$. В выпадающем контекстном меню выбираем Параметры полей значений. Выражение не слишком осмысленное. Оно пытается сообщить нам, что мы можем выбрать способ агрегирования данных в виде суммы, или минимума, или среднего значения. Видимо, слово «агрегирование» не слишком понравилось разработчикам программы. А может быть, в то время (лет эдак тридцать тому назад) этот термин ещё не был таким популярным, как сейчас. Попутно отметим, что точно такие же действия доступны и в базах данных. В нашей работе мы используем Excel для большей наглядности.

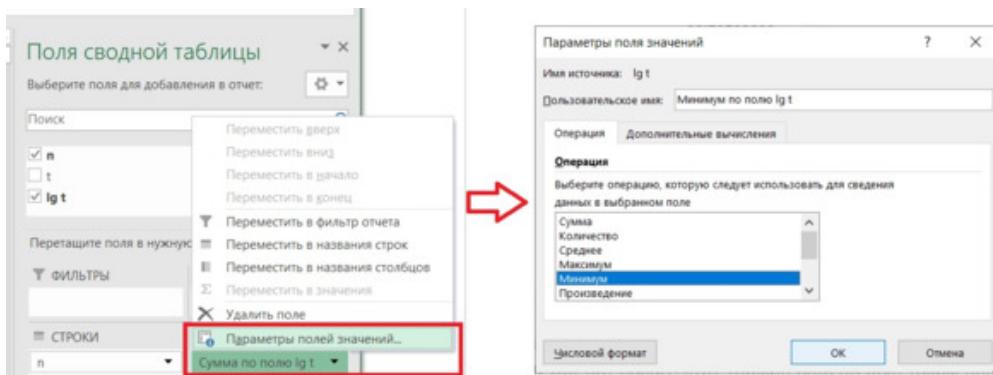


Рис. Выбор способа агрегирования

Выбираем minimum в качестве «операции для сведения данных». В переводе на русский язык это означает, что мы будем искать минимальное значение

логарифма времени по всем прогонам нашей программы для одного и того же значения n .

Здесь также есть возможность задать «пользовательское имя». Это будет заголовок столбца с нашими подсчитанными итогами.

Нажимаем OK и наблюдаем автоматическое перестроение сводной таблицы.

Теперь в оконке Значения, и в заголовке столбца сводной таблицы указано «минимум по полю $\lg t$ ».

The screenshot shows the 'Поля сводной таблицы' (PivotTable Fields) dialog box in Excel. On the left, there's a table titled 'Минимум по полю lg t'. The column header 'Минимум по полю lg t' is highlighted with a red box. On the right, under 'СТОЛБЦЫ' (Columns), the same header is also highlighted with a red box. The 'Поля сводной таблицы' pane shows fields 'n', 't', and 'lg t' with checkboxes; 'lg t' is checked. The 'Перетащите поля в нужную область:' (Drag fields to the required area:) section has 'ФИЛЬТРЫ' (Filters) and 'ЗНАЧЕНИЯ' (Values) sections. In the 'ЗНАЧЕНИЯ' section, 'n' is listed under 'СТРОКИ' (Rows) and 'Минимум по полю lg t' is listed under 'ЗНАЧЕНИЯ'.

N	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	-6,321629909	-6,321629909	-6,14553865	-6,020599913	-5,719569918	-5,543478659	-5,321629909	-5,066357404	-4,890266145	-4,663618512	-4,426760252	-4,201055978	-4,024964719	-3,862237421	-3,752841697	-3,545655578

Рис. Минимум как итоговое значение

Задание. Для своей сводной таблицы выберите минимальное значение как способ агрегирования данных. Обратите внимание на изменения в сводной таблице.

«СВОДНЫЙ» ГРАФИК

Переходим к построению графика. В таблицах Excel есть готовый инструмент под названием «Сводная таблица». Но не ищем лёгких путей. К тому же, нас интересуют все функциональные возможности

визуализации. Так что мы продублируем сводные данные и поработаем над ними.

Выделим диапазон ячеек нашей сводной таблицы. Скопируем его в буфер обмена и вставим на свободном месте текущего, рабочего листа КАК ЗНАЧЕНИЯ. В этом случае у нас не будет связи с исходной сводной таблицей, зато мы сможем свободно обращаться с этими обобщёнными цифрами и, в том числе, строить графики.

The screenshot shows two Excel windows side-by-side. The left window displays a pivot table with data from row 0 to 10. A context menu is open over the cell containing the value -6,321629909. The menu items include 'Копировать' (Copy), 'Формат ячеек...' (Format Cells...), 'Обновить' (Update), 'Параметры сводной таблицы...' (Pivot Table Options...), and 'Скрыть список полей' (Hide Fields). The 'Copy' option is highlighted. A red arrow points from this menu to the right window. The right window shows the 'Paste Special' dialog box with the 'Values' option selected. Below it is a table with data from row 0 to 11, where the first two rows have the same value (-6,32163) and the remaining rows show a decreasing trend starting from -5,06636.

	R	S	T	U
0	-6,32163			
1	-6,32163			
2	-5,06636			
3	-4,89027			
4	-4,66362			
5	-4,42676			
6	-4,20106			
7				
8				
9				
10				
11				

Рис. Вставка значений

После вставки значений переименуем наши столбцы. Дадим понятные названия: n и $\lg t$. Удалим последнюю строку Общий итог. Она нам для графика не понадобится.

Выделяем этот диапазон ячеек. Вставляем новую диаграмму разброса. Здесь гораздо лучше просматривается линейная зависимость.

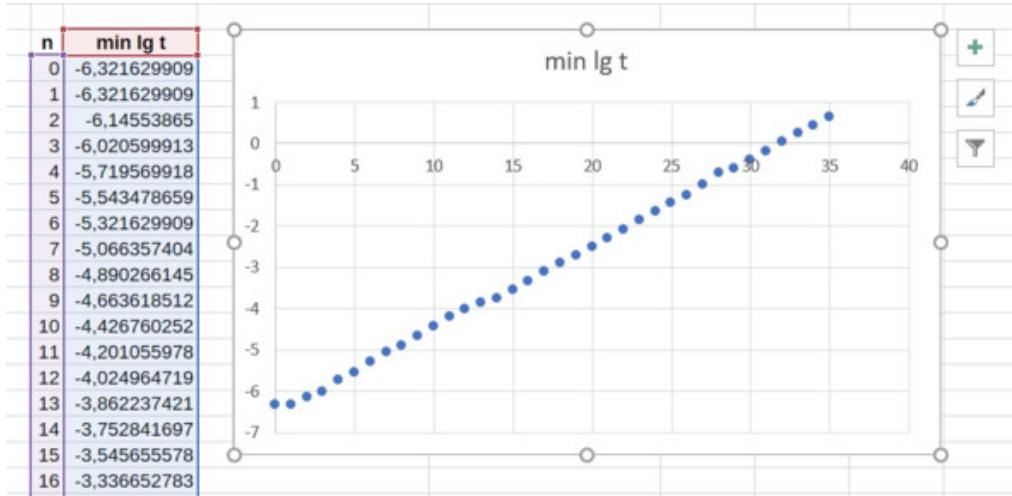


Рис. Новая диаграмма разброса

Задание. Постройте новую диаграмму разброса.
Обратите внимание на характер зависимости.

Теперь настроим оформление графика таким образом, чтобы он стал наиболее информативным, понятным, полезным. Зададим масштаб по оси X так, чтобы всё было занято нашим графиком.

В нашем случае максимальное значение X равно 35.

Дважды щёлкаем по оси X, и справа появляется панель Формат оси. Указываем Максимум 35 и нажимаем [Enter]. Теперь наши данные заняли всё поле графика.

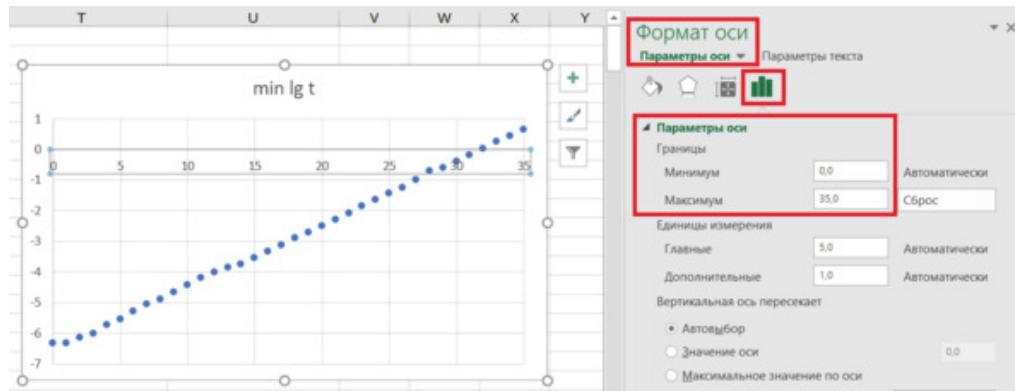


Рис. Настройка оси координат

Но и это ещё не всё. Мы можем получить уравнение линии, которая пройдет по нашим точкам. Нажимаем кнопку [+] Элементы диаграммы справа сверху, рядом с нашим графиком. Эта кнопка появляется только при выборе данного объекта. Если щёлкнуть в другое место рабочего листа, пропадает выбор графика и пропадает его контекстное меню.

Далее в меню выбираем Линия тренда – Дополнительные параметры.

Название «тренд» вообще-то означает «основная, долгосрочная тенденция». Это если мы рассматриваем, как что-то меняется во времени. В других случаях надо бы это по-другому называть. На самом деле, это линия, которая проходит в среднем по точкам. Как бы «средняя, приблизительная зависимость». Подробнее мы это изучаем в других курсах.

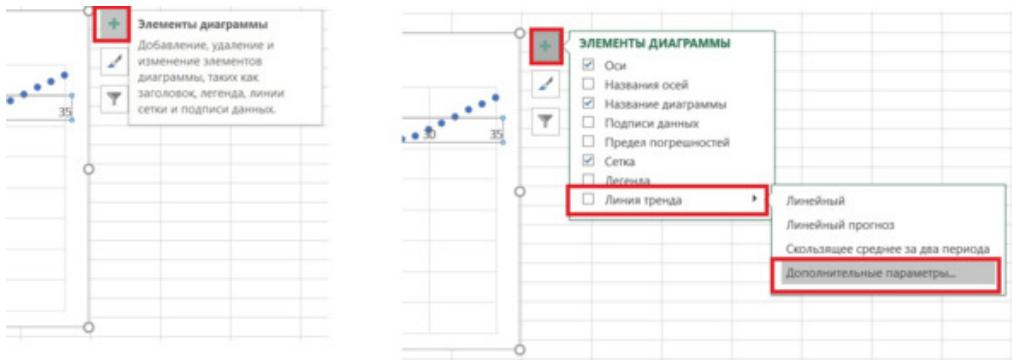


Рис. Включаем линию на графике

Затем мы выбираем характер уравнения для нашей линии. В нашем примере это прямая линия. И эта зависимость будет называться «линейная». Рядом можно рассмотреть схематическое изображение: «облако» точек, а через них как-то в среднем проведена прямая линия. Так что линия окружена этими точками.

Ставим галочку и выбираем пункт: показывать уравнение на диаграмме.

На графике появляется линия и уравнение. Это уравнение можно перетащить на свободное место, чтобы формула лучше читалась.

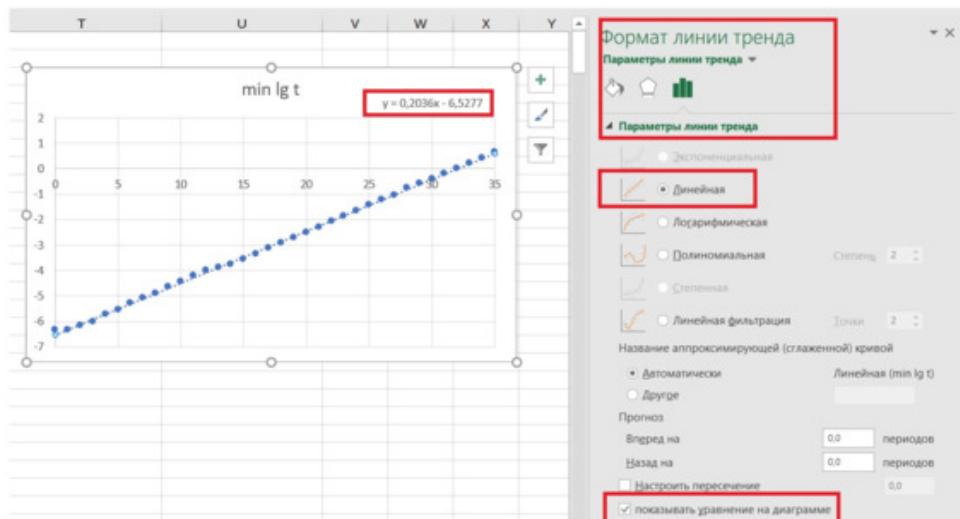


Рис. Прямая линия и уравнение на графике

Задание. Включите построение прямой линии и вывод уравнения на график.

Для окончательного оформления нашего графика нужно сделать заголовки по осям и общий заголовок для диаграммы. На каждой оси мы указываем название переменной, а ещё лучше – объясняем это словами. Общий заголовок графика должен сообщить нам что-то полезное об этом рисунке в целом. Мы также можем включить Легенду. Это объяснение условных обозначений – в тех случаях, когда на графике имеется несколько линий или несколько разных маркеров.

Пример оформления – см. на рисунке ниже.

Чтобы включить отображение дополнительных средств оформления, мы используем меню Оформление диаграммы. Там же мы задаём расположение легенды.

Можно настроить расположение меток по осям – через точку пересечения осей.

Шрифт для каждой надписи на рисунке тоже можно подобрать.

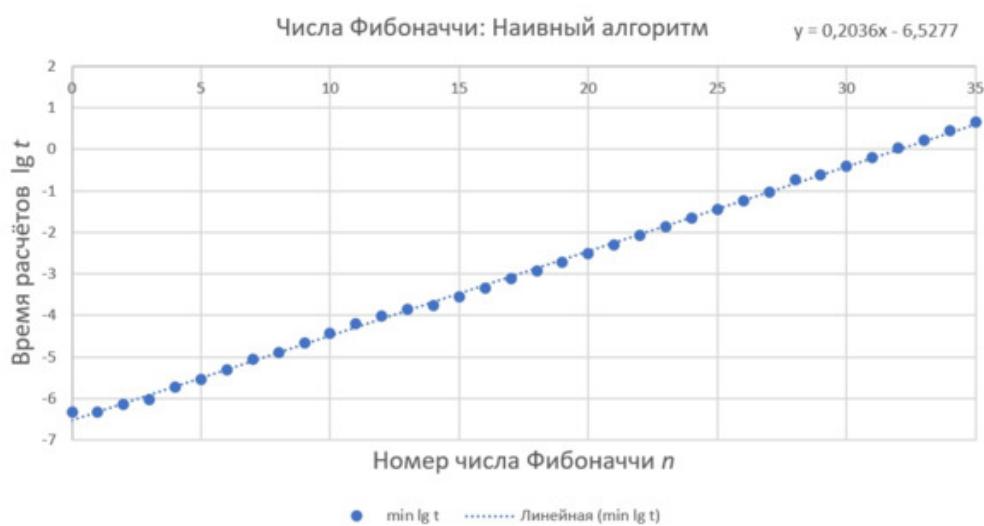


Рис. Оформление рисунка

Задание. Оформите рисунок таким образом, чтобы он легко воспринимался и содержал самую необходимую информацию. Можете использовать наш рисунок в качестве примера или сделать всё по-своему. В конечном счёте, за результат отвечает автор произведения, а не составитель инструкции или методички.

КРАСИВОЕ УРАВНЕНИЕ

Мы получили уравнение прямой линии для логарифма времени.

Теперь пора вернуться к исходным переменным.

Мы получим уравнение для времени вычислений t в зависимости от размера задачи, то есть от номера числа Фибоначчи n .

Оформим наши выкладки в новой текстовой ячейке Colab.

Нам дополнительно понадобится выполнить несложные расчёты с возведением числа 10 в нужную степень. Эта операция обозначается в Питоне двумя звёздочками.

Получается, что время расчётов пропорциональное 1,6 в степени, равной номеру числа n . Такая зависимость называется экспоненциальной.

<p>Уравнение линии на графике</p> $y = 0.2036x - 6.5277$ $\lg t = 0.2036n - 6.5277$ <p>Переходим от логарифма к исходным переменным</p> $t = 10^{-6.5277} \cdot (10^{0.2036})^n$ $t = 2.967 \cdot 10^{-7} \cdot 1.598^n$	<p>Уравнение линии на графике</p> $y = 0.2036x - 6.5277$ $\lg t = 0.2036n - 6.5277$ <p>Переходим от логарифма к исходным переменным</p> $t = 10^{-6.5277} \cdot (10^{0.2036})^n$ $t = 2.967 \cdot 10^{-7} \cdot 1.598^n$
<pre>[5] 10 ** -6.5277 → 2.966880130021001e-07</pre>	<pre>[5] 10 ** -6.5277 → 2.966880130021001e-07</pre>
<pre>[6] 10**0.2036 → 1.5980854594730916</pre>	<pre>[6] 10**0.2036 → 1.5980854594730916</pre>

Рис. Оценка уравнения

Задание. Получите выражение для зависимости времени вычислений t от номера числа n . Оформите выкладки в ячейках блокнота Colab.

ДЕРЕВО РЕКУРСИИ

Переходим к анализу нашего алгоритма.

Построим диаграмму, которая позволит нам определить количество обращений к нашей функции в процессе расчётов. Такую схему можно назвать деревом вызовов. И это характерно для рекурсии. Первые два базовых случая $F(0)$ и $F(1)$ потребуют сделать всего один вызов нашей функции $F()$. Для нахождения числа Фибоначчи под номером 2 потребуется сделать уже 3 вызова. Число под номером 3 потребует пять вызовов.

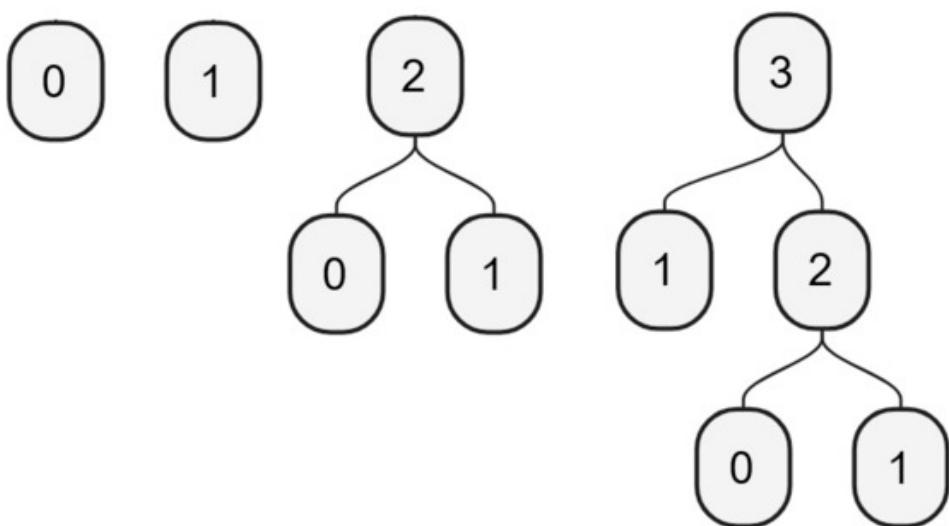


Рис. Дерево вызовов

Задание. Постройте диаграмму, как на рисунке выше. Добавьте ещё два дерева вызовов – для $F(4)$ и $F(5)$. Можете использовать уже изученные инструменты PlantUML и Mermaid. Найдите количество вызовов для этих случаев.

В процессе построения дерева мы обнаружили, что количество рекурсивных вызовов растёт, как снежный ком.

Фактически, каждый новый вызов $F(n)$ обращается к результатам двух предыдущих вызовов. Таким образом, количество вызовов $T(n)$ определяется рекурсивным соотношением, как показано на рисунке ниже.

Попробуем оценить скорость этого роста. Промоделируем рекурсивные вызовы в Excel. Будем предполагать, что время расчётов пропорционально количеству вызовов нашей функции.

Число рекурсивных вызовов $F(n)$:	Число рекурсивных вызовов $F(n)$:
$T(0) = 1$	$T(0) = 1$
$T(1) = 1$	$T(1) = 1$
$T(n) = T(n - 1) + T(n - 1) + 1$	$T(n) = T(n - 1) + T(n - 1) + 1$

Рис. Число вызовов функции

Задание. Составьте рекурсивное соотношение для $T(n)$ и опишите его в новой текстовой ячейке блокнота.

Мы исходим из предположения, что у нас будет экспоненциальный рост времени вычислений. Это означает, что отношение двух соседних значений будет постоянным, и оно будет равно основанию степени, см. рис.

Предполагаем экспоненциальный рост	Предполагаем экспоненциальный рост
$T(n) = C \cdot A^n$	$T(n) = C \cdot A^n$
Тогда	Тогда
$\frac{T(n)}{T(n - 1)} = \frac{C \cdot A^n}{C \cdot A^{n-1}} = A$	$\frac{T(n)}{T(n - 1)} = \frac{C \cdot A^n}{C \cdot A^{n-1}} = A$

Рис. Основание экспоненты

ЗОЛОТОЕ СЕЧЕНИЕ

Теперь переходим к моделированию вычисления $F(n)$ в Excel.

В первом столбце у нас будет номер числа $n = 0, 1, 2, 3\dots$

Во втором столбце – время вычислений $T(n)$. Сначала два первых базовых случая. Затем определяем время по рекурсивной формуле, построенной выше.

В третьем столбце будем определять отношение двух соседних значений $T(n) / T(n-1)$. Это будет оценка основания.

Как выясняется, отношение соседних чисел Фибоначчи тоже имеет особые свойства. Построим два дополнительных столбца: число Фибоначчи $F(n)$ под номером n и отношение соседей $F(n) / F(n-1)$.

Оба отношения достаточно быстро выходят на установившийся уровень 1.618.

Значение времени вычислений $T(n)$ и значение чисел Фибоначчи $F(n)$ различаются, но при этом их отношения очень похожи. Это одно и то же значение скорости роста экспоненты.

	D6	:	X	✓	f _x	=C6/C5
	A	B	C	D	E	F
1						
2		n	$T(n)$	$T(n) / T(n-1)$	$F(n)$	$F(n) / F(n-1)$
3		0	1		0	
4		1	1		1	
5		2	3		1	
6		3	5	1,6666667	2	2,000000
7		4	9	1,8000000	3	1,500000
8		5	15	1,6666667	5	1,666667
9		6	25	1,6666667	8	1,600000
22		19	13529	1,6181079	4181	1,618034
23		20	21891	1,6180797	6765	1,618034
24		21	35421	1,6180622	10946	1,618034

Рис. Предел отношения соседних значений

Задание. Промоделируйте в Excel рост времени вычислений $T(n)$ и рост чисел Фибоначчи $F(n)$. Найдите предельное значение, к которому стремится отношение двух соседних значений для обоих рядов.

Числа Фибоначчи – это достаточно хорошо исследованная последовательность чисел. Уже было доказано, к чему стремится отношение соседних чисел Фибоначчи. Это число «фи», его называют Золотое сечение или Золотая пропорция – Golden ratio.

Попробуем вывести формулу для «фи» и определить значение этой величины.

Решив квадратное уравнение, мы можем получить то самое число 1,618, которое появляется в наших оценках.

Золотое сечение (golden ratio)

$$F_n = F_{n-1} + F_{n-2}$$

$$\frac{F_n}{F_{n-1}} = \frac{F_{n-1}}{F_{n-2}} = \varphi > 0$$

$$\frac{F_{n-1} + F_{n-2}}{F_{n-1}} = \frac{F_{n-1}}{F_{n-2}}$$

$$1 + \frac{1}{\varphi} = \varphi$$

$$\varphi^2 + \varphi + 1 = 0$$

$$\varphi = ?$$

Рис. Постановка задачи

Задание. Решите приведённое на рисунке квадратное уравнение и определите формулу, по которой можно вычислить число «фи». Внесите свои выкладки в новую текстовую ячейку блокнота.

Поскольку отношение соседних значений T (n) должно приближаться к золотому сечению «фи», мы можем проверить наши оценки по предыдущим результатам.

Если мы возьмём время вычислений, которое мы измеряли в наших опытах и найдём отношение соседних значений, то мы можем с практической стороны подобраться к этому числу.

Задание. Вернитесь на предыдущую страницу своего отчёта. На основе результатов измерений постройте новую сводную таблицу. Определите минимальное время T (n) для каждого значения n . Скопируйте полученную сводную таблицу и вставьте содержимое как значения. Добавьте новый столбец, в котором определите отношение соседних значений времени. Постройте график, на котором можно будет увидеть постепенное приближение этого отношения к золотому сечению. Дополнительно нанесите на этот график горизонтальную линию, показывающую теоретический предел. Для этого можно создать две вспомогательные точки и соединить их прямой линией – это точечный график, на котором точки соединены прямыми отрезками.

На рисунке ниже показан пример того, что вам предстоит получить. Попутно, при выполнении этого задания, вы сможете более основательно освоить

электронные таблицы как инструмент расчётов и визуализации.

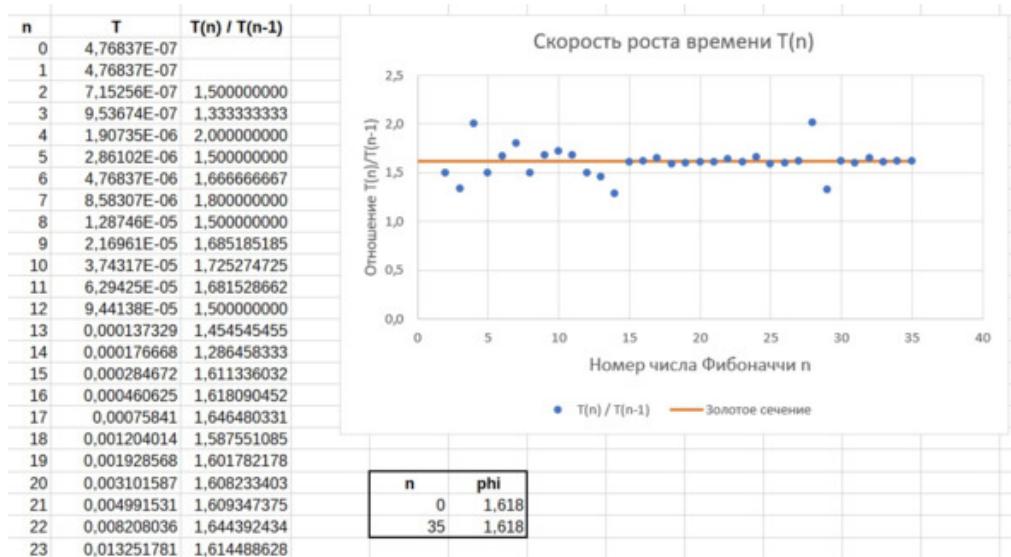


Рис. Предел отношения соседних значений $T(n)$

Если подходить к названиям более строго и серьёзно, то отношение текущего значения к предыдущему называют коэффициентом роста. Если это отношение выразить в процентах, тогда это будут темпы роста. Но можно, выражаясь разговорным языком, говорить про относительную скорость. А ещё лучше будет поставить кавычки – на всякий случай, если наш отчёт вдруг случайно попадётся на глаза специалисту в области статистики и анализа данных.

Задание. Проведите оценку относительной скорости роста для времени расчётов. Оформите график так, как показано на рисунке.

Подведём первые итоги. Мы составили программу, в точности соответствующую описанию последовательности чисел Фибоначчи. Мы сделали всё

именно так, как написано в Википедии и других источниках. Наши опыты показали, что время вычислений растёт экспоненциально. Наше решение задачи «в лоб» — это настолько неудачная программа, что мы не сможем продвинуться даже до числа под номером 40 или 50.

Экспонента, которые появляется при рекуррентном вызове двух экземпляров функции на каждом шаге, соответствует основному свойству чисел Фибоначчи. Это золотое сечение или золотая пропорция. Говорят, что это число появляется в живой и неживой природе, его можно найти в искусстве и архитектуре. Оно также имеет отношение к спиральной форме различных объектов и многим другим закономерностям.

Задание. Выясните, где применяют ряды чисел Фибоначчи, а также где обнаруживается золотое сечение.

НОТАЦИЯ «О БОЛЬШОЕ»

Главное свойство алгоритма — это продолжительность вычислений. Как время вычислений T растёт при увеличении размера задачи n ? Это свойство называют вычислительной сложностью.

В нашем примере мы исследовали и наблюдали экспоненциальную сложность. Время вычислений для нашей программы растёт по экспоненте с увеличением размера задачи. Мы даже выяснили основание степени, это особое число «фи» = 1,618.

Чтобы условно обозначить это свойство вычислительной сложности алгоритма и кратко его записать, используется система обозначений под

названием «Нотация О большое» – Big O Notation. То есть это большая, заглавная буква О. Первоначально это была первая буква немецкого слова *Ordnung*, что означает «порядок». Английское слово *order* (порядок) тоже начинается на букву О.

Слово «порядок» используется здесь в его специальном, математическом понимании. Это примерное, общее описание формы зависимости, внешний вид графика. «Порядок» в этом случае отвечает на вопрос: График похож на экспоненту или график похож на прямую линию?

Если график напоминает экспоненту, то в обозначении будет степень двойки: 2^n . Даже если основание степени – другое число. Главная идея – подчеркнуть, что это именно экспонента, а не прямая линия. Двойка выбрана для обозначения только потому, что у программистов часто используется двоичная система и очень популярны степени двойки.

Ещё раз напомним, что все наши объяснения очень приблизительные. Мы обсуждаем самые общие представления и понятия. После такого первого предварительного знакомства вы сможете почитать что-нибудь более основательное, например, официальные учебники по предмету.

Задание. Просмотрите статью Временная сложность алгоритма на Википедии и постройте таблицу по образцу, приведённому на рисунке выше. Дополните таблицу постоянной (константной), квадратичной и логарифмической сложностью. Заголовки можно позаимствовать из Википедии.

Идею вычислительной сложности алгоритма упоминает руководитель комиссии по разработке заданий для ЕГЭ по информатике С.С.Крылов. Видеозапись его выступления можно найти на сайте ФИПИ fipi.ru: Федеральный институт педагогических измерений – ЕГЭ – Видеоконсультации – Информатика. Судя по этому выступлению, выпускники средней школы должны иметь общее представление о проблеме вычислительной сложности и основных приёмах её решения. Эти знания и навыки требуются для решения некоторых задач и успешного прохождения ЕГЭ по информатике.

Задание. Изучите видеоконсультацию Сергея Крылова и обратите внимание на его комментарии по поводу сложности алгоритма. Какую именно вычислительную сложность он упоминает и для какой задачи? Кратко опишите его пояснения в своём отчёте.

Для того, чтобы закрепить понимание нотации « O большое», лучше всего построить графики. Мы зададим столбец значений n . Заполним его числами от одного до десяти. В следующей колонке будут значения времени, равные n . Эта колонка будет изображать из себя линейную сложность. Затем организуем колонку единиц. Это будет «константная» сложность. Аналогично заполним остальные столбцы для квадратичной и экспоненциальной сложности. По этим данным нужно будет построить совмещённый «точечный» график в виде плавных линий.

Задание. Постройте совмещённый график для стандартных видов вычислительной сложности.

Настройте масштаб по осям таким образом, чтобы можно было рассмотреть общую форму графиков.

МАССИВ ЗНАЧЕНИЙ

Вычисление по исходному алгоритму требует многократного вызова нашей функции с одними и теми же параметрами. Было бы разумно сохранить результаты вызова функции и использовать их повторно. Самое простое решение – создать массив или список. Здесь порядковый номер числа Фибоначчи (то есть аргумент функции) будет выступать в роли индекса элемента списка.

Такой подход в программировании называется «мемоизация». Можно сказать, что это разновидность кэширования вызовов функции.

Задание. Просмотрите статью Мемоизация на Википедии. Обратите внимание на происхождение этого странного названия. Выясните, каким способом сохраняют результаты предыдущих вызовов в рамках данного способа.

Наша новая программа будет достаточно простой. Её можно сгенерировать с помощью чат-бота и немного доработать вручную. Скорее всего, наш интеллектуальный помощник сгенерирует программу, которая будет проверять наличие искомого значения $F(n)$ в списке. При отсутствии значения программа переходит к вычислениям.

Задание. Сгенерируйте текст программы по описанию, приведённому выше. Сформулируйте подробное задание чат-боту. Убедитесь путём тестирования, что программа выдаёт правильные решения. Исследуйте поведение программы и постройте график зависимости времени вычислений от размера задачи.

В нашем случае мы заранее знаем схему вычислений. Поэтому мы можем заполнять наш список от начала к концу. Мы записываем два начальных значения 0 и 1, затем продвигаемся в сторону увеличения номера. Каждое новое полученное значение $F(n)$ добавляем в список. Продолжаем это движение до тех пор, пока не доберёмся до нужного числа под номером n .

При такой схеме вычислений можно ожидать линейную сложность алгоритма. С увеличением номера n нам нужно будет пропорционально увеличить количество операций. Соответственно, и время вычислений будет пропорционально n .

Когда мы говорим, что алгоритм имеет линейную сложность, это условно записывают таким образом:

$$T(n) = O(n).$$



Рис. Линейная сложность

Задание. Составьте программу, которая применяет мемоизацию, но при этом в процессе вычислений постепенно добавляет новые значения к списку. Постройте график по примеру, приведённому выше.

Мы получили программу, которая работает гораздо быстрее первоначальной версии. Теперь вместо экспоненциальной сложности мы наблюдаем линейную. Однако, для этого нам придётся хранить все предыдущие значения в виде списка или массива. И это означает, что теперь у нас линейно возрастает необходимый объём памяти. Если его обозначить буквой М (от английского слова Memory – память), можно будет записать так называемую пространственную сложность алгоритма:

$$M(n) = O(n).$$

Здесь «пространство» намекает на оперативную память, в которой размещаются значения переменных на время выполнения программы.

В зависимости от решаемой задачи и доступных вычислительных ресурсов, разработчик программы

старается улучшить то или иное свойство алгоритма: или время вычислений, или объём оперативной памяти, или обе характеристики одновременно.

КЭШИРОВАНИЕ ВЫЗОВОВ

Мы вручную запрограммировали сохранение результатов предыдущих вызовов нашей функции. Во многих языках программирования есть готовые инструменты для этой цели. В частности, в языке Python имеется модуль `functools`. Это набор инструментов для работы с функциями.

Нам потребуется `lru_cache`. Это декоратор, который указывают перед объявлением функции. Декоратор – это своеобразная «обёртка» вокруг нашей функции. Служит этот инструмент для изменения поведения нашей функции. При этом нам не потребуется вносить изменения в текст программы. Всё скрыто в этой «обёртке».

Можно сказать, что этот декоратор обеспечивает сохранение результатов предыдущих вызовов нашей функции. При очередном вызове он находит готовый результат и выдаёт его, не обращаясь к самой функции.

The screenshot shows a code editor with Python code and its documentation for the `lru_cache` decorator. The code is:

```
from functools import lru_cache

@lru_cache(maxsize=2000)
def f(n):
    if n < 2:
        return n
    else:
        return f(n - 1) + f(n - 2)

lru_cache?
```

The documentation pane shows:

- Signature:** `lru_cache(maxsize=128, typed=False)`
- Docstring:** Least-recently-used cache decorator.
- If `*maxsize*` is set to `None`, the LRU features are disabled and the cache can grow without bound.
- If `*typed*` is `True`, arguments of different types will be cached separately. For example, `f(3.0)` and `f(3)` will be treated as distinct calls with distinct results.

Рис. Кэширование функции

Мы можем вывести на экран описание — краткую справку по данному инструменту. Для этого в новой кодовой ячейке мы вводим интересующее нас название и ставим знак вопроса. После выполнения этой ячейки в правой части окна браузера выводится справка. Здесь описаны входные и выходные переменные и даётся расшифровка названия LRU.

Задание. Выясните, как работает организация кэша по принципу LRU (least recently used).

Проверяем, как ведёт себя наша улучшенная версия программы. Во-первых, убеждаемся в правильности расчётов. Во-вторых, нас интересует скорость вычислений.

После небольшого тестирования мы собираем весь код в одну ячейку и запускаем очередной эксперимент. Кстати, у нас есть возможность просматривать содержимое файла CSV не выходя из Колаба. Достаточно открыть облачный проводник и дважды щёлкнуть по нужной строчке в списке файлов. В правой части окна браузера открывается окно просмотра таблицы. Убеждаемся, что таблица выглядит правдоподобно. Даже заголовки столбцов в первой строке файла распознаны как заголовки.

The screenshot shows a Jupyter Notebook interface. On the left, there's a file tree with a red box around the 'lru.csv' file. In the center, cell [1] contains Python code for generating a CSV file named 'lru.csv'. Cell [2] shows the command to run the script and output the CSV file.

```
[1] %%writefile fib.py
from functools import lru_cache
@lru_cache(maxsize=2000)
def f(n):
    if n < 2:
        return n
    else:
        return f(n - 1) + f(n - 2)

from time import time

print("n,t")
for n in range(0, 100):
    for _ in range(3):
        t1 = time()
        x = f(n)
        t2 = time()
        print(f"{n},{t2 - t1}")

Writing fib.py
```

```
[2] !python fib.py > lru.csv
```

On the right, a CSV file viewer titled 'lru.csv' displays the following data:

n	
0	2.86102294921875e-06
0	1.1920928955078125e-06
0	2.384185791015625e-07
1	1.1920928955078125e-06
1	4.76837158203125e-07
1	2.384185791015625e-07
2	1.430511474609375e-06
2	2.384185791015625e-07
2	2.384185791015625e-07
3	9.5367431640625e-07

Show 10 per page

Рис. Просмотр результатов

Задание. Проведите описанный выше эксперимент и откроите полученный файл CSV для просмотра.

При решении одной из задач в рамках ЕГЭ по информатике ученики часто вынуждены использовать кэширование вызовов функции. Без использования этого инструмента программа будет работать слишком долго.

Задание. Выясните с помощью поиска в интернет, какое именно задание ЕГЭ по информатике требует использования кэширования. Изучите готовое решение для этой задачи. Попробуйте запустить эту программу без кэширования и с использованием кэширования. Обратите внимание на продолжительность вычислений.

ЗАГРУЗКА ЧЕРЕЗ POWERQUERY

Скачиваем файл CSV и загружаем его в Excel. В новой версии Excel загрузка данных из файла может отличаться. интерфейс немного другой, но смысл остается тот же. Обращаем внимание на разделители и на окошко предварительного просмотра.

После настройки нажимаем кнопку Load – Загрузить.

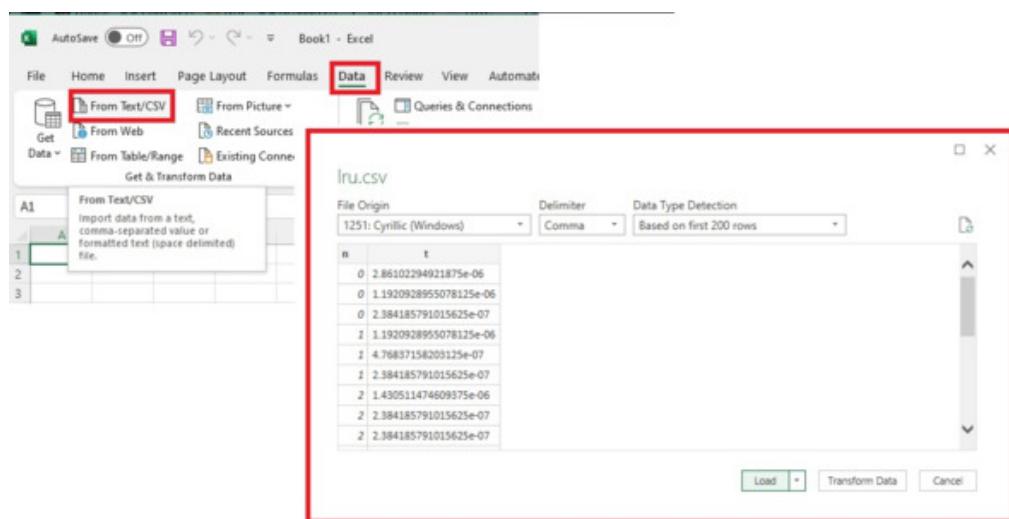


Рис. Загрузка файла CSV

К сожалению, действия по умолчанию могут привести к загрузке текста, а не числовых данных. В нашем примере не было настройки десятичного разделителя.

Повторим выбор файла, но вместо кнопки Load нажимаем Transform – Выполнить преобразование данных.

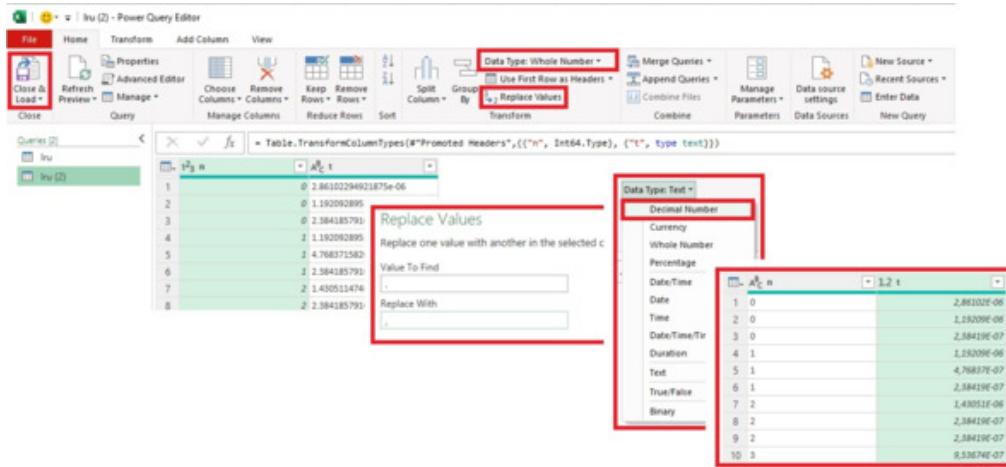


Рис. Преобразование данных

Придется почти вручную заменить точки на запятые и выбрать числовой тип данных: Replace Values – Value To Find – Replace With, Data Type – Decimal Number.

В первом столбце должно быть целое число – Whole Number.

Обращаем внимание на тип данных, который очень символично показан в заголовках столбцов:

Целое число – 1 2 3

Текст – А В С

Вещественное число – 1.2

В этом диалоговом окне Power Query Editor форматирование ячеек тоже намекает на тип данных: текст прижат влево, числа прижаты вправо.

После всех настроек нажимаем главную кнопку Close & Load – Закрыть окно и загрузить данные в Excel.

После загрузки данных можно наблюдать таблицу на рабочем листе. Данные из первой строки использованы как заголовки. Весь набор данных оформлен как один объект – Таблица Excel.

Строим диаграмму разброса по всему набору данных и изучаем график. Похоже, что программа стала работать слишком быстро. Здесь пока не просматривается общая тенденция.

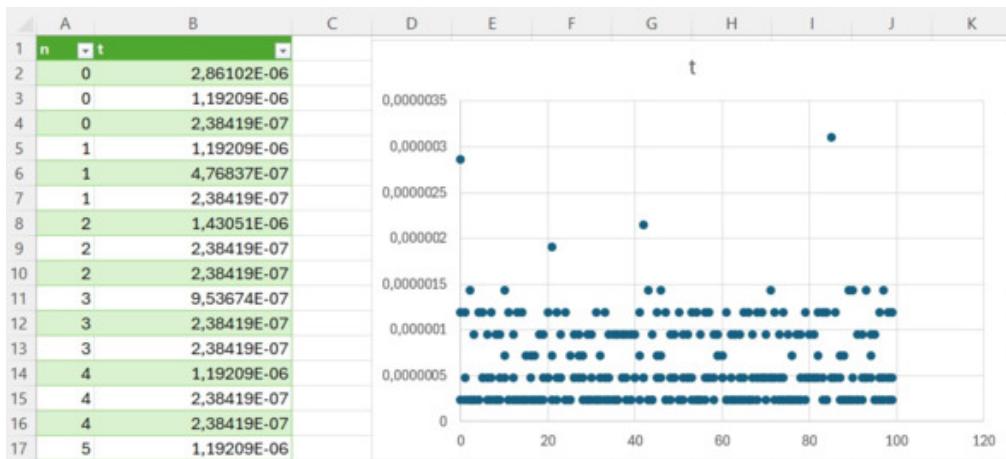


Рис. Результаты загрузки

Задание. Загрузите данные в Excel и постройте диаграмму разброса.

ГЛУБИНА РЕКУРСИИ

Чтобы всё-таки исследовать форму зависимости $T(n)$, попробуем увеличить размер задачи. Подбираем размер n так, чтобы по этим данным можно было построить информативный график и рассмотреть его форму.

Постепенно увеличиваем значение n и вначале получаем очень длинное число. Это число целое, но уж слишком много разрядов. Про длинные числа мы ещё поговорим. Продолжительность вычислений — доли миллисекунды.

Увеличиваем размер задачи и получаем сообщение об ошибке:

RecursionError: maximum recursion depth exceeded in comparison

Мы превысили максимально допустимую глубину рекурсии.

Это означает, что мы превысили максимальное количество вложенных вызовов, по умолчанию этот предел составляет около 1000.

```
t1 = time()
x = f(500)
x, time() - t1
(139423224561697880139724382870407283950070256587697307264108962948325571622863290691557658876222521294125,
0.00013256072998046875)

t1 = time()
x = f(1000)
x, time() - t1
-----
RecursionError: maximum recursion depth exceeded in comparison
Traceback (most recent call last)
<ipython-input-21-99105432ddaa4> in <cell line: 2>()
    1 t1 = time()
----> 2 x = f(1000)
    3 x, time() - t1
    -----
    ^ 479 frames
<ipython-input-1-6f9f90d88d20> in f(n)
    3 @lru_cache(maxsize=2000)
    4 def f(n):
----> 5     if n < 2:
    6         return n
    7     else:
RecursionError: maximum recursion depth exceeded in comparison
```

Рис. Глубина рекурсии

Мы можем увеличить допустимую глубину рекурсии с помощью команды `setrecursionlimit()`. Такое решение в обычной работе не рекомендуется. Но мы учимся, и нам эта настройка нужна только для наших экспериментов.

Подбираем глубину рекурсии побольше. Проводим опыты и убеждаемся, что программа работает «слишком быстро». В обычной жизни это хорошо. А вот для измерения времени вычислений – не очень.

Дальнейшее увеличение размера задачи приводит к более серьёзным ошибкам. Придется нам использовать что-то ещё.

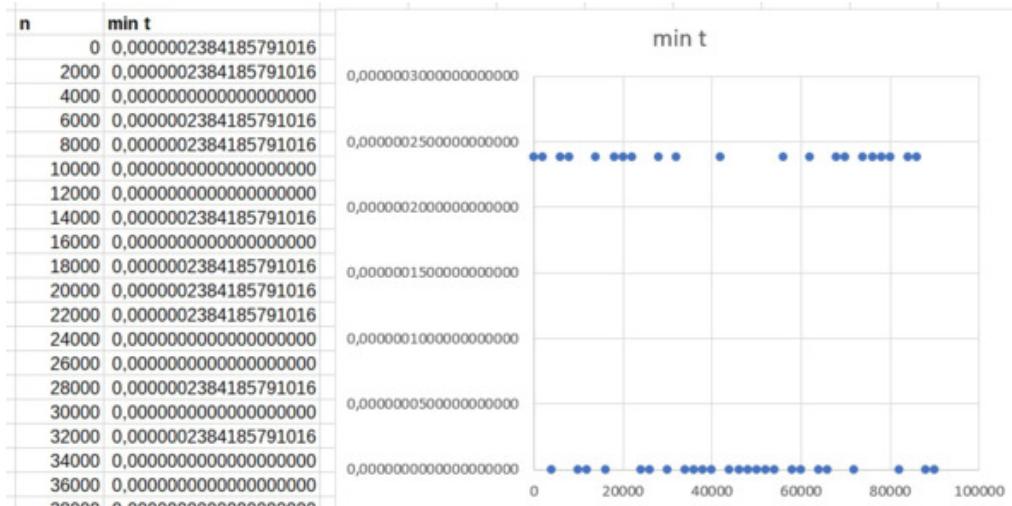


Рис. Работа на грани погрешности

Задание. Проведите описанный выше эксперимент, задавая разное значение допустимой глубины рекурсии. Увеличьте размер задачи до появления сообщений о новой ошибке. Ознакомьтесь с этой проблемой с помощью чат-бота.

Глубина рекурсии – это не просто очередная системная настройка. При каждом вызове функции интерпретатор использует часть оперативной памяти под названием «стек». При рекурсивном вызове можно довести ситуацию до переполнения стека и возникновения системной ошибки. Кроме того, кэширование значений функции увеличивает потребный размер памяти, и это нежелательно.

В конечном счёте, мы хотим найти значение функции за очень небольшое или приемлемое время и с небольшим расходом оперативной памяти. Это

идеальное поведение программы. И мы с вами попытаемся к этому идеалу приблизиться.

Поскольку требуется найти только значение $F(n)$, нам не обязательно будет хранить все предыдущие результаты. Это означает, что мы не обязаны создавать большой массив, или список, или кэш. И мы даже не обязаны использовать рекурсию. Обычно рекомендуют вместо рекурсии использовать что-нибудь ещё. Например, очень хорошо себя зарекомендовали вычисления в цикле.

ЦИКЛ ВМЕСТО РЕКУРСИИ

Если мы попросим нашего интеллектуального помощника сгенерировать самый простой и самый быстрый вариант программы для нахождения энного числа Фибоначчи, мы получим что-то действительно новое. Конечно, эта программа будет новой для нас. Для всего остального интернета – это будет вещь известная, а для кого-то даже очевидная. Мы же рассматриваем эту программу как инструмент для наших очередных опытов. И как пример того, что алгоритм расчётов не всегда дословно и буквально реализует исходную постановку задачи. Главное, чтобы результаты расчётов был правильным.

Если дополнительно попросить помощника не использовать мемоизацию, мы получим следующую программу, показанную на рисунке ниже.

The screenshot shows a GigaChat interface. At the top, there's a back arrow, the GigaChat logo, and a search bar with placeholder text "Измени программу таким образом запятая чтобы не использовать мемоизацию". Below this, a user message is shown with a profile icon and the text: "Конечно! Можно написать программу для нахождения n\$-го числа Фибоначчи без использования мемоизации. Однако стоит отметить, что такая версия будет медленнее для больших значений \$n. Вот простой вариант реализации:". A code block follows:

```
1 def fibonacci(n):
2     a, b = 0, 1
3     for _ in range(n):
4         a, b = b, a + b
5     return a
6
7 # Пример использования
8 n = 10
9 print(fibonacci(n))
```

Рис. Генерируем программу без мемоизации

Приведённый выше ответ получен с помощью нейросетевого помощника ГигаЧата (Sber). Дата обращения: 2 ноября 2024 года.

Обратим внимание на форматирование ответа с помощью символа доллара. После изучения предыдущей темы вы сможете догадаться, что в этом ответе можно было бы подправить, чтобы улучшить отображение на экране.

Программа действительно стала гораздо проще. Посмотрим, что покажут наши эксперименты.

Начинаем подбирать размер задачи точка следим за тем, как увеличивается время расчётов. Теперь мы добираемся до новых ограничений и новых сообщений об ошибках. На этот раз нам говорят:

`ValueError: Exceeds the limit (4300) for integer string conversion; use sys.set_int_max_str_digits () to increase the limit.`

Похоже, что очередное число Фибоначчи оказалось слишком длинным. даже стандартный интерпретатор питона не смог его «переварить».

```
t1 = time()
x = f(10000)
x, time() - t1
336476487643178326662161200510754331030214846068006390656476997468008144216666236815559551363373402558206533268083615937373
0.0038661956787109375

<

t1 = time()
x = f(100000)
x, time() - t1
-----
ValueError                                Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/IPython/core/formatters.py in __call__(self, obj)
    700         type_pprinters=self.type_printers,
    701         deferred_pprinters=self.deferred_printers)
--> 702         printer.pretty(obj)
    703         printer.flush()
    704         return stream.getvalue()

-----  
3 frames-----
/usr/local/lib/python3.10/dist-packages/IPython/lib/pretty.py in _repr_pprint(obj, p, cycle)
    698     """A pprint that just redirects to the normal repr function."""
    699     # Find newlines and replace them with p.break_()
--> 700     output = repr(obj)
    701     lines = output.splitlines()
    702     with p.group():

ValueError: Exceeds the limit (4300) for integer string conversion; use sys.set_int_max_str_digits() to increase the limit
```

Рис. Новая программа и новые ошибки

Задание. Исследуйте поведение новой программы и найдите границу её работоспособности. Просмотрите англоязычную страницу Википедии про числа Фибоначчи и выясните, как оценить количество разрядов очередного числа Фибоначчи (number of digits in F (n)).

Наше очередное ограничение в действительности связано с выводом результатов на экран. Если нас будет интересовать только время вычислений, мы можем ещё увеличить размер задачи. Убираем вывод числа Фибоначчи на экран и оставляем только вывод времени.

Дальше 10–20 секунд мы не будем забираться. Ведь нам предстоит провести многократный запуск программы с разными размерами задачи.

The screenshot shows three separate command-line executions. Each execution consists of a command followed by its output. The first two executions are identical, while the third is slightly longer.

Execution	Command	Output
1	[59] t1 = time() f(10000) time() - t1	0s 0.0036885738372802734
2	[57] t1 = time() f(100000) time() - t1	0s 0.1277143955230713
3	[58] t1 = time() f(1000000) time() - t1	12s 12.620739698410034

Рис. Подбираем размер задачи

Повторяем наш эксперимент. На этот раз мы добираемся до очень больших значений n . Повторяем основные шаги по записи файла и загрузки данных в Excel. Настраиваем сводную таблицу для определения минимальных значений времени расчетов. Строим график и видим, что на этот раз вычислительная сложность отличается от линейной. На этот раз мы столкнулись с растущими вычислительными затратами на обслуживание операций с длинными целыми числами. Насколько это будет важно при решении конкретной практической задачи, будет зависеть от масштабов решаемой задачи.

Возможно, мы и не дойдём до тех пределов, где начинаются настоящие вычислительные сложности. Асимптотические свойства алгоритма начинают проявляться при больших размерах задачи. На практике можно ограничиться не самыми лучшими алгоритмами, если время вычислений будет приемлемым.

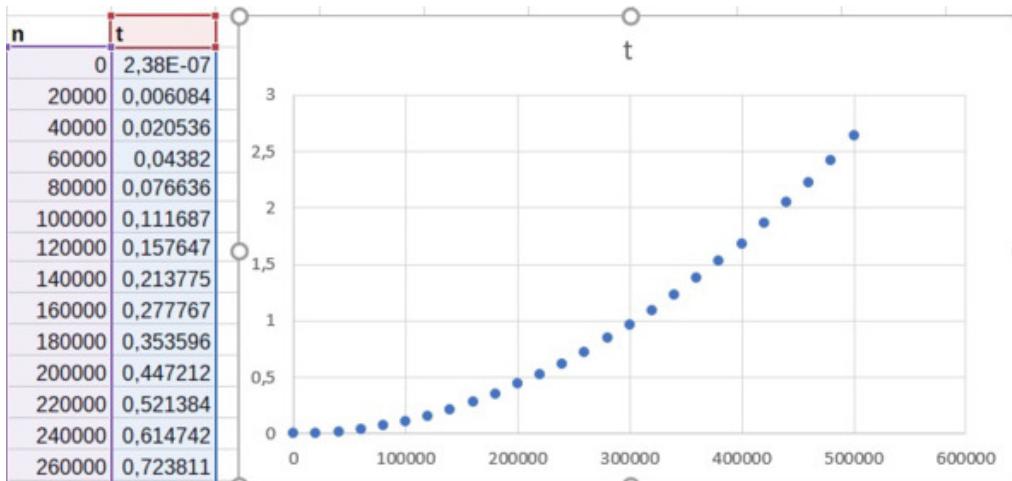


Рис. «Не очень прямая» линия

Задание. Проведите описанный выше эксперимент. Оцените характер зависимости (квадратичная или экспоненциальная). Выясните с помощью интеллектуального помощника, Какая вычислительная сложность ожидается в теории при работе с длинными целыми числами.

Мы с вами проделали ряд экспериментов в облаке на платформе Google Colab. Здесь не гарантируется время выполнения программы. Мы можем наблюдать ощутимый случайный разброс продолжительности работы нашей программы. На работу сервера может влиять обслуживание других задач других пользователей. Тем более, что мы в наших опытах используем бесплатный доступ. Многие облачные провайдеры предлагают минимальный или ограниченный функционал на бесплатном тарифе и более качественный набор услуг на платном варианте.

При запуске той же самой программы на локальном компьютере Случайный разброс результатов может быть гораздо меньше. Предлагаем вам в этом убедиться.

Задание. Проведите локальном компьютере все вычислительные эксперименты, начиная с самого первого. Постройте те же графики и обратите внимание на величину случайного разброса значений.

Наш основной отчёт оформляется в виде электронной таблицы. Здесь у нас должны быть тексты программ, результаты загрузки файлов CSV, графики и выводы. Поскольку часть работы производилась в Google Colab, нам нужно отразить этого отчёте. Желательно было бы приложить весь блокнот к отчету. Мы сделаем это через ссылку на блокнот, чтобы проверяющий мог с ним ознакомиться.

Задание. Откройте доступ на чтение к своему облачному блокноту. Вставьте на последней странице своего отчёта ссылку на облачный блокнот Google Colab.

Естественно, все программы, результаты экспериментов, полученные графики и выводы нужно привести в отчёте. В самом начале мы оформляли отчёт в облачной версии в Таблиц Яндекса. Затем мы скачали этот файл и продолжали работать на локальном компьютере в электронной таблице Excel. Окончательный вариант отчёта нужно разместить на облачном диске Яндекса и предоставить ссылку для чтения через форму на гитхабе. В этом случае у нас появляется возможность для просмотра отчёта через таблицы Яндекс, а также для скачивания этого файла и просмотра его на локальном компьютере. Обращаем ваше внимание, что облачный «офис» имеет ограниченный функционал, и не все элементы отчёта

могут правильно воспроизводиться при просмотре через браузер.

Задание. Загрузите свой файл формата XLSX на облачный диск Яндекс. Проверьте, как выводится на экран каждая страница отчёта при просмотре его через браузер. Сравните с внешним видом отчёта на локальном компьютере.

ЗАКЛЮЧЕНИЕ

В этом разделе мы познакомились с основным, самым важным свойством алгоритмов – вычислительной сложностью. Мы определили, как меняется время выполнения программы. Мы измерили это «явление природы» и построили соответствующие графики.

На примере нахождения чисел Фибоначчи мы с вами увидели, что решение задачи «в лоб» не всегда даёт приемлемые результаты. Мы также познакомились с тем, как можно улучшить работу программы за счёт небольшого изменения алгоритма расчётов. Попутно мы рассмотрели общую схему проведения эксперимента и обработки полученных данных: перенаправление вывода в файл, загрузка файла CSV, создание сводной таблицы, построение и настройка графиков.

Всё это – элементы современных информационных технологий. Это наша с вами компьютерная грамотность.

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

ССЫЛКИ

Fibonacci sequence

https://en.wikipedia.org/wiki/Fibonacci_sequence

CSV

<https://ru.wikipedia.org/wiki/CSV>

ASCII

<https://ru.wikipedia.org/wiki/ASCII>

Золотое сечение

https://en.wikipedia.org/wiki/Golden_ratio

«O» большое и «o» малое

https://ru.wikipedia.org/wiki/«O»_большое_и_«o»_малое

Теория алгоритмов

https://ru.wikipedia.org/wiki/Теория_алгоритмов

Вычислительная сложность

https://ru.wikipedia.org/wiki/Вычислительная_сложность

Временная сложность алгоритма

https://ru.wikipedia.org/wiki/Временная_сложность_алгоритма

Федеральный институт педагогических измерений

<https://fipi.ru/>

Онлайн-консультация «На все 100» по подготовке к ЕГЭ по информатике с участием Сергея Крылова, руководителя комиссии по разработке КИМ ГИА по информатике

https://vk.com/video-36510627_456239982

Мемоизация

<https://ru.wikipedia.org/wiki/Мемоизация>

functools – Higher-order functions and operations on callable objects

<https://docs.python.org/3/library/functools.html>

Модуль functools

<https://pythonworld.ru/moduli/modul-functools.html>

LRU

[https://en.wikipedia.org/wiki/Cache_replacement_policies#Least_Recently_Used_\(LRU\)](https://en.wikipedia.org/wiki/Cache_replacement_policies#Least_Recently_Used_(LRU))

ОБЛАЧНЫЕ ОТЧЕТЫ

Отчеты по всем выполненным работам мы оформляем в облаке.

Затем отправляем преподавателю на проверку ссылку на свой отчет. Ссылка сама по себе не занимает много места. Отчет лежит на вашем облаке.

Отчет по первой работе, в которой мы осваиваем интеллектуальных чат-ботов, оформляем в Документах Яндекса. Это пример текстового редактора. В текст мы вставляем картинки и используем форматирование – в разумных пределах.

Отчет по второй работе, в которой мы знакомимся с форматом Markdown, оформляется в виде интерактивного блокнота Google Colab. Это пример сочетания программного кода и форматированного текста. Причем форматирование задается дополнительными инструкциями в самом тексте.

Отчет по третьей работе, в которой мы создаем диаграммы с помощью кода, оформляем в виде странички на GitHub. Здесь мы закрепляем свои навыки работы в Markdown. В качестве упражнений мы составляем план своего обучения в рамках выбранной профессии.

Отчет по четвертой работе, в которой мы исследуем свойства алгоритмов, оформляем в Таблицах Яндекса. В него же вставляем ссылку на Блокнот Google Colab, в котором проводятся основные эксперименты с программами.

С каждым занятием материал становится сложнее. Первая и вторая работы относительно простые и занимают одно занятие. Третья и четвертая работы более объёмные. Для них отводится по два занятия. Каждое новое занятие основано на предыдущих темах. Поэтому лучше проходить данные материалы в той последовательности, как они здесь изложены.

Как видим, разные темы оказываются связаны между собой. Кроме того, сам процесс оформления отчетов разными средствами – это ещё один аспект обучения. Это ещё один набор полезных инструментов. Все рассмотренные темы развивают вашу способность осваивать новые программы и средства, новые правила и стандарты. В результате, способность учиться может оказаться самым ценным навыком из всего, чтобы вы здесь получите.

КАРТИНКА НА ОБЛОЖКЕ

Котик на обложке нашего пособия сгенерирован с помощью нейросети Кандинский 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. Но об этом мы поговорим в следующий раз.

БЛАГОДАРНОСТИ

Если вы успешно изучили материал данного пособия, у вас была возможность получить новые знания и науки, а также попрактиковаться в их применении.

Данное произведение составлено автором по его личной инициативе и в свободное от работы время.

Если вы посчитаете этот проект полезным и захотите его поддержать, у вас есть такая возможность. Вы можете приобрести данное пособие на сайте издательства Ридеро в разделе Магазин по цене читателя. Издательство оплатит подоходный налог и перечислит ваше пожертвование автору книги.

Можно также оставить отзывы на сайте издательства или на платформе, где вы познакомились с нашим произведением. Мы в любом случае продолжим нашу деятельность на благо тех, кто ещё не утратил желание и способность учиться. Следите за нашими выпусками.