



НТО ЮНИОРЫ 2026/27 · СФЕРА

ИИтех

Задачник-справочник для подготовки к
отборочному этапу

5-7 класс

Автор-составитель

Артём Осинowskiй, методист и разработчик детского технопарка «Кванториум»

Детский технопарк «Кванториум», Новосибирск · 2026

ИИтех

Задачник-справочник для подготовки к отборочному этапу НТО Юниоры. 5–7 класс

Автор-составитель – **Артём Осиновский**, методист и разработчик детского технопарка «Кванториум».

Нашли ошибку или неточность? Напишите на osinovsky.aa@kvantorium54.ru (рабочая почта): сфера, версия книги (1.0) и номер страницы – исправим в следующей версии. Актуальная версия и список изменений – nto.kvantorium54.ru/knigi. Личная почта для других вопросов – artem081033@gmail.com.

Важно. Это не официальное пособие олимпиады, а книга методиста «Кванториума». Каких заданий ждать на отборе 2026/27, мы не знаем: организаторы их заранее не показывают, а задания разных лет не повторяются. Книга готовит к темам из программы сферы и приёмам решения, а не к конкретным задачам.

Книга помогает школьнику самостоятельно подготовиться к отборочному этапу сферы «ИИтех» Национальной технологической олимпиады «НТО Юниоры»: теория с нуля, подробные разборы задач и задачи для самопроверки с ответами.

Большинство задач составлены автором по образовательной программе сферы, часть – по мотивам открытых заданий прошлых лет; источник указан у каждой задачи. Уровни сложности – шкала этой книги, а не баллы олимпиады. Правила, сроки и формат отбора – на сайте junior.ntcontest.ru.

Версия 1.0 от 02.10.2026, сборка 02.10.2026. © Артём Осиновский, 2026.

Содержание

Как устроена эта книга	4
Для кого эта книга	4
Как устроен отбор	4
Что пригодится на отборе и где это в книге	5
Как пользоваться книгой	6
Как вводить ответы на отборе	6
Что понадобится	7
План на месяц до отбора	34
Родителям и наставникам	35
Честно об этой книге	35
Глава 1 Что такое искусственный интеллект	36
1.1. Что такое ИИ и чем он не является	37
1.2. Машинное обучение: учимся на примерах	39
1.3. Типы задач: классификация, регрессия, кластеризация	41
1.4. Обучающая и проверочная выборка	43
1.5. Нейрон – веса с порогом	44
1.6. Точность и ошибка	47
1.7. Разбираем задачи	48
Глава 2 Множества и логика	60
2.1. Множество и его элементы	60
2.2. Пересечение и объединение: круги Эйлера	62
2.3. Считаем элементы: формула включений и исключений	64
2.4. Логика: И, ИЛИ, НЕ	66
2.5. Запросы к поисковику	68
2.6. Разбираем задачи	69
Глава 3 Комбинаторика и вероятность	79
3.1. Дерево возможностей и правило умножения	80
3.2. Правило сложения: «или» против «и»	82
3.3. Перестановки: кто за кем	83
3.4. Вероятность – доля подходящих исходов	85
3.5. Частота в эксперименте	87
3.6. Вероятность в ИИ: «модель уверена на 80 %»	89
3.7. Разбираем задачи	91
Глава 4 Закономерности, последовательности, уравнения	100
4.1. Ряды чисел: ищем правило	101
4.2. Арифметическая прогрессия: n-й член без перебора	103
4.3. Выражения с переменными	105
4.4. Линейные уравнения	106
4.5. Линейная зависимость: правило предсказания	108
4.6. Прямая через облако точек и ошибка прогноза	110
4.7. Разбираем задачи	112

Глава 5 Данные: собираем и наводим порядок	123
5.1. Данные, датасет и таблица	124
5.2. Типы данных: число, текст, да/нет, дата	125
5.3. Форматы файлов: CSV и JSON	127
5.4. Откуда берут данные и что такое API	129
5.5. Ошибки в данных и чистка	132
5.6. Фильтр, сортировка и подсчёт	134
5.7. Разбираем задачи	137
Глава 6 Графики и анализ данных	152
6.1. Четыре главных графика и когда какой	152
6.2. Как читать график: оси, единицы, масштаб	154
6.3. Числа-сводки: среднее, медиана, минимум, максимум, размах	156
6.4. Гипотеза: проверяем догадку графиком	159
6.5. Обманчивые графики	161
6.6. Дашборд и вычислительное эссе	162
6.7. Разбираем задачи	164
Глава 7 Нейросети своими руками	176
7.1. Из нейронов – сеть: слои и связи	177
7.2. Считаем выход сети	179
7.3. Как сеть учится: ошибка, веса, эпохи	181
7.4. Переобучение: выучил наизусть, а новое не узнаёт	184
7.5. Нейросеть без программирования	186
7.6. Разбираем задачи	188
Глава 8 Python, диалоги и промнты	199
8.1. Первая программа: переменные, ввод и вывод	199
8.2. Условия и циклы	202
8.3. Списки, словари и строки	205
8.4. Диалог – это граф	207
8.5. Промпт: как попросить нейросеть	211
8.6. Вайбкодинг: код от нейросети и его проверка	213
8.7. Разбираем задачи	214
Ответы и решения	226
Справочник	235
Формулы и правила	235
Мини-шпаргалка Python	238
Словарик	240
Шпаргалка: всё «Запомни» из глав	242
Где что найти	248

Как устроена эта книга

Эта книга – тренажёр для отборочного этапа сферы «ИИтех» олимпиады «НТО Юниоры». Здесь есть всё, чтобы готовиться самостоятельно: теория с нуля, подробные разборы задач и похожие задачи для самопроверки с ответами в конце книги. Мощный компьютер и знакомый программист не нужны: половина задач решается на бумаге, остальное – в браузере и в бесплатных программах сферы.

Для кого эта книга

- Для школьников **5–7 классов**, которые выбрали (или думают выбрать) сферу «ИИтех».
- Для тех, кто **ни разу не программировал** и не знает, как устроена нейросеть. Каждая тема начинается с самого начала, а первый код мы разбираем построчно.
- Для **наставников и родителей**, которые хотят помочь. Для них есть отдельный раздел в конце введения.

Как устроен отбор

Таблица В.1

Когда	Что происходит
до 9 ноября	регистрация на олимпиаду (наш совет – успеть до 20 октября, тогда будут обе попытки)
20–30 октября	отбор, попытка 1 (3 часа)
31 октября – 10 ноября	отбор, попытка 2 (засчитывают лучшую из двух)
7–17 декабря	финалы в регионах

- Отбор проходит **дома, онлайн, с компьютера**. Задания открываются в личном кабинете на my.ntcontest.ru и решаются на платформе «Орбита» (orbita.education). Каждый решает сам за себя.
- Задания – это задачи с ответом: ввести число, выбрать верные варианты, сопоставить, расставить шаги по порядку, прочитать график или таблицу и сделать вывод.
- Разработчик сферы на ознакомительном вебинаре 2026 года сказала: в образовательном курсе изложен весь материал, который ждёт на отборе и в финале, и ничего нового на отборе не будет. Поэтому книга построена по программе сферы «Основы искусственного интеллекта и анализа данных».
- На вебинаре для наставников в сентябре 2026 года разработчики описали путь, по которому идут и курс, и отбор, и финал: понять задачу, записать её на языке математики, получить данные, исследовать их, построить модель и оценить, насколько хорошо она работает. Главы книги идут в том же порядке.
- В курсе, на отборе и в финале сферы используются инструменты, где программу собирают **из блоков**, без набора кода. Писать на Python с нуля от тебя не потребуют. Но

Python прямо назван на странице сферы в списке того, что пригодится, а в блочном инструменте курса за каждым блоком стоит строчка на Python. Поэтому в книге есть глава про Python.

- За тест бесплатного курса своей сферы дают **+5 баллов**. Курс – на academy.junior.online, вход через «Талант». Бонус нужно включить самому в личном кабинете («Мои привилегии») до конца отбора. Учти: после этого сферу уже не поменять.
- Финал – **командный** и очный: команда из двух-трёх человек около трёх часов решает задачи на компьютерах, на той же «Орбите». Команду можно собрать заранее, а можно прямо на месте.

Совет. Сроки и правила могут уточняться. Главный источник – сайт олимпиады junior.ntcontest.ru и памятка для Новосибирской области nto.kvantorium54.ru.

Что пригодится на отборе и где это в книге

На странице сферы перечислено, что пригодится на отборе. Вот где это в книге:

Таблица В.2

Что пригодится на отборе	Глава
Базовые понятия ИИ на уровне определений	1. Что такое искусственный интеллект
Математика: множества и логика	2. Множества и логика
Математика: комбинаторика (и вероятность – из программы сферы)	3. Комбинаторика и вероятность
Математика: уравнения, выражения с переменными; поиск закономерностей	4. Закономерности, последовательности, уравнения
Информатика: информация и информационные процессы	5. Данные: собираем и наводим порядок
Визуализация данных, графики по таблицам, умение читать графики и делать выводы	6. Графики и анализ данных
Нейросети: классификация, регрессия, точность модели (из программы сферы)	7. Нейросети своими руками
Основы программирования (Python); обработка текстовой информации	8. Python, диалоги и промпты

Главы 1–4 можно читать в любом порядке, для них хватит тетради и карандаша. Главы 5, 6 и 7 лучше читать подряд: в пятой мы собираем данные и наводим в них порядок, в шестой смотрим на них глазами, в седьмой учим на них нейросеть. Главу 8 можно начать когда угодно. Если ты ни разу не писал программ, загляни сначала в мини-шпаргалку Python в конце книги: код встречается уже в главах 5 и 6.

Как пользоваться книгой

В каждой главе теория и разборы идут вперемешку: за каждым разбором сразу следуют две похожие задачи «Реши сам» – чтобы закрепить приём, пока он свежий в голове, не долистывая до конца главы.

1. **Теория.** Коротко и по делу, с рисунками. Каждая новая тема начинается с большой картинки «на пальцах», а самое важное собрано в рамках «Запомни». Все «Запомни» из всех глав повторяются в конце книги, в шпаргалке.
2. **Разбираем задачи.** Задачи такие же, как на отборе, и к каждой – решение по шагам: что дано, что делаем и почему, где обычно ошибаются, какой ответ получился и похож ли он на правду.
3. **Реши сам.** Сразу после разбора – две похожие задачи для самостоятельного решения, тем же приёмом, но с другими числами.
4. **Ответы** – в конце книги, в разделе «Ответы и решения». Там только ответ, иногда – короткая подсказка, если задача с подвохом; полного решения нет специально: если подсказки не хватило, вернись к разбору перед задачей.

Не готов читать всю главу целиком? Есть отдельная короткая книжка «Быстрый старт» (около 50 страниц): из каждой главы главное, один разбор и несколько задач «Реши сам». Её удобно пройти за неделю, а эту книгу держать под рукой.

У каждой задачи есть уровень – как в обычном задачнике: у трудных задач звёздочка рядом с номером («задача 3*»), лёгкие подписаны словом «разминка». Отдельных значков и таблицы уровней в книге нет.

Звёздочка и «разминка» – это уровни нашей книги, а не баллы олимпиады. Сколько задач будет на отборе и сколько баллов стоит каждая, заранее не объявляют.

Совет. Разбор читай с листком бумаги. Закрой решение, попробуй решить сам, а потом сверяйся по шагам. Так ты запомнишь в три раза больше, чем если просто прочитаешь.

У каждой задачи внизу написано, откуда она взялась: «по мотивам задачи отборочного этапа» – значит, похожая задача была на отборе прошлых лет (мы пересказали её своими словами и поменяли числа), «своя задача по программе сферы» – значит, мы придумали её по теме из программы 2026/27 года. Программа сферы сама предлагает такой приём: решить знакомую задачу с другими входными данными. Если ты справился с задачей из разбора, а потом с обеими задачами «Реши сам», – тему ты понял.

Как вводить ответы на отборе

Так было на отборочных этапах прошлых лет. На «Орбите» перед каждой задачей читай инструкцию: там сказано, в каком виде нужен ответ.

- **Только число**, без единиц измерения и без знака процента, если не сказано иначе.
- **Дробную часть** на сайте обычно отделяют **точкой**: 0.85 , а не $0,85$. В книге мы пишем по-русски, через запятую.

- **Доля или проценты?** Точность модели 85 % можно записать как 85 и как 0.85 . Смотри, что просит условие.
- **Округляй так, как сказано в условии.** «До сотых» – два знака после точки. Промежуточные результаты не округляй – только ответ.
- **«Сколько способов»** – всегда целое число. Если получилось 12,5 варианта, ошибка где-то раньше.
- **У каждого могут быть свои числа.** В некоторых задачах условие или набор данных генерируется для каждого участника отдельно, так что списать ответ не выйдет.
- **Попытки ограничены.** Сначала реши на бумаге или проверь программу в онлайн-Python, и только потом вводи ответ.

Что понадобится

- Тетрадь в клетку, карандаш, линейка – для глав 1–4, кругов и таблиц.
- Калькулятор. Хватит обычного на телефоне: проценты, дроби, умножение.
- Компьютер с интернетом и браузером – для глав 5–8. Планшета мало: программы курса ставятся на компьютер (в программе сферы указан Windows 10 или новее). GraphNet и Датавиз-конструктор выпускают только для Windows.
- **Python** – любой онлайн-интерпретатор, ничего ставить не нужно. Для глав про данные и графики удобнее **Google Colab** (colab.research.google.com): там уже есть библиотеки для таблиц и графиков. Полистать готовые примеры можно и так, а чтобы запустить код, нужен аккаунт Google.
- **Сервисы из программы сферы**, по мере того как до них дойдёт курс:
 - **GraphNet** – конструктор нейросетей из блоков;
 - **Датавиз-конструктор** – графики, дашборды и «вычислительные эссе»;
 - **Teachable Machine** и **Machine Learning for Kids** – модели, которые учатся на твоих картинках и звуках без программирования;
 - **CodeChef** (codechef.ru) – блочная среда, в которой в курсе сферы собирают программу сбора данных: блоки HTTP, Pandas, списки, переменные; проект сохраняется файлом «.chef»;
 - **Scratch** – дополнительный материал кружка, не часть курса: в нём можно встроить обученную модель в свою программу;
 - ИИ-чаты **GigaChat** и **Алиса** – для главы про промпты (со взрослым, см. ниже).

Где скачать GraphNet и Датавиз-конструктор и как их поставить, рассказывают в курсе сферы – ссылки собраны в справочнике в конце книги.

CodeChef: программа из блоков

CodeChef (codechef.ru) – бесплатный конструктор программ: их не печатают, а собирают из цветных блоков, как из деталей. Справа тут же виден настоящий код на Python, который получился. В курсе сферы в CodeChef собирают программу сбора данных (модуль 5 «Данные», шаг 5 и задачи 4–5). Ниже – как начать и четыре задачи: от «привет» до сбора таблицы по страницам. Рамка с номером на картинке – куда смотреть и куда нажимать.

Шаг 1. Открыть. Зайди на codechef.ru: страница открывается сразу, входить и регистрироваться не нужно. Python работает прямо в твоём браузере (мы проверяли в Chromium – на

нём построены Chrome, Яндекс Браузер и Edge). Он загружается при открытии страницы, это несколько секунд; во время запуска вверху справа горит «Выполняется».

Шаг 2. Окно программы.

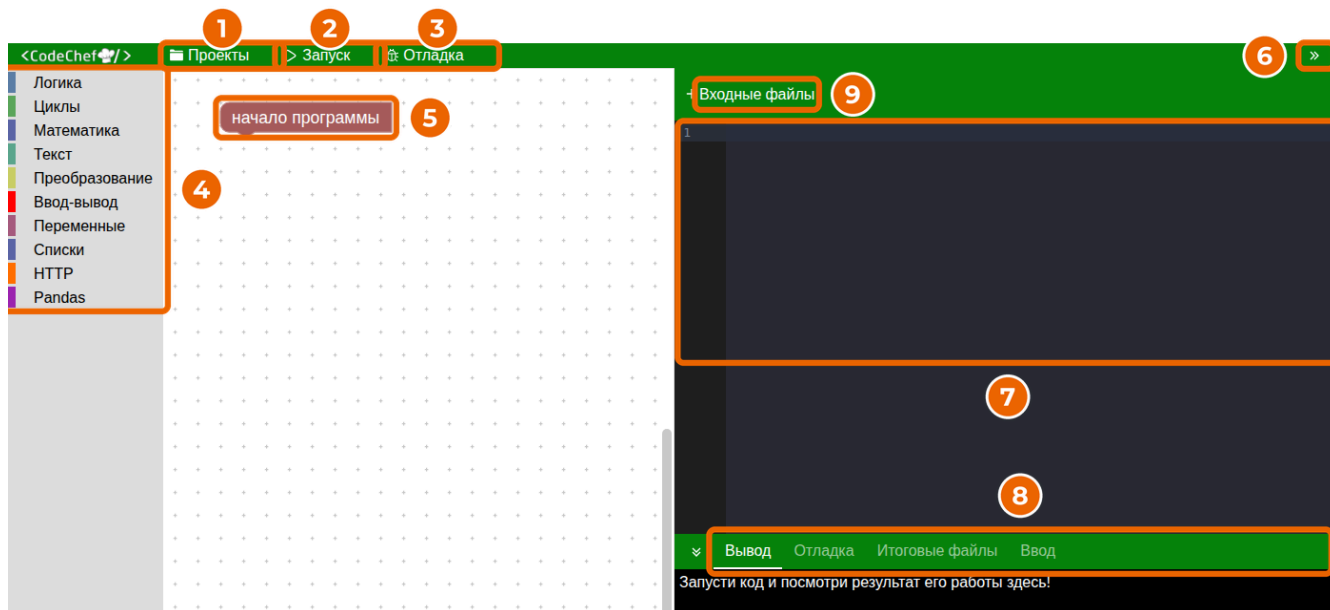


Рисунок В.1 – Окно CodeChef

1. **«Проекты»** – меню: «Открыть», «Создать проект», «Сохранить» (шаг 6).
2. **«Запуск»** – запустить программу. Что она напечатала, видно на вкладке «Вывод» (8).
3. **«Отладка»** – запуск по шагам: видно, как меняются переменные (задача П.3).
4. **Палитра блоков**, десять групп: «Логика», «Циклы», «Математика», «Текст», «Преобразование», «Ввод-вывод», «Переменные», «Списки», «HTTP», «Pandas». Щёлкни по названию – рядом откроются блоки группы.
5. **Рабочее поле**. На нём уже лежит блок «начало программы» – с него всё начинается. Блок, который ни к чему не прицеплен, в код не попадает.
6. **Двойная стрелка** – спрятать правую часть окна (код и вывод) или показать её снова. Когда блоки не помещаются в ширину, спрячь её: поле станет во всё окно.
7. **Код на Python**. Он сам меняется вместе с блоками; печатать в нём нельзя.
8. **Вкладки**: «Вывод» – что напечатала программа; «Отладка» – переменные при пошаговом запуске; «Итоговые файлы» – файлы, которые создала программа; «Ввод» – ответы на вопросы программы.
9. **«Входные файлы»** – добавить с компьютера файл `.txt`, `.json` или `.csv` для программы. В задачах ниже не нужен.

Шаг 3. Как ставить блоки. Открой группу, например «Ввод-вывод» (1 на левой половине), и перетаски блок мышью – берись за его левый край, а не за поле внутри. Блок-команду (с выемкой сверху и выступом снизу) подноси к нижнему краю предыдущего: он защёлкивается. Блок-значение (с выступом слева – число, текст, переменная, вычисление) вставляют в белое гнездо другого блока.

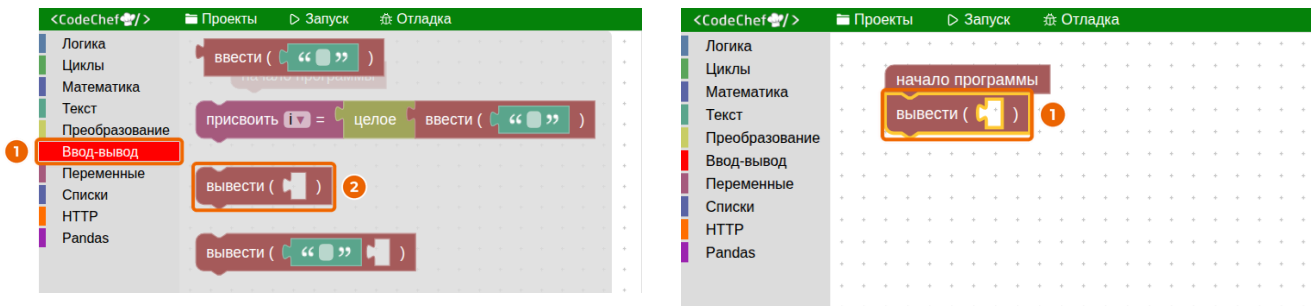


Рисунок В.2 – Слева – группа «Ввод-вывод» и блок «вывести» (2), справа – блок защёлкнутся под «началом программы» (1)

- Число или слово в блоке меняют щелчком по полю: щёлкни, впиши, нажми Enter. Стрелка ▼ открывает список – выбери нужное.
- Блок не защёлкнут – он чуть отъезжает и висит сам по себе. Перетащи его ещё раз.
- Блок тянет за собой всё, что под ним. Лишний блок: щёлкни по нему и нажми Delete.
- Поле двигают мышью за пустое место. Если в углу всплыла английская подсказка «Use → to navigate inside of blocks» – закрой её крестиком, на программу она не влияет.

Шаг 4. Переменные. Переменная – это коробочка с именем, в которую программа кладёт число, слово или список. В группе «Переменные» нажми «Создать переменную...», впиши имя и нажми Enter.

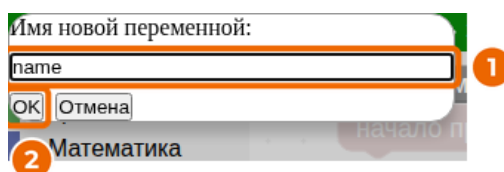


Рисунок В.3 – Окно создания переменной: имя (1), OK (2)

Имя пиши латиницей: `name`, `total`, `marks`. Русские буквы CodeChef превращает в код вида `_D0_BE_D1_86...`: программа работает, но читать её и смотреть переменные при отладке невозможно. В группе три блока: «присвоить ... =» (положить значение), «увеличить ... на» и блок с самим именем (взять значение).

Шаг 5. Ввод и запуск. Ответы на вопросы программа берёт не из окна, а с вкладки «Ввод»: впиши их заранее, по одному в строке (1 – вкладка, 2 – поле). Потом нажми «Запуск» вверху и смотри результат на вкладке «Вывод».

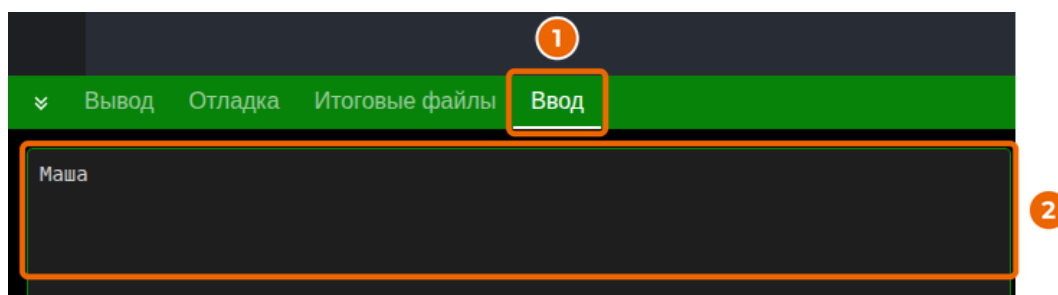


Рисунок В.4 – Вкладка «Ввод»

Если ответов меньше, чем вопросов, программа остановится с сообщением `E0FError: EOF when reading a line` – допиши строки. Если там, где нужно число, стоит слово, будет

`ValueError: invalid literal for int()` – впиши цифры. Зависшую программу (например, бесконечный цикл) останавливает крестик вверх справа, рядом с надписью «Выполняется».

Шаг 6. Сохранить и открыть. Всё, что ты собрал, живёт только на этой странице: обновил её или закрыл вкладку – поле пустое. Поэтому сохраняйся.

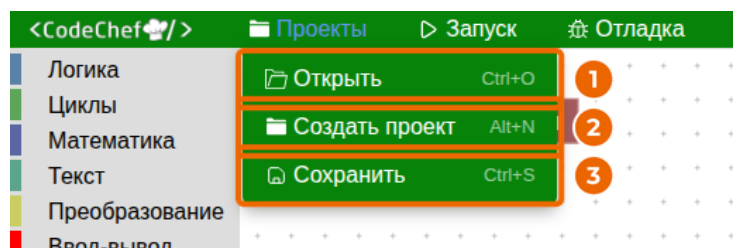


Рисунок В.5 – Меню «Проекты»: открыть (1), создать (2), сохранить (3)

«Сохранить» (3) скачивает файл `project.chef` : переименуй его, чтобы не перепутать с другими. «Открыть» (1) просит выбрать такой файл – так открывают и проект, который дают в курсе. «Создать проект» (2) очищает поле и сначала переспрашивает: «Все несохранённые изменения будут потеряны».

Задача П.1. Привет. Программа спрашивает имя и здороваются. Создай переменную `name` . Из «Переменных» возьми «присвоить ... =», поставь под «начало программы» и выбери в списке `name` . Из «Ввод-вывод» возьми верхний блок «ввести ()» и вставь его в гнездо справа от «=»; в кавычках впиши вопрос `Как тебя зовут?` (пробел в конце оставь, чтобы ответ не слипся с вопросом). Под ним поставь блок «вывести» с текстом и пустым гнездом, впиши `Привет,` , а в пустое гнездо вставь блок `name` из «Переменных».

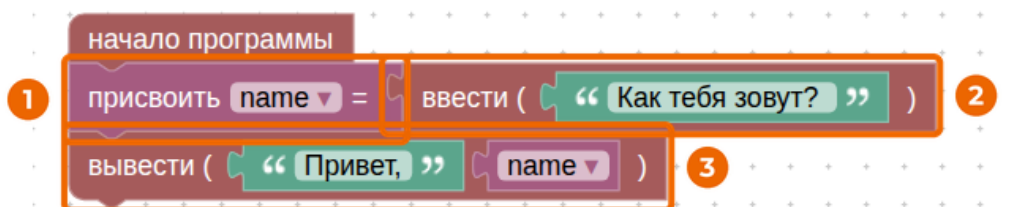


Рисунок В.6 – Программа П.1: переменная (1), вопрос (2), ответ на экран (3)

Открой вкладку «Ввод», впиши любое имя и нажми «Запуск».

```

1 name = None
2
3
4 name = input('Как тебя зовут? ')
5 print('Привет, ', name)

```

Вывод: Как тебя зовут? Маша
Привет, Маша

Рисунок В.7 – Результат: справа в коде – те же команды (1), в выводе – вопрос с твоим ответом и приветствие (2)

Что должно получиться: в «Вывод» две строки – Как тебя зовут? Маша (твое имя после вопроса) и Привет, Маша. В коде справа – `name = input('Как тебя зовут? ')` и `print('Привет, ', name)`. Впиши другое имя и запусти ещё раз: вторая строка изменится.

Задача П.2. Оценки: сколько, сумма, среднее. Программа просит три оценки, складывает их в список и считает. Создай переменные `marks` (список), `total` (сумма) и `x` (очередная оценка). Собирай по кадру сверху вниз.

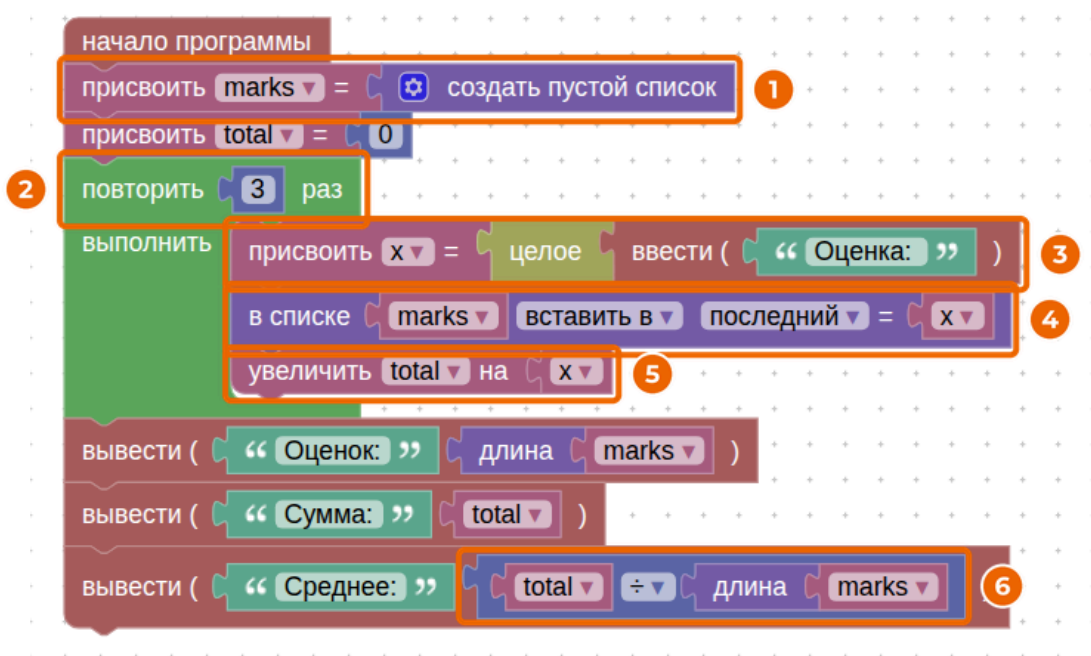


Рисунок В.8 – Программа П.2

1. «присвоить `marks` =» и блок «создать список из» из группы «Списки». Нужен пустой список, поэтому нажми на шестерёнку и потяни верхний «элемент» в правой части

влево, в серое поле: вместе с ним уйдут и нижние, а надпись станет «создать пустой список».

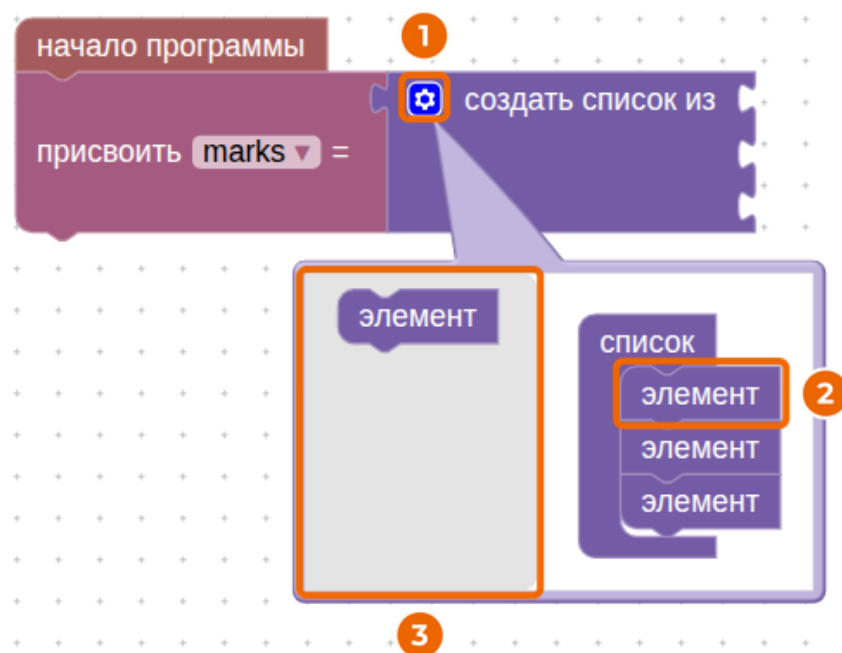


Рисунок В.9 – Шестерёнка: сама шестерёнка (1), «элемент», который тянут (2), серое поле, куда его тянут (3)

Под ним – «присвоить `total` =» с числом 0 из «Математики». (2) Из «Циклов» – «повторить ... раз» и число 3 (первый блок «Математики» – впиши в него 3). Внутри, в «выполнить», ставь по очереди три блока. (3) Из «Ввод-вывод» – второй блок, «присвоить ... = целое ввести ()»: выбери в нём `x` и впиши вопрос `Оценка: .` Слово «целое» превращает ответ в число. (4) Из «Списков» – «в списке ... присвоить № ... =»: выбери «вставить в» и «последний», слева поставь `marks`, справа `x` – оценка ляжет в конец списка. (5) «увеличить `total` на»: выбери `total`, а число 1 замени блоком `x`.

После цикла – три блока «вывести». Первый: текст `Оценок:` и блок «длина» из «Списков» со списком `marks`. Второй: `Сумма:` и `total`. Третий (6): `Среднее:` и блок «+» из «Математики» – выбери в нём `÷`, слева поставь `total`, справа «длину» списка `marks`.

На вкладке «Ввод» впиши три строки: `5`, `4`, `3` – и нажми «Запуск».

```

1  from numbers import Number
2
3  marks = None
4  total = None
5  x = None
6
7
8  marks = []
9  total = 0
10 for count in range(3):
11     x = int(input('Оценка: '))
12     marks.append(x)
13     total = (total if isinstance(total, Number) else 0) + x
14     print('Оценок:', len(marks))
15     print('Сумма:', total)
16     print('Среднее:', total / len(marks))

```

Вывод:

```

Оценка: 5
Оценка: 4
Оценка: 3
Оценок: 3
Сумма: 12
Среднее: 4.0

```

Рисунок В.10 – Результат П.2: вывод (1) и тот же код на Python (2)

Что должно получиться: Оценок: 3, Сумма: 12, Среднее: 4.0. Попробуй 5, 5, 4: сумма 14, а среднее – 4.666666666666667. Длинный хвост – это нормально, округлять здесь нечем. Проверь себя: $5 + 4 + 3 = 12$, $12 : 3 = 4$.

Задача П.3. Копилка. В копилке 0 рублей, каждый день в неё кладут 15. Через сколько дней накопится хотя бы 100? Создай `money` и `days`, обеим дай значение 0 («присвоить ... =», число 0). Из «Циклов» возьми «повторять, пока» и вставь в его гнездо блок сравнения из «Логики»: выбери в нём знак `<`, слева `money`, справа число 100. Внутри цикла поставь два блока «увеличить»: `money` на 15 (число 1 замени на 15) и `days` на 1. После цикла выведи Дней: с `days` и В копилке: с `money`.

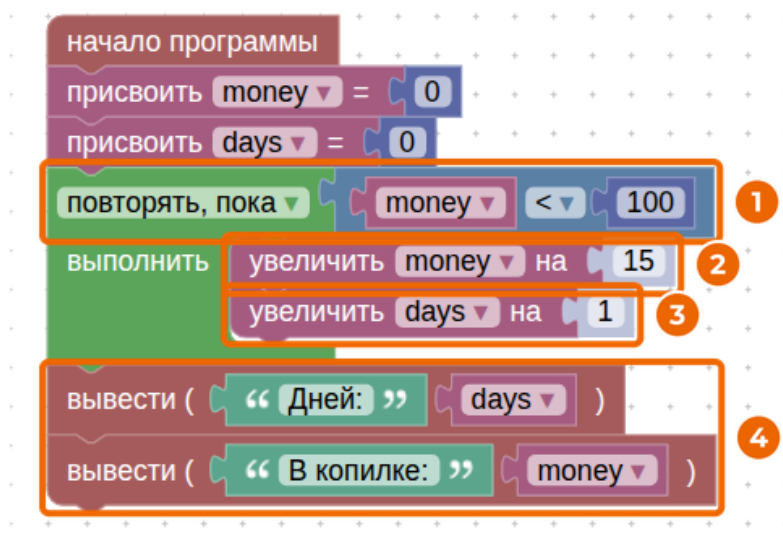


Рисунок В.11 – Программа П.3: условие (1), копим деньги (2), считаем дни (3), ответ (4)

Что должно получиться: Дней: 7 и В копилке: 105. Проверь пальцем: за 6 дней накопится 90 – это ещё меньше 100, значит нужен седьмой. Если вместо 15 по ошибке оставить 0, копилка не растёт и цикл не кончится никогда – останови его крестиком сверху справа.

Теперь посмотри, как цикл крутится. В узкой колонке слева от кода, правее номеров строк, щёлкни напротив строки `days = ...` – появится красная точка (1): здесь программа остановится. Нажми «Отладка» вверху и открой вкладку «Отладка»: в таблице будут `money` и `days` (2). «Продолжить» (3) пускает программу до следующей остановки, «Шаг вперед» – на одну строку, «Завершить» – конец. Строка `Number` в таблице служебная, её не читай. Крестик (4) тоже останавливает программу.

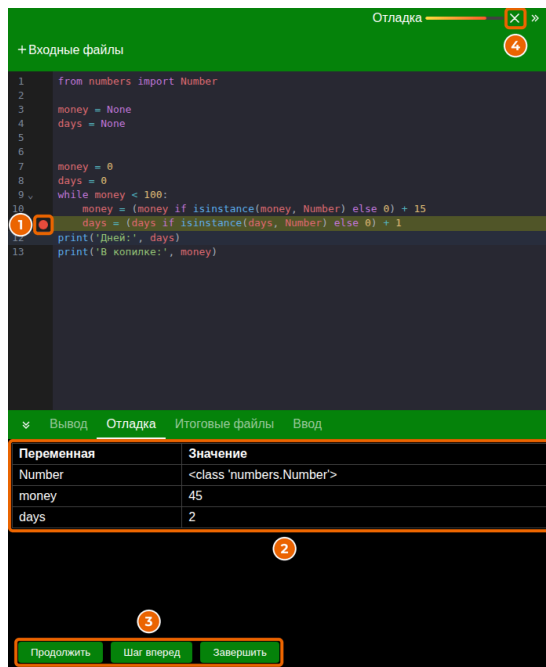


Рисунок В.12 – Отладка: точка останова (1), таблица переменных (2), кнопки (3), крестик (4)

Что должно получиться: при первой остановке `money` равно 15, а `days` – 0 (до `days` программа ещё не дошла). После двух «Продолжить» – `money` 45 и `days` 2: деньги растут по 15 за круг, дни – по 1.

Задача П.4. Таблица по страницам. Сервер отдаёт 200 записей страницами по 50. Программа обойдёт все четыре страницы, склеит их в одну таблицу и сохранит файл. Так же устроена программа курса, только там страниц больше и другой адрес. Для тренировки возьмём открытый учебный сервер с адресом <https://jsonplaceholder.typicode.com/todos>: он отдаёт 200 дел, у каждого четыре поля – `userId`, `id`, `title` (заголовок) и `completed` («сделано»). Параметр `_page` – номер страницы, `_limit` – сколько записей на странице. Нужен интернет. Заранее создай переменные `df_list`, `df`, `data` и `i`.

Главный блок задачи – «HTTP GET» из группы «HTTP»: он ходит на сервер за данными.

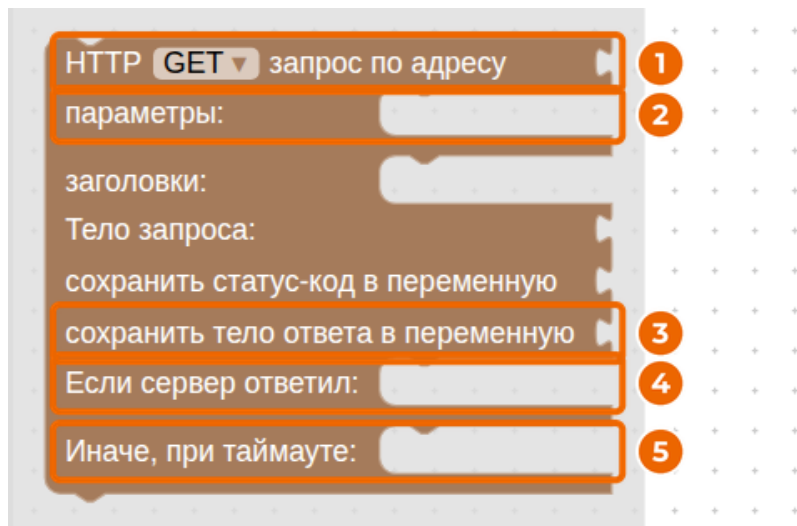


Рисунок В.13 – Блок «HTTP GET» в палитре: адрес (1), параметры (2), тело ответа (3), «Если сервер ответил» (4), «Иначе» (5)

Начало. Из «HTTP» – «подключить библиотеку requests», из «Pandas» – «подключить библиотеку pandas как pd» (1 на кадре ниже). Первая ходит за данными в интернет, вторая умеет работать с таблицами. Потом «присвоить `df_list = []` и пустой список, как в П.2 (2).

Цикл. Из «Циклов» – «цикл по ... от ... до ... шаг ...»: выбери переменную `i` (в новом блоке стоит другая буква), впиши числа от 1, до 5, шаг 1 (3). Последнее число в перебор не входит, поэтому для четырёх страниц пишем 5.

Запрос. Внутри цикла – «HTTP GET». В адрес (4) вставь текстовый блок с адресом. В «параметры» (5) поставь два блока «HTTP-параметр» друг под другом: ключ `_limit` со значением 50 (оба – пустые кавычки из группы «Текст») и ключ `_page`, а значение – переменная `i`. В «сохранить тело ответа в переменную» (6) вставь `data`. Остальные гнёзда оставь пустыми.

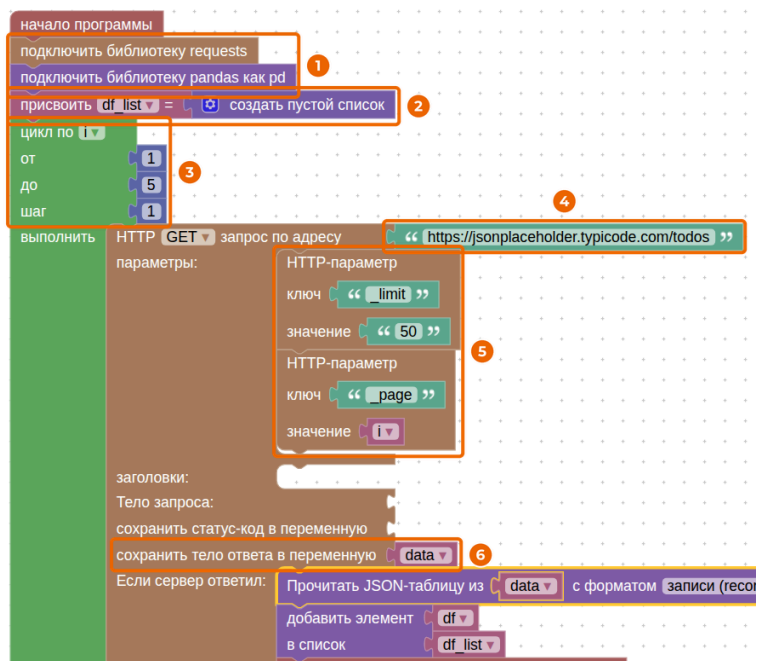


Рисунок В.14 – Начало программы П.4

Ответ сервера. В ветке «Если сервер ответил»: «Прочитать JSON-таблицу из `data` с форматом записи (records) в переменную `df`» из «Pandas» (7), потом «добавить элемент `df` в спи-

сок `df_list` » (8) из «Pandas» и «вывести» с текстом Готова страница и `i` (9). В ветке «Иначе, при таймауте» – «вывести» Ошибка (10). Ветки лежат близко: если блок прыгает не в ту, подними или опусти его чуть выше или ниже.

После цикла – не внутри него. «присвоить `df` => «объединить список таблиц `df_list` в одну таблицу» (11); «вывести» с блоком «первые 3 строк таблицы `df` » из «Pandas» (12); «показать информацию о таблице `df` » (13); «сохранить таблицу `df` в CSV-файл с именем» `todos.csv` (14).

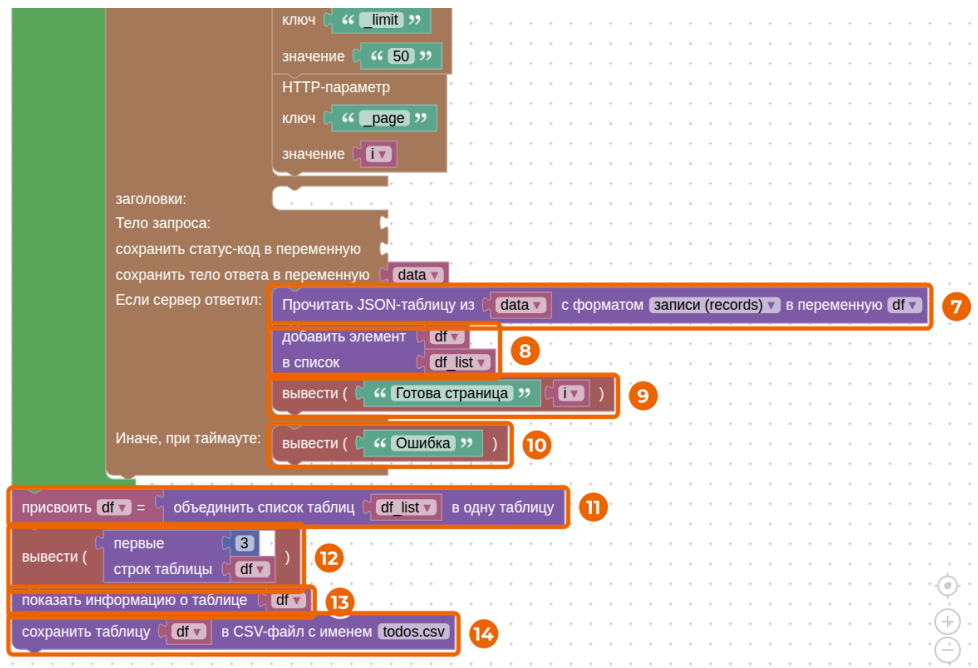


Рисунок В.15 – Конец программы П.4

Нажми «Запуск». Несколько секунд программа ходит за страницами, потом печатает результат.

```

Вывод
Готова страница 1
Готова страница 2
Готова страница 3
Готова страница 4
 1
  userId  id      title  completed
 0      1   1  delectus aut autem  False
 1      1   2  quis ut nam facilis et officia qui  False
 2      1   3  fugiat veniam minus  False
<class 'pandas.core.frame.DataFrame'>
Index: 200 entries, 0 to 49
Data columns (total 4 columns):
#   Column  Non-Null Count  Dtype
---  -
0  userId  200 non-null   int64
1  id      200 non-null   int64
2  title   200 non-null   object

```

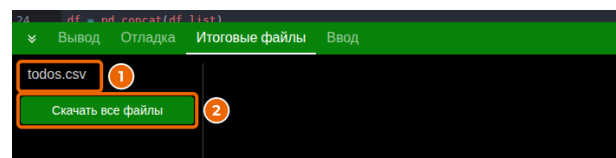


Рисунок В.16 – Слева – вывод программы, справа – вкладка «Итоговые файлы»

Что должно получиться: четыре строки Готова страница 1 ... Готова страница 4 (1), таблица из колонок `userId`, `id`, `title`, `completed` с тремя первыми строками, под ней сводка: `Index: 200 entries` и `total 4 columns` (2) – 200 записей и 4 колонки. Это сходится: 4 страницы по 50. На вкладке «Итоговые файлы» – `todos.csv`, а кнопка «Скачать все файлы» скачает архив `result_files.zip` с ним. Если вместо страниц печатается «Ошибка» – сервер не ответил: проверь интернет и адрес, а если они в порядке, повтори позже. Если записей получилось 150, а не 200 – в цикле стоит «до 4», и последняя страница потерялась.

Датавиз-конструктор: таблица, статистика, диаграммы

Датавиз-конструктор – программа для исследования данных без кода: загружаешь таблицу, смотришь статистику, строишь диаграммы, собираешь дашборды и «вычислительные эссе» – диаграммы с пояснениями. В курсе сферы ИИтех в нём делают задания на наборе пассажиров космического лайнера Spaceship Titanic. Кадры сняты в версии 2.0.0 для Windows.

Шаг 1. Поставить. Инструкция разработчиков – teletype.in/@irk.digitalwave/dataviz-install, ссылка на неё есть и в курсе сферы (модуль «Исследовательский анализ данных»). Программа работает только под Windows и ставится без установщика: распаковал архив – запускаешь `DatavizConstructor.exe`.

Шаг 2. Окно программы.

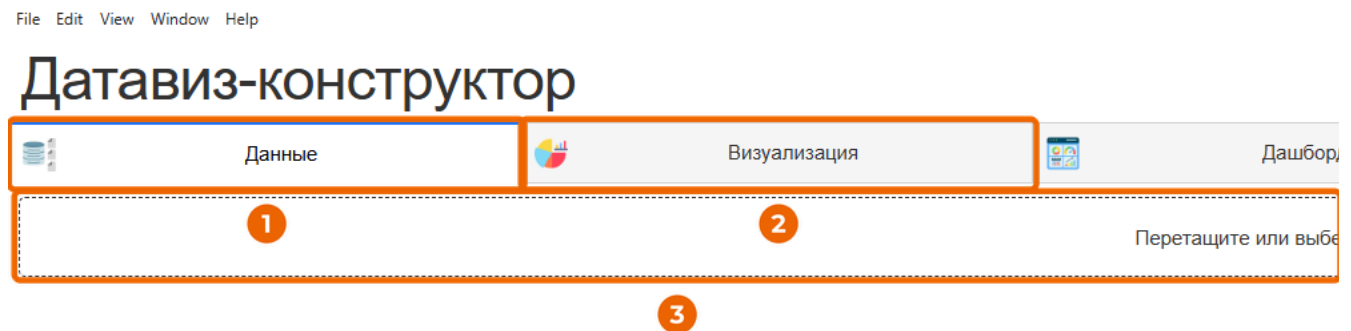


Рисунок В.17 – Окно Датавиз-конструктора, левая часть: вкладки и зона загрузки

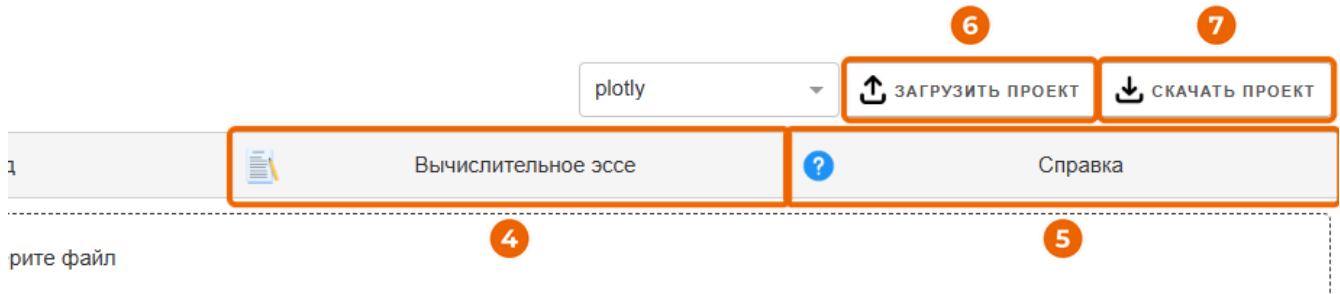


Рисунок В.18 – Окно Датавиз-конструктора, правая часть: вкладки и кнопки проекта

1. **Данные** – загрузить таблицу, отфильтровать, посмотреть статистику, скачать.
2. **Визуализация** – листы с диаграммами: «+» добавляет лист.
3. **Зона загрузки** – перетащи сюда файл `.csv` или щёлкни и выбери его.
4. **Вычислительное эссе** – диаграммы и текст одной лентой, как отчёт.
5. **Справка** – подсказки по программе.
6. **Загрузить проект** – открыть сохранённую работу (файл `.dviz`).
7. **Скачать проект** – сохранить всю работу в файл `.dviz`. Сохраняй почаще.

Между вкладками «Визуализация» и «Вычислительное эссе» есть ещё «Дашборд» – несколько диаграмм на одном экране.

Задача П.1. Сколько пассажиров без возраста? Скачай в курсе файл `train.csv` и загрузи его во вкладке «Данные».

СКРЫТЬ/ПОКАЗАТЬ ПОЛЯ

	NA	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP
	filter data.							
x	0	101	Europa	0	B/0/P	TRAPPIST-1e	39	
x	0	201	Earth	0	F/0/S	TRAPPIST-1e	24	
x	0	301	Europa	0	A/0/S	TRAPPIST-1e	58	
x	0	302	Europa	0	A/0/S	TRAPPIST-1e	33	
x	0	401	Earth	0	F/1/S	TRAPPIST-1e	16	
x	0	501	Earth	0	F/0/P	PSO J318.5-22	44	
x	0	601	Earth	0	F/2/S	TRAPPIST-1e	26	
x	1	602	Earth	1	G/0/S	TRAPPIST-1e	28	
x	0	701	Earth	0	F/3/S	TRAPPIST-1e	35	
x	0	801	Europa	1	B/1/P	55 Cancri e	14	
x	1	802	Europa	1	B/1/P	TRAPPIST-1e	34	
x	0	803	Europa	0	B/1/P	55 Cancri e	45	
x	0	901	Mars	0	F/1/P	TRAPPIST-1e	32	
x	0	1001	Earth	0	G/1/S	TRAPPIST-1e	48	
x	0	1101	Earth	0	F/2/P	TRAPPIST-1e	28	

Рисунок В.19 – Таблица загружена: служебный столбец NA

Программа сама добавила первым столбец NA (1): в нём 1, если в строке есть хоть один пропуск, и 0, если пропусков нет. Сами пустые ячейки подсвечены красным.

483	0	291	0	Sandie Hi
1539	3	0	0	Billex Jaci
0	0	0		Candra Jaci
785	17	216	0	Andon
0	0	0	0	Erraia
0		0	0	Altard
7295	589	110	124	Wezen

Рисунок В.20 – Пустые ячейки подсвечены красным

Над таблицей – кнопки: «Статистика» (1), поле «Название файла» (2) и «Скачать» (3). Нажми «Статистика».

ДАННЫЕ

1 СТАТИСТИКА

2 Название файла

ЭНКОДЕР

3 СКАЧАТЬ

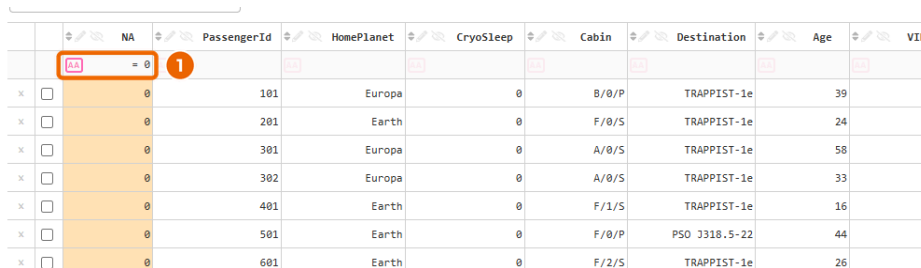
Рисунок В.21 – Кнопки над таблицей

index	Age	RoomService
count	8514	8512
mean	28.8279304675	224.6876174812
std	14.4890214239	666.7176629281
min	0	0
25%	19	0
50%	27	0
75%	38	47
max	79	14327

Рисунок В.22 – Статистика: count у Age (1)

Что должно получиться: строк в таблице 8693, а у Age count (1) – 8514. **count считает только непустые значения**, поэтому возраст не указан у 8693 – 8514 = 179 пассажиров. Посмотри count у других столбцов – пропуски есть почти везде.

Задача П.2. Чистая таблица. Вернись к таблице кнопкой «Данные». В строке фильтра под заголовком NA (1) впиши = 0 и нажми Enter – останутся строки без пропусков. Порядок строк и столбцы не трогай.



The screenshot shows a table with columns: NA, PassengerId, HomePlanet, CryoSleep, Cabin, Destination, Age, and VIP. The NA column has a filter dropdown set to 0. The table contains 7 rows of data, all of which have a value of 0 in the NA column.

	NA	PassengerId	HomePlanet	CryoSleep	Cabin	Destination	Age	VIP
x	0	101	Europa	0	B/0/P	TRAPPIST-1e	39	
x	0	201	Earth	0	F/0/S	TRAPPIST-1e	24	
x	0	301	Europa	0	A/0/S	TRAPPIST-1e	58	
x	0	302	Europa	0	A/0/S	TRAPPIST-1e	33	
x	0	401	Earth	0	F/1/S	TRAPPIST-1e	16	
x	0	501	Earth	0	F/0/P	PSO 3318.5-22	44	
x	0	601	Earth	0	F/2/S	TRAPPIST-1e	26	

Рисунок В.23 – Фильтр NA = 0

Впиши название файла (например, train_clean) и нажми «Скачать»: Windows спросит, куда сохранить, – имя (1) и «Сохранить» (2).

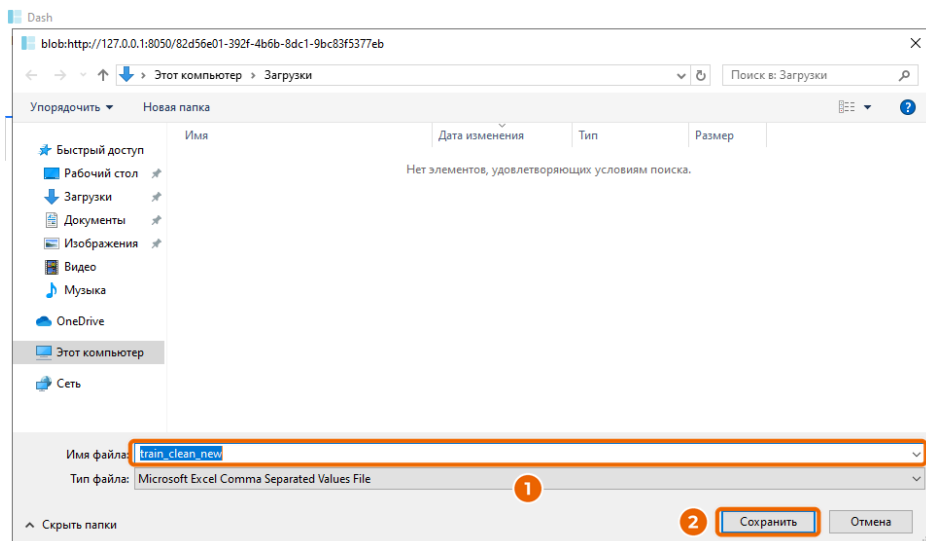
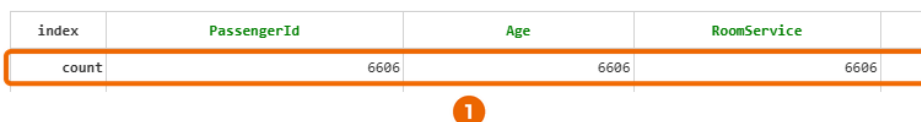


Рисунок В.24 – Сохраняем отфильтрованную таблицу

Загрузи сохранённый файл снова и открой «Статистику».



The screenshot shows a table with columns: index, PassengerId, Age, and RoomService. The 'count' row shows the count for each column: PassengerId (6606), Age (6606), and RoomService (6606).

index	PassengerId	Age	RoomService
count	6606	6606	6606

Рисунок В.25 – В чистой таблице count одинаковый у всех столбцов

Что должно получиться: count у всех столбцов – **6606** (1). Ровно столько строк должно остаться, так проверяет себя и курс. Не совпало – в курсе есть готовый файл data_new.csv, работай с ним. Важно: фильтр таблицы не действует на диаграммы – поэтому чистый файл и загружают заново.

Задача П.3. Кого «унесло» чаще? В столбце Transported 1 – пассажира переместило, 0 – нет; в CryoSleep 1 – он был в криосне. Вкладка «Визуализация» → «Выберите тип диаграммы» →

«Столбчатая» (1).

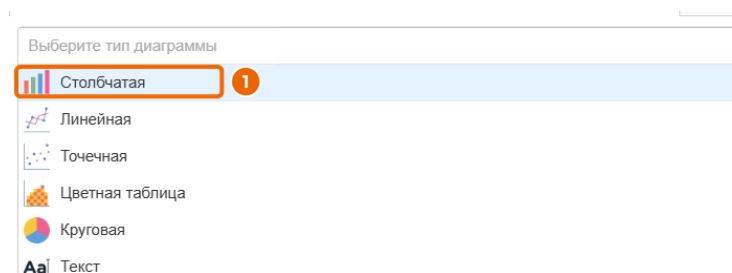


Рисунок В.26 – Тип диаграммы

Ось X – CryoSleep (1), ось Y – Transported (2), агрегация (3) пока «Сумма».

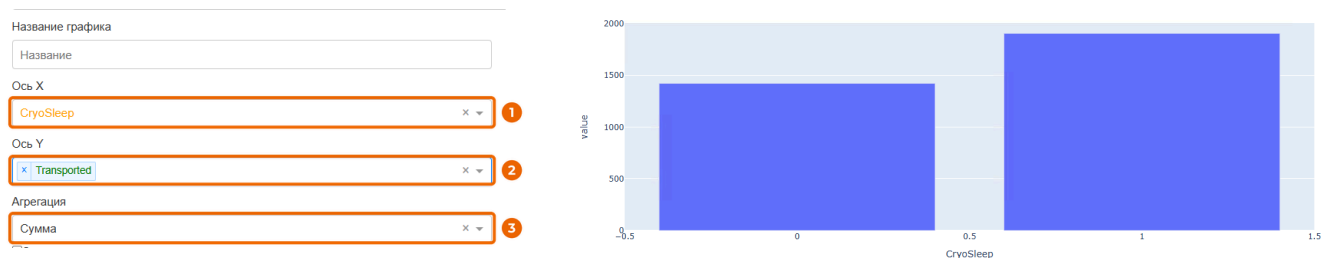


Рисунок В.27 – Слева – настройки диаграммы, справа – агрегация «Сумма»: сколько пассажиров переместило

Теперь смени агрегацию на «Среднее» (1).

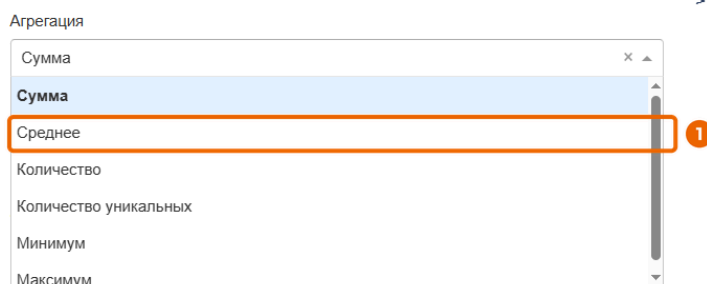


Рисунок В.28 – Агрегация «Среднее»

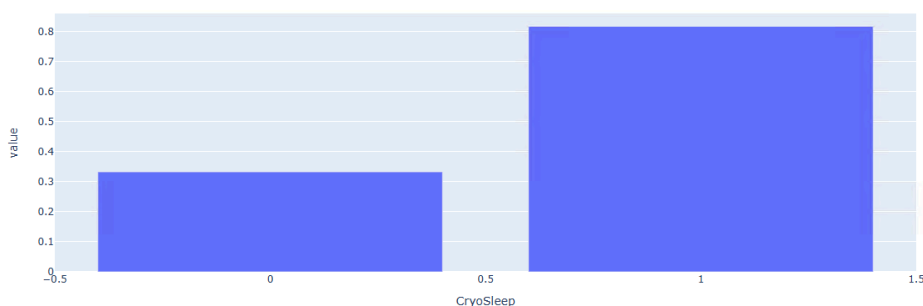


Рисунок В.29 – Среднее Transported: доля перемещённых

Что должно получиться: с «Суммой» столбцы показывают, **сколько** пассажиров переместило – около 1420 без криосна и около 1900 в криосне. Со «Средним» – **какая доля**: 0,33 и 0,82. Среднее столбца из нулей и единиц – это и есть доля единиц. Вывод: из спавших в криосне перемещены 82 % пассажиров, из бодрствовавших – только треть. Сравнить группы разного размера честнее по доле, а не по сумме.

GraphNet: собираем нейросеть из блоков

GraphNet – конструктор нейросетей: сеть собирают из блоков и соединяют их линиями, а учит её библиотека Keras. На сфере ИИтех с него начинают модуль про нейросети. Кадры сняты в версии 0.1.0 для Windows. В курсе модуль GraphNet открывается позже (4 октября), поэтому здесь – первая сеть из раздела 7.3: линейная регрессия $y = w \cdot x + b$.

Шаг 1. Поставить. Инструкция автора – teletype.in/@yupest/graphnet: там ссылка на файл GraphNet_v010.exe (около 650 МБ). Установка не нужна – скачал и запустил. Windows может предупредить о неизвестном издателе: «Подробнее» → «Выполнить в любом случае» (вместе со взрослым). Первый запуск долгий – программа распаковывается 15–40 секунд. Процессор Pentium или Celeron не подойдёт.

Шаг 2. Окно. Слева – список блоков (1), на поле уже стоит блок Input (2) – вход сети. Внизу кнопка «Собрать модель»: она соединяет блоки, обучает сеть и выполняет всё, что нарисовано.

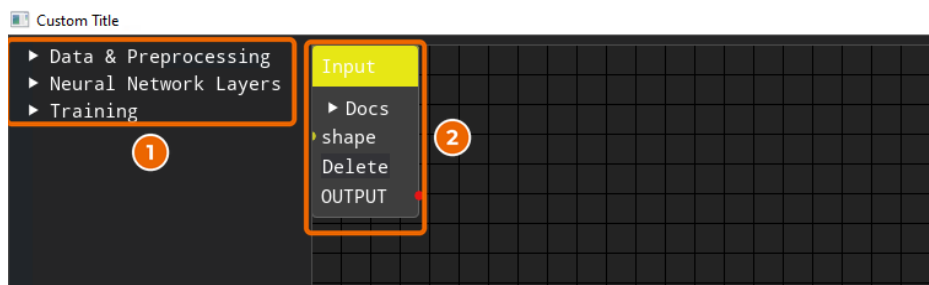


Рисунок В.30 – Окно GraphNet: список блоков (1) и блок Input (2)

Разверни разделы списка треугольниками – увидишь все блоки. Блок ставят на поле, **перетаскивая мышью** (щелчок по названию ничего не добавляет). Удалить – кнопка Delete в самом блоке.

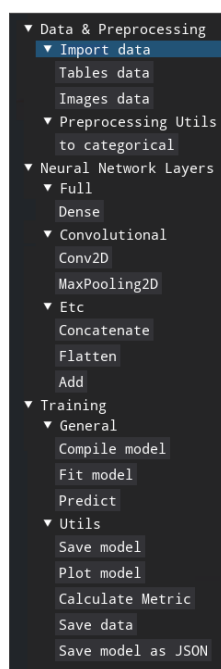


Рисунок В.31 – Все блоки GraphNet

- **Tables data** – загрузить таблицу из файла; **Images data** – папку с картинками.

- **Dense** – полносвязный слой нейронов; **Conv2D**, **MaxPooling2D** – слои для картинок.
- **Compile model** – выбрать, как учить: оптимизатор и функцию ошибки.
- **Fit model** – обучить: что подавать на вход (x), какие правильные ответы (y), сколько эпох.
- **Predict** – прогноз обученной сети на новых данных; **Calculate Metric** – посчитать точность.
- **Save model**, **Save data** – сохранить модель и результаты в файл.

Важно. Схему блоков GraphNet пока не сохраняет: закрыл программу – схему придётся собирать заново. Сделай снимок экрана, а обученную модель сохрани блоком Save model.

Задача П.1. Сколько вёдер на грядки. Дачник поливает грядки: на каждую уходит 3 ведра и ещё 1 ведро на дорожку – $y = 3x + 1$. Пусть сеть сама найдёт эти числа. Создай в Блокноте три файла, по одному числу в строке, без заголовка: `x.csv` – числа от 1 до 10 (грядки); `y.csv` – 4, 7, 10 и так до 31 (ведра); `new.csv` – 12 и 25 (новые грядки). Собери схему:

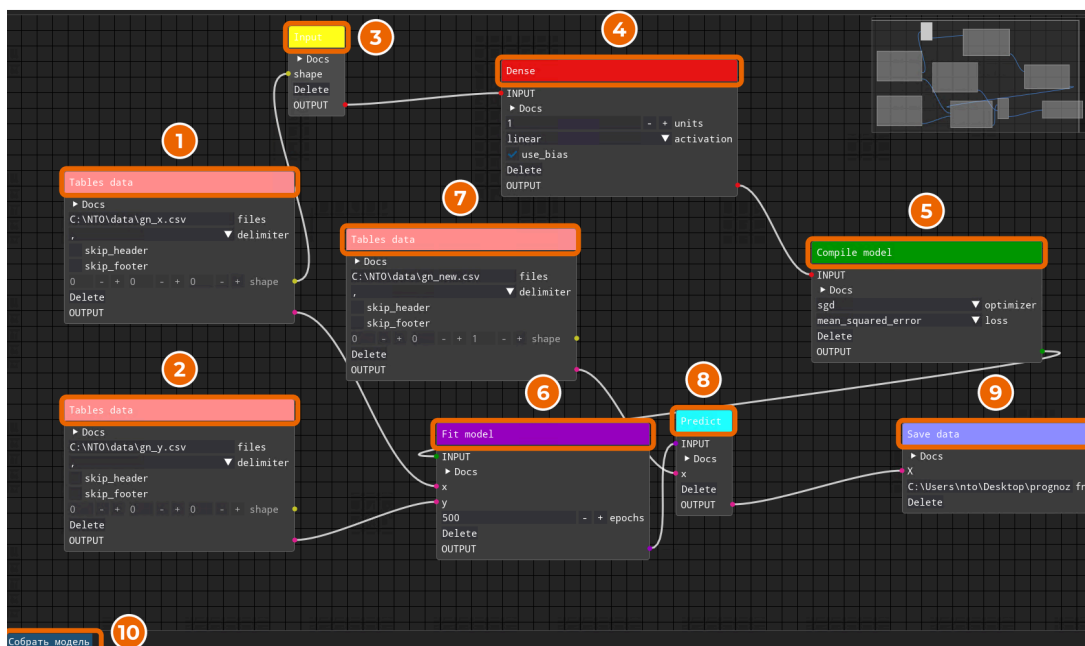


Рисунок В.32 – Собранная сеть: порядок блоков – как номера

1. **Tables data** с файлом `x.csv` – в поле files впиши полный путь, например `C:\Users\Ученик\Desktop\x.csv` (1 на рисунке ниже). Её выход **shape** (2) соедини со входом **shape** блока Input (3) – так сеть узнает размер входа.
2. **Tables data** с файлом `y.csv` – правильные ответы.
3. **Input** – уже стоит на поле.
4. **Dense**: units – 1 (один нейрон), activation – linear, галочка **use_bias** – чтобы у нейрона было смещение b . Соедини OUTPUT блока Input со входом INPUT блока Dense.
5. **Compile model**: optimizer – sgd, loss – mean_squared_error (средний квадрат ошибки). Вход – от OUTPUT Dense.
6. **Fit model**: вход INPUT – от Compile model, x – от таблицы `x.csv`, y – от таблицы `y.csv`, epochs – 500.
7. **Tables data** с файлом `new.csv` – новые грядки для прогноза.
8. **Predict**: INPUT – от Fit model, x – от таблицы `new.csv`.
9. **Save data**: X – от Predict, в поле имени файла – полный путь, например `C:\Users\Ученик\Desktop\prognoz.txt`.

10. Нажми «Собрать модель».

Линию тянут мышью от цветной точки-выхода одного блока к точке-входу другого.

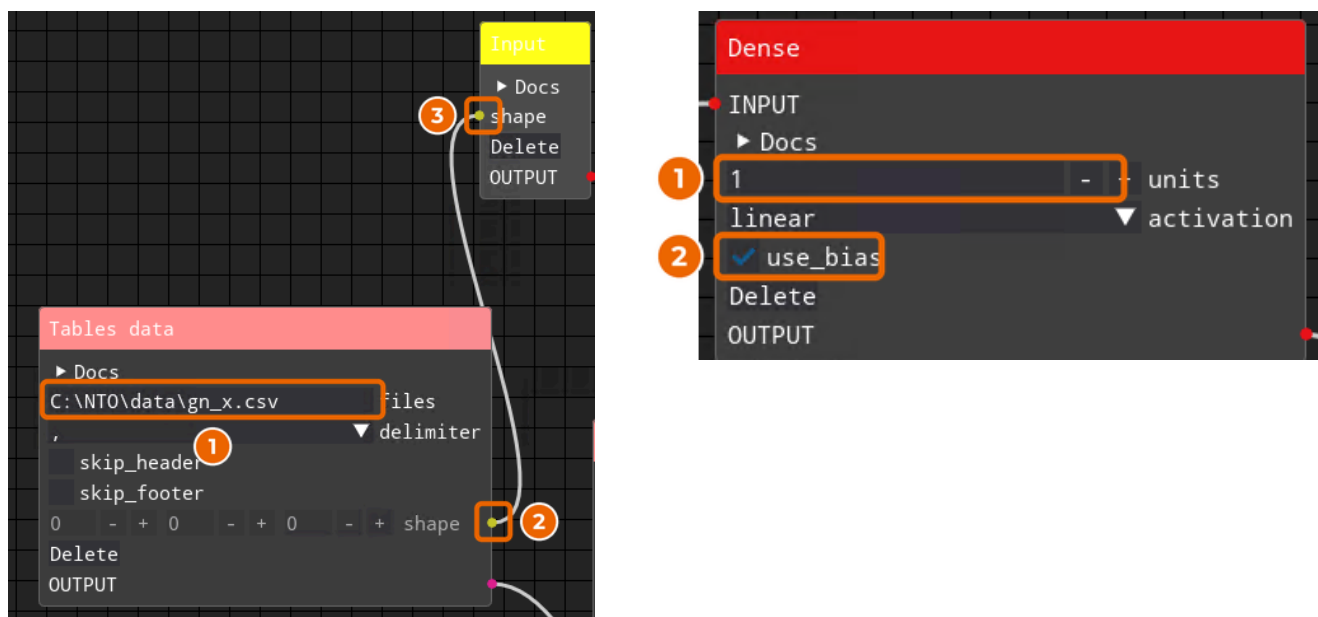


Рисунок В.33 – Слева – таблица и вход: путь к файлу (1), выход shape (2), вход shape у Input (3); справа – Dense: units (1), use_bias (2)

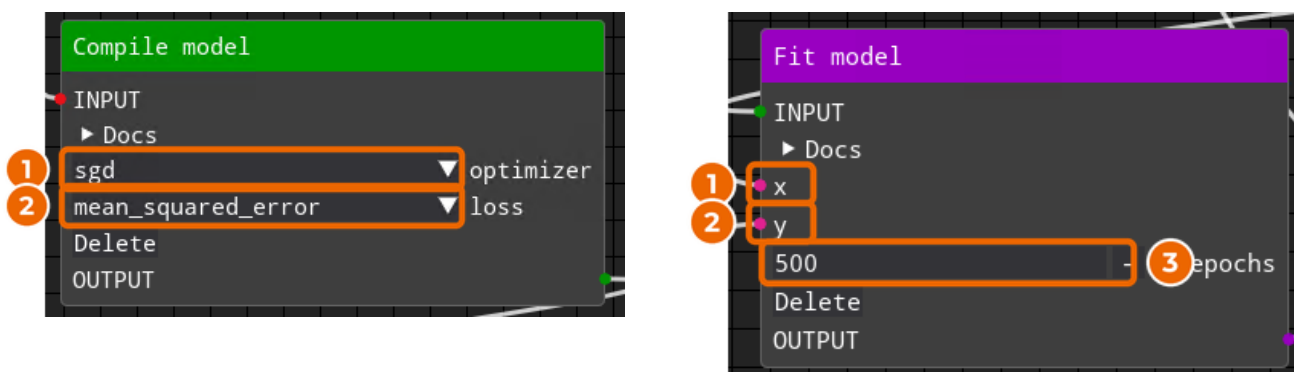


Рисунок В.34 – Слева – Compile model: оптимизатор (1) и ошибка (2); справа – Fit model: x (1), y (2), эпохи (3)

Что должно получиться: на несколько секунд появится окошко «Обучение», потом в файле `prognoz.txt` окажутся два числа – около 37 и около 76. У нас вышло 37,06 и 76,21: правильные ответы $3 \cdot 12 + 1 = 37$ и $3 \cdot 25 + 1 = 76$. Значит, сеть нашла $w \approx 3$ и $b \approx 1$ сама, по десяти примерам. Числа в файле записаны так: `3.705e+01` – это $3,705 \cdot 10^1 = 37,05$.

Задача П.2. Оптимизатор решает. В Compile model смени sgd на adam (1), эпохи оставь 500, снова «Собери модель» и открой `prognoz.txt`.

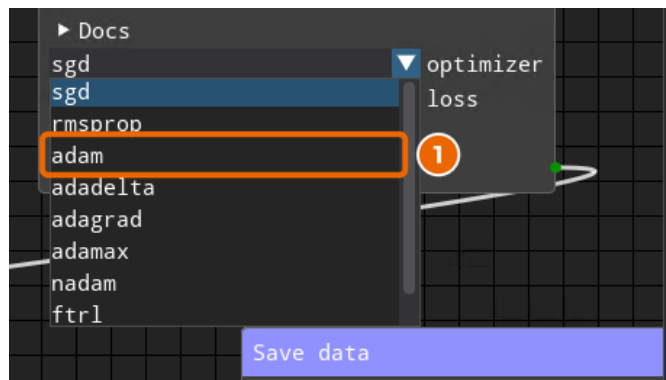


Рисунок В.35 – Список оптимизаторов

Что должно получиться: прогноз намного меньше правильного (у нас вышло 7,8 и 15,8 вместо 37 и 76; у тебя будут другие числа – веса сети в начале случайные). Adam с настройками по умолчанию делает слишком маленькие шаги и за 500 эпох не успевает дойти до $w = 3$. Поставь 5000 эпох – прогноз заметно приблизится (у нас 34,8 и 69,6), но учиться сеть будет несколько минут. Вывод для отбора: обученность сети зависит и от оптимизатора, и от числа эпох (раздел 7.3).

Teachable Machine: научи компьютер отличать круг от квадрата

Teachable Machine (teachablemachine.withgoogle.com) – бесплатный сервис Google: модель-классификатор в нём собирают мышкой, без единой строчки кода. Работает в браузере, регистрироваться не нужно. В курсе сферы 2026/27 Teachable Machine не используется – это инструмент кружка, чтобы руками понять, как работает классификация. Ниже весь путь по шагам и четыре задачи. Рамка с номером на картинке – куда нажимать.

Надписи в сервисе – на языке твоего браузера. Если они английские, внизу страницы есть выпадающий список языков – выбери «русский». Главные кнопки по-английски: «Get Started» – «Начать», «Image Project» – «Проект с изображениями», «Train Model» – «Обучить модель», «Advanced» – «Дополнительные», «Under the hood» – «Дополнительные сведения», «Preview» – «Посмотреть», «Input» – «Ввод», «Output» – «Выходные данные».

Шаг 0. Подготовка картинки. Веб-камера не нужна: примеры мы загрузим файлами. Нарисуй их сам в Paint или в любом редакторе, где можно сохранить PNG или JPG.

- Холст квадратный, например 400 × 400, фон белый: сервис всё равно обрежет картинку до квадрата.
- На холсте одна закрашенная фигура. Инструмент «Овал» с зажатой клавишей Shift рисует круг, «Прямоугольник» с Shift – квадрат. Овал без Shift тяни так, чтобы одна сторона была в полтора-два раза длиннее другой.
- У каждой картинки свои размер, место на холсте и цвет (красный, синий, зелёный, жёлтый, фиолетовый). Цвета и размеры у кругов и квадратов должны быть одинаково разными, чтобы по ним нельзя было угадать фигуру.
- Нужно четыре папки: «круг» – 30 картинок, «квадрат» – 30, «овал» – 30 и «проверка» – по 5 новых кругов, квадратов и овалов, которые в классы мы не положим (назови их «новый-круг-1», «новый-квадрат-1» и так далее).

Рисовать 105 картинок в одиночку долго, поэтому разделите работу в кружке. Если рисовать лень, можно фотографировать: крышки и монеты – круги, стикеры – квадраты, овалы вырежи

из бумаги. Снимай на светлом ровном фоне.

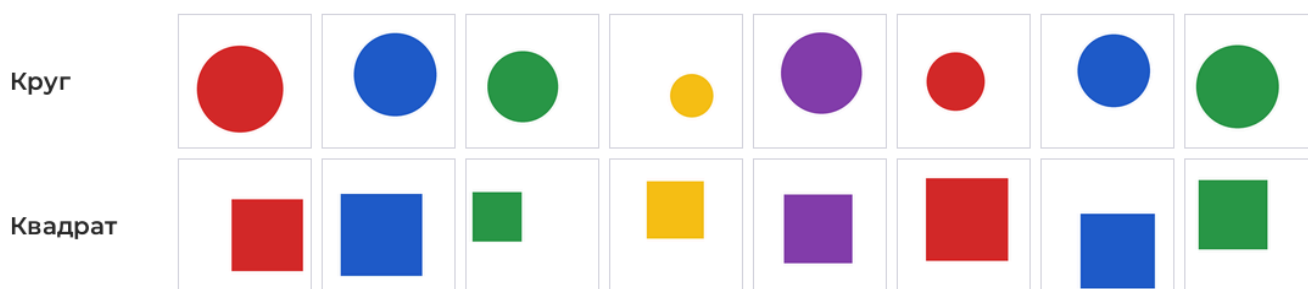


Рисунок В.36 – Образец выборки: по восемь картинок из папок «круг» и «квадрат»

Шаг 1. Открыть сайт. Зайди на teachablemachine.withgoogle.com и нажми «Начать» (1).



Рисунок В.37 – Главная страница Teachable Machine: кнопка «Начать»

Шаг 2. Выбрать проект. На странице «Новый проект» нажми «Проект с изображениями» (1 на левой половине рисунка). В окне выбери «Стандартную модель изображения» (1 на правой половине): она работает с цветными картинками и нужна нам. «Встроенная модель» – для микроконтроллеров, её не трогай.

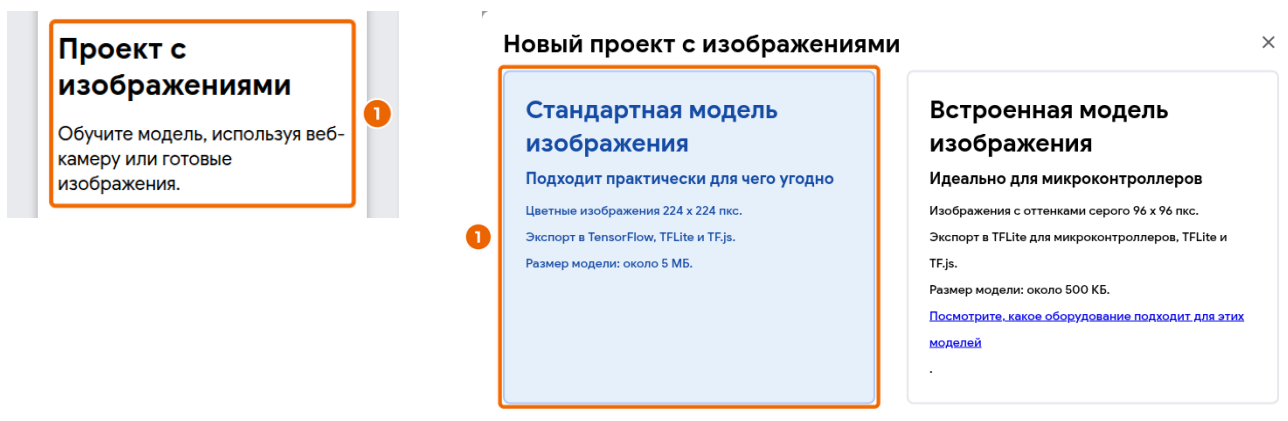


Рисунок В.38 – Новый проект: слева – карточка «Проект с изображениями», справа – выбор модели

Шаг 3. Окно редактора. Что где лежит, видно на рисунке. Сверху слева может всплыть синяя подсказка с видео – закрой её крестиком.

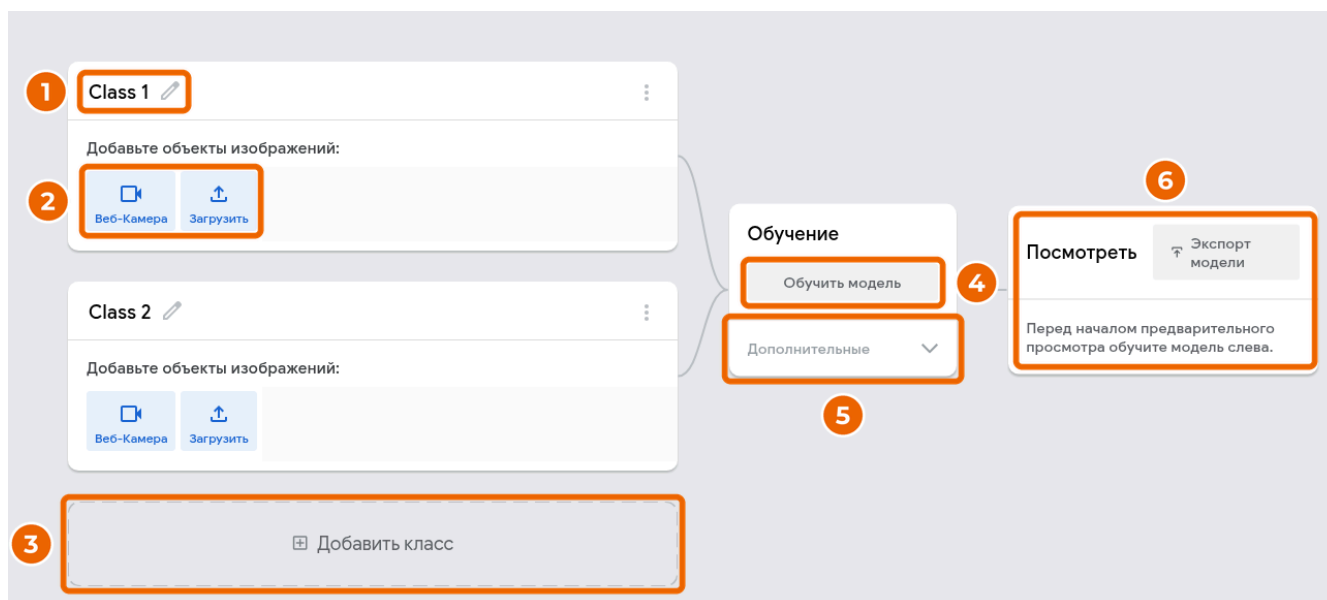


Рисунок В.39 – Редактор Teachable Machine

1. **Название класса.** Класс – это коробка для картинок одного вида. Название меняется карандашом.
2. **«Веб-камера» и «Загрузить»** – два способа положить в класс примеры. Нам нужен «Загрузить».
3. **«Добавить класс»** – ещё одна коробка. Сразу есть два класса, «Class 1» и «Class 2».
4. **«Обучить модель»** – запуск обучения.
5. **«Дополнительные»** – настройки обучения (эпохи, размер пакета, скорость обучения; они знакомы тебе из главы 7) и кнопка «Дополнительные сведения» с графиками.
6. **«Посмотреть»** – сюда после обучения показывают новые картинки и смотрят, что ответит модель.

Шаг 4. Назвать классы. Нажми на карандаш рядом с названием класса, сотри «Class 1», впиши «Круг» и нажми Enter. Второй класс назови «Квадрат»: так выглядит поле для названия, пока ты печатаешь (1 на левой половине рисунка ниже). Названия на панели «Посмотреть» обрезаются многоточием («Квад...») – это только подпись, на модель она не влияет.

Шаг 5. Загрузить примеры. В классе «Круг» нажми «Загрузить». Слева откроется панель: нажми «Выберите изображения из файлов» (1 на правой половине рисунка) и в окне выбора файлов найди папку «круг». Нажми Ctrl+A, чтобы выделить все 30 картинок, потом «Открыть». Сервис предупреждает (2 на правой половине), что обрежет картинки до квадрата: наши и так квадратные.

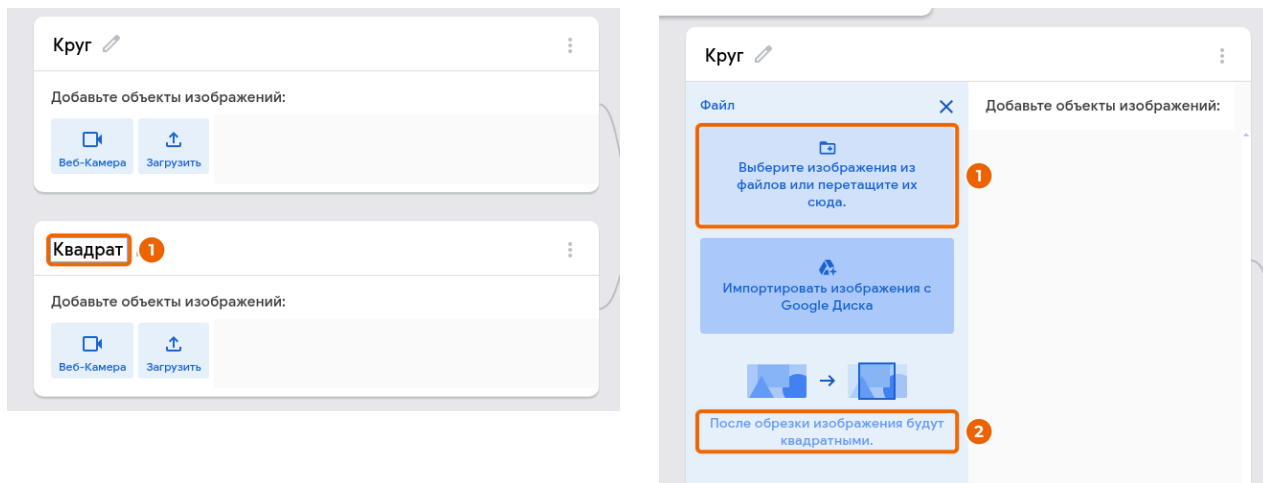


Рисунок В.40 – Слева – переименование класса, справа – панель «Загрузить»

Закрой панель крестиком, то же самое сделай со вторым классом и папкой «квадрат». Проверь, что в каждом классе написано «Образцов изображений: 30» (1 и 2). Если число не такое, значит, в классе лишние или пропавшие картинки – добавь или найди их.

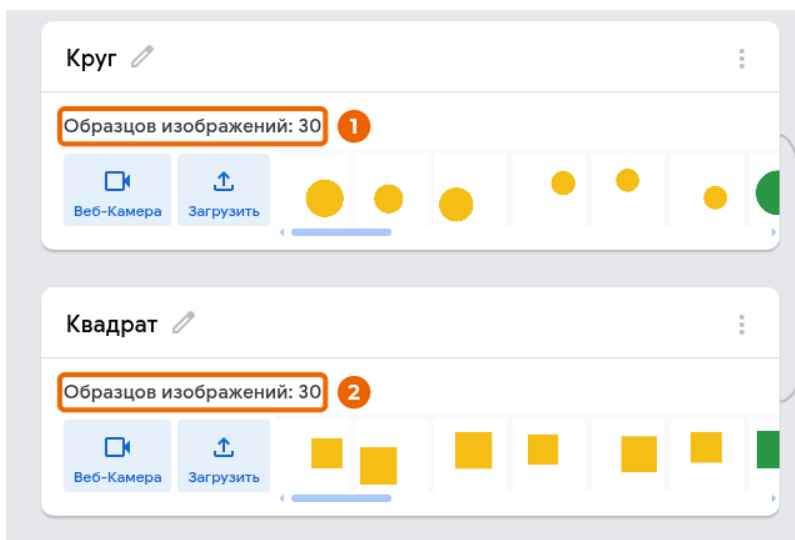


Рисунок В.41 – Оба класса готовы: по 30 образцов

Шаг 6. Обучить. Нажми «Обучить модель» (на рисунке шага 3 – кнопка 4). Кнопка превратится в полоску «Обучение», а под ней побежит счётчик (1): время и номер эпохи из 50. Сверху всплывёт сообщение (2): не переключайся на другие вкладки и не закрывай эту, пока идёт обучение. Оно занимает от нескольких секунд до минуты. Когда на кнопке появится «Модель обучена», модель готова.

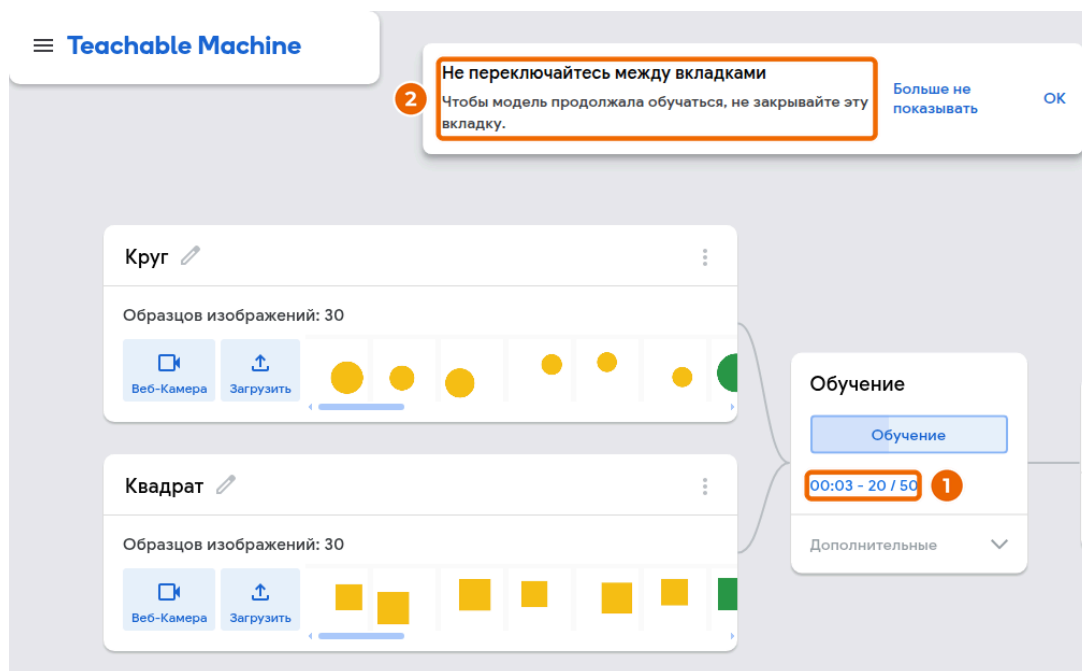


Рисунок В.42 – Обучение идёт: 20-я эпоха из 50

Шаг 7. Проверить на новой картинке. По умолчанию панель «Посмотреть» ждёт веб-камеру: браузер попросит к ней доступ или сервис напишет «Не удалось включить веб-камеру». Нам камера не нужна. Открой выпадающий список справа от слова «Ввод» (1) и выбери «Файл». Нажми «Выберите изображения из файлов» (2) и возьми любую картинку из папки «проверка». Под ней в «Выходных данных» (3) появятся полоски: для каждого класса – процент уверенности. Проценты всех классов в сумме дают 100, а ответ модели – класс с самой длинной полоской.

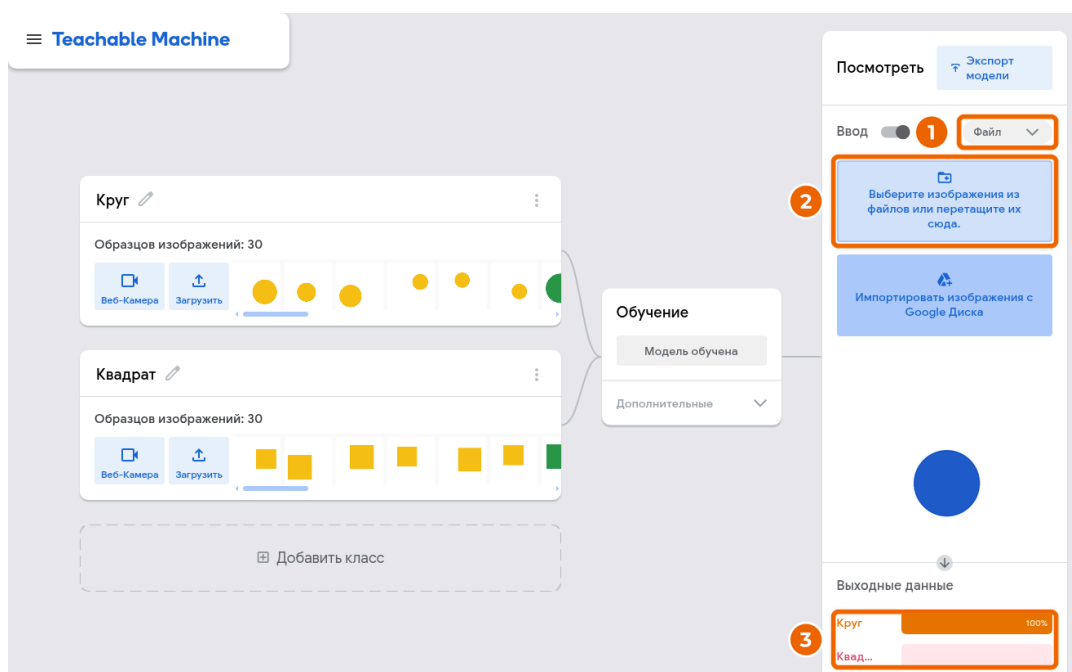


Рисунок В.43 – Проверка на файле: новый синий круг, ответ «Круг» со 100 %

Не потерять работу. Закроешь вкладку – проект пропадёт. Чтобы сохранить, нажми три полоски слева вверху и выбери «Скачать проект как файл»: на компьютер придёт файл `project.tm`. Открывается он на странице выбора проекта кнопкой «Открыть существующий

проект из файла». Входить в аккаунт Google для этого не нужно. Пункты про Google Диск пропусти.

Практика: четыре опыта.

Опыты идут по порядку: первые три – в одном проекте, для четвёртого начни новый. Числа в твоём проекте будут немного другими: при каждом обучении сеть стартует с других случайных весов. Важно, куда числа сдвигаются, а не какими они получились у нас.

Задача П.1. Круг или квадрат? Сделай по шагам 1–7 модель из двух классов «Круг» и «Квадрат» по 30 картинок. Проверь четыре картинки из папки «проверка»: два новых круга и два новых квадрата разных цветов. Для каждой запиши, какой класс выбрала модель и сколько процентов получил этот класс.

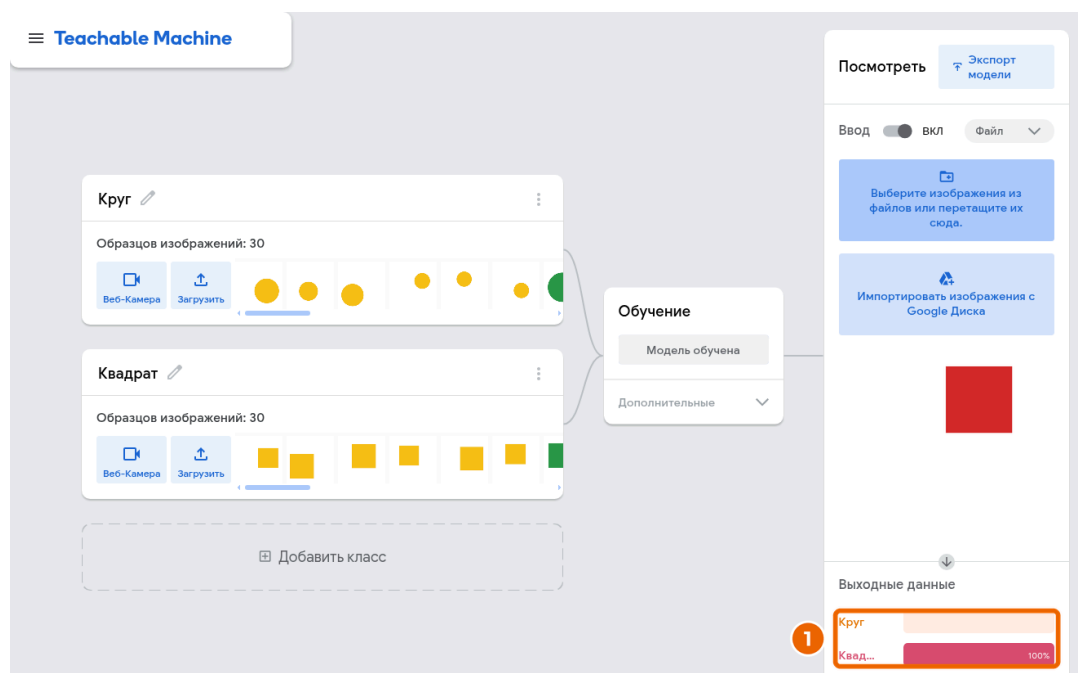


Рисунок В.44 – Новый красный квадрат: ответ «Квадрат» со 100 %

Что должно получиться: на всех четырёх картинках правильный класс набрал больше 90 %. У нас получилось 100 % и 0 %: фигуры такие разные, что модель не сомневалась. Связь с книгой: 60 картинок, которые ты положил в классы, – обучающая выборка, а картинки из «проверки» – проверка на новых данных: их модель до этого не видела (глава 1 и раздел 7.5).

Задача П.2. Заглянуть под капот. Не закрывая проект, раскрой в блоке «Обучение» список «Дополнительные» и нажми «Дополнительные сведения» (1). Справа выедет панель. Нажми «Вычислить точность на каждый класс» (2) и «Вычислить матрицу ошибок» (3). Потом прокрути панель вниз к графикам «Точность на каждую эпоху» и «Потеря на каждую эпоху». Ответь: сколько картинок каждого класса сервис отложил для проверки? Какая получилась точность по классам? Что стоит в клетках матрицы вне главной диагонали?

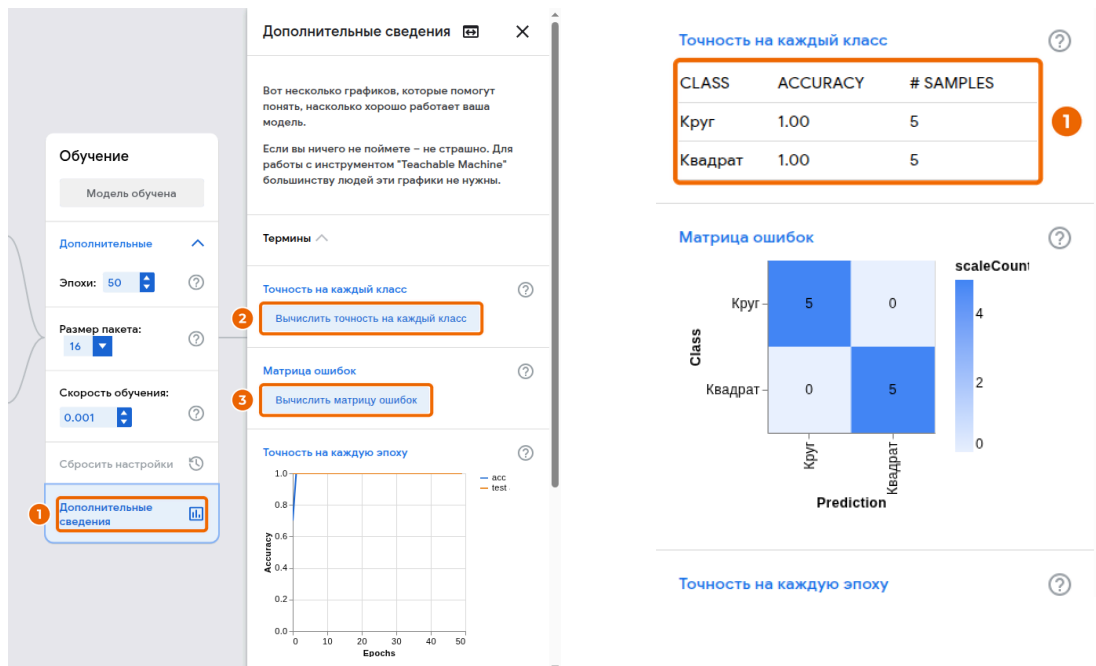


Рисунок В.45 – Слева – кнопки панели «Дополнительные сведения», справа – таблица точности и матрица ошибок хорошей модели

Что должно получиться:

- Сервис сам отложил 15 % картинок и не учил на них модель: по 5 из каждой 30 (столбец «# Samples»). В Teachable Machine они называются тестовыми, а в книге – проверочными.
- Точность у обоих классов 1.00: из 5 отложенных картинок верно названы все 5.
- В матрице ошибок строка – настоящий класс, столбец – ответ модели, число в клетке – сколько картинок. У нас по 5 на главной диагонали и нули вне её: модель не ошиблась ни разу.
- На графике точности две линии: «acc» – точность на обучающих картинках, «test» – на отложенных. У хорошей модели обе почти сразу упираются в 1.0.
- График потерь показывает ошибку (функцию потерь): она падает почти до нуля за несколько эпох.

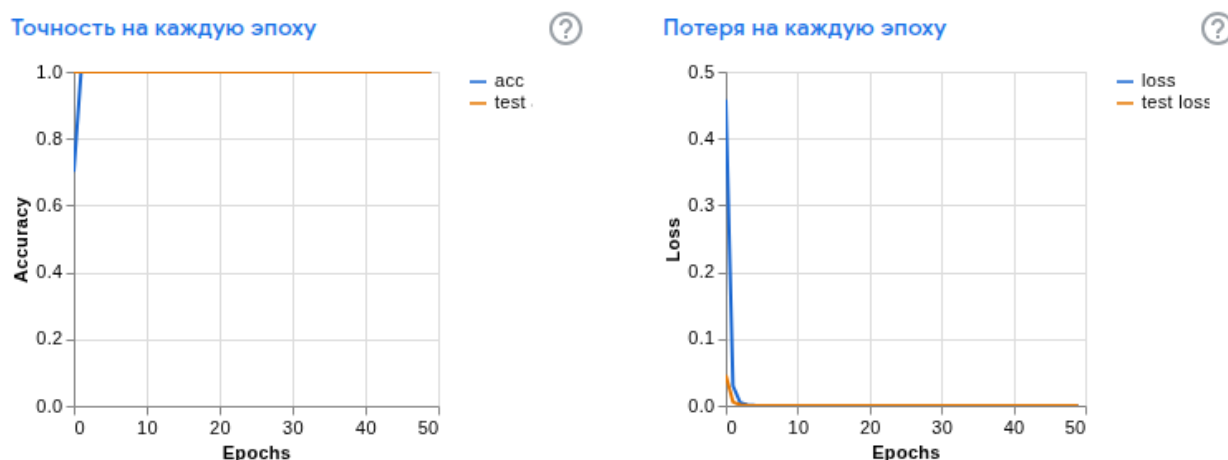


Рисунок В.46 – Хорошая модель: слева точность по эпохам («acc» – обучающие, «test» – отложенные), справа потеря

Задача П.3. Перепутанные подписи. В классе «Круг» нажми «Загрузить» и добавь ещё 10 картинок, но квадратов: любые десять из папки «квадрат». Теперь в «Круге» 40 образцов, и 10 из них подписаны неправильно – будто в кружке кто-то перепутал папки. Нажми «Обучить модель» ещё раз: после любого изменения в классах кнопка снова становится активной. Когда модель обучится, в списке «Ввод» может опять появиться сообщение о камере: переключи список на «Веб-камера» и обратно на «Файл». Потом: а) проверь на новых картинках те же четыре; б) открой «Дополнительные сведения», вычисли точность по классам и матрицу ошибок; в) посмотри оба графика и сравни их с графиками из П.2.

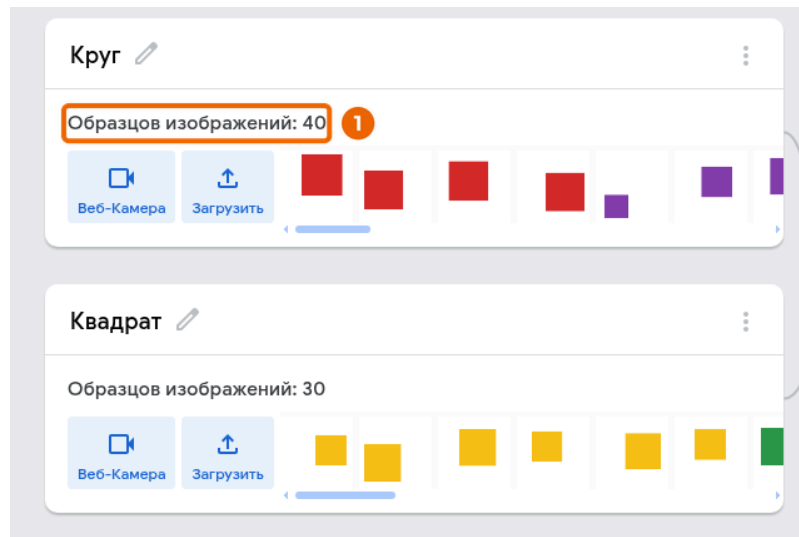


Рисунок В.47 – В классе «Круг» теперь 40 образцов, среди них красные и фиолетовые квадраты

Что должно получиться:

- На четырёх новых картинках модель ещё часто права: у нас круги получили 100 %, квадраты от 81 до 100 %. По одной-двум картинкам порчи почти не видно.
- В таблице точность по классам упала: у нас 0.50 для круга и 0.60 для квадрата, в других запусках выходило от 0.20 до 0.80. Одинаковых чисел не жди, но 1.00 уже не будет. В матрице вне главной диагонали больше нет одних нулей.
- На графике точности линии разошлись: «асс» держится около 0.9, «test» падает ниже, у нас примерно до 0.55.
- На графике потерь «loss» всё ещё снижается, а «test loss» растёт.

Модель подгоняется под обучающие картинки вместе с их ошибками и всё хуже справляется с отложенными. Это то самое переобучение из раздела 7.4. Значит, ошибки в подписях не видны на одной-двух проверочных картинках: их находят по точности по классам и по расхождению графиков.



Рисунок В.48 – Испорченная модель: слева таблица и матрица ошибок, справа график точности – «test» ушла вниз от «acc»



Рисунок В.49 – Потеря на каждую эпоху: «test loss» растёт, пока «loss» падает

Задача П.4. Мало примеров. Открой сайт заново в новой вкладке и начни новый проект (шаги 1 и 2). Назови классы «Круг» и «Овал». В «Круг» загрузи всего 3 любые картинки, в «Овал» – все 30. Обучи модель. Проверь пять новых кругов и один новый квадрат из папки «проверка», запиши ответы. Потом загрузи в «Круг» остальные 27 картинок, обучи модель ещё раз (список «Ввод» переключи так же, как в П.3) и проверь те же пять кругов. Сколько новых кругов модель назвала овалами до и после добора? Что она ответила про квадрат?

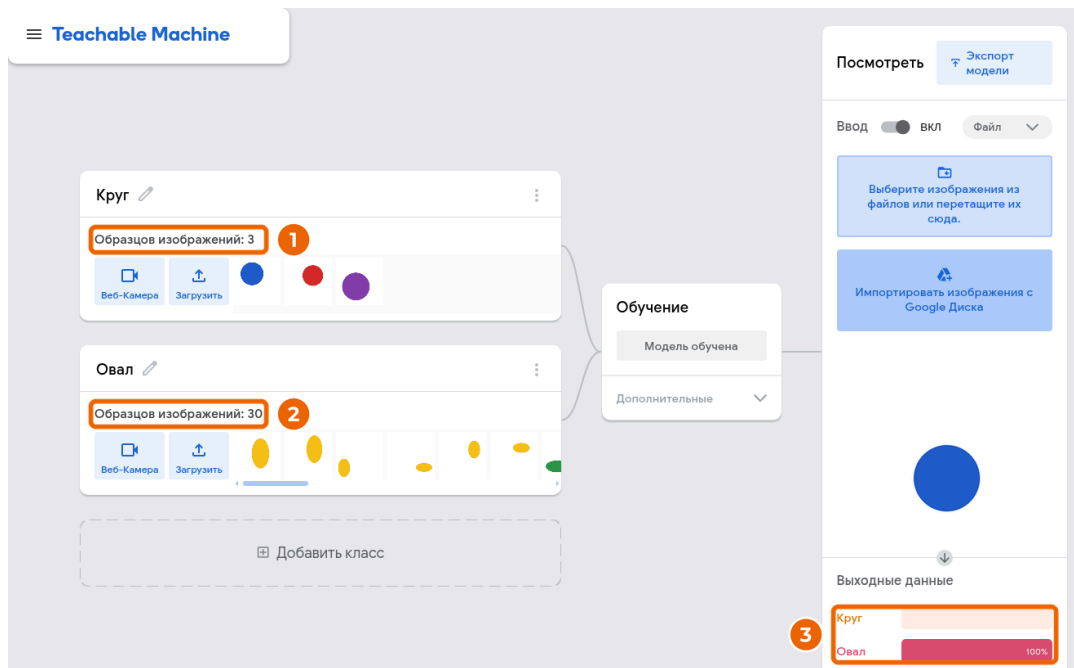


Рисунок В.50 – В «Круге» 3 образца, в «Овале» 30: новый синий круг назван «Овалом» на 100 %

Что должно получиться:

- Пока кругов три, модель называет «Овалом» почти всё: из пяти новых кругов у нас «Кругом» назван один, остальные четыре – «Овал» с уверенностью 95–100 %.
- В таблице точности у круга всего одна отложенная картинка, и по ней ничего не скажешь: у нас вышло 0.00, в другой раз было 1.00. Проверка на новых картинках показала больше любой таблицы.
- Квадрат модель тоже назвала «Овалом» на 100 %, хотя таких картинок никогда не видела. «Не знаю» она сказать не умеет и выбирает из тех классов, что есть. Уверенность – это не правильность.
- После добора до 30 кругов все пять новых кругов стали «Кругом» (97–100 %), овалы остались овалами.

Так работает перекося классы из раздела 7.5: три картинки против тридцати – это даже не один порядок.

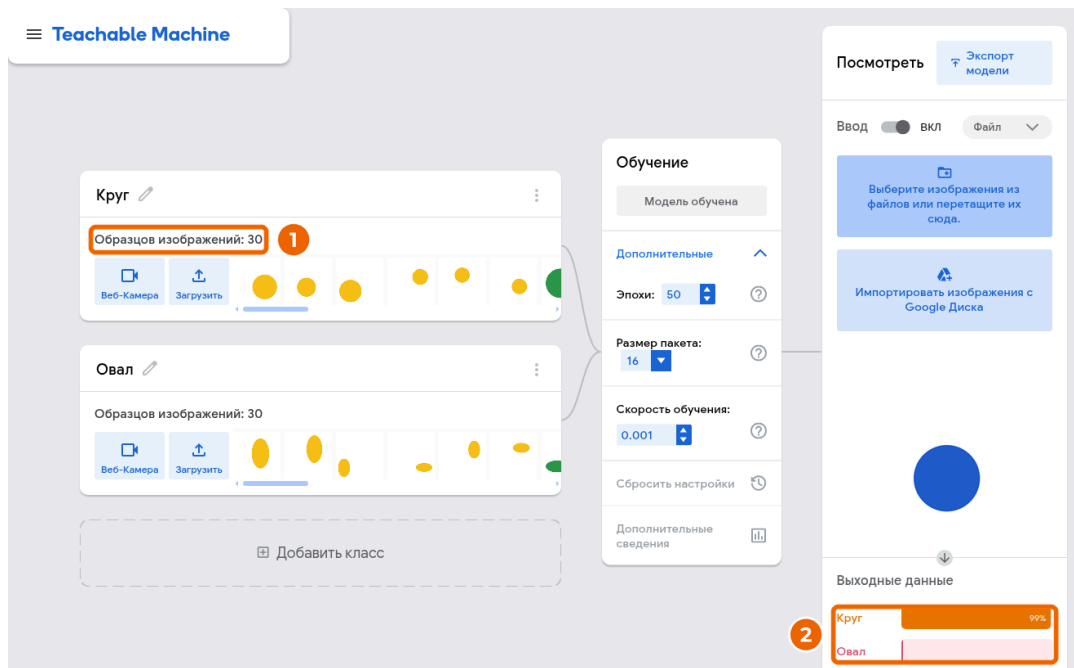


Рисунок В.51 – После добора: в «Круге» 30 образцов, новый синий круг назван «Кругом» на 99 %

Если взять вместо овалов квадраты, перекося почти незаметно: у нас три круга и тридцать квадратов модель различила без единой ошибки, фигуры слишком разные. Перекося ломает модель там, где классы похожи: круг и овал, синица и снегирь у кормушки.

План на месяц до отбора

Если начинаешь в конце сентября, вот спокойный план: примерно по часу в день.

Таблица В.3

Неделя	Что читаем	Что делаем
1	главы 1–3	каждый день 3–4 разбора с задачами «Реши сам»; зарегистрироваться на курс сферы
2	главы 4–5	открыть онлайн-Python или Colab, повторить код из разборов главы 5
3	главы 6–7	поставить GraphNet и Датавиз-конструктор по инструкциям курса, построить графики из разборов
4	глава 8 и шпаргалка	заново решить задачи, где были ошибки; пройти тест курса сферы (+5 баллов)

Отбор длится до 10 ноября, и попыток две. Если первая попытка прошла неудачно, вернись к главам, где потерял баллы, и иди на вторую.

Родителям и наставникам

- **Помогайте разбираться, но не решайте за ребёнка.** И не давайте решать нейросети: на командном финале сразу видно, кто разобрался сам, а за кого решили. Лучше пусть ребёнок ошибётся на отборе и научится, чем попадёт в финал и растеряется там.
- **Садитесь рядом на первых задачах.** Детей чаще всего останавливает не сложность, а «программа не ставится», «что-то зависло» или «непонятно, куда нажать».
- **ИИ-чаты – вместе со взрослым.** Прочитайте правила сервиса: GigaChat, Алиса и DeepSeek рассчитаны на взрослых, а детям до 18 лет разрешают пользоваться ими с согласия родителей и под их присмотром. Договоритесь, что в чат не пишут имя, адрес, школу и фотографии. И проверяйте ответы вместе: нейросеть умеет уверенно ошибаться.
- **Всё про регистрацию, бонусы и финал** для Новосибирской области – на nto.kvantorium54.ru (вкладки «Я наставник» и «Я участник или родитель»).
- **Региональный координатор НТО Юниоры в Новосибирской области** – детский технопарк «Кванториум», info@kvantorium54.ru.

Честно об этой книге

Главное, что нужно знать про эту книгу: заданий отбора 2026/27 мы не видели – организаторы их заранее не показывают. Угадывать их бесполезно: наставники, которые следят за олимпиадой не первый год, говорят, что задания из года в год не повторяются. Поэтому книга учит темам из программы сферы и приёмам решения, а не готовит к конкретным задачам. Когда организаторы выложат пробные задания, мы сверим с ними книгу и выпустим новую версию.

Это не официальное издание НТО. Книгу составил методист «Кванториума» по открытым материалам олимпиады: программе сферы «Основы искусственного интеллекта и анализа данных» 2026/27, записям вебинаров сферы и разборам задач отборочных этапов прошлых лет (сфера тогда называлась «Технологии и виртуальные помощники», а потом «Технологии и искусственный интеллект»). Задачи прошлых лет пересказаны своими словами, с новыми числами и новыми ответами.

Разработчики предупредили, что GraphNet и Датавиз-конструктор к этому сезону обновились. Поэтому кнопки и окна этих программ мы описываем в общих чертах, а смысл – подробно: смысл от версии к версии не меняется. Точную инструкцию к новой версии смотри в курсе сферы.

Каждое число в ответах мы пересчитали на компьютере, а каждую программу запустили. Если ты всё-таки нашёл ошибку или что-то осталось непонятным, напиши автору: osinovsky.aa@kvantorium54.ru. Следующие версии книги станут лучше благодаря тебе.

Что такое искусственный интеллект

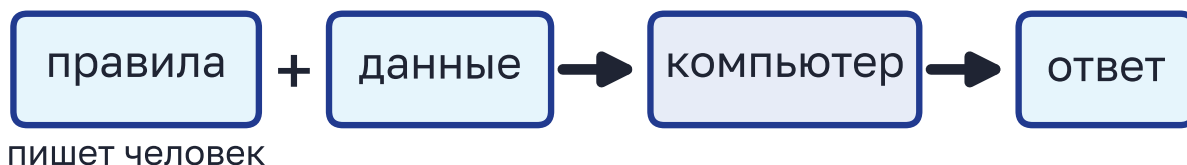


С этой главы начинается вся сфера. Формул здесь почти нет, зато есть слова, без которых дальше не прочесть ни одного условия: «признак», «обучающая выборка», «классификация», «вес нейрона», «точность». На отборе их проверяют так: сопоставь термины, определи тип задачи по описанию, посчитай точность модели по таблице ответов или выход нейрона по его весам. Задачи несложные, а баллы на них теряют из-за одного – путают похожие слова. Поэтому у каждого слова в главе есть свой пример.

1.1. Что такое ИИ и чем он не является

НА ПАЛЬЦАХ

Обычная программа



Машинное обучение

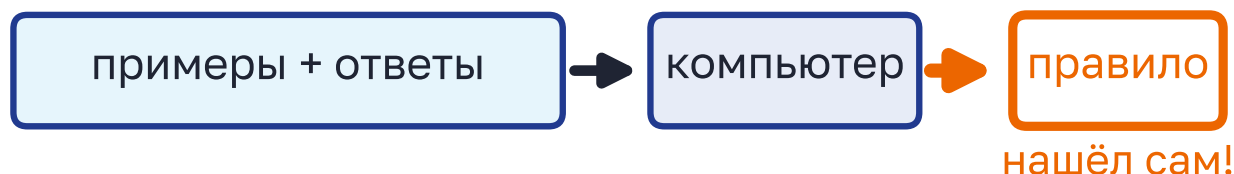


Рисунок 1.1 – Обычная программа и машинное обучение

В обычной программе правило придумал человек. В машинном обучении компьютер находит правило сам – по примерам с ответами.

Сначала о слове «интеллект». Так называют умение решать новые задачи: узнавать, понимать, делать выводы. Ты узнаёшь друга в толпе, хотя он в новой шапке. Понимаешь фразу, которую никогда раньше не слышал. Это и есть интеллект.

Искусственный интеллект (ИИ) – это программы, которые решают задачи, для которых человеку нужен интеллект: узнать лицо на фото, понять речь, перевести текст, сыграть в шахматы, нарисовать картинку по описанию. Сами слова «искусственный интеллект» предложил американский учёный Джон Маккарти в 1955 году, когда готовил заявку на научный семинар. Сам семинар прошёл летом 1956 года.

Не путай ИИ с любой программой. Возьми школьный звонок: он звенит по расписанию, которое вбил завхоз. Это **алгоритм** – точная последовательность шагов, записанная человеком. Звонок ничего не узнаёт и ничему не учится. Так же устроены калькулятор, будильник, светфор, который переключается каждые 40 секунд. Всё это не ИИ.

Большинство нынешних ИИ-программ устроены по-другому. Им не пишут правило «кошка – это когда уши треугольником». Им показывают тысячи фотографий с подписями «кошка» и «не кошка», и программа сама находит, чем они отличаются. Такой способ называется **машинное обучение** (machine learning): компьютер учится на примерах. А самый мощный сегодня вид машинного обучения – **нейросети**, о них раздел 1.5.



Рисунок 1.2 – Нейросети – часть машинного обучения, машинное обучение – часть ИИ. Калькулятор и будильник в этот круг не попадают

Получаются круги один в другом. Все нейросети – это машинное обучение, и всё машинное обучение – это ИИ. Но **не наоборот**: не весь ИИ – нейросети. Старые шахматные программы перебирали ходы по правилам, которые записали люди, – это ИИ, но без обучения. Бывают и модели, которые учатся на примерах, но обходятся без нейронов, например «дерево решений» – цепочка вопросов «да или нет», которую компьютер сам составил по таблице примеров.

Ещё два деления, которые любят спрашивать:

- **Узкий и общий ИИ.** Узкий умеет свои задачи: переводчик переводит, но не играет в шахматы. Общий ИИ умел бы всё, что умеет человек, и учился бы любому новому делу. Все системы, которые существуют сегодня, – узкие. Общий ИИ пока не создан, и учёные спорят, когда и как он появится.
- **Генеративный и негенеративный ИИ.** Генеративный **создаёт новое**: пишет текст, рисует картинку, сочиняет музыку. Так работают чат-боты GigaChat, Алиса и DeepSeek, когда пишут ответ на твой вопрос. Негенеративный **распознаёт и предсказывает**: узнаёт лицо, отличает спам от обычного письма, прогнозирует погоду. Он выбирает ответ, а не сочиняет его.

Где ИИ уже есть в твоей жизни:

- телефон разблокируется по твоему лицу;
- клавиатура подсказывает следующее слово;
- Алиса понимает, что ты сказал, и отвечает голосом;
- навигатор в машине родителей прикидывает, сколько минут ехать до школы с учётом пробок;
- почта сама убирает рекламу в папку «Спам»;
- лента видео подбирает ролики по тому, что ты досмотрел до конца;
- галерея телефона сама собирает фото в альбомы по лицам, хотя имён ты не вводил.

Чем ИИ не является. Он не думает, как человек, и ничего не «знает» о мире. Он находит закономерности в примерах, которые ему дали. Отсюда его слабости. Учили на плохих примерах – будет плохо работать. Попал в ситуацию, которой в примерах не было, – может ошибиться, причём очень уверенно. Чат-бот способен выдумать несуществующую книгу с автором и годом издания и выдать это спокойным тоном. Поэтому ответы ИИ всегда проверяют.

Запомни. ИИ – программы для задач, где человеку нужен интеллект. Машинное обучение – та часть ИИ, которая учится на примерах, а нейросети – часть машинного обучения. Если программа только выполняет правила, записанные человеком (звонок, калькулятор, светофор по таймеру), – это не ИИ.

1.2. Машинное обучение: учимся на примерах

НА ПАЛЬЦАХ



Рисунок 1.3 – Примеры с ответами и новый предмет

Никто не объяснял правило словами. Посмотрели на примеры с ответами – и новый предмет узнаётся сам.

Вспомни, как малыш учится отличать собаку от кошки. Ему не читают лекцию про форму ушей. Ему много раз показывают: «Это собака. А это кошка». Через месяц он уверенно тычет пальцем в незнакомую собаку во дворе. Машинное обучение устроено так же. Нужны **примеры** и **правильные ответы** к ним.

Примеры для машины записывают в **таблицу**. Одна строка – один **объект**: один день, один ученик, одна фотография, один астероид. Столбцы – то, что мы знаем про объект.

Дата	Температура, °С	Облачность, %	Давление, мм рт. ст.	Снег завтра
12.12	–8	90	742	да
13.12	–15	40	758	нет
14.12	–3	100	735	да
15.12	–21	10	765	нет

← один объект (одна строка)

не признак

признаки – по ним угадываем

ответ

Рисунок 1.4 – Таблица для обучения. Каждый столбец играет свою роль

Столбцы бывают трёх видов:

- **Признаки** – то, по чему модель угадывает ответ. Здесь это температура, облачность и давление. Признак может быть числом (–8 °С) или словом (отсек «А», цвет «красный»).

- **Ответ**, или **целевая переменная**, – то, что модель должна научиться угадывать. Здесь – «снег завтра: да или нет». В обучающих примерах ответ известен, а для нового дня его как раз и предсказывают.
- **Идентификаторы** – номер, фамилия, код пропуска. Они нужны людям, чтобы отличать строки друг от друга, но ничего не говорят об ответе. Снег идёт не потому, что сегодня 12-е число. Перед обучением такие столбцы убирают: модель может «зацепиться» за номер строки и начать угадывать по нему, а это ерунда. Но если в коде защиты свойства (в номере каюты – палуба и сторона борта), такой код может быть признаком: удаляют только то, что лишь называет строку.

Слово **модель** будет встречаться постоянно. Модель – это то, что получилось после обучения: правило, которое программа нашла в примерах и теперь применяет к новым объектам.

Учиться можно по-разному:

- **Обучение с учителем** – в примерах есть правильные ответы. Как в задачнике с ответами в конце: решил, сверился, исправился. Так решают классификацию и регрессию (раздел 1.3).
- **Обучение без учителя** – ответов нет совсем. Модели дают кучу объектов и просят найти в них порядок: разбить на похожие группы (кластеризация) или оставить меньше признаков, сохранив главное (снижение размерности: например, выбрать несколько главных цветов картины).
- **Обучение с подкреплением** – ответов нет, но есть награда и штраф. Так учат играть в игры: выиграл – молодец, упал в яму – минус очки. Играющий – **агент**, игра вокруг него – **среда**, очки – **награда** и **штраф**. Модель пробует снова и снова и выбирает действия, за которые хвалят.

Запомни. Строка таблицы – объект. Столбцы, по которым угадываем, – признаки, а столбец, который надо угадать, – ответ (целевая переменная). Номера и фамилии, которые лишь называют строку, – не признаки: перед обучением их удаляют вместе с ответом.

Совет. Спроси себя про каждый столбец: «Если я узнаю это значение, мне станет легче угадать ответ?» Температура – да, значит, признак. Номер строки – нет, значит, идентификатор.

1.3. Типы задач: классификация, регрессия, кластеризация

НА ПАЛЬЦАХ

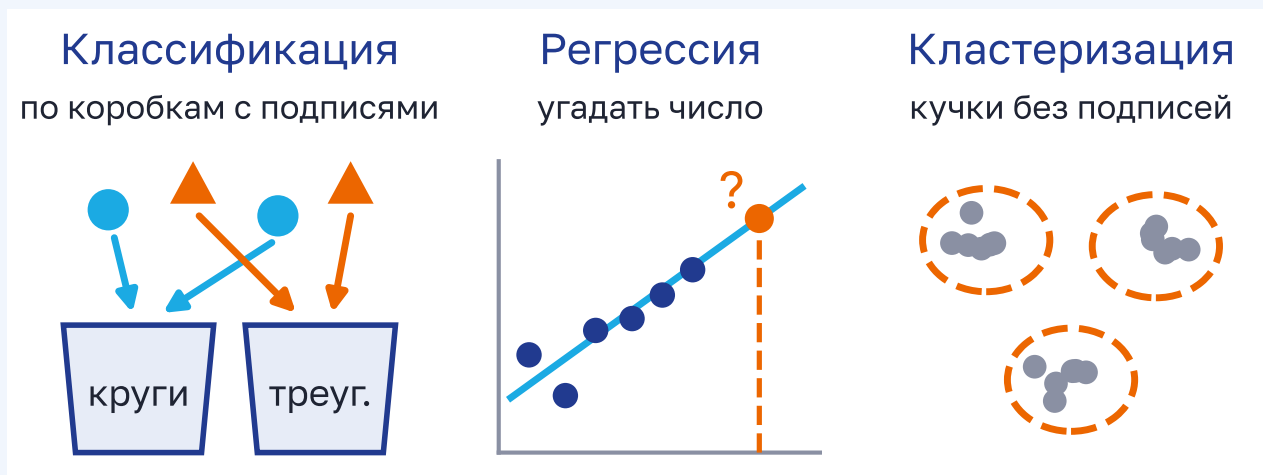


Рисунок 1.5 – Три типа задач машинного обучения

Классификация раскладывает по коробкам, на которых уже есть подписи. Регрессия угадывает число. Кластеризация сама находит кучки, и подписей у них нет.

Задачи машинного обучения делят на типы по тому, **какой ответ** нужно получить. На отборе тип часто спрашивают прямо: дано описание, выбери тип.

Классификация (classification) – отнести объект к одной из **заранее известных групп** (их называют классами). Письмо: спам или не спам. Снимок: кошка, собака или попугай. Астероид: опасен или нет. Классов может быть два (тогда ответ «да или нет») или больше, но список классов известен заранее.

Регрессия (regression) – предсказать **число**, которое можно измерить или посчитать: сколько сантиметров снега выпадет, сколько минут ехать до школы, сколько стоит квартира, какой будет температура. Такие числа можно сравнивать: 13 см больше 12 см, а прогноз 12 см при настоящих 13 см – почти попадание.

Кластеризация (clustering) – разбить объекты на группы похожих, когда **правильных ответов нет**. Никто заранее не говорил, какие будут группы и сколько их. Галерея телефона собирает фото одного и того же лица в альбом, хотя имени человека не знает, – это кластеризация. Магазин делит покупателей на группы по покупкам, чтобы понять, кто к ним ходит, – тоже она.

Как узнать тип по условию – два вопроса по порядку:



Рисунок 1.6 – Как по условию узнать тип задачи

Смотри на ответ, а не на слова вокруг. Слово «прогноз» ничего не решает: «прогноз, пойдёт ли завтра снег» – классификация (да или нет), а «прогноз, сколько сантиметров снега выпадет» – регрессия.

Спотыкаются тут в двух местах. Бывает, что **ответ записан цифрой, но это номер класса**. «Отнести астероид к классу опасности 1, 2 или 3» – классификация: класс 2 не «вдвое больше» класса 1, а класса 2,5 не бывает. А бывает, что **группы есть, но подписи неизвестны**. «Разложи 500 снимков на кошек и собак» – классификация, если в обучающих снимках подписано, где кто. «Разложи 500 снимков на группы похожих» без подписей – кластеризация.

Запомни. Готовые группы и ответы в примерах – классификация. Число – регрессия. Ответов нет, ищем группы сами – кластеризация. Первым делом смотри, **что должно получиться на выходе**.

1.4. Обучающая и проверочная выборка

НА ПАЛЬЦАХ



Рисунок 1.7 – Одни примеры – для учёбы, другие – для проверки

На одних примерах модель учится и видит ответы. На других – пишет контрольную. Задания контрольной заранее не показывают.

Допустим, у нас 1000 примеров с ответами. Если учить модель на всех 1000 и на них же проверять, мы ничего не узнаем. Модель могла просто запомнить ответы. Это как выучить наизусть ответы из конца задачника: на тех же задачах – пятёрка, а на контрольной с новыми числами – двойка.

Поэтому примеры заранее делят на две части. Каждую часть называют **выборкой**:

- **Обучающая выборка** (её ещё называют тренировочной) – на ней модель учится, ответы видны. Обычно это большая часть: 70–80 %.
- **Проверочная выборка** (тестовая) – её откладывают и во время обучения модели не показывают. Когда обучение закончено, модель отвечает на эти примеры, а мы сравниваем её ответы с настоящими.

Пример. Из 1000 примеров 80 % идут в обучающую выборку: $1000 \cdot 80/100 = 800$. В проверочную – остальные: $1000 - 800 = 200$.

Когда модель отлично отвечает на обучающих примерах и плохо на новых, говорят, что она **переобучилась**: выучила примеры вместо того, чтобы понять закономерность. Заметить это можно только на новых примерах – вот почему **точность модели считают на проверочной выборке**, а не на обучающей.

Запомни. Обучающая выборка – учимся, проверочная – сдаём экзамен. Проверочные примеры модель при обучении не видит. Точность, которой можно верить, – только на проверочной выборке.

1.5. Нейрон – весы с порогом

НА ПАЛЬЦАХ

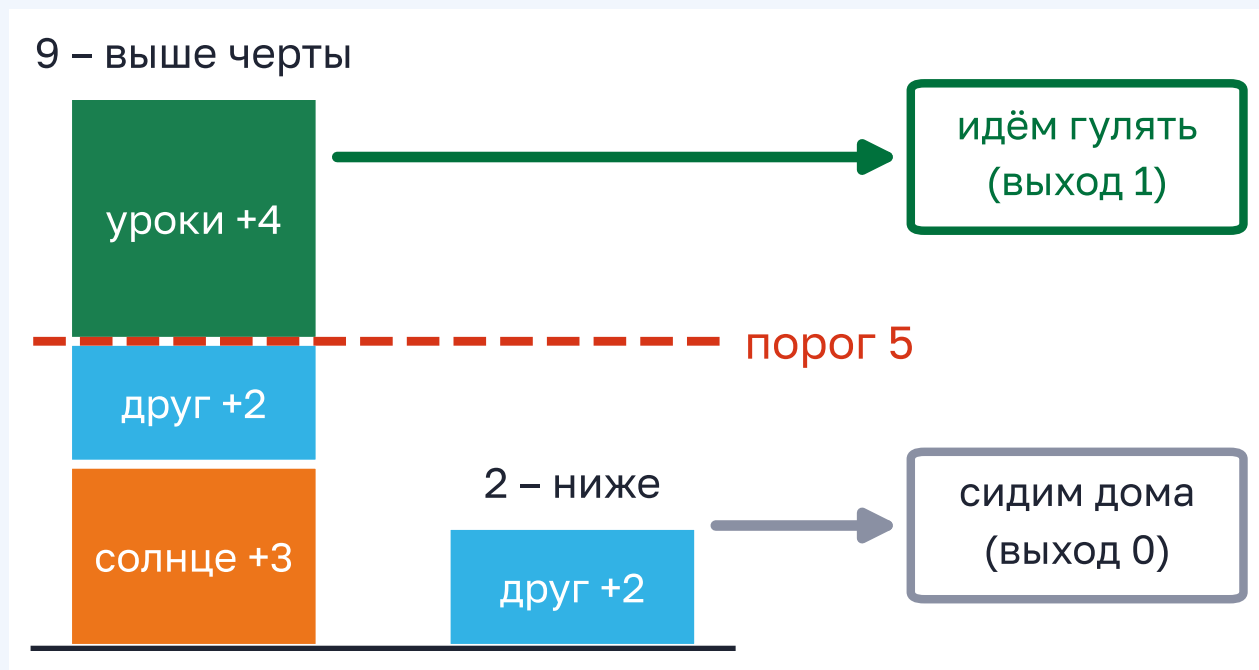


Рисунок 1.8 – Кубики «за» ставим друг на друга и сравниваем с чертой

Каждая причина «за» – кубик своей высоты. Башня выросла выше черты – решение «да». Не доросла – «нет».

Как ты решаешь, пойти ли гулять? У каждой причины свой «вес». Солнце – приятно, +3. Друг свободен – +2. Уроки уже сделаны – сразу +4, это важнее всего. Складываем то, что совпало, и сравниваем с внутренней чертой – **порогом**. Набралось 9, а порог 5 – идёшь. Свободен только друг – всего 2, сидишь дома.

Так же устроен **искусственный нейрон** – маленький кирпичик нейросети:

1. На вход приходят числа $x_1, x_2, x_3, \dots$ – это признаки объекта. Часто это 1 («да, есть») и 0 («нет»), но бывают и любые числа: температура, высота камня.
2. У каждого входа есть **вес** $w_1, w_2, w_3, \dots$ – число, которое говорит, насколько этот вход важен. Большой вес – сильно влияет, маленький – слабо. **Отрицательный** вес тянет против: мороз -30°C – повод не идти на улицу.
3. Нейрон умножает каждый вход на его вес и всё складывает. Получается **сумма** S .
4. Сумму сравнивают с **порогом** T . Если S не меньше порога – на выходе 1, иначе 0.

Словами: **сумма равна первому входу, умноженному на свой вес, плюс второму, умноженному на свой вес, и так далее.** Буквами:

$$S = x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3.$$

Если $S \geq T$ – на выходе 1. Если $S < T$ – на выходе 0. Здесь x – входы, w – веса, T – порог, знак $\geq$ читается «больше или равно».

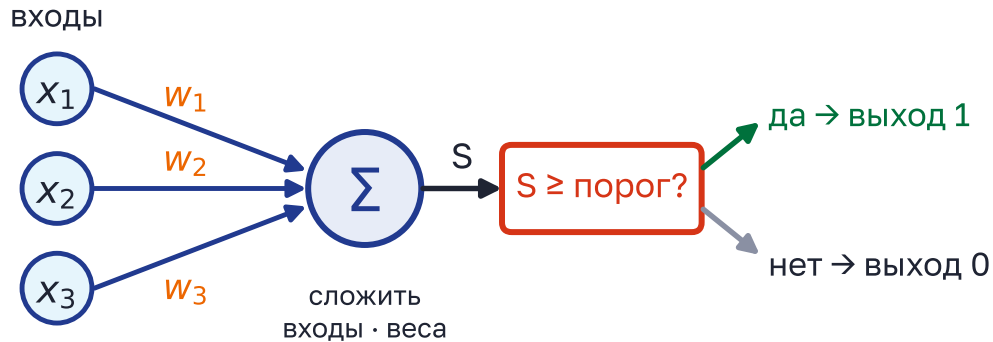


Рисунок 1.9 – Искусственный нейрон: входы умножаются на веса, складываются, сумма сравнивается с порогом

Пример. Нейрон решает, идти ли гулять: $w_1 = 3$ (солнце), $w_2 = 2$ (друг свободен), $w_3 = 4$ (уроки сделаны), порог $T = 5$. Сегодня солнца нет, друг свободен, уроки сделаны: $x = (0, 1, 1)$. $S = 0 \cdot 3 + 1 \cdot 2 + 1 \cdot 4 = 6$. $6 \geq 5$ – выход 1, идём гулять.

Тот же расчёт на Python. Если ты ещё не писал программ, читай по строчке:

```
w = [3, 2, 4]           # веса: солнце, друг, уроки
x = [0, 1, 1]           # входы: 1 – да, 0 – нет
T = 5                   # порог

S = 0                   # сумма пока равна нулю
for xi, wi in zip(x, w): # берём пары «вход – вес» по очереди
    S = S + xi * wi      # добавляем к сумме вход, умноженный на вес
if S ≥ T:
    print(S, "выход 1")
else:
    print(S, "выход 0")
# 6 выход 1
```

- **w** и **x** – **списки**: несколько чисел в квадратных скобках через запятую.
- Всё после знака **#** – комментарий, пояснение для человека. Компьютер его пропускает.
- **zip(x, w)** складывает два списка в пары: (0 и 3), (1 и 2), (1 и 4). **for** проходит по этим парам, и строка с отступом под ним повторяется для каждой пары. После трёх повторов в **S** лежит $0 + 2 + 4 = 6$.
- **if S ≥ T**: – «если сумма больше или равна порогу», **else**: – «иначе». **print** выводит на экран.

Опытные программисты пишут сумму в одну строку: `S = sum(xi * wi for xi, wi in zip(x, w))`. Это то же самое, **sum** складывает все произведения. Такая запись встретится в проверках ниже.

Как нейрон учится. Сначала веса ставят наугад, и нейрон отвечает неважно. На каждом обучающем примере его ответ сравнивают с правильным. Ошибся – веса чуть-чуть подкручивают так, чтобы ответ стал ближе к верному. Тысячи таких поправок – и веса становятся

«правильными». Вот что значит «модель обучилась»: подобрались веса. Первый такой обучаемый нейрон – **перцептрон** – построил американский учёный Фрэнк Розенблатт в 1957 году.

Нейросеть – это много нейронов, соединённых слоями. Выход одних нейронов – вход для следующих.

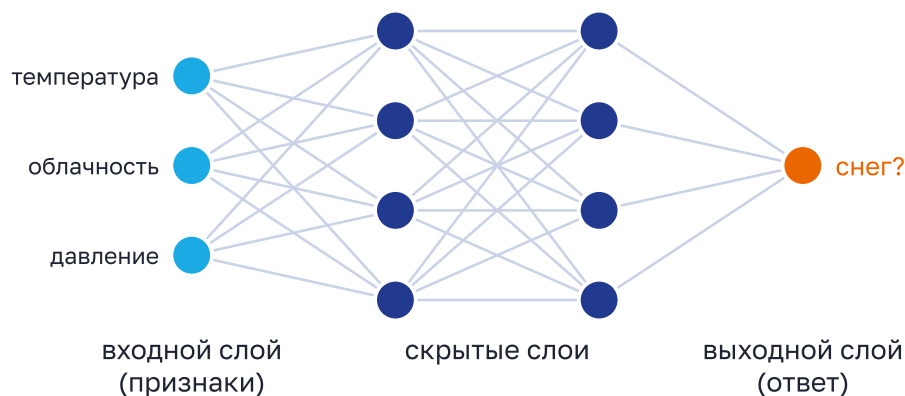


Рисунок 1.10 – Нейросеть из слоёв: входной слой получает признаки, выходной выдаёт ответ

- **Входной слой** получает признаки объекта: сколько признаков, столько входов.
- **Скрытые слои** – нейроны внутри сети. Их не видно снаружи, отсюда название. Чем их больше, тем более сложные закономерности сеть может уловить.
- **Выходной слой** выдаёт ответ: «снег да или нет», номер класса, число.

Правило, по которому нейрон превращает сумму в свой выход, называется **функцией активации**. Наш порог «1 или 0» – самая простая функция активации, ступенька.

Для любопытных. В настоящих нейросетях ступенька встречается редко: она слишком резкая, и подкручивать веса по ней неудобно. Часто берут функцию ReLU: отрицательную сумму она заменяет нулём, а положительную пропускает как есть. Сумма -3 даст 0, сумма 2,5 даст 2,5. Вместо порога часто пишут **смещение** b : выход 1, если $S + b \geq 0$. Это одно и то же: порог 5 – всё равно что смещение -5 .

Запомни. Нейрон: умножь каждый вход на его вес, сложи, сравни сумму с порогом. $S \geq T$ – выход 1, иначе 0. Отрицательный вес работает «против». Обучение – это подбор весов.

Совет. Считай нейрон по строчке для каждого входа: « $1 \cdot 5 = 5$, $0 \cdot 2 = 0$, $1 \cdot (-3) = -3$ ». Отрицательный вес пиши в скобках – так минус не потеряется. И проверь по условию, что делать при S ровно равном порогу.

Если проверяешь кодом, а веса дробные, есть ловушка. Компьютер хранит десятичные дроби чуть-чуть неточно: $1.5 * 1.2$ он считает как 1.7999999999999998 . Обычно это незаметно. Но когда сумма ровно на пороге, ошибка решает ответ: $0.9999999999999998 \geq 1$ – это False, выход 0 вместо 1. Так что на границе досчитай вручную или округли сумму перед сравнением: $S = \text{round}(S, 6)$ оставит 6 знаков после точки.

1.6. Точность и ошибка

НА ПАЛЬЦАХ

модель ответила на 10 новых примеров



8 верных из 10 = $8 : 10 = 80 \%$

Рисунок 1.11 – Десять ответов модели: восемь верных, два неверных

Точность – доля верных ответов. Восемь верных из десяти – это 80 %.

Обучили модель – как понять, хороша ли она? Даём ей проверочную выборку, записываем её ответы рядом с настоящими и отмечаем, где угадала, а где нет.

Таблица 1.1

№	На самом деле	Ответ модели	Угадала?
1	спам	спам	да
2	не спам	не спам	да
3	не спам	спам	нет
4	спам	спам	да
5	не спам	не спам	да

Точность (accuracy) – доля верных ответов. Словами: **точность равна числу верных ответов, делённому на число всех ответов**. Обычно её переводят в проценты:

$$\text{точность} = \frac{\text{верных}}{\text{всего}} \cdot 100 \%$$

В таблице верных ответов 4 из 5: $4/5 \cdot 100 = 80 \%$. **Доля ошибок** – всё остальное: $100 - 80 = 20 \%$, то есть одна ошибка из пяти.

Ту же точность можно посчитать программой – удобно, когда примеров сотня. Счётчик `vernyh` сначала равен 0. Запись `vernyh += 1` – короткое `vernyh = vernyh + 1`: прибавить единицу. Двойное `==` значит «равно ли?». `len` – длина списка, то есть сколько в нём элементов:

```
nastoyashchie = ["спам", "не спам", "не спам", "спам", "не спам"]
otvety_modeli = ["спам", "не спам", "спам", "спам", "не спам"]

vernyh = 0
for a, b in zip(nastoyashchie, otvety_modeli):
    if a == b:
        # ответ модели совпал с настоящим
        vernyh += 1
print(vernyh, "из", len(nastoyashchie))
print("точность", vernyh / len(nastoyashchie) * 100, "%")
```

И ошибки не все одинаковые. Модель может **пропустить** – сказать «не спам» про настоящий спам. А может поднять **ложную тревогу** – назвать спамом письмо от бабушки. В таблице выше одна ошибка, и это ложная тревога (строка 3). Для почты ложная тревога хуже: письмо бабушки уйдёт в спам, и его никто не прочтёт.

Ловушка редких случаев. Представь: из 100 астероидов опасен один. «Ленивая» модель на всё отвечает «не опасен» – и угадывает 99 раз из 100, точность 99 %. Звучит отлично, а толку ноль: единственный опасный астероид она пропустила. Поэтому, когда один класс встречается редко, смотрят не только на точность, но и на то, сколько редких случаев модель нашла.

Ошибка в регрессии. Когда модель предсказывает число, «угадала или нет» не подходит: прогноз -12°C при настоящих -11°C почти верный. Здесь считают, **на сколько** модель ошиблась в каждом примере (разность без знака минус), и находят **среднюю ошибку**. Прогноз -10 , -9 , -13 при настоящих -12 , -8 , -13 : ошибки 2, 1 и 0, средняя $(2 + 1 + 0) / 3 = 1$ градус. Чем меньше средняя ошибка, тем лучше модель. Правило, по которому считают ошибку модели, называют **функцией потерь** (loss), а число, по которому модели сравнивают, – **метрикой**. Точность и средняя ошибка – две самые простые метрики.

Запомни. Точность = верных : всего · 100 %. Доля ошибок = 100 % – точность. Считаем на проверочной выборке. Для регрессии – средняя ошибка: насколько в среднем промахнулись, без учёта знака.

Совет. Сначала заполни столбец «угадала?» для каждой строки, потом считай галочки. Не пытайся найти ошибки, скользая взглядом по двум столбцам сразу, – одну обязательно пропустишь. Итог проверь: верных + неверных = всего.

1.7. Разбираем задачи

Задача 1.1 Какой это тип задачи

разминка · типы задач машинного обучения

Сопоставь каждую задачу с её типом: 1 – классификация, 2 – регрессия, 3 – кластеризация.

А. По скорости и размеру астероида отнести его к одному из трёх классов опасности: 1, 2 или 3 (в обучающих примерах класс указан).

Б. Предсказать, сколько сантиметров снега выпадет завтра в Новосибирске.

- В. Разбить 5000 снимков звёздного неба на группы похожих; какие будут группы, заранее неизвестно.
- Г. По письму определить, спам это или нет.
- Д. По температуре и заряду аккумулятора лунохода предсказать, сколько минут он проработает до разрядки.
- Е. Разделить покупателей магазина на группы по их покупкам, ничего не зная о покупателях заранее.

Ответ запиши как шесть цифр подряд – по порядку для А, Б, В, Г, Д, Е.

Дано: шесть описаний задач.

Найти: тип каждой.

Решение

Шаг 1. Правило. Для каждой задачи задаём два вопроса из раздела 1.3. Есть ли в примерах правильные ответы? Если есть – что получается на выходе: число или одна из готовых групп?

Шаг 2. А. Ответ – класс 1, 2 или 3, список классов известен, в примерах он указан. Цифры здесь – имена классов, а не величины: класса 2,5 не бывает. Классификация – **1**.

Шаг 3. Б. Ответ – число сантиметров, может быть и 12,5. Регрессия – **2**.

Шаг 4. В. Правильных ответов нет, группы надо найти самим. Кластеризация – **3**.

Шаг 5. Г. Два готовых класса: «спам» и «не спам». Классификация – **1**.

Шаг 6. Д. Ответ – число минут. Регрессия – **2**.

Шаг 7. Е. Ответов нет, группы неизвестны. Кластеризация – **3**.

Перепроверь кластеризацию. Только в В и Е прямо сказано, что групп заранее никто не знает. В остальных четырёх ответы в примерах есть, значит, там классификация или регрессия.

 **Ошибка номер один** – записать задачу А в регрессию, потому что ответ – цифра. Цифра – ещё не число-величина: номер класса, номер автобуса, номер отсека – это имена. А задачу В часто принимают за классификацию из-за слова «группы». Группы в классификации известны заранее и подписаны, а здесь их надо найти.

Ответ: 123123.

Своя задача по программе сферы, модуль 1

Реши сам

1.1. Определи тип задачи (1 – классификация, 2 – регрессия, 3 – кластеризация): а) по росту и весу собаки предсказать её возраст в годах; б) по записи голоса птицы определить, кто поёт: синица, воробей или снегирь (в обучающих записях это подписано); в) разложить 300 снимков комет на группы похожих, не зная заранее, какие будут группы; г) предсказать, сколько человек придёт на новогоднюю ёлку в Академгородке. Ответ – четыре цифры подряд.

1.2. Какие из этих устройств и программ используют машинное обучение (учились на примерах)? 1) калькулятор в телефоне; 2) разблокировка телефона по лицу; 3) светофор, который переключается каждые 40 секунд; 4) голосовой помощник, который распознаёт твою речь; 5) электрочайник, который выключается при закипании; 6) лента видео, которая подбирает ролики по тому, что ты смотрел. Запиши номера в порядке возрастания.

Задача 1.2 Какие столбцы убрать

разминка · признаки, целевая переменная, идентификаторы

Врач лунной базы хочет научить модель предсказывать, заболит ли житель базы простудой в ближайший месяц. У него есть таблица:

Таблица 1.2

Номер пропуска	Имя	Возраст	Отсек	Спорт, ч в неделю	Сон, ч в сутки	Заболел за месяц
1041	Ирина	34	А	5	7	нет
1057	Олег	41	Б	1	6	да
1063	Самир	28	А	3	8	нет

а) Какой столбец – целевая переменная? б) Какие столбцы нужно удалить, чтобы остались только признаки? в) Какой это тип задачи?

Дано: таблица из семи столбцов.

Найти: целевую переменную, лишние столбцы, тип задачи.

Решение

Шаг 1. Что угадываем. Врач хочет знать, заболит ли человек. В таблице это столбец «Заболел за месяц» – наша целевая переменная (ответ).

Шаг 2. Идентификаторы. Проверяем каждый столбец вопросом «помогает ли это угадать ответ?». Номер пропуска – нет: 1057 не болеет чаще, чем 1041. Имя – тоже нет: простуда не выбирает людей по имени. Это идентификаторы.

Шаг 3. Признаки. Возраст, часы спорта, часы сна – могут влиять на здоровье, это признаки. Отсек записан буквой, но и он может влиять: вдруг в отсеке Б сквозняк. Буква – не повод удалять столбец. Признак.

Шаг 4. Что удалить. Чтобы остались только признаки, убираем ответ и два идентификатора: «Номер пропуска», «Имя», «Заболел за месяц». Остаются четыре признака.

Шаг 5. Тип. Ответ – «да» или «нет», два готовых класса, в примерах они указаны. Классификация.

Пересчитаем. $7 - 3 = 4$ признака: возраст, отсек, спорт, сон. И про каждый легко представить, как он связан с простудой.

❗ **Типичная ошибка.** Удалить «Отсек», потому что там не число, и забыть удалить ответ. Если оставить «Заболел за месяц» среди признаков, модель будет «угадывать» ответ, подглядывая в сам ответ, – на обучении у неё будет 100 %, а на новом человеке (у которого этого столбца ещё нет) она работать не сможет.

Ответ: а) «Заболел за месяц»; б) «Номер пропуска», «Имя», «Заболел за месяц»; в) классификация.

Своя задача по программе сферы, модуль 3

Реши сам

1.3. Наставник кружка хочет по таблице предсказать, пройдёт ли ученик в финал олимпиады. Столбцы таблицы: «Фамилия», «Номер в журнале», «Класс», «Часов занятий в неделю», «Решено задач на пробном туре», «Прошёл в финал». Какие столбцы нужно удалить, чтобы остались только признаки? Какой это тип задачи?

1.4. Сопоставь термины и определения. Термины: 1) обучающая выборка; 2) проверочная выборка; 3) признак; 4) целевая переменная; 5) идентификатор. Определения: А) столбец, значение которого модель должна угадать; Б) примеры, которые модели не показывают во время обучения, чтобы потом оценить её; В) столбец, по которому различают строки, но который ничего не говорит об ответе; Г) примеры с ответами, на которых модель учится; Д) свойство объекта, по которому модель угадывает ответ. Ответ запиши как пять букв по порядку для терминов 1–5.

Задача 1.3 Модель ищет метеоры

точность, таблица предсказаний, пропуски и ложные тревоги

Камера обсерватории снимает ночное небо, а модель решает, есть на снимке метеор (М) или нет (Н). Модель проверили на 12 снимках проверочной выборки:

Таблица 1.3

Снимок	На самом деле	Ответ модели
1	М	М
2	Н	Н
3	Н	М
4	М	М
5	М	Н
6	Н	Н
7	Н	Н
8	М	Н

Снимок	На самом деле	Ответ модели
9	Н	Н
10	Н	Н
11	М	М
12	Н	Н

а) Найди точность модели в процентах. б) Сколько настоящих метеоров модель пропустила?

Дано: 12 снимков, настоящие ответы и ответы модели.

Найти: точность, %; число пропущенных метеоров.

Решение

Шаг 1. Отмечаем каждую строку. Идём по снимкам сверху вниз и сравниваем два столбца таблицы:

- совпали – снимки 1, 2, 4, 6, 7, 9, 10, 11, 12;
- **не совпали** – снимок 3 (на самом деле Н, модель – М), снимки 5 и 8 (на самом деле М, модель – Н).

Шаг 2. Считаем. Неверных ответов 3 (снимки 3, 5, 8), верных $12 - 3 = 9$.

Шаг 3. Точность. $9/12 \cdot 100 = 0,75 \cdot 100 = 75 \%$.

Шаг 4. Какие это ошибки. На снимке 3 метеора не было, а модель сказала «М» – ложная тревога. На снимках 5 и 8 метеор был, а модель сказала «Н» – это пропуски. Пропущено **2** метеора.

Сверим с другой стороны. Всего настоящих метеоров 5 (снимки 1, 4, 5, 8, 11). Модель нашла 3 из них и пропустила 2: $3 + 2 = 5$ – сходится. Ошибок 3 из 12 – четверть, и точность 75 % – как раз три четверти.

 **Где ошибаются.** Считают только снимки с метеорами: «нашла 3 из 5 – точность 60 %». Точность – доля верных ответов **среди всех** снимков, и снимки без метеора, где модель правильно промолчала, тоже считаются верными. А в пункте б) все 3 ошибки записывают в пропуски, хотя снимок 3 – это ложная тревога.

Ответ: а) 75 %; б) 2 метеора.

Своя задача по программе сферы, модуль 1

Реши сам

1.5. Модель 10 дней предсказывала, пойдёт ли снег (С – снег, Н – нет):

Таблица 1.4

День	На самом деле	Прогноз модели
1	С	С
2	Н	С
3	Н	Н
4	С	С
5	С	Н
6	Н	Н
7	С	С
8	Н	С
9	Н	Н
10	С	С

а) Найди точность модели в процентах. б) Сколько раз модель обещала снег, а его не было?

1.6. В проверочной выборке 400 астероидов, из них опасных только 12. «Ленивая» модель про любой астероид отвечает «не опасен». а) Какова её точность в процентах? б) Сколько опасных астероидов она пропустит?

1.7. Две модели 5 дней предсказывали температуру ночью. Настоящая температура: $-12, -8, -15, -20, -5$ °С. Модель А предсказала: $-10, -9, -12, -21, -7$. Модель Б: $-12, -6, -15, -16, -5$. Найди среднюю ошибку каждой модели (в градусах, без учёта знака) и назови лучшую модель.

Задача 1.4 Открывать ли купол телескопа

искусственный нейрон, веса, порог

Нейрон решает, открывать ли купол телескопа на ночь. Входы (1 – да, 0 – нет) и их веса:

- x_1 – небо ясное, вес $w_1 = 5$;
- x_2 – ветер слабый, вес $w_2 = 2$;
- x_3 – полная Луна (засвечивает небо), вес $w_3 = -3$;
- x_4 – в обсерватории есть дежурный, вес $w_4 = 3$.

Порог $T = 6$. Если сумма не меньше порога, выход 1 – купол открывают, иначе 0. Что ответит нейрон в три ночи?

- Ночь 1: ясно, ветер сильный, полная Луна, дежурный есть.
- Ночь 2: ясно, ветер слабый, Луна не полная, дежурный есть.

- Ночь 3: облачно, ветер слабый, Луна не полная, дежурный есть.

Ответ запиши тремя цифрами подряд – выходы нейрона в ночи 1, 2, 3.

Дано: $w = (5; 2; -3; 3)$; $T = 6$; три набора условий.

Найти: выход нейрона для каждой ночи.

Решение

Шаг 1. Переводим слова во входы. Каждое условие – 1, если оно выполнено, и 0, если нет. Внимательно с x_2 : вход означает «ветер **слабый**», значит, сильный ветер – это 0.

- Ночь 1: $x = (1, 0, 1, 1)$.
- Ночь 2: $x = (1, 1, 0, 1)$.
- Ночь 3: $x = (0, 1, 0, 1)$.

Шаг 2. Ночь 1. $S = 1 \cdot 5 + 0 \cdot 2 + 1 \cdot (-3) + 1 \cdot 3 = 5 + 0 - 3 + 3 = 5$. $5 < 6$ – выход **0**. Полная Луна «отняла» 3, и ясного неба не хватило.

Шаг 3. Ночь 2. $S = 1 \cdot 5 + 1 \cdot 2 + 0 \cdot (-3) + 1 \cdot 3 = 5 + 2 + 0 + 3 = 10$. $10 \geq 6$ – выход **1**.

Шаг 4. Ночь 3. $S = 0 \cdot 5 + 1 \cdot 2 + 0 \cdot (-3) + 1 \cdot 3 = 0 + 2 + 0 + 3 = 5$. $5 < 6$ – выход **0**.

Проверка кодом.

```
w = [5, 2, -3, 3]
T = 6
for x in ([1, 0, 1, 1], [1, 1, 0, 1], [0, 1, 0, 1]):
    S = sum(xi * wi for xi, wi in zip(x, w))
    print(S, 1 if S ≥ T else 0)    # 1, если S ≥ T, иначе 0
# 5 0
# 10 1
# 5 0
```

А если без чисел? Купол открыли только в ясную безлунную ночь. Через облака смотреть нечего, так что ночь 3 – правильно «нет». Нейрон ведёт себя как разумный астроном.

❗ **Главная ловушка** – потерять минус у веса -3 и получить в ночь 1 сумму $5 + 3 + 3 = 11$ – «открываем». Бывает и так: ставят $x_2 = 1$ в ночь 1, потому что «про ветер что-то сказано». Вход равен 1, только когда выполнено именно то, что написано в таблице.

Ответ: 010.

Своя задача по программе сферы, модуль 1

Реши сам

1.8. Нейрон решает, ехать ли в выходной кататься на лыжах. Входы (1 – да, 0 – нет) и веса: выпал свежий снег – 4; мороз сильнее -25°C – (–6); сегодня выходной – 3; лыжи смазаны – 1. Порог 5, при сумме не меньше порога выход 1. Что ответит нейрон в три дня? День 1: свежий снег, мороз слабый, выходной, лыжи не смазаны. День 2: свежий снег, мороз -30°C , выходной, лыжи смазаны. День 3: снега нового нет, мороз слабый, выходной, лыжи смазаны. Ответ – три цифры подряд.

1.9. У нейрона три входа: $x_1 = 0,5$, $x_2 = 2$, $x_3 = 1,5$. Веса: $w_1 = 1,5$, $w_2 = -0,5$, $w_3 = 1,5$. Порог $T = 2$; если сумма не меньше порога, выход 1, иначе 0. Найди сумму S и выход нейрона.

Задача 1.5 Облака на снимках

обучающая и проверочная выборка, проценты, точность

Метеостанция собрала 2400 снимков неба с подписями «будет дождь» или «не будет». 75 % снимков отдали в обучающую выборку, остальные – в проверочную. На проверочной выборке модель ошиблась на 42 снимках. Найди точность модели на проверочной выборке в процентах.

Дано: всего 2400 снимков; обучающая выборка – 75 %; ошибок на проверочной – 42.

Найти: точность, %.

Решение

Шаг 1. Где считаем точность. На проверочной выборке. Значит, сначала узнаём, сколько в ней снимков.

Шаг 2. Обучающая выборка. $2400 \cdot 75/100 = 1800$ снимков.

Шаг 3. Проверочная выборка. $2400 - 1800 = 600$ снимков. (Или сразу: $100 - 75 = 25$ %, $2400 \cdot 25/100 = 600$.)

Шаг 4. Верные ответы. $600 - 42 = 558$.

Шаг 5. Точность. $558/600 \cdot 100 = 93$ %.

Проверка разумности. 42 ошибки из 600 – ошибается примерно на каждом четырнадцатом снимке. $42/600 \cdot 100 = 7$ % ошибок, и $93 + 7 = 100$.

 **Типичная ошибка.** Разделить на все 2400 снимков: $(2400 - 42)/2400 \cdot 100 = 98,25$ %. На обучающих снимках модель не проверяли, ошибки на них никто не считал. Ещё обиднее – ответить «7 %»: это доля ошибок, а спрашивали точность.

Ответ: 93 %.

Своя задача по программе сферы, модуль 1

Реши сам

1.10. В наборе данных 1500 записей о погоде. 80 % из них – обучающая выборка, остальные – проверочная. Сколько записей в проверочной выборке?

1.11. Точность модели на проверочной выборке из 240 примеров – 85 %. На скольких примерах модель ошиблась?

Задача 1.6 Чат-бот школьной библиотеки

классификация, точность, проверочная выборка

Чат-бот библиотеки должен понимать, чего хочет читатель. Разработчики задали ему три **интента** (намерения): «Посоветуй книгу» (П), «Часы работы» (Ч), «Продлить книгу» (Д). Для каждого интента написали по десятку тренировочных фраз, и бот на них обучился. Потом бота проверили на 10 новых фразах (после фразы – нужный интент и то, что решил бот):

1. «Что бы почитать про космос?» – нужен П, бот решил П.
2. «Во сколько вы открываетесь в субботу?» – нужен Ч, бот решил Ч.
3. «Можно поддержать «Незнайку» ещё неделю?» – нужен Д, бот решил П.
4. «Библиотека работает в каникулы?» – нужен Ч, бот решил Ч.
5. «Посоветуй фантастику» – нужен П, бот решил П.
6. «Хочу сдать книгу позже» – нужен Д, бот решил Д.
7. «До сколько сегодня можно прийти?» – нужен Ч, бот решил Ч.
8. «Есть что-нибудь смешное для 6 класса?» – нужен П, бот решил П.
9. «Продли мне, пожалуйста, книгу» – нужен Д, бот решил Д.
10. «Когда можно прийти за книгой про динозавров?» – нужен Ч, бот решил П.

а) Какой тип задачи машинного обучения решает бот, когда выбирает интент? б) Найди точность бота на этих фразах в процентах. в) Разработчики добавили фразу № 3 в тренировочные фразы интента «Продлить книгу», снова обучили бота и проверили на тех же 10 фразах. Получилось 90 %. Можно ли считать, что бот стал понимать читателей лучше на 10 %? Ответь «да» или «нет».

Дано: 3 интента; 10 проверочных фраз с нужным интентом и ответом бота.

Найти: тип задачи; точность, %; можно ли верить 90 %.

Решение

Шаг 1. Тип задачи. Бот относит фразу к одной из трёх заранее известных групп, а в тренировочных фразах для каждой указан правильный интент. Это классификация (на три класса).

Шаг 2. Отмечаем ответы. Сравниваем нужный интент с ответом бота. Не совпали фразы 3 (нужно Д, бот – П) и 10 (нужно Ч, бот – П). Верных $10 - 2 = 8$.

Шаг 3. Точность. $8/10 \cdot 100 = 80 \%$.

Шаг 4. Почему бот ошибся. В фразе 3 есть название книги, и бот зацепился за него: «про книгу – значит, советовать». В фразе 10 тоже есть «книга», хотя спрашивают о времени. Бот часто реагирует на отдельные слова, а не на смысл.

Шаг 5. Можно ли верить 90 %. Фраза 3 теперь стала **тренировочной**: бот видел её вместе с правильным ответом. Когда мы снова проверяем на ней, это уже не контрольная, а задача, ответ к которой бот подсмотрел. Выросла точность только за счёт подсмотренной фразы. Узнать, стал ли бот лучше понимать **новых** читателей, можно только на фразах, которых он не видел. Ответ – **нет**.

Много это или мало? 2 ошибки из 10 – каждая пятая фраза уходит не туда. Для чат-бота это заметно: из пяти читателей один получит странный ответ. Есть что улучшать.

 **Самая частая ошибка** – в пункте в) ответить «да, ведь 90 больше 80». Проверочные примеры нельзя подкладывать в обучение: иначе точность выходит красивой, но ничего не говорит о новых фразах. В пункте а) – назвать задачу кластеризацией, потому что «бот раскладывает фразы по группам». Группы здесь заданы заранее и подписаны – значит, классификация.

Ответ: а) классификация; б) 80 %; в) нет.

По мотивам задачи отборочного этапа НТО Junior 2023/24 «Dialogflow. Книгомания»

Совет. В исходной задаче отбора бота-рекомендателя книг собирали в сервисе Dialogflow: создавали интенды, задавали тренировочные фразы и проверяли бота новыми вопросами в окне справа. Сервиса у тебя может и не быть, но идея та же, что в любой классификации: тренировочные фразы – обучающая выборка, новые вопросы – проверочная.

Реши сам

1.12. Модель, которая отличает снимки галактик от снимков звёзд, проверили на 50 снимках. Верно она ответила на 43. Найди её точность в процентах.

1.13. Верно (В) или неверно (Н)? 1) Все нейросети относятся к искусственному интеллекту. 2) Любая программа с командой «если... то...» – это искусственный интеллект. 3) Для кластеризации в примерах нужны правильные ответы. 4) Проверочную выборку не показывают модели во время обучения. 5) Если модель ответила верно на все примеры обучающей выборки, она точно будет хорошо работать на новых данных. Ответ – пять букв подряд.

Задача 1.7* Порог для лунохода

искусственный нейрон, подбор порога, неравенства

Луноход решает, объезжать камень или ехать прямо. Нейрон получает два числа: x_1 – высота камня в сантиметрах, x_2 – крутизна склона в градусах. Веса: $w_1 = 2$, $w_2 = 1$. Если сумма не меньше порога T , выход 1 – «объезжать», иначе 0 – «ехать прямо». Порог – целое число, его надо подобрать по пяти примерам, где инженеры уже знают правильный ответ:

Таблица 1.5

№	Высота камня, см	Крутизна, °	Ответ
1	10	5	ехать прямо
2	15	12	объезжать
3	8	20	объезжать
4	12	6	ехать прямо
5	5	18	ехать прямо

Сколько существует целых значений порога, при которых нейрон отвечает верно на все пять примеров? Укажи также наименьшее и наибольшее из них.

Дано: $w_1 = 2$, $w_2 = 1$; пять примеров с ответами; T – целое.

Найти: сколько целых T подходят, наименьшее и наибольшее T .

Решение

Шаг 1. Что происходит. Порог – черта. Для «объезжать» сумма должна быть не ниже черты, для «ехать прямо» – ниже. Значит, сначала посчитаем сумму для каждого примера, а потом найдём, куда можно провести черту.

Шаг 2. Суммы. $S = 2 \cdot x_1 + 1 \cdot x_2$:

- № 1: $2 \cdot 10 + 5 = 25$ (ехать прямо);
- № 2: $2 \cdot 15 + 12 = 42$ (объезжать);
- № 3: $2 \cdot 8 + 20 = 36$ (объезжать);
- № 4: $2 \cdot 12 + 6 = 30$ (ехать прямо);
- № 5: $2 \cdot 5 + 18 = 28$ (ехать прямо).

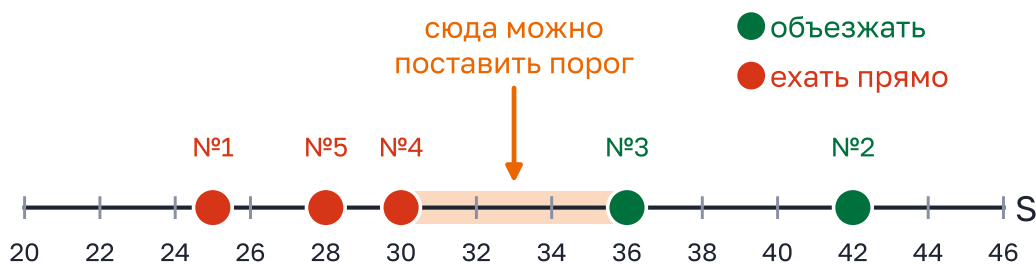


Рисунок 1.12 – Суммы пяти примеров на числовой прямой. Порог должен лечь между самой большой «красной» и самой маленькой «зелёной» суммой

Шаг 3. Условие для «объезжать». Нужно $S \geq T$ для примеров 2 и 3. Самая маленькая из этих сумм – 36. Если $36 \geq T$, то и $42 \geq T$. Значит, $T \leq 36$.

Шаг 4. Условие для «ехать прямо». Нужно $S < T$ для примеров 1, 4, 5. Самая большая из этих сумм – 30. Если $30 < T$, то и 25, и 28 меньше T . Значит, $T > 30$, то есть целое T не меньше 31.

Шаг 5. Собираем. Подходят целые T от 31 до 36: 31, 32, 33, 34, 35, 36. Их $36 - 31 + 1 = 6$.

Шаг 6. Проверяем края. $T = 31$: пример 4 даёт $30 < 31$ – «прямо», верно; пример 3 даёт $36 \geq 31$ – «объезжать», верно. $T = 36$: пример 3 даёт $36 \geq 36$ – «объезжать», верно (равен-

ство считается за «да»). $T = 37$ уже не подходит: $36 < 37$, и нейрон скажет про камень № 3 «прямо». $T = 30$ тоже не подходит: $30 \geq 30$, и про камень № 4 выйдет «объезжать».

Проверка кодом.

```
# (высота, крутизна, правильный ответ): 1 - объезжать, 0 - ехать прямо
primery = [(10, 5, 0), (15, 12, 1), (8, 20, 1), (12, 6, 0), (5, 18, 0)]
podhodyat = [] # сюда соберём подходящие пороги
for T in range(0, 100): # пробуем все пороги от 0 до 99
    vse_verno = True
    for h, k, otvet in primery:
        S = 2 * h + 1 * k
        vyhod = 1 if S ≥ T else 0
        if vyhod ≠ otvet: # ≠ значит «не равно»: нейрон ошибся
            vse_verno = False
    if vse_verno:
        podhodyat.append(T) # append добавляет число в конец списка
print(podhodyat, len(podhodyat)) # [31, 32, 33, 34, 35, 36] 6
```

Почему порогов целых шесть. Примеров всего пять, и на этих камнях все шесть порогов отвечают одинаково. Разница появится на новом камне: высота 11 см, склон 10° дают $S = 32$. При пороге 31 нейрон скажет «объезжать», при пороге 33 – «ехать прямо». Вот инженеры и собирают побольше примеров: каждый новый пример сужает промежуток.

❗ **Частый промах** – записать $30 \leq T$ и включить 30 в ответ – тогда пример 4 окажется «объезжать», потому что $30 \geq 30$. Где стоит знак «не меньше», а где «меньше», решает условие задачи, – проверяй границы подстановкой. А на самом финише теряют балл так: считают количество как $36 - 31 = 5$ и забывают, что оба края входят.

Ответ: 6 значений, от 31 до 36.

Своя задача по программе сферы, модуль 1

Реши сам

1.14. Дрон доставки решает, лететь или переждать непогоду. Нейрон получает два числа: x_1 – скорость ветра в м/с, x_2 – сила дождя в баллах. Веса: $w_1 = 3$, $w_2 = 2$. Если сумма не меньше порога T , выход 1 – «переждать», иначе 0 – «лететь». Порог – целое число. Примеры с правильными ответами: ветер 4, дождь 1 – лететь; ветер 6, дождь 3 – переждать; ветер 2, дождь 5 – лететь; ветер 7, дождь 1 – переждать; ветер 5, дождь 2 – лететь. Сколько существует целых значений порога, при которых нейрон отвечает верно на все пять примеров? Укажи наименьшее и наибольшее из них.

1.15.* У нейрона два входа, вес первого $w_1 = 1$, вес второго w_2 неизвестен, но это целое число. Порог $T = 10$, при сумме не меньше порога выход 1. Известны три примера: при входах (2; 3) выход 1; при входах (6; 1) выход 0; при входах (1; 4) выход 1. Найди w_2 . Что ответит нейрон на входы (7; 1)?

Множества и логика



Когда ты выбираешь в интернет-магазине фильтры «синий» и «до 1000 рублей», сайт ищет пересечение двух множеств. Когда спам-фильтр решает, пустить письмо во «Входящие» или нет, он проверяет правило из связок И, ИЛИ, НЕ. В этой главе научимся считать элементы, когда группы пересекаются, рисовать круги Эйлера и читать логические правила. На отборе такие задачи обычно короткие: история про книги, кружки или поисковые запросы, а в ответ вводится одно число или выбирается вариант.

2.1. Множество и его элементы

НА ПАЛЬЦАХ

Множество «игрушки в коробке»



Рисунок 2.1 – Множество – как обруч, в который сложили предметы

Всё, что лежит внутри обруча, – элементы множества. Обруч поменьше внутри большего – подмножество.

Множество – это набор каких-то предметов, собранных вместе по одному признаку. Игрушки в коробке, ученики твоего класса, буквы в слове, книги в школьной библиотеке – всё это множества. Каждый предмет из набора называется **элементом** множества. Мяч – элемент множества «игрушки в коробке», а твой сосед по парте – элемент множества «ученики 6 «Б»».

Множество записывают в фигурных скобках: $A = \{\text{мяч, кубик, машинка, самокат}\}$. Два правила отличают множество от обычного списка:

- **порядок не важен:** $\{1, 2, 3\}$ и $\{3, 1, 2\}$ – одно и то же множество;
- **повторов нет:** каждый элемент либо есть в множестве, либо его нет. В слове «КОЛОКОЛ» семь букв, но множество его букв – всего $\{К, О, Л\}$, три элемента.

Число элементов множества A обозначают $n(A)$. Для игрушек $n(A) = 4$, для букв слова «КОЛОКОЛ» – 3. Бывает и **пустое множество** – в нём нет ни одного элемента, $n = 0$. Например, множество учеников твоего класса ростом больше трёх метров.

Множество можно задать двумя способами. Можно **перечислить** элементы: $\{2, 4, 6, 8\}$. А можно **описать правилом**: «чётные числа от 1 до 9». Получится то же самое. Правилом удобнее, когда элементов много: «все книги библиотеки с отметкой 10+» перечислять долго, а описать можно одной строкой.

Подмножество – часть множества: каждый его элемент есть и в большом множестве. Игрушки с колёсами – подмножество игрушек в коробке. Девочки 6 «Б» – подмножество учеников 6 «Б». Два особых случая математики тоже считают подмножествами: всё множество целиком и пустое множество – в нём нет ни одного «чужого» элемента.

Зачем это ИИ? Нейросеть, которая учится узнавать котов на фото, получает **набор данных** – множество картинок. У части из них стоит метка «кот» – это подмножество. Когда говорят «модель ошиблась на 15 картинках из 200», считают элементы подмножества «ошибки». Слова «множество», «подмножество», «пересечение» будут встречаться во всей книге.

В Python для множеств есть готовый тип `set`. Он сам выбрасывает повторы:

```
bukvy = set("КОЛОКОЛ")      # множество букв слова
print(bukvy)                 # {'К', 'О', 'Л'} – порядок при печати может быть любым
print(len(bukvy))            # 3 – число элементов, наше  $n(A)$ 
print("Л" in bukvy)          # True – буква «Л» есть в множестве
print("М" in bukvy)          # False – буквы «М» нет
```

Построчно: `set("КОЛОКОЛ")` разбирает слово на буквы и оставляет каждую по одному разу; `len` считает элементы; `in` проверяет, есть ли элемент в множестве, и отвечает `True` («да, истина») или `False` («нет, ложь»).

Запомни. Множество – набор без повторов, порядок в нём не важен. $n(A)$ – сколько в нём элементов. Подмножество – часть, все элементы которой есть в большом множестве.

2.2. Пересечение и объединение: круги Эйлера

НА ПАЛЬЦАХ

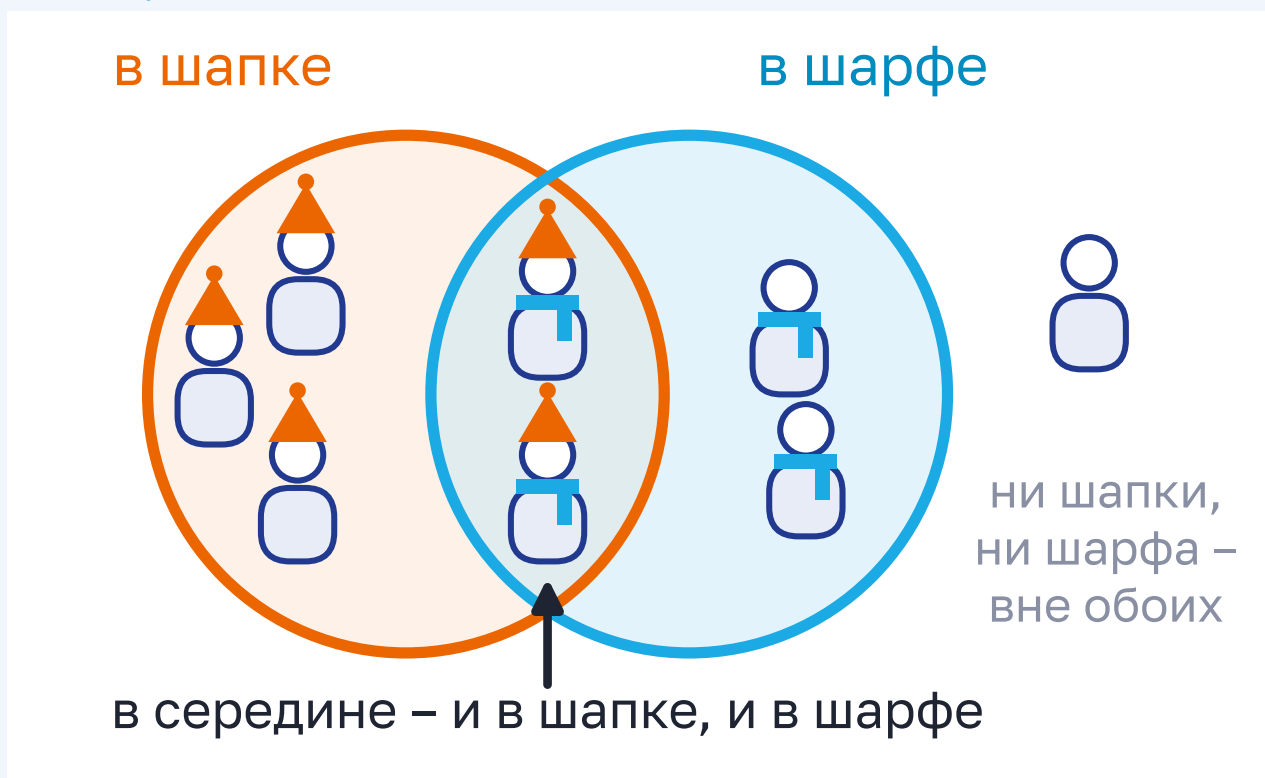


Рисунок 2.2 – Два обруча на полу: в левый встали те, кто в шапке, в правый – те, кто в шарфе

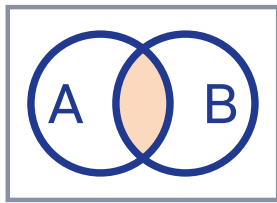
Тот, кто и в шапке, и в шарфе, должен стоять сразу в двух обручах. Поэтому обручи кладут внахлест, а он встаёт в середину.

Представь зимнюю перемену. Учитель кладёт на пол два обруча и просит: «Кто в шапке – в левый, кто в шарфе – в правый». Тем, у кого есть и то, и другое, придётся встать туда, где обручи перекрываются. А кто без шапки и без шарфа, останется снаружи. Вот и весь смысл кругов Эйлера: каждое множество рисуют кругом, и сразу видно, кто где.

Рисовать множества кругами придумал Леонард Эйлер – великий математик, который много лет работал в Петербургской академии наук. Поэтому их так и называют: **круги Эйлера**. В книгах встречается и название «диаграммы Эйлера – Венна».

Круги рисуют внутри прямоугольника. Прямоугольник – это **всё, о чём речь в задаче**: весь класс, все книги библиотеки, все страницы интернета. Без него непонятно, что значит «снаружи». Те, кто без шапки и без шарфа, стоят в прямоугольнике, но вне обоих кругов.

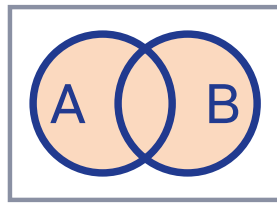
Два круга режут прямоугольник на четыре кусочка: «только A », «только B », общая середина и всё, что снаружи. Из этих кусочков, как из деталей конструктора, собирают четыре главных понятия. Некоторые занимают один кусочек, а некоторые – сразу несколько.



пересечение

$$A \cap B$$

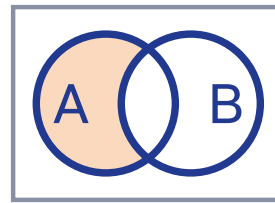
1 кусочек



объединение

$$A \cup B$$

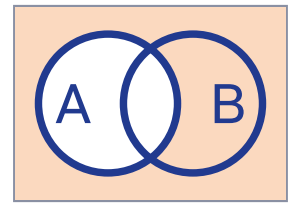
3 кусочка



разность

A , но не B

1 кусочек



дополнение

не A

2 кусочка

Рисунок 2.3 – Закрашено то, что подписано под картинкой. Под названием – сколько кусочков оно занимает

- **Пересечение** $A \cap B$ – элементы, которые есть **и** в A , **и** в B . Это один кусочек – общая серединка. Дети и в шапке, и в шарфе.
- **Объединение** $A \cup B$ – элементы, которые есть **хотя бы в одном** из множеств: в A , **или** в B , или в обоих сразу. Это три кусочка вместе: оба круга целиком. Все, кто стоит хоть в каком-то обруче.
- **Разность** « A , но не B » – один кусочек, левый «полумесяц»: в шапке, но без шарфа.
- **Дополнение** «не A » – всё, что внутри прямоугольника, но снаружи круга A . Это два кусочка: «только B » и «вне обоих». Все, кто без шапки: и те, кто в одном шарфе, и те, у кого нет ни шапки, ни шарфа.

Как не перепутать значки? $\cap$ похож на букву «П» – **П**ересечение. $\cup$ похож на чашку, в которую сливают всё вместе, – **о**бъединение.

Пример на числах. $A = \{1, 2, 3, 4, 6\}$, $B = \{2, 4, 5, 8\}$. Общие элементы – 2 и 4, значит, $A \cap B = \{2, 4\}$. Всё вместе, без повторов: $A \cup B = \{1, 2, 3, 4, 5, 6, 8\}$ – семь элементов. Только в A : $\{1, 3, 6\}$.

В Python пересечение пишут значком `&`, объединение – значком `|`, разность – минусом:

```
A = {1, 2, 3, 4, 6}
B = {2, 4, 5, 8}
print(A & B)           # {2, 4} – пересечение
print(A | B)           # {1, 2, 3, 4, 5, 6, 8} – объединение
print(A - B)           # {1, 3, 6} – есть в A, но нет в B
print(len(A | B))      # 7
```

Запомни. $A \cap B$ – «и то, и другое», $A \cup B$ – «хотя бы одно», « A , но не B » – круг A без общей части, «не A » – всё за пределами круга A . Одно понятие может занимать и один кусочек картинки, и несколько.

Совет. Как только в задаче появились две группы, которые могут пересекаться, рисуй два круга внахлест. Даже если кажется, что и так всё понятно. Рисунок занимает полминуты и спасает от самой частой ошибки – о ней следующий раздел.

2.3. Считаем элементы: формула включений и исключений

НА ПАЛЬЦАХ

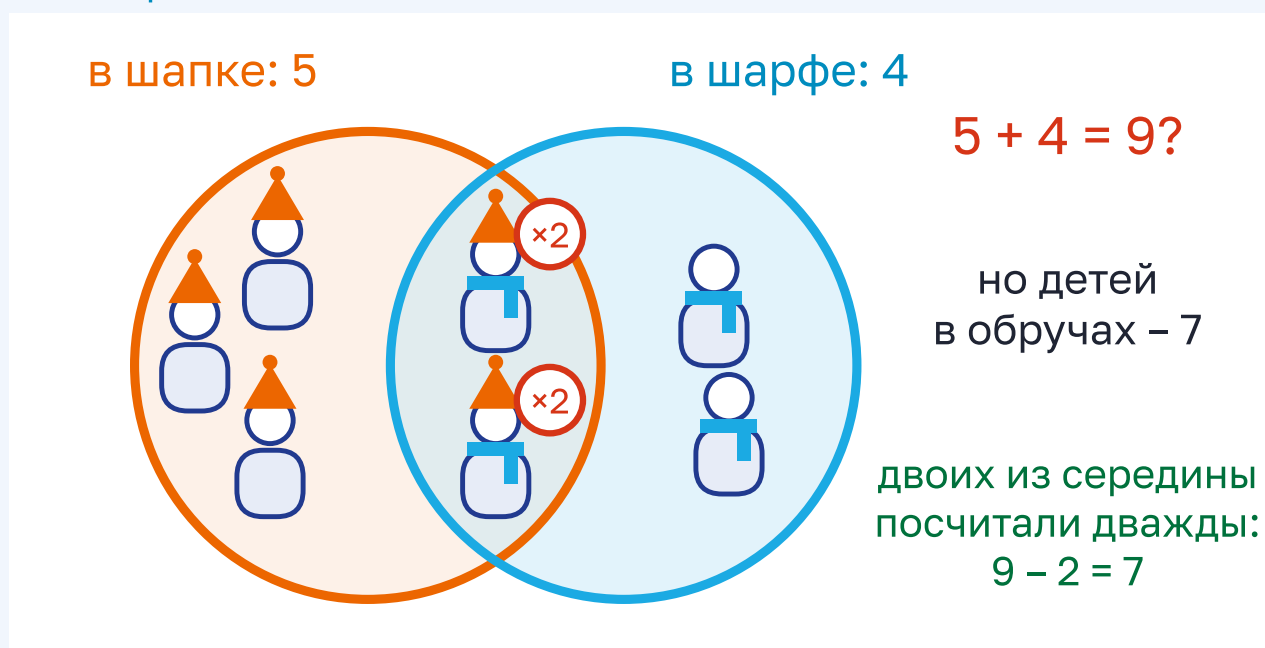


Рисунок 2.4 – Пятеро в шапке, четверо в шарфе. Но детей в обручах не девять, а семь

Двое из середины попали и в «шапки», и в «шарфы». Сложили – посчитали их дважды. Один лишний раз надо вычесть.

Главная ловушка задач на множества – **двойной счёт**. Если просто сложить, сколько детей в шапке и сколько в шарфе, тех, кто в середине, мы посчитаем два раза. Поэтому после сложения пересечение вычитают один раз:

$$n(A \cup B) = n(A) + n(B) - n(A \cap B).$$

Словами: **хотя бы в одном** = в первом + во втором – в обоих. Сначала всех «включили» (сложили), потом лишних «исключили» (вычли), отсюда и название – **формула включений и исключений**.

Пример. В 6 «А» 27 учеников. На плавание ходят 15 человек, на лыжи – 11, и туда, и туда – 6. Сколько человек ходят хотя бы в одну секцию и сколько – ни в одну?

- Хотя бы в одну: $15 + 11 - 6 = 20$ человек.
- Ни в одну: из всего класса вычитаем тех, кто хотя бы в одной: $27 - 20 = 7$ человек.
- Только на плавание: $15 - 6 = 9$. Только на лыжи: $11 - 6 = 5$. Проверка: $9 + 5 + 6 = 20$ – сходится.

Эту формулу можно «вертеть» как угодно: из четырёх чисел $n(A)$, $n(B)$, $n(A \cap B)$ и $n(A \cup B)$ по любым трём находится четвёртое. Чаще всего на отборе спрашивают пересечение:

$$n(A \cap B) = n(A) + n(B) - n(A \cup B).$$

Если про плавание знаем 15, про лыжи 11 и что хотя бы в одну секцию ходят 20 – в обе ходят $15 + 11 - 20 = 6$.

Три множества. Это место многие перечитывают дважды, и это нормально. Три круга внахлёст режут картинку на семь кусочков внутри и один снаружи. Складывать и вычитать наугад тут легко запутаться, поэтому круги **заполняют числами от центра наружу**.

Опросили 30 учеников 6 «Б», у кого какие питомцы. Кошка есть у 12, собака у 10, рыбки у 6. Кошка и собака – у 4, кошка и рыбки – у 3, собака и рыбки – у 2, все три – у одного.



Рисунок 2.5 – Круги заполняем по порядку: сначала центр, потом кусочки «ровно два», в конце «только один». В кругах – число учеников

- 1. Центр** – все три питомца: 1.
- 2. Ровно два питомца.** Кошка и собака есть у 4, но один из них – в центре, у него ещё и рыбки. Значит, только кошка и собака, без рыбок: $4 - 1 = 3$. Так же кошка и рыбки: $3 - 1 = 2$, собака и рыбки: $2 - 1 = 1$.
- 3. Только один питомец.** Из всего круга вычитаем то, что в нём уже вписано. Только кошка: $12 - 3 - 2 - 1 = 6$. Только собака: $10 - 3 - 1 - 1 = 5$. Только рыбки: $6 - 2 - 1 - 1 = 2$.
- 4. Сумма всех кусочков** – сколько учеников хотя бы с одним питомцем:
 $6 + 5 + 2 + 3 + 2 + 1 + 1 = 20$.
- 5. Снаружи** – без питомцев: $30 - 20 = 10$ учеников.

Когда все кусочки вписаны, можно ответить на любой вопрос. «Сколько учеников держат ровно одного питомца?» – $6 + 5 + 2 = 13$. «Ровно двух?» – $3 + 2 + 1 = 6$.

Проверка формулой. Для трёх кругов тоже есть формула включений и исключений, только длиннее:

$$\begin{aligned}
 n(A \cup B \cup C) = & \\
 & n(A) + n(B) + n(C) \\
 & - n(A \cap B) - n(A \cap C) - n(B \cap C) \\
 & + n(A \cap B \cap C).
 \end{aligned}$$

Словами: сложи три круга, вычти три пары и прибавь центр. Для питомцев:
 $12 + 10 + 6 - 4 - 3 - 2 + 1 = 20$ – столько же, сколько при заполнении.

Откуда в конце плюс? Посмотри на единственного ученика в центре – у него все три питомца. Когда сложили три круга, его посчитали **три** раза. Потом вычли три пары, а он есть в

каждой – вычли его тоже **три** раза. Вышло $3 - 3 = 0$: ученик пропал из подсчёта совсем. Возвращаем его, прибавив один раз.

Запомни. Хотя бы в одном $= n(A) + n(B) - n(A \cap B)$. Ни в одном $=$ все минус хотя бы в одном. Для трёх кругов заполняй кусочки от центра наружу, а в конце проверь сумму формулой.

Совет. Кусочек круга не может быть отрицательным. Если при заполнении вышло «только собака: -2 », ты где-то вычел лишнее или перепутал числа в условии.

2.4. Логика: И, ИЛИ, НЕ

НА ПАЛЬЦАХ



Рисунок 2.6 – Дверь с двумя замками, комната с двумя дверями и фонарь во дворе

И – как два замка на одной двери: войдёшь, только если открыты оба. **ИЛИ** – как две двери в комнату: хватит одной открытой. **НЕ** – как фонарь во дворе: он делает всё наоборот и горит, когда НЕ светло.

Высказывание – фраза, про которую можно сказать «верно» или «неверно». «Новосибирск стоит на Оби» – верно. «В неделе 8 дней» – неверно. «Который час?» – не высказывание: это вопрос, он не бывает верным или неверным. В логике вместо «верно» говорят **истина**, вместо «неверно» – **ложь**. Коротко истину пишут 1, ложь – 0.

Из простых высказываний собирают сложные с помощью **связок**:

- **И** – истинно, только когда **оба** высказывания истинны. «Идёт снег И дует ветер» – верно, только если есть и снег, и ветер.
- **ИЛИ** – истинно, когда **хотя бы одно** истинно (или оба). «Возьму на дачу книгу ИЛИ планшет» – обещание выполнено, если взял книгу, планшет или и то, и другое. Ложно, только если не взял ничего.

- **НЕ** – переворачивает: истину делает ложью, ложь – истиной. «НЕ идёт снег» верно ровно тогда, когда снега нет.

Осторожно с ИЛИ. В обычной речи «чай или компот?» значит «что-то одно». В логике ИЛИ разрешает и оба сразу. В задачах всегда логическое ИЛИ, если не сказано иначе.

Таблица истинности – таблица, где перебраны все возможные сочетания «истина/ложь» и для каждого написан результат. У двух высказываний A и B сочетаний четыре: оба ложны, ложно только одно (два случая), оба истинны.

Таблица 2.1

A	B	$A \text{ и } B$	$A \text{ или } B$	$\text{НЕ } A$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Столбец «И» – одна единица, только в последней строке. Столбец «ИЛИ» – один ноль, только в первой. Больше тут запоминать нечего.

Посмотри на круги Эйлера из раздела 2.2 – это та же логика. Элемент попадает в пересечение, если он в A **и** в B . В объединение – если он в A **или** в B . В дополнение – если он **НЕ** в A . Поэтому задачи про множества и задачи про логику решаются одними и теми же картинками.

Сложные выражения. Связки можно соединять: «(дома есть хлеб ИЛИ есть деньги) И магазин НЕ закрыт». Порядок такой же строгий, как у действий в арифметике: сначала то, что в **скобках**, потом **НЕ**, потом **И** и в конце **ИЛИ**.

Это как в арифметике, где умножение делают раньше сложения. Связка И и правда похожа на умножение: $0 \cdot 1 = 0$, и «ложь И истина» – тоже ложь. Сомневаешься в порядке – ставь скобки, лишними они не бывают.

Пример. $A = 1, B = 0, C = 0$. Чему равно A ИЛИ B И НЕ C ? По порядку: НЕ $C = 1$. Потом И: B И $1 = 0$ И $1 = 0$. В конце ИЛИ: A ИЛИ $0 = 1$ ИЛИ $0 = 1$. Ответ: 1, истина.

Если высказываний три, строк в таблице истинности будет $2 \cdot 2 \cdot 2 = 8$: каждое из трёх может быть 0 или 1 (это правило умножения из главы 3). Для четырёх – 16 строк. Перебирать руками долго, а компьютер делает это мгновенно. В Python связки пишут по-английски: И – `and`, ИЛИ – `or`, НЕ – `not`:

```
# печатаем таблицу истинности для выражения A и не B
for A in [0, 1]:
    for B in [0, 1]:
        rez = A and not B
        print(A, B, int(rez))
```

Как это читать: `for A in [0, 1]:` – «по очереди подставь в A сначала 0, потом 1». Второй `for` внутри первого перебирает B для каждого A – так получаются все четыре строки. `rez = A and not B` считает выражение, а `int(rez)` печатает результат цифрой 0 или 1. Программа вы-

ведет четыре строки: 0 0 0, 0 1 0, 1 0 1, 1 1 0. Истина только в одной строке: $A = 1$, $B = 0$.

Для любопытных. В учебниках информатики связки часто пишут значками: И – $\wedge$, ИЛИ – $\vee$, НЕ – $\neg$. Правила с ними те же. Науку о вычислениях с истиной и ложью в XIX веке придумал английский математик Джордж Буль, поэтому её называют **булевой алгеброй**. Внутри процессора твоего телефона – миллиарды крошечных переключателей, собранных в элементы И, ИЛИ, НЕ.

Запомни. И – истина, только если истинны оба. ИЛИ – ложь, только если ложны оба. НЕ – наоборот. Порядок: скобки, НЕ, И, ИЛИ. Для k высказываний в таблице истинности 2^k строк: 4, 8, 16...

2.5. Запросы к поисковику

НА ПАЛЬЦАХ

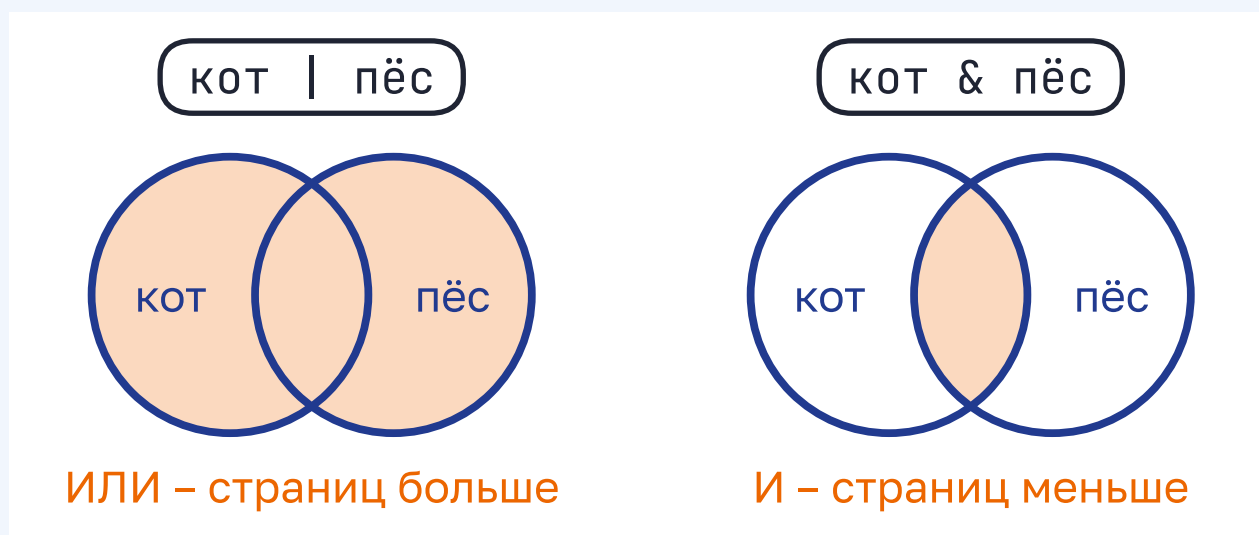


Рисунок 2.7 – Один поисковик, два запроса

ИЛИ собирает оба круга – находится больше. И оставляет только общую часть – находится меньше.

Поисковик – это огромная машина для работы с множествами. На каждое слово у него есть множество страниц, где это слово встречается. Запрос из нескольких слов – это пересечение или объединение таких множеств. В задачах (и в школьных учебниках) обычно пишут так:

- кот | пёс – ИЛИ: страницы, где есть хотя бы одно из слов, – объединение;
- кот & пёс – И: страницы, где есть оба слова, – пересечение.

В настоящих поисковиках значки могут быть другими, да и работают они сложнее. Но в задаче всегда сказано, что значит каждый значок, – решай по условию.

Нарисуй круги Эйлера – и сразу увидишь два правила:

- **И сужает.** Каждое новое слово через & может только убрать страницы. кот & пёс найдёт не больше, чем кот, а кот & пёс & попугай – не больше, чем кот & пёс.
- **ИЛИ расширяет.** Каждое новое слово через | может только добавить. кот | пёс найдёт не меньше, чем кот.

Поэтому запросы легко выстроить по числу найденных страниц: меньше всего – у запроса с самым длинным «И», больше всего – у запроса с самым длинным «ИЛИ». А если даны числа, работает формула из раздела 2.3. Нашлось «кот» – 800 страниц, «пёс» – 600, «кот | пёс» – 1100. Тогда «кот & пёс»: $800 + 600 - 1100 = 300$ страниц.

Так же устроены фильтры в каталогах, интернет-магазинах и базах данных. В одной из задач отбора прошлых лет участникам давали книжный сервис: можно было выбрать сразу несколько жанров, и сервис показывал книги **хотя бы одного** из них – то есть объединение. А нужно было найти книги **обоих** жанров. Пересечения сервис не давал – его надо было досчитать самому по формуле. Такую задачу мы разберём ниже (задача 2.3).

Запомни. & – И – пересечение: найдётся меньше. | – ИЛИ – объединение: найдётся больше. Пересечение по трём числам: $n(A \cap B) = n(A) + n(B) - n(A \cup B)$.

Совет. Прежде чем считать, прочитай, что делает фильтр или значок. Выбрал в каталоге два жанра – это «ИЛИ» или «И»? От ответа зависит, какую формулу брать.

2.6. Разбираем задачи

Задача 2.1 Кружки после уроков

разминка · объединение, формула включений и исключений

В 6 «В» 28 учеников. На робототехнику ходят 16 человек, на программирование – 11, на оба кружка – 5. а) Сколько учеников ходят хотя бы в один кружок? б) Сколько не ходят ни в один?

Дано: всего 28; $n(P) = 16$; $n(\Pi) = 11$; $n(P \cap \Pi) = 5$.

Найти: $n(P \cup \Pi)$ – ?; ни в один – ?

Решение

Шаг 1. Что происходит. Два множества: «ходят на робототехнику» и «ходят на программирование». Они пересекаются: 5 человек ходят туда и туда. Рисуем два круга внахлест внутри прямоугольника «весь класс».

Шаг 2. Хотя бы в один. По формуле включений и исключений: $16 + 11 - 5 = 22$ ученика.

Шаг 3. Ни в один. Из всего класса вычитаем тех, кто хотя бы в одном: $28 - 22 = 6$ учеников.

Проверка. Только робототехника: $16 - 5 = 11$, только программирование: $11 - 5 = 6$, оба: 5, ни одного: 6. Всего $11 + 6 + 5 + 6 = 28$ – как раз весь класс.

И прикинь на глаз: тех, кто ходит хотя бы в один кружок, не может быть меньше, чем в самом большом кружке (16), и больше, чем во всём классе (28). 22 – между ними.

! **Самая частая ошибка.** Сложить $16 + 11 = 27$ и ответить «27 ходят хотя бы в один, 1 не ходит никуда». Пятерых, кто ходит в оба кружка, посчитали два раза.

Ответ: а) 22 ученика; б) 6 учеников.

Своя задача по программе сферы, модуль 2

Реши сам

2.1. В классе 25 учеников. Математику любят 14, информатику – 12, а 4 ученика не любят ни то, ни другое. Сколько учеников любят оба предмета?

2.2.* В классе 30 учеников. 20 ходят на плавание, 17 – на футбол. Какое **наименьшее** и какое **наибольшее** число учеников может ходить и туда, и туда? Запиши два числа через точку с запятой: сначала наименьшее.

Задача 2.2 Буквы в словах

разминка · множество, пересечение, объединение

A – множество букв слова «НЕЙРОСЕТЬ», B – множество букв слова «РОБОТ». Найди $n(A)$, $n(B)$, $n(A \cap B)$ и $n(A \cup B)$. В ответ запиши $n(A \cup B)$.

Дано: A – буквы слова «НЕЙРОСЕТЬ»; B – буквы слова «РОБОТ».

Найти: $n(A \cup B)$ – ?

Решение

Шаг 1. Выписываем множества без повторов. В слове «НЕЙРОСЕТЬ» девять букв, но «Е» встречается дважды: $A = \{Н, Е, Й, Р, О, С, Т, Ь\}$, $n(A) = 8$. В слове «РОБОТ» пять букв, «О» – дважды: $B = \{Р, О, Б, Т\}$, $n(B) = 4$.

Шаг 2. Пересечение. Идём по буквам B и ищем каждую в A : Р – есть, О – есть, Б – нет, Т – есть. $A \cap B = \{Р, О, Т\}$, $n(A \cap B) = 3$.

Шаг 3. Объединение. По формуле: $8 + 4 - 3 = 9$. Проверим перечислением – к буквам A добавляем те буквы B , которых там нет (только Б): $\{Н, Е, Й, Р, О, С, Т, Ь, Б\}$ – 9 букв.

Проверка кодом.

```
A = set("НЕЙРОСЕТЬ")
B = set("РОБОТ")
print(len(A), len(B), len(A & B), len(A | B))  # 8 4 3 9
```

! **Ловушка с повторами.** Посчитать буквы вместе с повторами: $n(A) = 9$, $n(B) = 5$ – и получить $9 + 5 - 3 = 11$. В множестве каждая буква – один раз, сколько бы раз она ни встречалась в слове.

Ответ: $n(A) = 8, n(B) = 4, n(A \cap B) = 3, n(A \cup B) = 9$. В ответ – 9.

Своя задача по программе сферы, модуль 2

Реши сам

2.3. A – множество букв слова «МАТЕМАТИКА», B – множество букв слова «ИНФОРМАТИКА». Сколько элементов в $A \cup B$?

2.4. A – чётные числа от 1 до 20, B – числа от 1 до 20, которые делятся на 3. Найди $n(A \cap B)$ и $n(A \cup B)$.

Задача 2.3 Каталог школьной библиотеки

пересечение, формула включений и исключений, фильтры

У школьной библиотеки есть сайт-каталог. У каждой книги указан один или несколько жанров и одна возрастная отметка (6+, 10+, 12+ и так далее). В фильтре можно выбрать сразу несколько жанров – тогда сайт показывает книги, у которых есть **хотя бы один** из выбранных жанров. Катя попробовала разные фильтры и записала, сколько книг нашлось:

Таблица 2.2

Фильтр	Отметка 10+	Любой возраст
фантастика	9	24
приключения	7	15
оба жанра	14	33

В строке «оба жанра» в фильтре выбраны сразу фантастика и приключения.

Сколько в библиотеке книг, у которых есть **оба** жанра – и фантастика, и приключения, – а возрастная отметка **не** 10+?

Дано: отметка 10+: $n(\Phi) = 9, n(\Pi) = 7, n(\Phi \cup \Pi) = 14$; все книги: $n(\Phi) = 24, n(\Pi) = 15, n(\Phi \cup \Pi) = 33$.

Найти: книги с обоими жанрами и отметкой не 10+ – ?

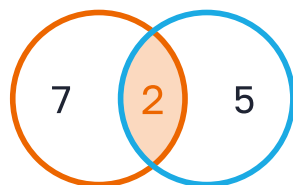
Решение

Шаг 1. Что происходит. Когда выбраны оба жанра, сайт показывает книги хотя бы одного жанра – это **объединение** $\Phi \cup \Pi$. А нам нужны книги обоих жанров – **пересечение** $\Phi \cap \Pi$. Сайт его не показывает, досчитаем сами. Сделаем это два раза: для отметки 10+ и для всех книг.

Шаг 2. Оба жанра с отметкой 10+. $n(\Phi \cap \Pi) = 9 + 7 - 14 = 2$ книги.

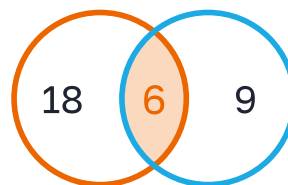
Шаг 3. Оба жанра, все возрасты. $n(\Phi \cap \Pi) = 24 + 15 - 33 = 6$ книг.

фантастика: 9 приключения: 7



отметка 10+: всего 14 книг

фантастика: 24 приключения: 15



все возрасты: всего 33 книги

Рисунок 2.8 – Те же книги на кругах Эйлера. Закрашено пересечение: книги обоих жанров

Шаг 4. Оба жанра, но не 10+. У каждой книги ровно одна возрастная отметка. Значит, 6 книг обоих жанров делятся на две группы: 2 с отметкой 10+ и все остальные. Остальных: $6 - 2 = 4$ книги.

❗ **На чём ловит эта задача.** Кажется, что фильтр с двумя жанрами показывает книги обоих жанров, – и тянет сразу ответить $33 - 14 = 19$. Но 33 и 14 – объединения, «хотя бы один жанр». Читай в условии, что делает фильтр: здесь выбор двух жанров – это ИЛИ, а не И.

Ответ: 4 книги.

По мотивам задачи отборочного этапа НТО Junior 2023/24 «Круги Эйлера»

Реши сам

2.5. В детском онлайн-кинотеатре фильтр жанров работает как в задаче 2.3: выбрал несколько жанров – показывает мультфильмы хотя бы одного из них. У каждого мультфильма одна возрастная отметка. С отметкой 6+: «комедия» – 12, «сказка» – 9, «комедия» и «сказка» (выбраны оба) – 17. Без фильтра возраста: «комедия» – 30, «сказка» – 22, оба жанра в фильтре – 45. Сколько мультфильмов одновременно и комедия, и сказка, но с отметкой **не** 6+?

2.6.* Хотя бы в одном из множеств A и B – 30 элементов. В A – 20 элементов, в B – 18. Сколько элементов есть **только** в B ?

Задача 2.4 Реки Сибири в поисковике

запросы к поисковику, И сужает, ИЛИ расширяет

В языке запросов поисковика значок `|` означает ИЛИ, значок `&` – И. Вот сколько страниц нашлось по трём запросам:

- Байкал `|` Обь – 780 тыс. страниц;
- Байкал – 500 тыс. страниц;
- Обь – 420 тыс. страниц.

а) Сколько тысяч страниц найдётся по запросу Байкал `&` Обь ? б) Расположи запросы в порядке **возрастания** числа найденных страниц: А) Обь `&` Байкал `&` Енисей ; Б) Обь `|` Байкал ; В) Обь ; Г) Обь `&` Байкал . Ответ запиши буквами.

Дано: $n(B \cup O) = 780$; $n(B) = 500$; $n(O) = 420$ (тыс. страниц).

Найти: $n(B \cap O)$ – ?; порядок запросов А–Г.

Решение

Шаг 1. Что происходит. Каждое слово – множество страниц. $|$ – объединение, $\&$ – пересечение. Три числа из формулы включений и исключений известны, четвертое найдём.

Шаг 2. Пересечение. $n(B \cap O) = 500 + 420 - 780 = 140$ тыс. страниц.

Шаг 3. Порядок. Для Б число есть: $Обь | Байкал$ – то же, что $Байкал | Обь$, 780 тыс. Для В оно тоже дано (420 тыс.), для Г мы его только что нашли (140 тыс.). А для А числа нет – и не нужно: хватит понять, кто в ком помещается.

- Г) $Обь \& Байкал$ – страницы, где есть оба слова. Каждая такая страница есть и среди страниц со словом «Обь», поэтому Г не больше В.
- А) $Обь \& Байкал \& Енисей$ – из страниц Г остаются только те, где есть ещё и «Енисей». А не больше Г.
- Б) $Обь | Байкал$ – все страницы со словом «Обь» плюс ещё страницы про Байкал. Б не меньше В.

Выходит цепочка: А внутри Г, Г внутри В, В внутри Б. По возрастанию: А, Г, В, Б.

Сверим с числами. Пересечение не может быть больше ни одного из множеств – и правда, 140 меньше и 500, и 420. А известные числа идут в том же порядке, что и в ответе: $140 < 420 < 780$ (Г, В, Б).

 **Частая путаница.** Многие думают, что чем длиннее запрос, тем больше он найдёт: «в А три слова – значит, страниц больше всего». Для $\&$ всё наоборот: каждое новое слово – ещё одно условие, и страниц становится меньше или столько же.

Ответ: а) 140 тыс. страниц; б) АГВБ.

Своя задача по программе сферы, модуль 2

Реши сам

2.7. По запросу $Обское\ море | Академгородок$ нашлось 5200 страниц, по запросу $Обское\ море \& Академгородок$ – 900, по запросу $Академгородок$ – 3400. Сколько страниц найдётся по запросу $Обское\ море$?

2.8. Расположи запросы в порядке **возрастания** числа найденных страниц: А) $дрон$; Б) $дрон \& Луна$; В) $дрон | Луна | ракета$; Г) $дрон | Луна$. Ответ запиши буквами.

Задача 2.5 Умный почтовый фильтр

И, ИЛИ, НЕ, порядок действий, скобки

Почтовая программа кладёт письмо во «Входящие» по правилу:

(отправитель есть в контактах ИЛИ в теме есть слово «кружок») И НЕ (в письме есть ссылка на незнакомый сайт).

Остальные письма уходят в «Спам». Пришло шесть писем:

Таблица 2.3

№	Отправитель	Тема	Ссылка на незнакомый сайт
1	одноклассник (в контактах)	«Что задали по математике?»	нет
2	незнакомый адрес	«Кружок робототехники переносится»	нет
3	незнакомый адрес	«Вы выиграли телефон!»	есть
4	мама (в контактах)	«Фото с дачи»	есть
5	незнакомый адрес	«Запись в кружок – по ссылке»	есть
6	учитель (в контактах)	«Кружок завтра в 15:00»	нет

Какие письма попадут во «Входящие»? Запиши номера по возрастанию, без пробелов.

Дано: К – «отправитель в контактах»; Т – «в теме есть слово «кружок»»; С – «есть ссылка на незнакомый сайт». Правило: (К ИЛИ Т) И НЕ С.

Найти: номера писем, для которых правило истинно.

Решение

Шаг 1. Переводим письма в нули и единицы. Слово «кружок» в любой форме («кружок», «кружка», «Кружок») считаем. Для каждого письма выписываем К, Т, С.

Шаг 2. Считаем по порядку: сначала скобки, потом НЕ, потом И.

Таблица 2.4

№	К, Т, С	К ИЛИ Т	НЕ С	итог
1	1, 0, 0	1	1	1
2	0, 1, 0	1	1	1
3	0, 0, 1	0	0	0
4	1, 0, 1	1	0	0
5	0, 1, 1	1	0	0
6	1, 1, 0	1	1	1

Шаг 3. Читаем столбец «итог». Единицы – у писем 1, 2 и 6.

А как же мама? Письмо от мамы со ссылкой на незнакомый сайт ушло в «Спам». Обидно, но правило честное: ссылки на незнакомые сайты фильтр не пропускает ни от кого. Если мамину почту взломают, мошенники будут рассылать такие ссылки как раз от её имени.

⚠ **Осторожно со скобками.** Если их потерять и прочитать правило как «К ИЛИ (Т И НЕ С)», письмо 4 от мамы пройдёт, ведь $K = 1$, а ИЛИ с единицей всегда даёт 1, – и получится 1246. Скобки меняют ответ, поэтому считай сначала то, что в них.

Ответ: 126.

Своя задача по программе сферы, модуль 2

Реши сам

2.9. В лабораторию пускают по правилу: (есть пропуск И пройден инструктаж) ИЛИ рядом наставник. Пришли пятеро: 1) пропуск есть, инструктаж пройден, наставника нет; 2) пропуск есть, инструктажа нет, наставника нет; 3) пропуска нет, инструктажа нет, наставник рядом; 4) пропуска нет, инструктаж пройден, наставник рядом; 5) всё есть. Кого пустят? Запиши номера по возрастанию, без пробелов.

2.10. $A = 1$, $B = 0$. Чему равно $(\text{НЕ } A) \text{ ИЛИ } B$? Ответ – 0 или 1.

Задача 2.6* Разметка фотографий для нейросети

три множества, заполнение кругов Эйлера

Чтобы научить нейросеть находить на фото кошек, собак и людей, ребята из кружка разместили 200 фотографий: поставили метки «кошка», «собака», «человек». На одном фото может быть несколько меток, а может не быть ни одной (например, пейзаж). Итог разметки:

- метка «кошка» – на 70 фото, «собака» – на 60, «человек» – на 90;
- «кошка» и «собака» вместе – на 20 фото, «кошка» и «человек» – на 30, «собака» и «человек» – на 25 (на части этих фото может стоять и третья метка);
- все три метки – на 10 фото.

а) Сколько фото остались без единой метки? б) Сколько фото получили **ровно одну** метку?

Дано: всего 200; $n(K) = 70$; $n(C) = 60$; $n(Ч) = 90$; $n(K \cap C) = 20$; $n(K \cap Ч) = 30$; $n(C \cap Ч) = 25$; $n(K \cap C \cap Ч) = 10$.

Найти: без меток – ?; ровно одна метка – ?

Решение

Шаг 1. Что происходит. Три множества, которые пересекаются. Рисуем три круга и заполняем кусочки от центра наружу – так ответим на оба вопроса сразу.

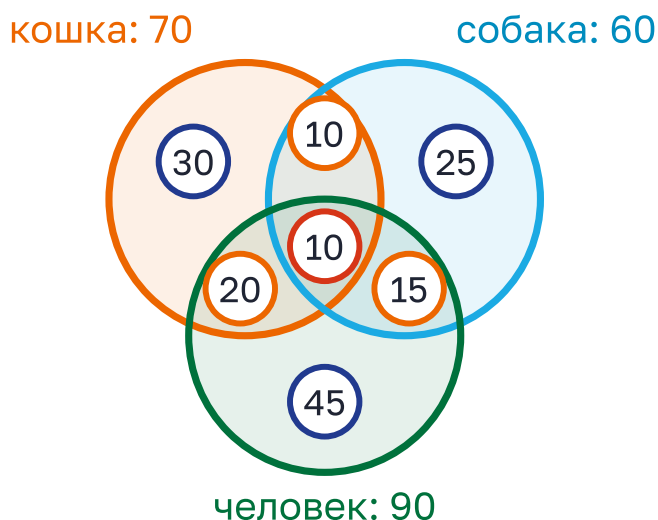
Шаг 2. Центр – все три метки: 10.

Шаг 3. Ровно две метки. Из каждой пары вычитаем центр: только кошка и собака $20 - 10 = 10$; только кошка и человек $30 - 10 = 20$; только собака и человек $25 - 10 = 15$.

Шаг 4. Ровно одна метка. Из каждого круга вычитаем всё, что в нём уже вписано:

- только кошка: $70 - 10 - 20 - 10 = 30$;

- только собака: $60 - 10 - 15 - 10 = 25$;
- только человек: $90 - 20 - 15 - 10 = 45$.



в кругах: 155 фото

без меток:

$$200 - 155 = 45$$

ровно одна метка:

$$30 + 25 + 45 = 100$$

Рисунок 2.9 – Заполненные круги для задачи 2.6. Числа – сколько фото в каждом кусочке

Шаг 5. Хотя бы одна метка – сумма всех семи кусочков:

$30 + 25 + 45 + 10 + 20 + 15 + 10 = 155$ фото. Проверим формулой:

$70 + 60 + 90 - 20 - 30 - 25 + 10 = 155$ – то же число.

Шаг 6. Ответы. а) Без меток: $200 - 155 = 45$ фото. б) Ровно одна метка:

$30 + 25 + 45 = 100$ фото.

Все ли фото на месте? 100 фото с одной меткой, $10 + 20 + 15 = 45$ – с двумя, 10 – с тремя, 45 – без меток. $100 + 45 + 10 + 45 = 200$ – ни одно не потерялось. И ни один кусочек не вышел отрицательным.

⚠ **Типичная ошибка.** Забыть в конце прибавить центр:

$70 + 60 + 90 - 20 - 30 - 25 = 145$, и тогда без меток «55». А в пункте б) рука тянется сразу написать $70 + 60 + 90 = 220$ «фото с одной меткой»: так каждое фото с двумя метками посчитано дважды, а с тремя – трижды. Да и 220 больше, чем всех фото, – верный знак, что что-то не так.

Ответ: а) 45 фото; б) 100 фото.

Своя задача по программе сферы, модуль 2

Реши сам

2.11. В детском технопарке 40 школьников. На робототехнику записаны 20, на программирование – 18, на 3D-моделирование – 15. Робототехника и программирование – у 7, робототехника и 3D – у 6, программирование и 3D – у 5, все три кружка – у 3. Сколько школьников не записаны ни в один из этих кружков?

2.12.* Для нейросети, которая узнаёт звуки, разместили 120 записей: метка «речь» – на 60, «музыка» – на 45, «шум» – на 40. «Речь» и «музыка» вместе – на 15 записях, «речь» и «шум» – на 12, «музыка» и «шум» – на 10, все три метки – на 5. Сколько записей получили **ровно две** метки?

Задача 2.7* Умная лампа

таблица истинности на три высказывания, Python

Умная лампа в прихожей включается по правилу: $(A \text{ И } B)$ ИЛИ НЕ C , где A – «на улице темно», B – «в прихожей кто-то есть», C – «включён режим «Экономия»». Каждое из трёх высказываний может быть истинным или ложным. При скольких сочетаниях A, B, C из всех возможных лампа будет гореть?

Дано: правило $(A \text{ И } B)$ ИЛИ НЕ C ; A, B, C – каждое 0 или 1.

Найти: число сочетаний, при которых правило истинно.

Решение

Шаг 1. Сколько всего сочетаний. У каждого из трёх высказываний два значения: $2 \cdot 2 \cdot 2 = 8$ сочетаний. Их удобно перебирать как числа от 000 до 111: C меняется чаще всех, A – реже.

Шаг 2. Таблица. Сначала скобка $A \text{ И } B$, потом НЕ C , в конце ИЛИ.

Таблица 2.5

A	B	C	$A \text{ И } B$	НЕ C	итог
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	0	1

Шаг 3. Считаем единицы в итоге: 5.

Короткий путь без таблицы. ИЛИ истинно, если истинна хотя бы одна часть. НЕ C истинно, когда $C = 0$, – это 4 сочетания из 8 (A и B любые, $2 \cdot 2 = 4$). Остались 4 сочетания с $C = 1$. Там лампу может включить только $A \text{ И } B$, а это одно сочетание: $A = 1, B = 1$. Всего $4 + 1 = 5$.

Проверка кодом.

```

schet = 0
for A in [0, 1]:
    for B in [0, 1]:
        for C in [0, 1]:
            if (A and B) or not C:
                schet = schet + 1
print(schet)    # 5

```

Три вложенных `for` перебирают все 8 сочетаний. Если правило истинно, счётчик `schet` увеличивается на 1.

Как это выглядит в прихожей. В режиме «Экономия» лампа горит только в одном случае – темно и кто-то пришёл. Без экономии – всегда.

⚠ Так теряют баллы. Читают правило в другом порядке: *А И (В ИЛИ НЕ С)*. Тогда при $A = 0$ лампа не горит никогда, и выходит 3 вместо 5. Или перебирают не все сочетания: выписывают «на глаз» 4–5 строк и теряют часть. Пиши строки строго по порядку, от 000 до 111.

Ответ: 5.

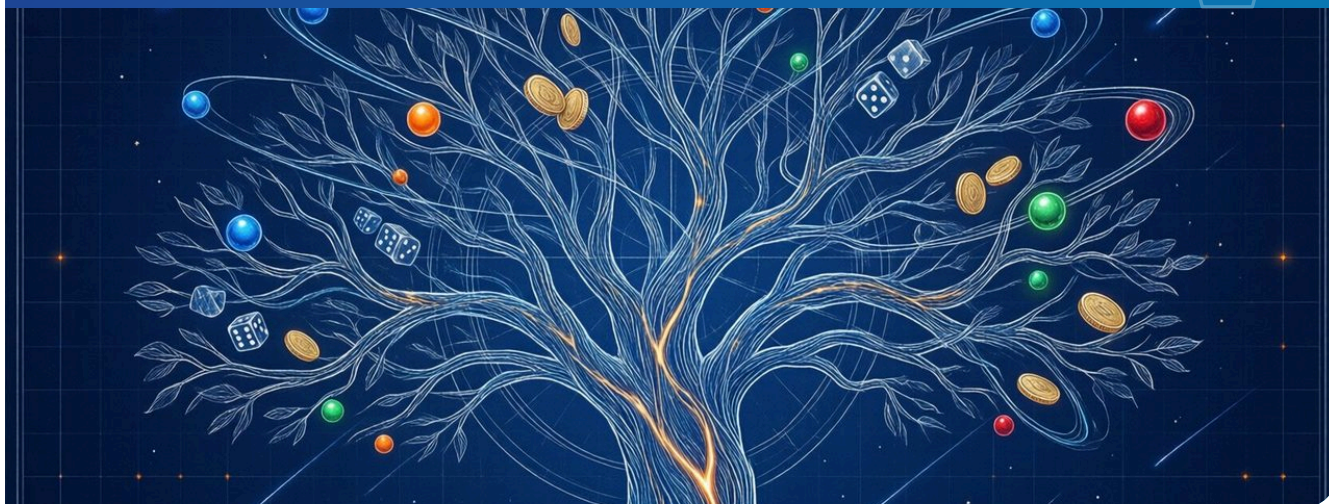
Своя задача по программе сферы, модуль 2

Реши сам

2.13.* При скольких сочетаниях A, B, C (каждое – 0 или 1) истинно выражение $(A \text{ ИЛИ } B) \text{ И } (\text{НЕ } A \text{ ИЛИ } C)$?

2.14. Сколько целых чисел X от 1 до 10 делают истинным высказывание: $(X > 4) \text{ И НЕ } (X \text{ чётное})$?

Комбинаторика и вероятность



Сколько разных разговоров может получиться у чат-бота, если на каждом шаге человек выбирает одну из нескольких кнопок? Какова вероятность выбросить на двух кубиках 7? И что имеет в виду программа, когда пишет «кошка – 80 %»? Все три вопроса решаются одним приёмом: аккуратно пересчитать варианты и не потерять ни одного. На отборе прошлых лет была как раз такая задача – «дерево возможностей» чат-бота, ответ вводился одним числом. Арифметика здесь на уровне таблицы умножения. Баллы теряют на другом: пропускают ветку или считают одну и ту же дважды.

3.1. Дерево возможностей и правило умножения

НА ПАЛЬЦАХ



Рисунок 3.1 – Две шапки и три шарфа

Каждая шапка ветвится на три шарфа. Веток две, в каждой по три – всего $2 \cdot 3 = 6$ нарядов.

Утром в прихожей висят две шапки – синяя и оранжевая – и три шарфа. Сколько разных нарядов можно собрать? Перебирать наугад рискованно: что-нибудь пропустишь или посчитаешь дважды. Надёжнее нарисовать **дерево возможностей** – схему, где каждый выбор разветвляется на все свои варианты. Из начальной точки выходят ветки «синяя шапка» и «оранжевая шапка», из каждой – по три ветки-шарфа. Концы веток называют **листьями**. Каждый лист – один готовый вариант. Листьев шесть, значит, и нарядов шесть.

Чат-боты устроены так же. Бот «Что почитать?» сначала спрашивает жанр: фантастика, детектив или приключения. Потом уточняет, какую книгу хочется – короткую или длинную. Каждый путь от начала до листа – один **сценарий** разговора.

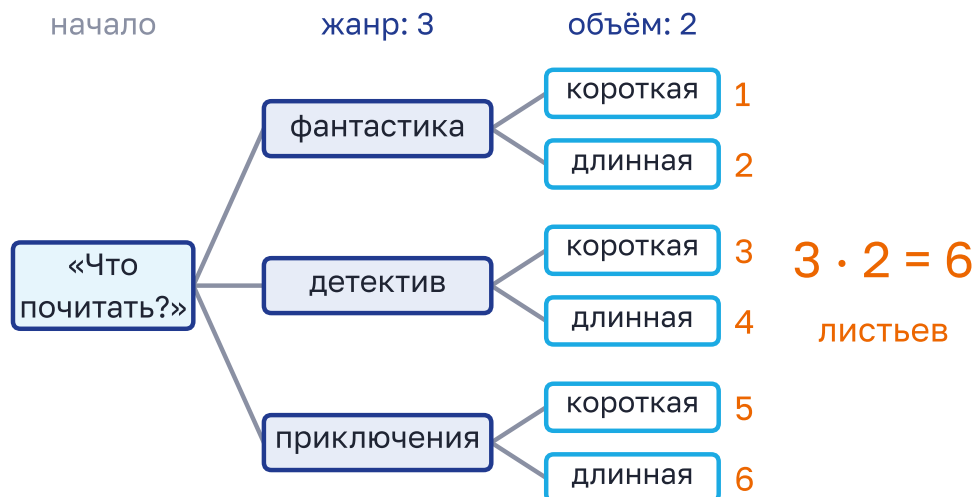


Рисунок 3.2 – Дерево бота «Что почитать?». Каждый путь от начала до листа – один сценарий разговора

Рисовать дерево каждый раз не обязательно – достаточно посмотреть на его форму. Из начала выходят 3 ветки, и **каждая** делится ещё на 2. Получается 3 раза по 2, то есть $3 \cdot 2 = 6$. Это **правило умножения**: если выбор идёт по шагам, на первом шаге a вариантов, а на втором – b вариантов при любом первом выборе, то всего вариантов $a \cdot b$. Шагов больше – множителей больше. Добавим боту третий вопрос «бумажная или электронная?» – каждый из шести листьев раздвоится, и получится $3 \cdot 2 \cdot 2 = 12$ сценариев.

Умножать можно, только если на втором шаге **всегда одно и то же** число вариантов, что бы ни выбрали на первом. Если после «фантастики» бот предлагает два объёма, а после «детектива» – один, правило ломается. Тогда листья считают по веткам и складывают. Об этом – следующий раздел.

Пересчитать листья может и программа. Два цикла `for`, один внутри другого, – это и есть дерево: внешний цикл идёт по веткам первого шага, внутренний – по веткам второго.

```
zhanry = ["фантастика", "детектив", "приключения"]
obyomy = ["короткая", "длинная"]

n = 0
for zh in zhanry:           # первый шаг дерева: жанр
    for ob in obyomy:        # второй шаг: для каждого жанра – оба объёма
        n += 1
        print(n, zh, ob)
print("всего:", n)          # всего: 6
```

Цикл `for zh in zhanry` по очереди берёт каждый жанр из списка. Для каждого жанра вложенный цикл перебирает оба объёма. Строка `n += 1` прибавляет к счётчику единицу на каждом листе, а `print` выводит номер листа и сам сценарий.

Запомни. Выбор по шагам – перемножаем, сколько вариантов на каждом шаге: $a \cdot b \cdot c$. Так можно, только если число вариантов на шаге не зависит от того, что выбрали раньше.

Совет. Сомневаешься – нарисуй начало дерева: первую ветку целиком, остальные – одной чертой с подписью «так же». Ошибку «сложил вместо того, чтобы умножить» такой набросок ловит сразу.

3.2. Правило сложения: «или» против «и»

НА ПАЛЬЦАХ



Рисунок 3.3 – «И» – умножаем, «или» – складываем

Кашу **и** напиток берут вместе – каждая каша с каждым напитком, $2 \cdot 3 = 6$ пар. Книгу **или** фильм выбирают одно – все варианты встанут в одну очередь, $5 + 3 = 8$.

Прислушивайся к союзам в условии. На завтрак берут кашу **и** напиток – обе вещи сразу, и каждая каша сочетается с каждым напитком. Это правило умножения. А вечером читают книгу **или** смотрят фильм – что-то одно. Тогда варианты просто выстраиваются в один ряд: 5 книг да 3 фильма – $5 + 3 = 8$ способов провести вечер. Это **правило сложения**: если выбираем **один** вариант из нескольких групп, которые не пересекаются, числа вариантов складываем.

Часто в одной задаче есть и то, и другое. Бот кинотеатра спрашивает: «Мультфильм или кино?» Мультфильмов в афише 4, фильмов 2. Потом предлагает к сеансу напиток – один из трёх. Первый шаг – «или»: $4 + 2 = 6$ сеансов. Второй шаг – «и»: к любому сеансу подходит любой напиток. Итого $(4 + 2) \cdot 3 = 6 \cdot 3 = 18$. Скобки тут обязательны: сначала собираем все варианты одного шага, потом перемножаем шаги.

Ловушка – **группы с общими элементами**. На полке 5 книг про космос и 4 фантастических, но одна из них – фантастика про космос. Сколько есть книг «про космос или фантастика»? Не 9, а $5 + 4 - 1 = 8$: общую книгу посчитали в обеих группах, и одну лишнюю надо вычесть. Это те же круги Эйлера из главы 2.

Запомни. «И» – берём по варианту на каждом шаге – умножаем. «Или» – берём одно из нескольких групп – складываем. Если у групп есть общие элементы, общее вычитаем один раз.

3.3. Перестановки: кто за кем

НА ПАЛЬЦАХ



все очереди:

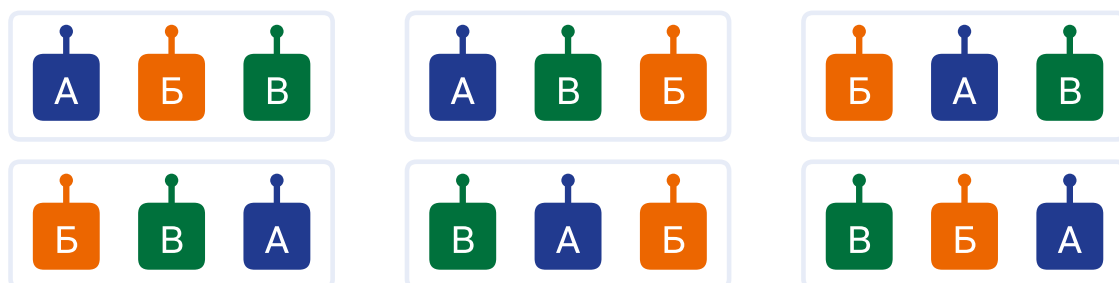


Рисунок 3.4 – Три робота встают в очередь на зарядку

На первое место – любой из трёх, на второе – любой из двух оставшихся, на третье – последний. $3 \cdot 2 \cdot 1 = 6$ очередей.

Три робота – А, Б и В – встают в очередь на зарядку. Сколькими способами это можно сделать? Выпишем все очереди: АВВ, АВВ, БАВ, БАВ, ВАВ, ВБА – шесть. Выписывай по порядку, иначе запутаешься: сначала все очереди, где первым стоит А, потом где первым Б, потом В.

А можно посчитать по местам – тем же правилом умножения. На первое место встаёт любой из 3 роботов. Когда первый встал, на второе место остаются 2. На третье – 1, последний. Всего $3 \cdot 2 \cdot 1 = 6$. Отличие от шапок и шарфов в том, что здесь каждый следующий шаг **беднее** предыдущего: робот, который уже стоит в очереди, второй раз в неё не встанет.

Такие расстановки всех предметов в ряд называют **перестановками**. Для четырёх роботов: $4 \cdot 3 \cdot 2 \cdot 1 = 24$. Для пяти: $5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$. Число растёт очень быстро. Десять школьников можно выстроить в очередь в столовой 3 628 800 способами. Если пробовать каждую минуту новый порядок без сна и выходных, уйдёт почти семь лет.

Произведение $1 \cdot 2 \cdot 3 \cdot \dots \cdot n$ встречается так часто, что у него есть имя – **факториал** – и значок $n!$ (читается «эн факториал»): $3! = 6$, $4! = 24$, $5! = 120$, $6! = 720$. Это просто короткая запись. Считать всё равно удобнее по местам.

Не все места. Из 10 команд надо выбрать призёров: золото, серебро и бронзу. Считаем по местам, но останавливаемся на третьем: $10 \cdot 9 \cdot 8 = 720$. Порядок здесь важен: «Альфа – золото,

Бета – серебро» и «Бета – золото, Альфа – серебро» – разные итоги турнира.

Когда порядок не важен. Пятеро друзей при встрече пожали руки – каждый каждому. Каждый жмёт руку четверым, и кажется, что рукопожатий $5 \cdot 4 = 20$. Но рукопожатие Ани с Борей и Бори с Аней – одно и то же, а мы посчитали его дважды. Делим на 2: $20/2 = 10$. Так же считают партии в турнире «каждый с каждым» и пары дежурных.

Перестановки умеет перебирать и Python. В модуле `itertools` – наборе готовых инструментов, который есть в любом Python, – есть функция `permutations` («перестановки»). Она выдаёт все расстановки сразу.

```
from itertools import permutations

ocheredi = list(permutations(["А", "Б", "В"]))
for o in ocheredi:
    print(*o)
print("всего:", len(ocheredi))           # всего: 6
print(len(list(permutations("АБВГД"))))  # 120 – очередей из пяти
```

`len` считает, сколько элементов в списке, а `print(*o)` печатает буквы одной очереди через пробел.

Запомни. Расставляем по местам: на каждое следующее место вариантов на один меньше. Все n предметов в ряд – $n! = n \cdot (n - 1) \cdot \dots \cdot 1$. Пары, где порядок не важен, – $\frac{n \cdot (n - 1)}{2}$.

Совет. Перед подсчётом спроси себя: «Если поменять двоих местами – это новый вариант?» Да – порядок важен (очередь, призовые места, код замка). Нет – дели на 2 (рукопожатия, пара дежурных).

3.4. Вероятность – доля подходящих исходов

НА ПАЛЬЦАХ

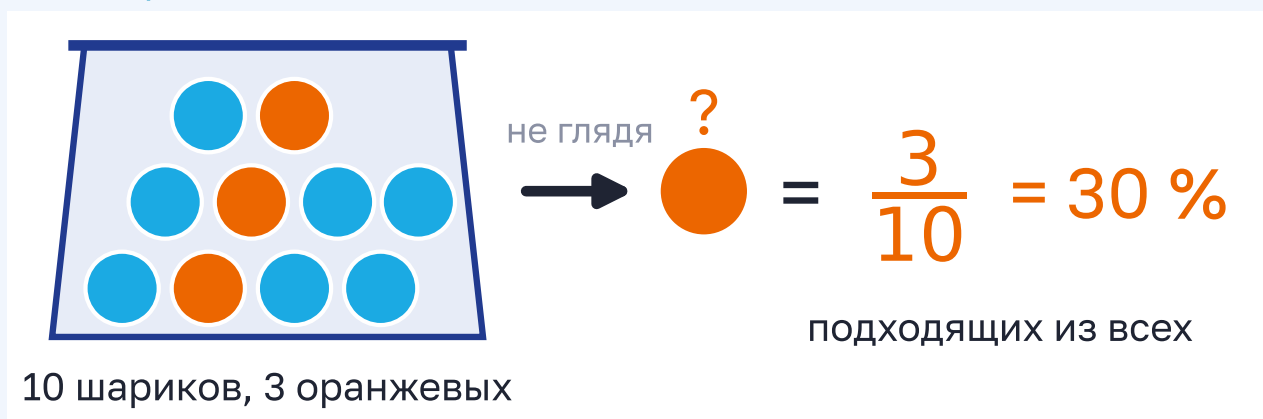


Рисунок 3.5 – Вытаскиваем шарик не глядя

Шариков 10, оранжевых среди них 3. Вероятность вытащить оранжевый – 3 из 10, то есть 30 %.

В мешке 10 одинаковых на ощупь шариков: 3 оранжевых и 7 голубых. Тянешь один не глядя. Может попасться любой из десяти, и ни у одного нет преимущества. Оранжевых – три из десяти. Поэтому говорят: **вероятность** вытащить оранжевый шарик равна $\frac{3}{10}$.

Несколько слов, без которых дальше не обойтись:

- **опыт** (эксперимент) – то, что мы делаем: бросаем монету, тянем шарик, включаем песню наугад;
- **исход** – чем опыт может закончиться: выпал орёл, попался шарик номер 7;
- **равновозможные** исходы – такие, что ни один не выпадает чаще других: монета ровная, кубик не кривой, шарики одинаковые на ощупь;
- **подходящие** (благоприятные) исходы – те, при которых случается нужное нам событие.

Вероятность – это доля подходящих исходов среди всех. Словами: **вероятность равна числу подходящих исходов, делённому на число всех исходов**. Буквами:

$$P = \frac{m}{n},$$

где P – вероятность, m – сколько исходов подходит, n – сколько всего равновозможных исходов. Дробь переводим в проценты умножением на 100: $\frac{3}{10} = 0,3 = 30 \%$.

- **Монета.** Исходов 2, орёл – один из них: $P = \frac{1}{2} = 50 \%$.
- **Кубик.** Исходов 6. Чётное число (2, 4 или 6) – 3 исхода: $\frac{3}{6} = \frac{1}{2} = 50 \%$. Шестёрка – один исход: $\frac{1}{6} \approx 16,7 \%$.
- **Выбор из списка.** Кого вызвать к доске, выбирает программа-жребий, и у всех 24 учеников класса шансы равные. Вероятность, что выпадет именно тебе, – $\frac{1}{24} \approx 4,2 \%$.

У вероятности две границы. Меньше 0 она не бывает: 0 – у события, которое никогда не случится (на кубике выпало 7). Больше 1 тоже не бывает: 1 – у события, которое случится всегда (на кубике выпало меньше 10). Вышло 1,3 или 130 % – ищи ошибку в решении.

Противоположное событие. «Вытащу оранжевый» и «не вытащу оранжевый» вместе покрывают все исходы. Поэтому $P(\text{не } A) = 1 - P(A)$. Не оранжевый: $1 - 0,3 = 0,7$, то есть 70 %. Иногда так считать заметно быстрее.

Две монеты. Сколько исходов, если бросить две монеты? Хочется сказать «три: два орла, две решки, орёл с решкой». Это главная ловушка темы, в неё попадали даже взрослые учёные. Покрась монеты мысленно: одна синяя, другая красная. Тогда исходы такие: ОО, ОР, РО, РР – четыре, и все равновозможные (дерево: $2 \cdot 2 = 4$). «Орёл с решкой» получается двумя способами – ОР и РО. Поэтому вероятность ровно одного орла – $\frac{2}{4} = 50\%$, а не $\frac{1}{3}$.

Вероятность двух орлов – $\frac{1}{4}$. Её можно найти и по-другому. В половине бросков синяя монета – орёл, и среди этих бросков в половине красная тоже орёл. Половина от половины: $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} = 25\%$. Это правило пригодится ещё не раз: вероятности двух **независимых** событий (одно никак не влияет на другое) перемножаются.

Два кубика. Исходов $6 \cdot 6 = 36$. Их удобно разложить в таблицу: строки – первый кубик, столбцы – второй, в клетке – сумма очков.

		второй кубик					
		1	2	3	4	5	6
первый кубик	1	2	3	4	5	6	7
	2	3	4	5	6	7	8
	3	4	5	6	7	8	9
	4	5	6	7	8	9	10
	5	6	7	8	9	10	11
	6	7	8	9	10	11	12

Рисунок 3.6 – Все 36 исходов броска двух кубиков. Сумма 7 выпадает в шести клетках из 36

Сумма 7 стоит в 6 клетках: $P = \frac{6}{36} = \frac{1}{6} \approx 16,7\%$. А сумма 2 – только в одной (1 и 1): $\frac{1}{36} \approx 2,8\%$. Суммы от 2 до 12 **не** равновозможные, поэтому рассуждать «сумм всего 11, значит, каждая – одна одиннадцатая» нельзя.

Программа проверит подсчёт перебором всех 36 клеток:

```
vse = 0
podhodyat = 0
for a in range(1, 7):          # первый кубик: 1, 2, ..., 6
    for b in range(1, 7):      # второй кубик
        vse += 1
        if a + b == 7:
            podhodyat += 1
print(podhodyat, "из", vse)    # 6 из 36
print(round(podhodyat / vse * 100, 1), "%") # 16.7 %
```

`range(1, 7)` – это числа от 1 до 6: правая граница в `range` не входит. `round(..., 1)` округляет до десятых.

Запомни. $P = \frac{\text{подходящие}}{\text{все}}$, и все исходы должны быть равновероятными. $0 \leq P \leq 1$. «Не случится» – $1 - P$. У двух монет 4 исхода, у двух кубиков – 36: каждую монету и каждый кубик считай отдельно.

Совет. Подходящие исходы выписывай в столбик парами (первый кубик, второй кубик): (1, 6), (2, 5), (3, 4)... Такой список легко проверить глазами, а в уме пары теряются. Если подходящих слишком много, пересчитай неподходящие – их бывает меньше.

3.5. Частота в эксперименте

НА ПАЛЬЦАХ

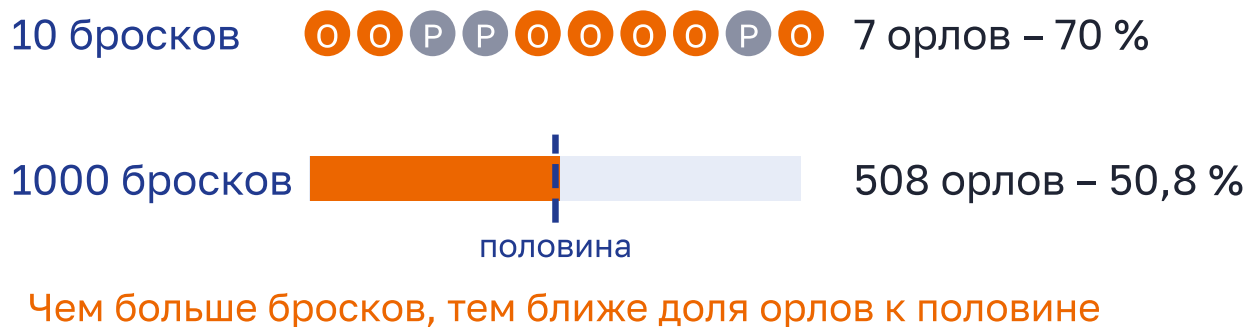


Рисунок 3.7 – Десять бросков и тысяча бросков

За 10 бросков орёл вполне может выпасть 7 раз. За 1000 бросков доля орлов подбегает к половине.

Вероятность орла – 50 %. Значит ли это, что из 10 бросков ровно 5 будут орлами? Попробуй – скорее всего, нет. Мы попросили компьютер бросить монету 10 раз и получили 7 орлов. Монета тут ни при чём: на коротком отрезке случай «гуляет».

Частота события – доля опытов, в которых оно случилось. Считают её так:

$$\text{частота} = \frac{\text{сколько раз случилось}}{\text{сколько было опытов}}.$$

7 орлов из 10 – частота $\frac{7}{10} = 70\%$. Бросаем дальше. После 100 бросков набралось 54 орла – 54 %. После 1000 – 508 орлов, 50,8 %.



Рисунок 3.8 – Доля орлов в одном опыте из 1000 бросков. Вначале линия скачет, потом прижимается к 50 %

Это одно из главных наблюдений теории вероятностей, его называют **законом больших чисел**: чем больше опытов, тем ближе частота к вероятности. Ближе, но не обязательно ровно: 508 орлов из 1000 – самое обычное дело.

Сколько раз ожидать. Если вероятность известна, то в N опытах событие случится примерно $P \cdot N$ раз. Кубик бросили 600 раз – шестёрка выпадет около $\frac{1}{6} \cdot 600 = 100$ раз. Может, 93, может, 108, но не 20 и не 300.

Вероятность по опыту. Бывает, что исходы не пересчитать. Вероятность того, что школьный автобус опоздает, по дереву не найдёшь. Зато можно вести записи: за 80 учебных дней автобус опоздал 12 раз. Частота $\frac{12}{80} = 0,15$, и вероятность опоздания завтра оценим в 15 %. Так же учится машинное обучение: модель смотрит на тысячи примеров и запоминает, как часто что встречается. Чем больше примеров, тем точнее оценка – тот же закон больших чисел.

Бросать монету тысячу раз руками долго, а компьютер справится за долю секунды. Модуль `random` (он тоже есть в любом Python) выдаёт случайные числа: `random.randint(0, 1)` – случайно 0 или 1, как монета.

```
import random

orly = 0
for n in range(1, 1001):
    if random.randint(0, 1) == 1:    # 1 - орёл, 0 - решка
        orly += 1
    if n in (10, 100, 1000):
        print(n, "бросков:", round(orly / n * 100, 1), "% орлов")
```

Запусти программу несколько раз. Числа каждый раз будут другими – на то он и случай. Но после 1000 бросков доля орлов почти всегда окажется где-то между 45 и 55 %.

Запомни. Вероятность – сколько должно выходить в среднем, частота – сколько вышло в нашем опыте. На малом числе опытов они могут сильно различаться, на большом – сближаются. Ожидаемое число случаев = $P \cdot N$.

Совет. «Вероятность 50 %» – не обещание «через раз». У монеты нет памяти: после трёх решек подряд вероятность орла на четвёртом броске всё те же 50 %.

3.6. Вероятность в ИИ: «модель уверена на 80 %»

НА ПАЛЬЦАХ



Рисунок 3.9 – Модель смотрит на фото и раздаёт проценты

Модель не говорит просто «кошка». Она выдаёт вероятность каждому классу, который знает, и вместе выходит 100 %.

В главе 1 модель отвечала одним словом: «спам» или «не спам». На самом деле модель-классификатор обычно сначала выдаёт **вероятности** для каждого класса: кошка – 0,80, собака – 0,15, лиса – 0,05. Все 100 % она делит между классами, которые знает. Ответом называет класс с наибольшей вероятностью, а это число – 80 % – называют **уверенностью** модели. В Teachable Machine (её проходят в модуле 9 программы) такие проценты видны при проверке модели: у каждого класса своя полоска.

Что значит «уверена на 80 %»? Не то, что на фото «на 80 % кошка»: кошка там либо есть, либо нет. Смысл другой, и тут выручает частота. Если модель хорошо обучена, то среди **всех** фото, про которые она сказала «кошка, 80 %», кошки окажутся примерно на 80 из каждых 100. На остальных двадцати она ошибётся. Уверенность – обещание про частоту верных ответов на многих примерах, а не гарантия для одного снимка.

Модель может и **переоценивать себя**: говорит «90 %», а права лишь в 70 случаях из 100. Проверяют это как в разделе 3.5: берут много ответов с одинаковой уверенностью и считают частоту верных. Такая модель похожа на одноклассника, который на каждый вопрос уверенно тянет руку, а отвечает верно через раз.

Порог уверенности. Часто модели разрешают отвечать, только если уверенность выше порога, а иначе зовут человека. Камера на въезде во двор распознала номер машины с уверенностью 55 % – шлагбаум не поднимается, фото уходит охраннику. Вот такое правило на Python:

3.7. Разбираем задачи

Задача 3.1 Сценарии бота-звездочёта

разминка · дерево возможностей, правило умножения

Для планетария делают чат-бота «Звездочёт». Сначала он предлагает тему: Луна, Марс, Солнце, кометы или МКС. Потом спрашивает, как рассказывать: текстом, картинками или голосом. В конце загадывает по теме загадку – лёгкую или трудную, на выбор посетителя. Сценарий разговора – вся цепочка выборов. Сколько разных сценариев может быть у «Звездочёта»? В ответ введи целое число.

Дано: тем – 5; способов рассказа – 3; уровней загадки – 2.

Найти: N – число сценариев.

Решение

Шаг 1. Что происходит. Разговор идёт в три шага, на каждом посетитель выбирает один вариант. Сценарий = тема + способ рассказа + уровень загадки. Это выбор «и»: по варианту на каждом шаге.

Шаг 2. Можно ли умножать. Число способов рассказа не зависит от темы: и про Луну, и про кометы бот рассказывает тремя способами. Уровней загадки после любой темы – два. Значит, правило умножения работает.

Шаг 3. Считаем. В дереве из начала выходят 5 веток-тем, каждая делится на 3 ветки-способа: $5 \cdot 3 = 15$ пар «тема + способ». Каждая пара делится ещё на 2 уровня загадки: $15 \cdot 2 = 30$.

Проверим с другого конца. Выпишем начало: «Луна – текст – лёгкая», «Луна – текст – трудная», «Луна – картинки – лёгкая»... На одну только Луну приходится $3 \cdot 2 = 6$ сценариев, а тем пять: $6 \cdot 5 = 30$. Сходится.

 **Типичная ошибка.** Сложить: $5 + 3 + 2 = 10$. Это число всех кнопок, которые бот может показать, а не число разговоров. Складывают, когда выбирают что-то одно из всех вариантов, а здесь выбирают на каждом шаге.

Ответ: 30.

По мотивам задачи отборочного этапа НТО Junior 2023/24 «Дерево возможностей»

Реши сам

3.1. У велосипедного замка три колёсика, на каждом – цифры от 0 до 9. Сколько разных кодов можно набрать на таком замке?

3.2. Бот-экскурсовод краеведческого музея предлагает выбрать зал (их 4), язык рассказа (русский или английский) и длительность экскурсии: 15, 30 или 45 минут. Сколько разных сценариев экскурсии может получиться?

Задача 3.2 Бот с развилкой

дерево возможностей, правила сложения и умножения

Чат-бот школьной библиотеки сначала спрашивает: «Подобрать книгу или фильм?» Для книги он уточняет, как подбирать: по автору, по жанру или по эпохе. Для фильма – по жанру или по году выхода. После рекомендации посетитель выбирает одно: мини-викторину по ней в одном из четырёх видов (текст, картинка, аудио или видео) или «спасибо, не надо» – тогда бот прощается. Сколько разных сценариев разговора у бота? В ответ введи целое число.

Дано: книга – 3 способа подбора; фильм – 2 способа; викторина – 4 вида; отказ – 1 вариант.

Найти: N – число сценариев.

Решение

Шаг 1. Делим разговор на шаги. Сначала бот выясняет, что и как подбирать, потом – чем закончить разговор.

Шаг 2. Первый шаг – это «или». Посетитель берёт книгу по одному из 3 признаков **или** фильм по одному из 2. Книгу и фильм сразу не подобрать, общих вариантов у веток нет. Складываем: $3 + 2 = 5$ вариантов первого шага.

Шаг 3. Второй шаг – тоже «или». Викторина одного из 4 видов или прощание: $4 + 1 = 5$ вариантов.

Шаг 4. Шаги между собой – «и». После любого из 5 вариантов подбора возможны все 5 окончаний. Умножаем: $5 \cdot 5 = 25$.

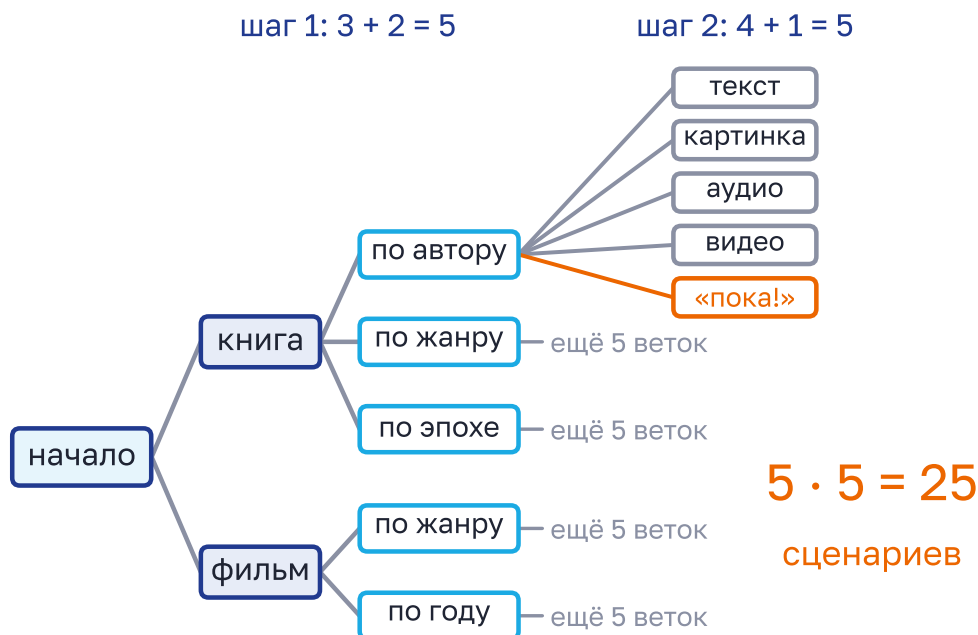


Рисунок 3.11 – Дерево бота с развилкой. Для одной ветки показаны все 5 окончаний, у остальных – такие же

А если по большим веткам? Ветка «книга» даёт $3 \cdot 5 = 15$ сценариев, ветка «фильм» – $2 \cdot 5 = 10$. Вместе $15 + 10 = 25$ – то же число.

❗ **Типичная ошибка.** Перемножить всё подряд: $3 \cdot 2 \cdot 4 = 24$. Но книга и фильм – ветки «или», их складывают. Коварство в том, что 24 очень похоже на 25, и прикидкой «на глаз» ошибку не поймать. Ещё часто забывают отказ от викторины и получают $5 \cdot 4 = 20$. Прощание – тоже конец разговора, то есть отдельный лист дерева.

Ответ: 25.

По мотивам задачи отборочного этапа НТО Junior 2023/24 «Дерево возможностей»

Реши сам

3.3. Бот-меню школьной столовой предлагает на первое суп (2 вида) или салат (3 вида), а к нему – один из 4 напитков. Сколько разных обедов «первое + напиток» можно заказать через бота?

3.4. Бот доставки сначала спрашивает: «Пицца или роллы?» Пицц в меню 6 видов, наборов роллов – 3. Потом бот предлагает один из двух соусов или «без соуса». Сколько разных заказов «еда + соус» можно оформить через бота?

Задача 3.3 Очередь на защиту

перестановки

На защите проектов выступают 5 команд, порядок определяет жребий. Команда «Сигма» просит поставить её последней: капитан едет из Бердска и успеет только к концу. Сколько существует порядков выступления, при которых «Сигма» – последняя?

Дано: команд – 5; «Сигма» – на 5-м месте.

Найти: N – число порядков.

Решение

Шаг 1. Что считаем. Порядок выступления – перестановка: команды занимают места с 1-го по 5-е, каждая по одному разу.

Шаг 2. Закрепляем «Сигму». Пятое место уже занято, выбора там нет, вариант один. Остаются 4 команды на 4 места.

Шаг 3. Считаем по местам. На 1-е место – любая из 4 оставшихся команд, на 2-е – любая из 3, на 3-е – из 2, на 4-е – последняя: $4 \cdot 3 \cdot 2 \cdot 1 = 24$.

Сверим со всеми порядками. Без условия их $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$. Последней может оказаться любая из пяти команд, и ни у одной нет преимущества. Значит, «Сигме» достаётся пятая часть порядков: $120/5 = 24$ – тот же ответ.

❗ **Типичная ошибка.** Считать «на первое место – любая из пяти» и получить 120. Но «Сигма» на первое место встать уже не может – она занята на пятом. Участника с закреплённым местом сразу убирают из выбора на остальные места.

Ответ: 24.

Своя задача по программе сферы, модуль 2

Реши сам

3.5. Шесть роботов встают в ряд для общей фотографии. Сколькими способами их можно расставить?

3.6. Петя и ещё трое ребят по очереди рассказывают о своих проектах. Петя очень не хочет выступать первым. Сколько существует порядков выступления, в которых Петя не первый?

3.7. В турнире по робофутболу играют 8 команд. Сколькими способами могут распределиться золотая, серебряная и бронзовая медали?

3.8. В шахматном кружке 7 человек. Каждый сыграл с каждым ровно одну партию. Сколько всего партий было сыграно?

Задача 3.4 Случайная песня

разминка · вероятность, выбор из списка

В плейлисте для поездки на дачу 20 песен: 8 – рок, 5 – рэп, 7 – поп. Включили режим «перемешать», первая песня выбирается случайно, у всех шансы одинаковые. Какова вероятность, что первой заиграет не рэп? Ответ дай в процентах.

Дано: песен – 20; из них рэп – 5.

Найти: $P(\text{не рэп})$ – ?

Решение

Шаг 1. Все исходы. Опыт – выбор первой песни. Исходов 20, все равновозможные.

Шаг 2. Подходящие исходы. «Не рэп» – это рок или поп: $8 + 7 = 15$ песен. То же самое проще: $20 - 5 = 15$.

Шаг 3. Вероятность. $P = \frac{15}{20} = \frac{3}{4} = 0,75$. В процентах: $0,75 \cdot 100 = 75\%$.

Можно и через противоположное событие: $P(\text{рэп}) = \frac{5}{20} = 0,25$, значит, $P(\text{не рэп}) = 1 - 0,25 = 0,75 = 75\%$.

⚠ Типичная ошибка. Ответить 25 % – это вероятность рэпа, а спрашивали «не рэп». Частичку «не» в условии полезно подчеркнуть. Другая ошибка – $\frac{5}{15}$: делить надо на число всех песен, а не только тех, что «не рэп».

Ответ: 75 %.

Своя задача по программе сферы, модуль 2

Реши сам

3.9. Бросают один кубик. Какова вероятность, что выпадет число больше 4? Ответ дай в процентах, округли до целых.

3.10. В классе 25 человек, из них 10 мальчиков. Дежурного на сегодня выбирает программа, случайно и с равными шансами для всех. Какова вероятность, что дежурной будет девочка? Ответ дай в процентах.

Задача 3.5 Выбраться из чёрной дыры

вероятность, два кубика

В настольной игре фишка вылетает из «чёрной дыры», только если на двух кубиках в сумме выпало не меньше 10. Какова вероятность вылететь с первого броска? Ответ дай в процентах, округли до целых.

Дано: кубиков – 2; нужна сумма ≥ 10 .

Найти: P – ?

Решение

Шаг 1. Все исходы. Кубики мысленно красим: красный и синий. Каждый даёт 6 вариантов, всего $6 \cdot 6 = 36$ равновозможных исходов.

Шаг 2. Подходящие исходы. «Не меньше 10» – это 10, 11 или 12. Выпишем пары (красный, синий):

- сумма 10: (4, 6), (5, 5), (6, 4) – 3 исхода;
- сумма 11: (5, 6), (6, 5) – 2 исхода;
- сумма 12: (6, 6) – 1 исход.

Всего $3 + 2 + 1 = 6$ подходящих исходов. В таблице двух кубиков из раздела 3.4 это маленький треугольник в правом нижнем углу.

Шаг 3. Вероятность. $P = \frac{6}{36} = \frac{1}{6} \approx 0,1667$. В процентах – 16,67 %, до целых – 17 %.

Много это или мало? Большие суммы на двух кубиках выпадают редко: нужны крупные числа сразу на обоих. Шесть клеток из 36 – меньше пятой части таблицы. Вот почему в таких играх из «дыры» обычно выбираются не с первого раза.

⚠ Типичная ошибка. Решить, что сумм всего 11 (от 2 до 12), а подходящих 3, и ответить $\frac{3}{11} \approx 27$ %. Суммы не равновозможные: 7 выпадает шестью способами, а 12 – одним. Вторая ошибка – посчитать (4, 6) и (6, 4) одним исходом. Тогда подходящих станет 4, пар без учёта порядка – 21, и выйдет $\frac{4}{21} \approx 19$ %. Тоже неверно: «5 и 5» выпадает одним способом, а «4 и 6» – двумя, такие исходы не равновозможные.

Ответ: 17 %.

Своя задача по программе сферы, модуль 2

Реши сам

3.11. Бросают два кубика. Какова вероятность «дубля» – одинаковых чисел на обоих кубиках? Ответ дай в процентах, округли до целых.

3.12. Бот для настольной игры бросает три виртуальные монеты. Какова вероятность, что выпадет хотя бы один орёл? Ответ дай в процентах.

Задача 3.6 Честна ли модель?

частота, уверенность модели

Фотоловушки в лесу под Новосибирском снимают всех, кто проходит мимо. Модель ищет на снимках лосей. Отобрали 60 снимков, про которые модель сказала «лось» с уверенностью 90 %. Экологи проверили их глазами: лось оказался на 42. Какова частота верных ответов модели на этих снимках в процентах? Честно ли модель оценивает свою уверенность?

Дано: снимков – 60; уверенность модели – 90 %; лось на самом деле – на 42 снимках.

Найти: частоту верных ответов в процентах; сравнить с уверенностью.

Решение

Шаг 1. Что проверяем. Уверенность 90 % обещает: среди многих таких ответов верных будет около 90 из 100. Проверить обещание можно только частотой – сразу на многих снимках.

Шаг 2. Частота. Верных ответов 42 из 60: $\frac{42}{60} = 0,7$. В процентах: $0,7 \cdot 100 = 70$ %.

Шаг 3. Сравниваем. Обещано 90 %, вышло 70 %: из каждых 100 таких ответов верных на 90 – 70 = 20 меньше, чем обещано. Если бы модель была честной, верных ответов было бы около $0,9 \cdot 60 = 54$, а их 42. Модель переоценивает себя – она самоуверенна.

Прикидка. 42 из 60 – больше половины, но далеко не «почти все». 70 % в эту вилку ложится.

⚠ Типичная ошибка. Сравнивать 42 с 90 или делить 42 на 90: это смешивание штук с процентами. Сначала переведи «42 из 60» в проценты, потом сравнивай. И не суди по одному снимку: «на этом фото лось нет, значит, модель врёт». Про один снимок уверенность ничего не обещает, судить можно только по многим.

Ответ: 70 %; модель самоуверенна – обещает 90 верных ответов из 100, а на деле выходит 70.

Своя задача по программе сферы, модули 1 и 2

Реши сам

3.13. Канцелярскую кнопку бросили на стол 400 раз, и 260 раз она упала остриём вверх. Сколько раз примерно она упадёт остриём вверх, если бросить её 1000 раз?

3.14. Модель оценивает себя честно: её уверенность совпадает с частотой верных ответов. Она дала 300 ответов, и у каждого уверенность 90 %. Сколько примерно ошибок среди этих ответов?

3.15. Модель различает на фото кошку, собаку и лису. Для очередного снимка она выдала: кошка – 70 %, собака – 20 %. Какую вероятность модель выдала для лисы? Какой ответ она назовёт?

Задача 3.7* Что допишет модель

вероятность цепочки событий, генеративный ИИ

Маленькая языковая модель продолжает фразу «Завтра в Новосибирске ...» двумя словами. Первое слово она выбирает так: «пойдёт» – с вероятностью 0,5, «будет» – 0,4, «ожидается» – 0,1. Второе слово зависит от первого. После «пойдёт»: «снег» – 0,45, «дождь» – 0,35, «град» – 0,2. После «будет»: «мороз» – 0,9, «оттепель» – 0,1. После «ожидается» – всегда «метель». Какое продолжение из двух слов модель пишет чаще всего и с какой вероятностью? Вероятность дай в процентах.

Дано: вероятности первого слова – 0,5; 0,4; 0,1; вероятности второго слова после каждого первого – по условию.

Найти: самую вероятную фразу из двух слов и её вероятность P .

Решение

Шаг 1. Рисуем дерево. Из начала – три ветки первого слова, на каждой подписана её вероятность. Из каждого первого слова – ветки второго.

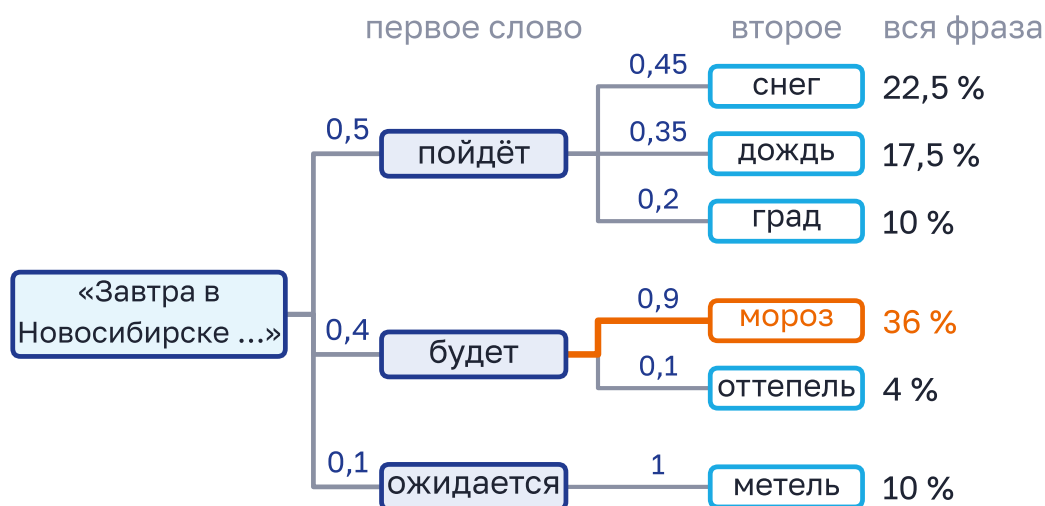


Рисунок 3.12 – Дерево продолжений фразы. У каждой ветки – вероятность, справа – вероятность всей фразы

Шаг 2. Как посчитать одну фразу. Возьмём «пойдёт снег». В половине запусков модель начинает со слова «пойдёт», и среди этих запусков в 45 % дописывает «снег». Доля от доли: $0,5 \cdot 0,45 = 0,225$, то есть 22,5 %. На пальцах: из 1000 запусков «пойдёт» появится при-

мерно в 500, а «снег» после него – в $500 \cdot 0,45 = 225$. Это тот же приём, что с сотней клеток в разделе 3.6.

Шаг 3. Считаем все шесть фраз.

Таблица 3.1

Фраза	Расчёт	Вероятность
пойдёт снег	$0,5 \cdot 0,45$	$0,225 = 22,5 \%$
пойдёт дождь	$0,5 \cdot 0,35$	$0,175 = 17,5 \%$
пойдёт град	$0,5 \cdot 0,2$	$0,1 = 10 \%$
будет мороз	$0,4 \cdot 0,9$	$0,36 = 36 \%$
будет оттепель	$0,4 \cdot 0,1$	$0,04 = 4 \%$
ожидается метель	$0,1 \cdot 1$	$0,1 = 10 \%$

Шаг 4. Выбираем наибольшую. Больше всех – «будет мороз», 36 %.

Не потеряли ли ветку? Модель всегда что-нибудь допишет, поэтому шесть фраз вместе обязаны дать 1: $0,225 + 0,175 + 0,1 + 0,36 + 0,04 + 0,1 = 1$. Все на месте. Ту же таблицу выдаст программа:

```
pervoe = {"пойдёт": 0.5, "будет": 0.4, "ожидается": 0.1}
vtoroe = {"пойдёт": {"снег": 0.45, "дождь": 0.35, "град": 0.2},
          "будет": {"мороз": 0.9, "оттепель": 0.1},
          "ожидается": {"метель": 1.0}}

for w1 in pervoe:
    for w2 in vtoroe[w1]:
        p = pervoe[w1] * vtoroe[w1][w2] # доля от доли
        print(w1, w2, round(p * 100, 1), "%")
```

 **Типичная ошибка.** Взять самое вероятное первое слово («пойдёт», 0,5), к нему самое вероятное второе («снег», 0,45) и ответить «пойдёт снег». Но у «пойдёт» вероятность размазана по трём продолжениям, а у «будет» почти вся досталась «морозу». Выбор «по одному лучшему слову за раз» не всегда даёт самую вероятную фразу. А кто-то складывает вероятности вдоль ветки: $0,4 + 0,9 = 1,3$. Больше 1 вероятность не бывает – это сигнал, что нужно было умножать.

Ответ: «будет мороз», 36 %.

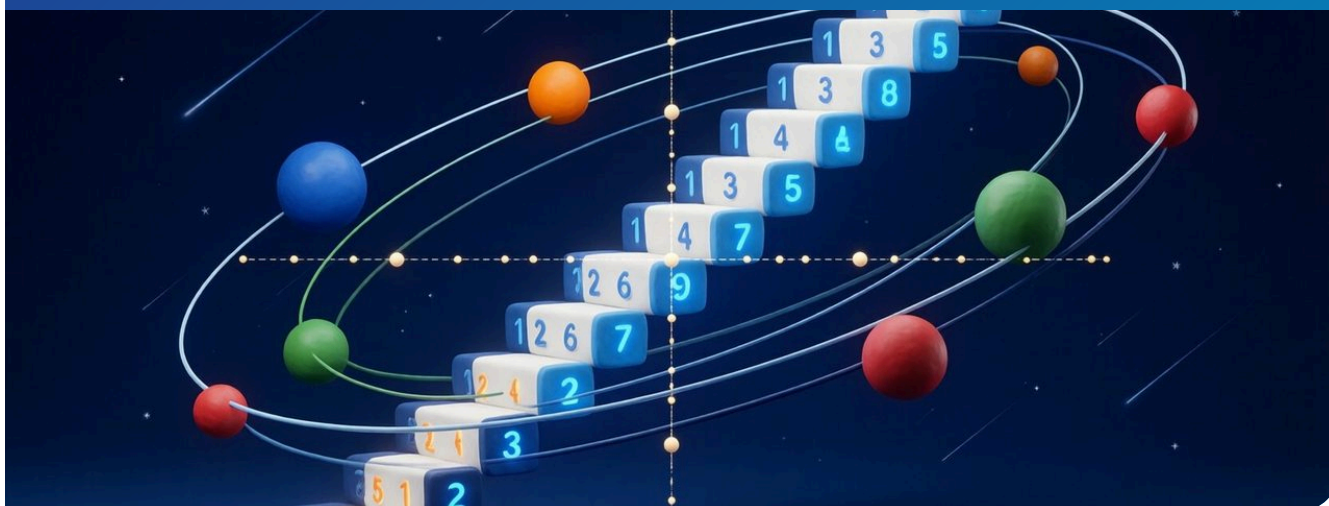
Своя задача по программе сферы, модули 1 и 2

Реши сам

3.16. Клавиатура телефона подсказывает продолжение фразы «Встретимся ...» двумя словами. Первое слово: «завтра» – с вероятностью 0,6, «вечером» – 0,4. После «завтра»: «утром» – 0,5, «днём» – 0,3, «вечером» – 0,2. После «вечером»: «дома» – 0,8, «там» – 0,2. Какое продолжение из двух слов клавиатура предложит чаще всего и с какой вероятностью? Вероятность дай в процентах.

3.17.* Чат-бот отвечает на вопрос «Как дела?» двумя словами. Первое слово: «всё» – с вероятностью 0,6, «отлично» – 0,3, «нормально» – 0,1. После «всё»: «хорошо» – 0,5, «отлично» – 0,3, «нормально» – 0,2. После «отлично»: «спасибо» – 0,8, «сегодня» – 0,2. После «нормально» – всегда «спасибо». Какова вероятность, что в ответе бота встретится слово «отлично» (первым или вторым)? Ответ дай в процентах.

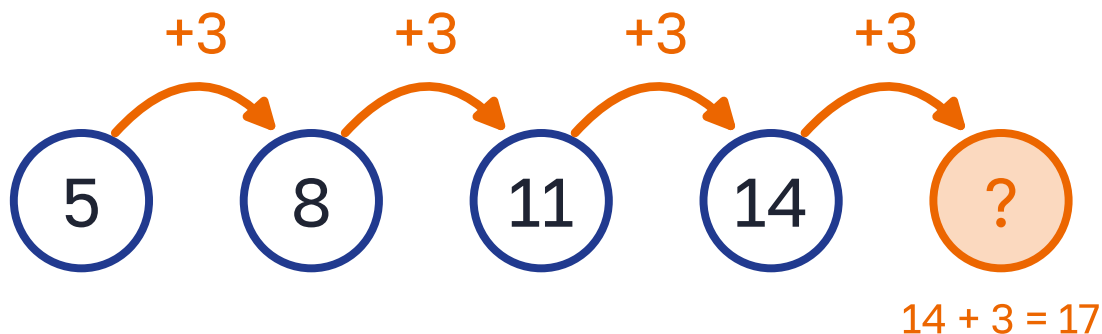
Закономерности, последовательности, уравнения



Прогноз погоды на завтра, подсказка следующего слова в телефоне, цена поездки в приложении такси – за всем этим одна идея: найти правило и по нему угадать то, чего ещё не видели. В этой главе мы ищем правила в рядах чисел, записываем их буквами, решаем уравнения, а в конце проводим прямую через облако точек. Так работает самая простая модель машинного обучения – линейная регрессия. Задачи на эти темы обычно короткие: ряд чисел с пропуском, история, из которой нужно составить уравнение, таблица или график, по которым надо сделать прогноз. Чаще всего в ответ вводят одно число.

4.1. Ряды чисел: ищем правило

НА ПАЛЬЦАХ



смотри не на сами числа, а на прыжки между соседями

Рисунок 4.1 – Ряд чисел, в котором спрятано правило

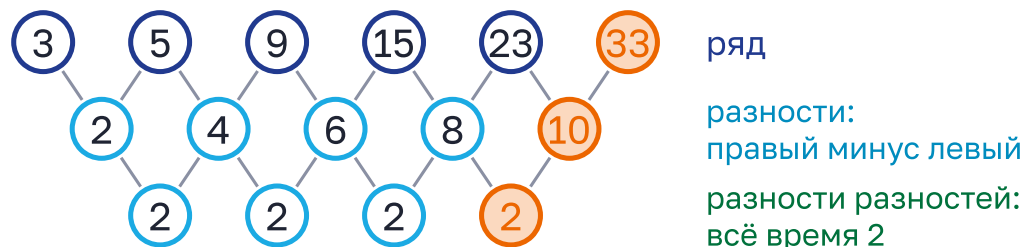
Между соседями каждый раз один и тот же прыжок: +3. Нашёл прыжок – знаешь, какое число будет дальше.

Числовая последовательность, или просто ряд, – числа, выписанные друг за другом по какому-то правилу. Каждое число ряда называют **членом**, и у каждого члена есть номер: первый, второй, третий... В ряду 5, 8, 11, 14 первый член – 5, а четвёртый – 14. Номер и сам член не путай: четвёртый член равен 14, а не 4.

Задача «найди следующее число» – это задача на поиск правила. Правило тебе никто не покажет. Его нужно угадать по нескольким числам и проверить на всех остальных. Так же учится и ИИ: ему дают примеры, он ищет в них закономерность и применяет её к новому. Компьютер может перебрать тысячи правил подряд, а тебе хватит пяти приёмов – пробуй их по порядку.

1. **Разности.** Из каждого числа вычти предыдущее. Если разности одинаковые, ряд каждый раз растёт (или уменьшается) на одно и то же число. Ряд 7, 12, 17, 22: разности 5, 5, 5, значит, дальше $22 + 5 = 27$.
2. **Отношения.** Раздели каждое число на предыдущее. Если выходит одно и то же, ряд каждый раз умножается на это число. Ряд 3, 6, 12, 24: разности 3, 6, 12 разные, зато $6:3 = 2$, $12:6 = 2$, $24:12 = 2$. Дальше $24 \cdot 2 = 48$. Подсказка, что пора делить: числа растут всё быстрее и быстрее.
3. **Разности разностей.** Если разности не равны, но сами идут по понятному правилу – ищи правило в них. Ряд 3, 5, 9, 15, 23: разности 2, 4, 6, 8 растут на 2. Следующая разность 10, следующий член $23 + 10 = 33$.
4. **Знакомые ряды.** Квадраты чисел: 1, 4, 9, 16, 25 – это $1 \cdot 1$, $2 \cdot 2$, $3 \cdot 3$ и так далее. Ряд, в котором каждое число – сумма двух предыдущих: 1, 1, 2, 3, 5, 8, 13, 21. Его называют рядом Фибоначчи, в честь итальянского математика, который описал его около восьмисот лет назад.
5. **Два ряда в одном.** Если числа скачут то вверх, то вниз, выпиши отдельно числа с нечётных мест и с чётных. Ряд 2, 10, 4, 20, 6, 30 – это 2, 4, 6 и 10, 20, 30 вперемешку. Дальше

идёт 8.



достраиваем снизу вверх: $2 \rightarrow 8 + 2 = 10 \rightarrow 23 + 10 = 33$

Рисунок 4.2 – Таблица разностей для ряда 3, 5, 9, 15, 23. Во второй строке – разности соседей, в третьей – разности разностей. Когда нижняя строка стала одинаковой, ряд достраивают снизу вверх

Одна честная оговорка. По трём-четырёх числам правило иногда можно придумать разными способами. Ряд 1, 2, 4 продолжится числом 8, если каждый раз удваивать, и числом 7, если разности растут: 1, 2, 3. Поэтому в задачах дают пять-шесть членов, а ты проверяй найденное правило на **каждом** данном числе, а не на первых двух. У ИИ та же беда: мало примеров – и модель находит не то правило.

Разности удобно считать и в Python:

```
ryad = [3, 5, 9, 15, 23]
raznosti = []
for i in range(1, len(ryad)):
    raznosti.append(ryad[i] - ryad[i - 1])
print(raznosti)          # [2, 4, 6, 8]
```

В первой строке мы кладем ряд в **список** – набор чисел в квадратных скобках. Во второй за-
водим пустой список для разностей. Цикл `for` проходит по номерам `i` от 1 до последнего:
`len(ryad)` – длина списка, 5, а `range(1, 5)` даёт номера 1, 2, 3, 4. Внимание: в Python счёт
идёт с нуля, поэтому `ryad[0]` – это первое число, 3. Строка `ryad[i] - ryad[i - 1]` вычитает из
числа его левого соседа, а `append` дописывает результат в конец списка.

Запомни. Порядок поиска: разности, отношения, разности разностей, знакомые ряды, два ряда вперемешку. Правило, которое подходит к первым двум числам, но не подходит к третьему, – не правило.

Совет. Выписывай разности прямо под рядом, мелкими цифрами в промежутках между чис-
лами. Следи за знаком: если число уменьшилось, разность отрицательная. У ряда 20, 17, 14
разности $-3, -3$.

4.2. Арифметическая прогрессия: n-й член без перебора

НА ПАЛЬЦАХ

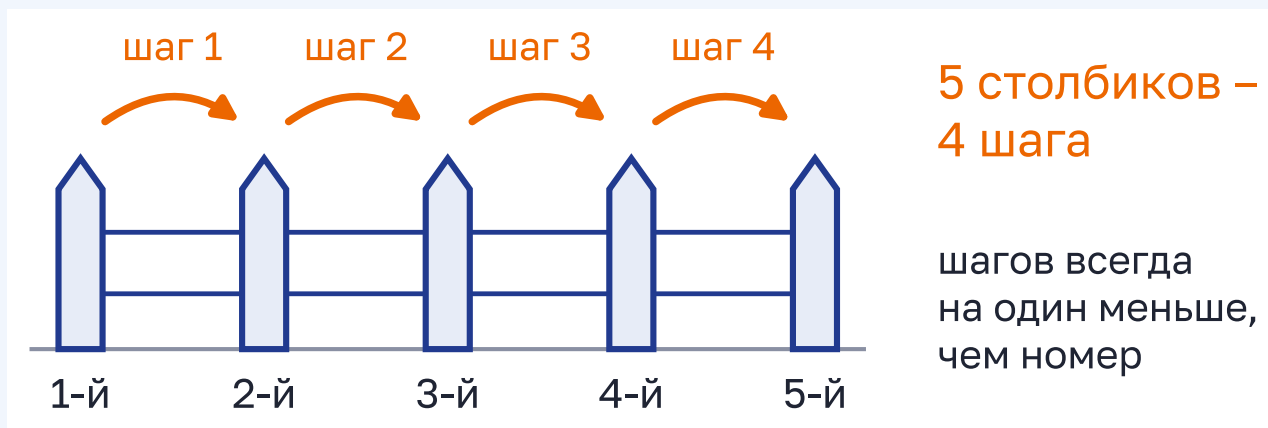


Рисунок 4.3 – Пять столбиков забора – четыре пролёта между ними

От первого члена до пятого – не пять шагов, а четыре. До сотого – 99.

Ряд, в котором к каждому числу прибавляют одно и то же, называется **арифметической прогрессией**. Число, которое прибавляют, – **шаг** прогрессии (в учебниках его зовут разностью), обозначают буквой d . Шаг бывает и отрицательным: 50, 46, 42, 38 – прогрессия с шагом -4 , она убывает.

Как найти, скажем, сотый член, не выписывая сто чисел? Встань на первый член и прыгай по ряду вперёд. Каждый прыжок прибавляет d . До второго члена – один прыжок, до третьего – два, до десятого – девять. До члена с номером n прыжков на один меньше, чем номер: $n - 1$.

Словами: **член с номером n равен первому члену плюс шаг, умноженный на $(n - 1)$** . Буквами:

$$a_n = a_1 + (n - 1) \cdot d,$$

где a_n – член с номером n (читается «а энное»), a_1 – первый член, d – шаг, n – номер.

Пример. Ряд 5, 8, 11, 14, ... Первый член 5, шаг 3. Сотый член:

$a_{100} = 5 + (100 - 1) \cdot 3 = 5 + 99 \cdot 3 = 5 + 297 = 302$. Проверим формулу на том, что видим своими глазами: четвёртый член $5 + 3 \cdot 3 = 14$ – как в ряду.

Обратная задача: какой у числа номер? Есть ли в ряду 5, 8, 11, ... число 65, и если есть, какое оно по счёту? От первого члена до 65 нужно пройти $65 - 5 = 60$. Это $60 : 3 = 20$ прыжков. Номер на единицу больше числа прыжков: $20 + 1 = 21$. Значит, 65 – двадцать первый член. Проверка: $5 + 20 \cdot 3 = 65$. А числа 66 в ряду нет: $66 - 5 = 61$ на 3 нацело не делится, на число 66 мы не попадём никаким числом прыжков.

Ряд можно построить и в Python – прыжок за прыжком или сразу по формуле:

```
a = 5      # первый член
d = 3      # шаг
for n in range(1, 6):
    print(n, a)      # номер и член
    a = a + d        # прыгаем к следующему члену
print(5 + (100 - 1) * 3)  # сотый член по формуле: 302
```

Программа печатает пары «номер – член»: 1 5, 2 8, 3 11, 4 14, 5 17, а последней строкой – 302. Запись `range(1, 6)` означает номера от 1 до 5: число 6 в счёт не входит. Строка `a = a + d` читается так: «возьми старое `a`, прибавь шаг и положи результат обратно в `a`».

Для любопытных. Сумму арифметической прогрессии можно найти без долгого сложения. Есть известная история про Карла Гаусса, будущего великого математика. Учитель велел классу сложить все числа от 1 до 100, а маленький Гаусс почти сразу дал ответ. Он заметил, что $1 + 100 = 101$, $2 + 99 = 101$, $3 + 98 = 101$... Таких пар 50, значит, сумма $50 \cdot 101 = 5050$. Для любой арифметической прогрессии: сумма равна (первый член + последний) $\cdot$ количество членов $: 2$.

НА ПАЛЬЦАХ

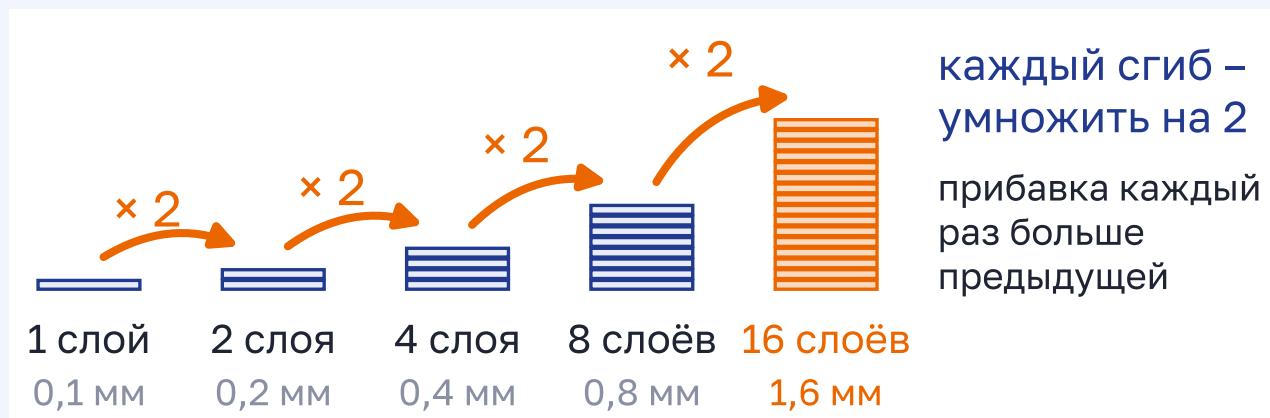


Рисунок 4.4 – Складываем лист пополам: слоёв каждый раз вдвое больше

Здесь каждый раз не прибавляют, а умножают на 2. Поэтому каждый следующий прирост больше предыдущего: сначала +0,1 мм, потом +0,2 мм, потом +0,4 мм.

Геометрическая прогрессия – ряд, в котором каждое число умножают на одно и то же. Это число называют **знаменателем** и обозначают q . Ряд 3, 6, 12, 24 – геометрическая прогрессия с $q = 2$. Логика прыжков та же, только прыжок теперь – умножение: до члена с номером n нужно $n - 1$ умножений, поэтому $b_n = b_1 \cdot q^{n-1}$. Запись q^{n-1} – степень: $n - 1$ множителей, и каждый равен q . Например, $2^3 = 2 \cdot 2 \cdot 2 = 8$.

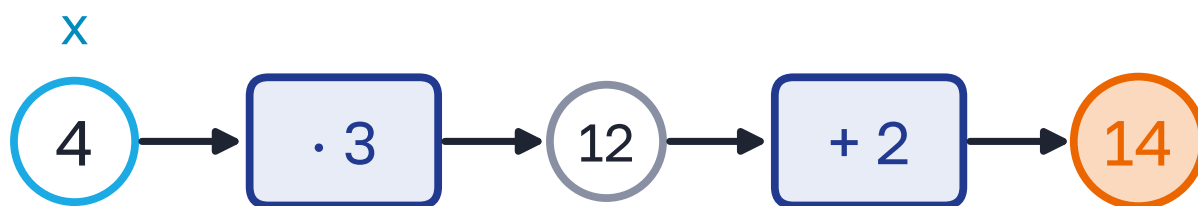
Такие ряды растут стремительно. Лист бумаги толщиной 0,1 мм после одного сгиба станет 0,2 мм, после второго – 0,4 мм. После десяти сгибов вышло бы $0,1 \cdot 2^{10} = 0,1 \cdot 1024 = 102,4$ мм – толще десяти сантиметров. Попробуй сложить настоящий лист хотя бы шесть раз подряд – к седьмому сгибу он почти перестаёт гнуться.

Запомни. $a_n = a_1 + (n - 1) \cdot d$: прыжков на один меньше, чем номер. Номер по числу – это (число – первый член) $: \text{шаг}$, и к результату прибавить 1. Не делится нацело – такого числа в ряду нет. В геометрической прогрессии вместо прибавления – умножение: $b_n = b_1 \cdot q^{n-1}$.

4.3. Выражения с переменными

НА ПАЛЬЦАХ

выражение $3 \cdot x + 2$



другое x – другой ответ: $x = 10 \rightarrow 3 \cdot 10 + 2 = 32$

Рисунок 4.5 – Выражение – машинка для чисел

Кладёшь в машинку число x – она выполняет действия по порядку и выдаёт ответ. Положил другое x – получил другой ответ, а машинка та же.

Переменная – буква, вместо которой можно подставить число. Представь коробку с надписью « x »: надпись одна и та же, а положить внутрь можно любое число. **Выражение с переменной**, например $3 \cdot x + 2$, – это рецепт: «возьми x , умножь на 3, прибавь 2». Подставить в него число и посчитать – значит найти **значение выражения**.

Считаем по обычному порядку действий: сначала скобки, потом умножение и деление, потом сложение и вычитание. Пример: $4 \cdot (x - 3) + x$ при $x = 10$. Скобка: $10 - 3 = 7$. Умножение: $4 \cdot 7 = 28$. Сложение: $28 + 10 = 38$.

Между числом и буквой знак умножения обычно не пишут: $3x$ – это $3 \cdot x$, а $2(x + 1)$ – это $2 \cdot (x + 1)$. В Python так нельзя. Там звёздочку ставят всегда, иначе программа выдаст ошибку:

```
x = 10
print(4 * (x - 3) + x)    # 38
```

Из слов – в выражение. Главное умение этого раздела – перевести историю на язык букв. Школьный автобус утром выезжает из гаража, в нём уже едут 6 человек, и на каждой остановке садятся ещё 4. Сколько людей в автобусе после n остановок? После одной остановки $6 + 4$, после двух $6 + 4 \cdot 2$, после n – $6 + 4n$. Подставим $n = 5$: $6 + 4 \cdot 5 = 26$ человек. Узнаёшь? Это арифметическая прогрессия из раздела 4.2, только номер теперь – переменная.

Таблица 4.1

Словами	Выражение
на 5 больше, чем x	$x + 5$
на 5 меньше, чем x	$x - 5$
в 5 раз больше, чем x	$5x$

Словами	Выражение
в 5 раз меньше, чем x	$x : 5$
x тетрадей по 45 Р и ручка за 30 Р	$45x + 30$

Подобные слагаемые. $2x + 3x = 5x$: две коробки с x да ещё три такие же – всего пять коробок. $7x - x = 6x$ (одинокое x – это $1 \cdot x$). Буквенные части складывают с буквенными, числа – с числами: $3x + 4 + 2x + 1 = 5x + 5$. Это пригодится в уравнениях, где x стоит с двух сторон.

Запомни. Переменная – место для числа. Подставил – посчитал в порядке: скобки, умножение и деление, сложение и вычитание. «На ...» – это плюс или минус, «в ... раз» – умножение или деление.

Совет. Составил выражение – проверь его на маленьком числе, которое легко посчитать в уме. Для автобуса: после одной остановки должно быть 10 человек, и правда $6 + 4 \cdot 1 = 10$.

4.4. Линейные уравнения

НА ПАЛЬЦАХ



Рисунок 4.6 – Уравнение – весы в равновесии

Снимешь с обеих чаш одно и то же – равновесие не нарушится. Сняли по 2 кг: остались три пакета против 12 кг, значит, один пакет весит $12 : 3 = 4$ кг.

Уравнение – равенство, в котором есть неизвестное число, обычно его обозначают x . **Решить уравнение** – найти, какое число вместо x делает равенство верным. Такое число называют **корнем** уравнения. Уравнения, где x просто умножается на число и к нему что-то прибавляется, вроде $4x + 9 = 37$, называют **линейными**. Откуда слово, станет ясно в разделе 4.5: у них график – прямая линия.

Правило весов: с обеими частями уравнения можно делать одно и то же – прибавлять, вычитать, умножать или делить на одно и то же число (только не делить на ноль). Равенство останется верным. Решаем, «снимая» с x всё лишнее в обратном порядке: что с x сделали последним, то снимаем первым.

Пример. $4x + 9 = 37$. С x сначала умножили на 4, потом прибавили 9 – значит, первой уходит девятка. Вычитаем 9 из обеих частей: $4x = 28$. Делим обе части на 4: $x = 7$. Проверка подстановкой: $4 \cdot 7 + 9 = 28 + 9 = 37$ – верно.

Таблица 4.2

Уравнение	Как найти x	Корень
$x + 8 = 20$	$x = 20 - 8$	12
$x - 8 = 20$	$x = 20 + 8$	28
$30 - x = 12$	$x = 30 - 12$	18
$6x = 42$	$x = 42 : 6$	7
$x : 4 = 9$	$x = 9 \cdot 4$	36
$60 : x = 5$	$x = 60 : 5$	12

Посмотри на третью и последнюю строки: там x стоит справа от знака минуса или деления, и правило другое. Если из 30 вычли что-то и осталось 12, то вычли $30 - 12 = 18$. Не уверен в строчке – подставь корень и проверь.

x с двух сторон. $5x + 3 = 2x + 18$. Снимем с обеих чаш по $2x$: $3x + 3 = 18$. Дальше как обычно: $3x = 15$, $x = 5$. Проверка: слева $5 \cdot 5 + 3 = 28$, справа $2 \cdot 5 + 18 = 28$ – равны.

Уравнение из истории. Сначала обозначь буквой то, что ищешь, и сразу напиши единицы: « x – масса одного датчика, г». Потом найди величину, про которую в условии сказано двумя способами, и приравняй. А решив уравнение, проверь ответ по условию задачи, а не по своему уравнению.

Пример. Коробка с пятью одинаковыми датчиками весит 640 г, пустая коробка – 90 г. Сколько весит один датчик? Пусть x – масса датчика в граммах. Массу полной коробки можно записать как $5x + 90$, и мы знаем, что она равна 640: $5x + 90 = 640$. Вычитаем 90: $5x = 550$. Делим на 5: $x = 110$ г. Проверка по условию: пять датчиков по 110 г – это 550 г, вместе с коробкой $550 + 90 = 640$ г. Сошлось.

Корень можно проверить и в Python:

```
x = 7
print(4 * x + 9 == 37)  # True – корень найден верно
```

Двойное `==` – это вопрос «равны ли?». Ответ `True` значит «да», `False` – «нет». Одинарное `=` в первой строке делает другое: кладёт число 7 в переменную `x`.

Запомни. Что делаешь с одной частью уравнения, делай и с другой. Действия с x снимаем в обратном порядке. Корень всегда проверяй подстановкой.

Совет. В задачах с историей проверяй ответ по тексту условия. Если уравнение составлено неверно, корень подойдёт к уравнению, но не к истории – и только проверка по условию это поймает.

4.5. Линейная зависимость: правило предсказания

НА ПАЛЬЦАХ

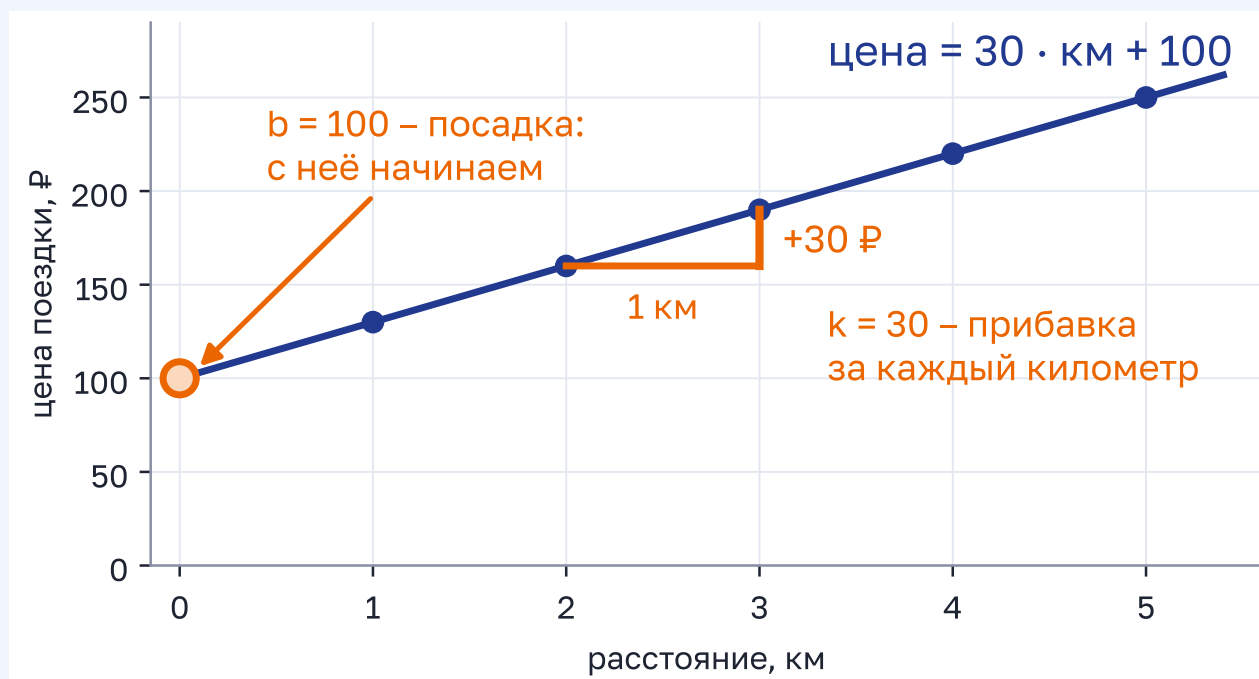


Рисунок 4.7 – Цена такси: плата за посадку и прибавка за каждый километр

b – с чего начинаем, цена поездки длиной 0 км. k – на сколько цена растёт с каждым километром. Точки ложатся на прямую линию.

Часто одна величина зависит от другой: цена поездки – от расстояния, заряд квадрокоптера – от времени в полёте, высота сугроба – от числа снежных дней. Если каждый шаг по одной величине прибавляет к другой одно и то же, зависимость называют **линейной** и записывают так:

$$y = k \cdot x + b.$$

- x – то, что мы знаем, вход: километры, минуты, градусы;
- y – то, что хотим узнать, выход или прогноз;
- k – на сколько меняется y , когда x увеличивается на 1 (его называют **коэффициентом наклона**);
- b – чему равен y , когда $x = 0$: стартовое значение.

Для такси с картинки $k = 30$ руб за километр, $b = 100$ руб: $y = 30x + 100$. Поездка 7 км стоит $30 \cdot 7 + 100 = 310$ руб. Подставь $x = 0, 1, 2, 3$ – получишь 100, 130, 160, 190. Это арифметическая прогрессия с шагом k ! Линейная зависимость – та же прогрессия, только x может быть и дробным: поездка 2,5 км стоит $30 \cdot 2,5 + 100 = 175$ руб.

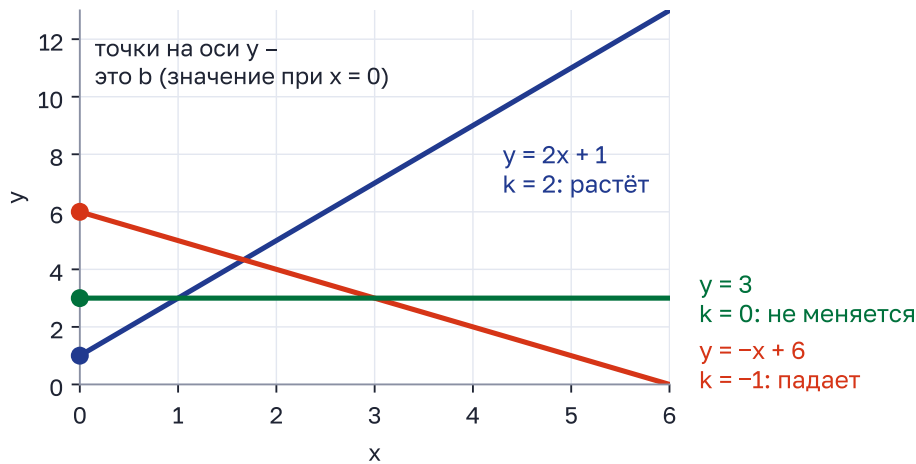


Рисунок 4.8 – Знак k показывает, куда идёт прямая. k больше нуля – растёт, меньше нуля – падает, равен нулю – стоит на месте

Как найти k и b по двум точкам. Сугроб у школы растёт равномерно: на 3-й день снегопадов в нём было 26 см, на 7-й день – 38 см. Сначала наклон: **k равно тому, на сколько изменился y , делённому на то, на сколько изменился x :**

$$k = \frac{y_2 - y_1}{x_2 - x_1} = \frac{38 - 26}{7 - 3} = \frac{12}{4} = 3 \text{ см в день.}$$

Здесь $(x_1; y_1) = (3; 26)$ – первая точка, $(x_2; y_2) = (7; 38)$ – вторая. Теперь b : подставим первую точку в правило, $26 = 3 \cdot 3 + b$, откуда $b = 26 - 9 = 17$ см. Правило: $y = 3x + 17$. Проверим второй точкой: $3 \cdot 7 + 17 = 38$ – верно. Что значит $b = 17$? Столько снега лежало в «нулевой» день, до начала подсчёта.

Прогноз – подставить x в правило. **Обратный вопрос** – «когда y станет таким-то?» – это уравнение. Когда сугроб дорастёт до 50 см? $3x + 17 = 50$, $3x = 33$, $x = 11$ – на 11-й день.

При чём тут ИИ. Правило $y = kx + b$ – самая простая модель машинного обучения. Задача «предскажи число» называется регрессией (глава 1), а модель-прямая – **линейной регрессией** (linear regression). Числа k и b – её **параметры**: k работает как вес нейрона, а b – как его смещение. Когда говорят «модель обучилась», это значит, что компьютер сам подобрал k и b по примерам. На сфере работу в конструкторе GraphNet начинают как раз с линейной регрессии – об этом глава 7.

Запомни. $y = kx + b$: k – прибавка на каждую единицу x (со знаком), b – значение при $x = 0$. $k = (y_2 - y_1) : (x_2 - x_1)$. Прогноз – подставить x . «Когда станет...» – решить уравнение.

Совет. Нашёл k и b по двум точкам – подставь третью точку из таблицы. Не сошлось – либо ошибка в счёте, либо зависимость не линейная. И считай разности всегда в одном порядке: вторая точка минус первая, и для y , и для x .

4.6. Прямая через облако точек и ошибка прогноза

НА ПАЛЬЦАХ

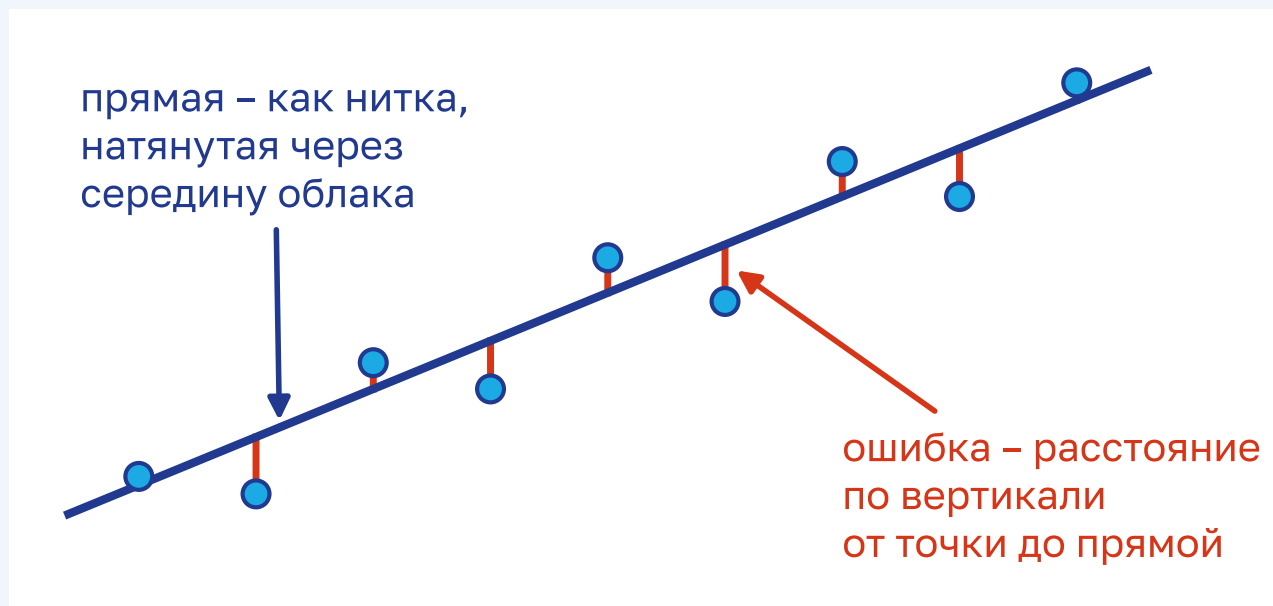


Рисунок 4.9 – Настоящие данные – облако точек, а не ровная линия

Модель – прямая, натянутая через середину облака, как нитка. Красные отрезки – ошибки: на сколько прогноз промахнулся в каждой точке.

В жизни точки почти никогда не ложатся на прямую ровно. Продажи кваса зависят от жары, но ещё и от дня недели, от того, пошёл ли дождь, приехал ли автобус с туристами. Такой разброс называют **шумом**. Задача линейной регрессии – провести одну прямую, которая проходит как можно ближе ко всем точкам сразу.

Как провести прямую на глаз:

1. Посмотри на облако целиком. Вытянуто вдоль линии – хорошо. Если точки лежат кругом или выгибаются дугой, прямая – плохая модель, и никакая линейка её не спасёт.
2. Приложи линейку так, чтобы точки были и сверху, и снизу, примерно поровну, а прямая шла вдоль всего облака – от левого края до правого.
3. Не гонись за одной странной точкой. **Выброс** – значение, которое сильно отличается от остальных: датчик заглянул, кто-то опечатался при вводе. Прямая, которая тянется к выбросу, хуже предсказывает все остальные точки.
4. Выбери на своей прямой две точки подальше друг от друга, лучше в узлах сетки, и найди по ним k и b , как в разделе 4.5.

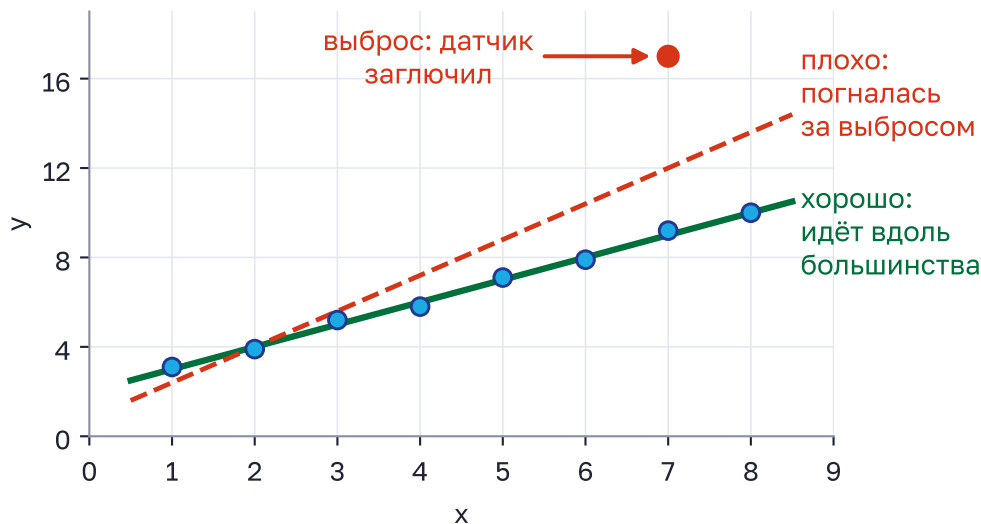


Рисунок 4.10 – Выброс тянет прямую за собой. Зелёная прямая идёт вдоль восьми нормальных точек, красная тянется к одной ошибочной – и уходит от остальных точек вверх

Два человека проведут прямые чуть по-разному, и их прогнозы разойдутся на пару единиц. Это нормально: «на глаз» – значит приблизительно. В задачах этой книги прямая уже проведена или названы две её точки, чтобы ответ был один.

Ошибка прогноза. Для каждой точки:

$$\text{ошибка} = \text{настоящее значение} - \text{прогноз}.$$

Ошибка со знаком плюс – модель **занизила**: прогноз меньше правды. Со знаком минус – **завысила**. Когда сравнивают модели, знак отбрасывают и смотрят только на размер промаха.

Средняя ошибка – та, что в главе 1: сложить ошибки без знака и разделить на число точек. Чем она меньше, тем лучше прямая описывает данные.

Пример. Прямая $y = 2x + 1$ и четыре точки: $(1; 4)$, $(2; 4)$, $(3; 8)$, $(4; 8)$. Прогнозы: 3, 5, 7, 9. Ошибки: $4 - 3 = +1$, $4 - 5 = -1$, $8 - 7 = +1$, $8 - 9 = -1$. Средняя ошибка без знака $(1 + 1 + 1 + 1) : 4 = 1$. А теперь сложи ошибки со знаком: $1 - 1 + 1 - 1 = 0$. Выходит, будто модель не ошиблась ни разу, хотя она промахнулась в каждой точке. Вот зачем знак отбрасывают.

Где прямой верить. Прогноз надёжен внутри облака и чуть-чуть за его краями. Далеко за краем прямая может выдать чушь. Прямая, которая хорошо предсказывает летние продажи мороженого, в январский мороз пообещает отрицательное число порций. Модель знает только то, что видела в примерах.

Для любопытных. Компьютер проводит прямую не на глаз. Чаще всего он подбирает k и b так, чтобы сумма **квадратов** ошибок была как можно меньше. Этот способ называют методом наименьших квадратов. Квадрат строже наказывает за большие промахи: одна ошибка на 10 даёт 100, а десять ошибок по 1 – всего 10. Нейросети учатся похоже, только постепенно: чуть-чуть меняют параметры, смотрят, уменьшилась ли ошибка, и повторяют так много раз.

Запомни. Ошибка = настоящее – прогноз: плюс – модель занизила, минус – завысила. Две модели сравнивают по средней ошибке **без знака** – у кого она меньше, та и лучше. Одна странная точка – не повод гнуть прямую, а далеко за краем облака прямой не верь.

4.7. Разбираем задачи

Задача 4.1 Какое число дальше

разминка · последовательности, разности, отношения

Найди следующее число в каждом ряду.

а) 7, 12, 17, 22, ...

б) 3, 6, 12, 24, ...

в) 2, 3, 6, 11, 18, 27, ...

Дано: три ряда чисел.

Найти: следующий член каждого ряда.

Решение

Шаг 1. Ряд а), разности. $12 - 7 = 5$, $17 - 12 = 5$, $22 - 17 = 5$. Все разности равны 5 – каждый раз прибавляем пять. Дальше $22 + 5 = 27$.

Шаг 2. Ряд б), разности. $6 - 3 = 3$, $12 - 6 = 6$, $24 - 12 = 12$ – разные. Числа растут всё быстрее – пробуем делить.

Шаг 3. Ряд б), отношения. $6 : 3 = 2$, $12 : 6 = 2$, $24 : 12 = 2$. Каждый раз умножаем на 2. Дальше $24 \cdot 2 = 48$.

Шаг 4. Ряд в), разности. $3 - 2 = 1$, $6 - 3 = 3$, $11 - 6 = 5$, $18 - 11 = 7$, $27 - 18 = 9$. Разности разные, но это нечётные числа подряд: 1, 3, 5, 7, 9. Разности разностей – всё время 2. Следующая разность $9 + 2 = 11$, следующий член $27 + 11 = 38$.

Ряд б) разгоняется, поэтому и ответ у него «прыгнул» сильнее всех: с 24 сразу на 48.

Посмотрим на все три ответа вместе. Ряды растут, и каждый ответ больше последнего числа своего ряда. В ряду б) рост ускоряется, поэтому и ответ «прыгнул» сильнее всех: с 24 сразу на 48.

 **Типичная ошибка.** Увидеть последнюю разность и прибавить её ещё раз. В ряду б) последняя разность 12, и выходит $24 + 12 = 36$. В ряду в) последняя разность 9, и выходит тоже $27 + 9 = 36$. Разность не обязана повторяться – сначала найди, по какому правилу меняются сами разности.

Ответ: а) 27; б) 48; в) 38.

Своя задача по программе сферы, модуль 2

Реши сам

4.1. Найди следующее число в ряду: 81, 27, 9, 3, ...

4.2. Найди следующее число в ряду: 1, 3, 7, 15, 31, ...

4.3.* В первый день ролик кружка посмотрели 5 человек. Каждый следующий день ролик смотрели втрое больше людей, чем накануне. В какой по счёту день число просмотров за день впервые станет больше 10 000?

Задача 4.2 Скачиваем каталог по страницам

арифметическая прогрессия, n -й член, округление вверх

Сервер с каталогом фильмов отдаёт данные страницами: на каждой странице 25 записей, записи идут по порядку номеров (на первой странице – записи с 1 по 25, на второй – с 26 по 50 и так далее). Всего в каталоге 318 записей. Один запрос к серверу приносит одну страницу.

а) Сколько запросов нужно сделать, чтобы скачать весь каталог?

б) С записи с каким номером начинается 9-я страница?

в) На какой странице лежит запись № 250?

Дано: 25 записей на странице; всего 318 записей.

Найти: число запросов; первую запись 9-й страницы; страницу записи № 250.

Решение

Шаг 1. Зачем это считать. Программы, которые собирают данные из интернета, почти всегда получают их порциями – страницами. Сколько страниц придётся попросить, считают заранее.

Шаг 2. а) Полные страницы. $318 : 25 = 12,72$. Значит, 12 страниц заполнены целиком – на них $12 \cdot 25 = 300$ записей.

Шаг 3. а) Хвост. Осталось $318 - 300 = 18$ записей. Им нужна ещё одна, 13-я страница, неполная. Всего 13 запросов. Число страниц округляем **вверх**: даже одна лишняя запись требует отдельного запроса.

Шаг 4. б) Первые записи страниц. Первые номера страниц – 1, 26, 51, 76, ... Это арифметическая прогрессия: первый член 1, шаг 25. До 9-й страницы – $9 - 1 = 8$ прыжков:
 $a_9 = 1 + 8 \cdot 25 = 201$.

Шаг 5. в) Страница записи № 250. Делим номер записи на число записей на странице:
 $250 : 25 = 10$ – ровно. Значит, запись № 250 – последняя на 10-й странице. Проверим: 10-я страница начинается с записи $1 + 9 \cdot 25 = 226$ и кончается записью $10 \cdot 25 = 250$.

Шаг 6. Проверка программой.

```
import math
vsego = 318      # записей в каталоге
na_str = 25      # записей на странице
print(math.ceil(vsego / na_str))    # а) 13
print(1 + (9 - 1) * na_str)         # б) 201
print(math.ceil(250 / na_str))      # в) 10
```

`import math` подключает набор математических функций, а `math.ceil` округляет число вверх до целого: $12,72 \rightarrow 13$.

Не многовато ли 13? 13 страниц по 25 – это место на 325 записей, чуть больше 318. А 12 страниц вместили бы только 300. Меньше 13 никак.

⚠ Типичная ошибка. В а) – округлить $12,72$ по обычным правилам или вниз и ответить 12: последние 18 записей так и не скачаются. В б) – сделать столько прыжков, сколько номер: $1 + 9 \cdot 25 = 226$. Это первая запись 10-й страницы. В в) ловушка появляется, когда деление не нацело: запись № 260 даёт $260 : 25 = 10,4$, и страница здесь 11-я (округляем вверх), а не 10-я.

Ответ: а) 13 запросов; б) с записи № 201; в) на 10-й странице.

По мотивам задачи отборочного этапа НТО Junior 2023/24 «Сбор данных»

Реши сам

4.4. Нейросеть рисует узор из клеток. В первом ряду 5 клеток, в каждом следующем – на 4 больше, чем в предыдущем. Сколько клеток в 30-м ряду?

4.5. Сервер отдаёт данные страницами по 40 записей, записи идут по порядку номеров (на первой странице – с 1 по 40). На какой странице лежит запись № 587?

4.6. Ряд 10, 16, 22, 28, ... – арифметическая прогрессия. Каким по счёту членом в нём стоит число 250?

Задача 4.3 Сколько рисует нейросеть

уравнение из истории, перевод единиц времени

Нейросеть-художник сначала загружается – на это уходит 12 секунд, – а потом рисует картинку по одной, каждую за одно и то же время. Лиза заказала 8 картинок, и от нажатия кнопки до появления последней картинки прошла 1 минута 40 секунд. Сколько секунд нейросеть рисует одну картинку?

Дано: загрузка 12 с; 8 картинок; всё вместе – 1 мин 40 с.

Найти: x – время на одну картинку, с.

Решение

Шаг 1. Переводим единицы. Загрузка дана в секундах, значит, и общее время нужно в секундах: $1 \text{ мин } 40 \text{ с} = 60 + 40 = 100 \text{ с}$.

Шаг 2. Обозначаем. Пусть x – время на одну картинку в секундах.

Шаг 3. Составляем уравнение. Общее время можно записать двумя способами. Через x : загрузка плюс 8 картинок, то есть $12 + 8x$. И по условию оно равно 100 с. Приравняем: $12 + 8x = 100$.

Шаг 4. Решаем. Вычитаем 12 из обеих частей: $8x = 88$. Делим обе части на 8: $x = 11$.

Шаг 5. Проверка по условию. Загрузка 12 с, восемь картинок по 11 с – это 88 с, всего $12 + 88 = 100 \text{ с} = 1 \text{ мин } 40 \text{ с}$, как в условии.

А если бы без загрузки? Тогда на картинку ушло бы $100 : 8 = 12,5 \text{ с}$. Загрузка съела часть времени, так что ответ обязан быть меньше 12,5 – и 11 с сюда подходит.

 **Типичная ошибка.** Записать 1 мин 40 с как 140 с или как 1,4 мин. В минуте 60 секунд, а не 100. А ещё забывают про загрузку и делят всё время на 8: получается 12,5 с.

Ответ: 11 с.

Своя задача по программе сферы, модуль 2

Реши сам

4.7. Реши уравнение $4x - 9 = 31$.

4.8. 3D-принтер в кружке сначала 6 минут разогревается, а потом печатает одинаковые фигурки, каждую за одно и то же время. Девять фигурок вместе с разогревом заняли 1 час 36 минут. Сколько минут печатается одна фигурка?

4.9. Сопоставь описание и выражение. 1) на 3 больше, чем удвоенное x ; 2) удвоенная сумма x и 3; 3) x уменьшили на 3, а результат удвоили; 4) x уменьшили в 3 раза, а потом прибавили 2. Выражения: А) $2(x - 3)$; Б) $2x + 3$; В) $x : 3 + 2$; Г) $2(x + 3)$. Ответ запиши парами «цифра – буква».

4.10. Найди значение выражения $5a - 2b$ при $a = 7$ и $b = 4$.

Задача 4.4 Заряд квадрокоптера

линейная зависимость, k и b по двум точкам, уравнение

Во время полёта заряд аккумулятора квадрокоптера падает равномерно. Через 2 минуты после взлёта пульт показал 88 %, через 5 минут – 70 %. По правилам кружка квадрокоптер сажают, когда заряд опускается до 10 %.

а) С каким зарядом квадрокоптер взлетел?

б) Через сколько минут после взлёта его нужно сажать?

Дано: $x_1 = 2$ мин, $y_1 = 88$ %; $x_2 = 5$ мин, $y_2 = 70$ %; посадка при 10 %.

Найти: b – заряд при взлёте; x – время до посадки, мин.

Решение

Шаг 1. Что происходит. «Падает равномерно» – значит, каждую минуту заряд уменьшается на одно и то же число процентов. Зависимость линейная: $y = kx + b$, где x – минуты полёта, y – заряд в процентах.

Шаг 2. Находим k . За $5 - 2 = 3$ минуты заряд изменился на $70 - 88 = -18$ %. За одну минуту: $k = -18 : 3 = -6$ % в минуту. Минус – потому что заряд падает.

Шаг 3. Находим b . Подставим первую точку: $88 = -6 \cdot 2 + b$, то есть $88 = -12 + b$, и $b = 88 + 12 = 100$ %. Это ответ на вопрос а): в момент взлёта ($x = 0$) заряд был 100 %.

Шаг 4. Проверяем правило. $y = -6x + 100$. Вторая точка: $-6 \cdot 5 + 100 = 70$ – верно.

Шаг 5. Когда посадить. Нужно, чтобы заряд стал 10 %: $-6x + 100 = 10$. Прибавим $6x$ к обеим частям, чтобы избавиться от минуса: $100 = 10 + 6x$. Вычтем 10: $6x = 90$. Разделим на 6: $x = 15$ мин.

Взлетели с полной батареей – логично. А четверть часа в воздухе – обычное время для небольшого квадрокоптера.

Много это или мало? От 100 % до 10 % – это 90 процентных пунктов, по 6 в минуту как раз выходит 15 минут. Четверть часа в воздухе – обычное время для небольшого квадрокоптера.

 **Типичная ошибка.** Потерять минус у k : тогда $b = 88 - 12 = 76$, и выходит, что заряд при взлёте был меньше, чем через 2 минуты полёта, – так не бывает. И не попадись на «10 минут»: столько осталось лететь **после** пятой минуты, а спрашивали время **после** взлёта.

Ответ: а) 100 %; б) через 15 минут после взлёта.

Своя задача по программе сферы, модуль 2

Реши сам

4.11. Воду в кастрюле греют на плите, температура растёт равномерно. В начале было 20 °С, через минуту – 28 °С, через две – 36 °С, через три – 44 °С. Через сколько минут от начала вода нагреется до 100 °С?

4.12.* Весной сугроб у крыльца тает равномерно. 1 апреля его высота была 60 см, 6 апреля – 45 см. Какого числа апреля от сугроба ничего не останется, если он будет таять так же? Ответ – число месяца.

Задача 4.5 Квас и жара

линейная регрессия, прямая по графику, ошибка прогноза

Хозяйка киоска на берегу Обского моря неделю записывала температуру воздуха в полдень и число стаканов кваса, проданных за день. Каждая точка на графике – один день. Аналитик

провёл через точки прямую, она проходит через два узла сетки, отмеченных оранжевыми кружками.

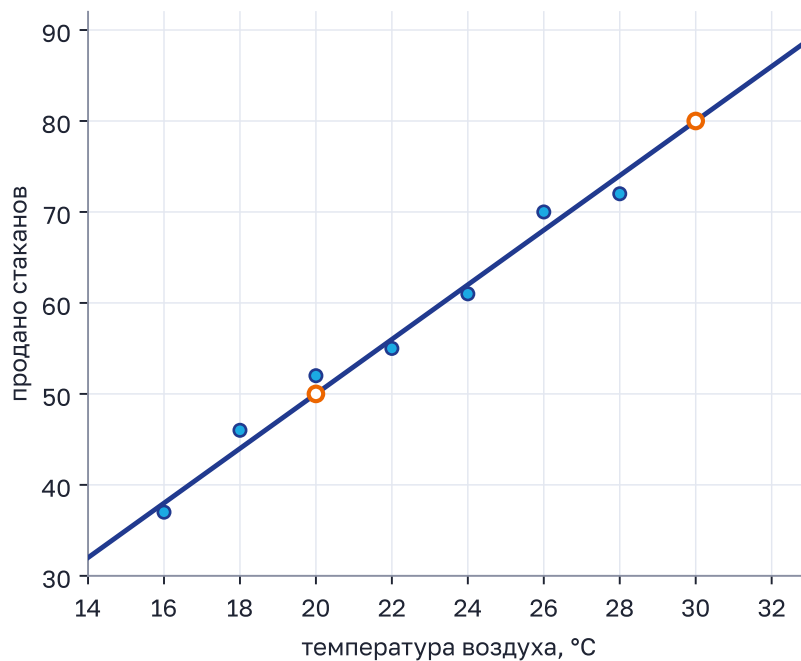


Рисунок 4.11 – Продажи кваса и температура воздуха. Голубые точки – семь дней, синяя прямая – модель

- а) Запиши правило прямой в виде $y = kx + b$, где x – температура в градусах, y – число стаканов.
- б) Сколько стаканов прямая предсказывает на день с температурой 32 °C?
- в) В день, когда было 26 °C, продали 70 стаканов. Какова ошибка прогноза в этот день?

Дано: график, прямая через два узла сетки.

Найти: k , b ; прогноз при 32 °C; ошибку при 26 °C.

Решение

Шаг 1. Читаем точки прямой. Кружки стоят в узлах (20; 50) и (30; 80): при 20 °C прямая показывает 50 стаканов, при 30 °C – 80.

Шаг 2. Находим k . $k = (80 - 50) : (30 - 20) = 30 : 10 = 3$. Каждый градус жары – плюс 3 стакана.

Шаг 3. Находим b . $50 = 3 \cdot 20 + b$, $50 = 60 + b$, $b = 50 - 60 = -10$. Правило: $y = 3x - 10$. Проверка второй точкой: $3 \cdot 30 - 10 = 80$ – верно.

Шаг 4. Прогноз на 32 °C. $y = 3 \cdot 32 - 10 = 96 - 10 = 86$ стаканов.

Шаг 5. Ошибка при 26 °C. Прогноз: $3 \cdot 26 - 10 = 78 - 10 = 68$ стаканов. Ошибка: $70 - 68 = +2$. Плюс – модель занизила: продали на 2 стакана больше, чем она обещала.

Сверься с графиком. Точка для 26 °C и правда чуть выше прямой. Прогноз 86 для 32 °C больше, чем в самый жаркий день недели (28 °C, 72 стакана), – в жару квас берут охотнее. 32 °C лежит чуть правее облака точек, так что прогнозу можно верить, но с осторожностью. А вот $b = -10$ не пугайся: при 0 °C прямая обещает минус десять стаканов, но 0 °C далеко за краем данных, туда прямую никто не применяет.

❗ **Типичная ошибка.** Взять для расчёта голубые точки вместо узлов на прямой. Точки – это данные, а правило модели задаёт прямая. Например, точки (20; 52) и (26; 70) дадут $k = 18:6 = 3$, но $b = 52 - 60 = -8$ – и все прогнозы сдвинутся. Вторая ошибка – найти b как $50 - 3 = 47$: забыли умножить k на $x = 20$.

Ответ: а) $y = 3x - 10$; б) 86 стаканов; в) +2 стакана (модель занизила на 2).

Своя задача по программе сферы, модуль 5

Реши сам

4.13. Модель предсказывает, сколько литров воды за день выпивают ребята в летнем лагере, по температуре воздуха x (в градусах). Прямая модели проходит через точки (20; 30) и (30; 50). а) Запиши правило прямой в виде $y = kx + b$. б) Сколько литров модель предскажет на день с температурой 27 °С? в) В день, когда было 24 °С, выпили 41 литр. Какова ошибка прогноза в этот день?

4.14. Маша тренируется бегать круг по стадиону. Модель предсказывает время круга (в секундах) по номеру тренировки x , её прямая проходит через точки (2; 70) и (6; 58). а) Запиши правило прямой. б) На какой тренировке модель обещает круг за 40 с? в) На 8-й тренировке Маша пробежала круг за 50 с. Какова ошибка прогноза?

Задача 4.6* Две модели для такси

средняя ошибка, сравнение моделей, Python

Две команды обучили модели, которые предсказывают цену поездки на такси по расстоянию. Модель А: $y = 25x + 130$. Модель Б: $y = 30x + 100$ (здесь x – расстояние в километрах, y – цена в рублях). Модели проверили на пяти настоящих поездках:

Таблица 4.3

Расстояние, км	Настоящая цена, ₽
2	180
4	235
6	290
8	330
10	395

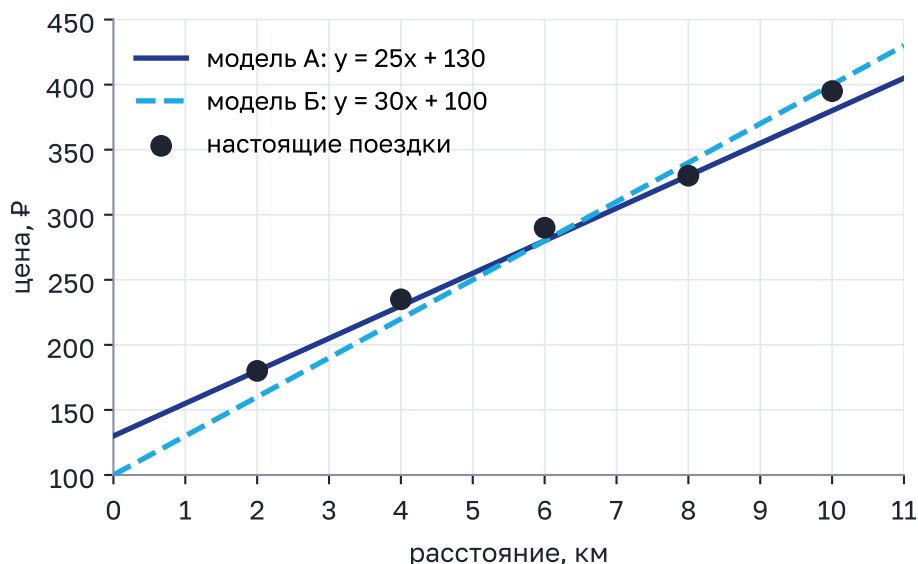


Рисунок 4.12 – Пять настоящих поездок и две модели. Сплошная прямая – модель А, пунктир – модель Б

а) Найди среднюю ошибку каждой модели (в рублях). б) Какая модель лучше? в) Какую цену она предскажет за поездку 12 км?

Дано: две модели, пять поездок.

Найти: средние ошибки А и Б; лучшую модель; её прогноз для 12 км.

Решение

Шаг 1. Прогнозы. Подставляем каждое расстояние в обе формулы. Для 2 км: А – $25 \cdot 2 + 130 = 180$, Б – $30 \cdot 2 + 100 = 160$. Остальные – так же, всё собрано в таблице.

Шаг 2. Ошибки без знака. Из настоящей цены вычитаем прогноз и отбрасываем знак.

Таблица 4.4

Км	Цена	Прогноз А	Ошибка А	Прогноз Б	Ошибка Б
2	180	180	0	160	20
4	235	230	5	220	15
6	290	280	10	280	10
8	330	330	0	340	10
10	395	380	15	400	5

Шаг 3. Средние ошибки. А: $(0 + 5 + 10 + 0 + 15) : 5 = 30 : 5 = 6$ руб. Б: $(20 + 15 + 10 + 10 + 5) : 5 = 60 : 5 = 12$ руб.

Шаг 4. Сравниваем. У модели А средняя ошибка меньше: 6 руб против 12 руб. Она лучше.

Шаг 5. Прогноз на 12 км. $25 \cdot 12 + 130 = 300 + 130 = 430$ руб.

Шаг 6. Проверка программой.

```

km = [2, 4, 6, 8, 10]
cena = [180, 235, 290, 330, 395]

def srednyaya_oshibka(k, b):
    summa = 0
    for i in range(len(km)):
        prognoz = k * km[i] + b
        summa = summa + abs(cena[i] - prognoz)
    return summa / len(km)

print(srednyaya_oshibka(25, 130))    # модель А: 6.0
print(srednyaya_oshibka(30, 100))   # модель Б: 12.0
print(25 * 12 + 130)                # прогноз А на 12 км: 430

```

Построчно: `def` создаёт свою **функцию** – маленькую программу с именем, в которую передают k и b . Внутри цикл проходит по всем поездкам, считает прогноз и прибавляет к сумме ошибку. `abs` – модуль, он отбрасывает знак: `abs(-20)` даёт 20. `return` возвращает ответ – сумму, делённую на число поездок. Python печатает 6.0 с точкой – в тетради это 6,0.

На графике то же самое: прямая А проходит точно через точки для 2 и 8 км и совсем рядом с точкой для 4 км, а прямая Б заметно промахивается на коротких поездках. На последней поездке (10 км) Б ошиблась меньше, чем А, – 5 ₽ против 15 ₽. Но модель оценивают по всем примерам, а не по одному удачному.

❗ **Типичная ошибка.** Складывать ошибки со знаком. У модели Б тогда выйдет $20 + 15 + 10 - 10 - 5 = 30$, средняя 6 ₽ – ровно как у А, и модели покажутся одинаковыми. Плюсы и минусы съели друг друга, хотя Б промахивается сильнее. Средняя ошибка считается **без знака**.

Ответ: а) А – 6 ₽, Б – 12 ₽; б) модель А; в) 430 ₽.

Своя задача по программе сферы, модуль 5

Реши сам

4.15. Модель предсказывает время поездки школьного автобуса (в минутах) по числу остановок x : $y = 2x + 10$. Настоящие поездки: 5 остановок – 21 мин, 10 остановок – 28 мин, 15 остановок – 41 мин, 20 остановок – 49 мин. Найди среднюю ошибку модели в минутах.

4.16.* Две модели предсказывают, сколько минут Коля делает домашнее задание по математике, по числу задач x . Модель А: $y = 4x + 10$. Модель Б: $y = 5x + 5$. Настоящие вечера: 2 задачи – 18 мин, 4 задачи – 27 мин, 6 задач – 34 мин, 8 задач – 43 мин. а) Найди среднюю ошибку каждой модели. б) Какая модель лучше? в) Сколько минут она предскажет на 10 задач?

Задача 4.7* Два тарифа для обучения нейросети

два линейных правила, уравнение с x в обеих частях

Чтобы обучить нейросеть, школьная команда арендует через интернет мощный компьютер. Тариф «Старт»: 300 ₽ за подключение и 12 ₽ за каждую минуту работы. Тариф «Про»: 60 ₽ за подключение и 20 ₽ за минуту.

а) При какой длительности обучения оба тарифа стоят одинаково?

б) Обучение займёт 45 минут. Какой тариф выгоднее и на сколько рублей?

Дано: «Старт» – 300 ₽ + 12 ₽/мин; «Про» – 60 ₽ + 20 ₽/мин; обучение 45 мин.

Найти: x – длительность, при которой цены равны, мин; выгодный тариф при 45 мин и экономию, ₽.

Решение

Шаг 1. Два правила. Пусть x – время обучения в минутах. «Старт» стоит $300 + 12x$ рублей, «Про» – $60 + 20x$ рублей. Это две линейные зависимости: у «Старта» $b = 300$, $k = 12$, у «Про» $b = 60$, $k = 20$.

Шаг 2. Уравнение. Цены равны: $300 + 12x = 60 + 20x$.

Шаг 3. Решаем. Снимаем с обеих частей по $12x$: $300 = 60 + 8x$. Снимаем по 60: $240 = 8x$. Делим на 8: $x = 30$ мин.

Шаг 4. Проверка. «Старт»: $300 + 12 \cdot 30 = 300 + 360 = 660$ ₽. «Про»: $60 + 20 \cdot 30 = 60 + 600 = 660$ ₽. Одинаково.

Шаг 5. 45 минут. «Старт»: $300 + 12 \cdot 45 = 300 + 540 = 840$ ₽. «Про»: $60 + 20 \cdot 45 = 60 + 900 = 960$ ₽. Выгоднее «Старт», экономия $960 - 840 = 120$ ₽.

Шаг 6. Проверка другим способом. После 30-й минуты тарифы стоят одинаково, а дальше каждая минута «Старта» дешевле на $20 - 12 = 8$ ₽. С 30-й до 45-й минуты – 15 минут, экономия $15 \cdot 8 = 120$ ₽ – те же деньги.

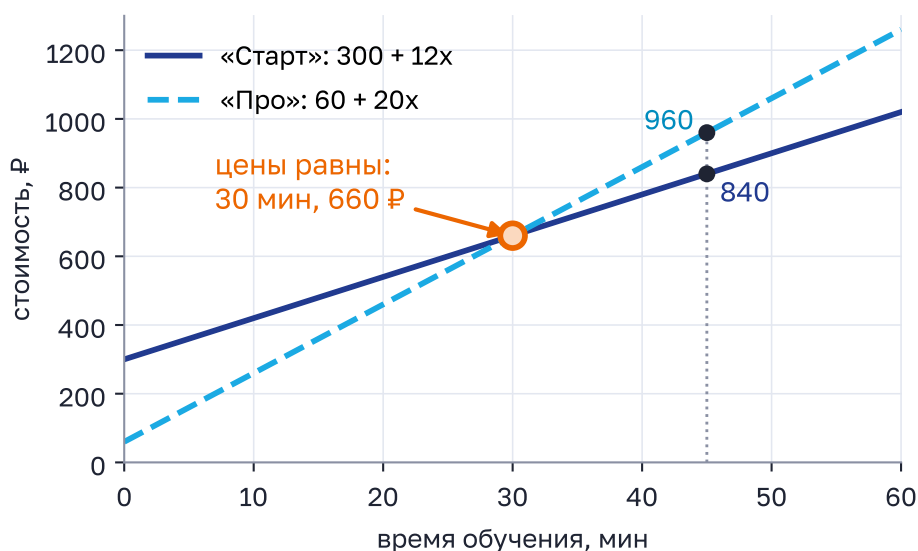


Рисунок 4.13 – Две прямые – два тарифа. До точки пересечения дешевле «Про», после – «Старт»

Кому какой тариф. «Про» дешевле на старте, но дороже за минуту. Значит, для коротких заданий выгоден он, а для долгих – «Старт». Точка, где тарифы меняются местами, – 30 минут, и 45 минут лежит после неё.

 **Типичная ошибка.** Сравнить только цену подключения («Про» дешевле – значит, он и выгоднее) или только цену минуты. Выгоду решают оба числа вместе. Ещё одна ошибка в счёте: $240 : 20 = 12$ – поделили не на разницу цен минуты ($20 - 12 = 8$), а на цену минуты одного тарифа.

Ответ: а) 30 минут; б) «Старт», дешевле на 120 ₽.

Своя задача по программе сферы, модуль 2

Реши сам

4.17. Кружок робототехники заказывает печать деталей. Мастерская А: 200 ₽ за заказ и 15 ₽ за каждую деталь. Мастерская Б: 50 ₽ за заказ и 25 ₽ за деталь. а) При скольких деталях заказ в обеих мастерских стоит одинаково? б) Нужно 22 детали. Где дешевле и на сколько рублей?

4.18.* Прокат самокатов. Тариф А: 50 ₽ за старт и 8 ₽ за минуту. Тариф Б: старт бесплатно, 10 ₽ за минуту. При какой длительности поездки (в минутах) тарифы стоят одинаково?

Данные: собираем и наводим порядок



Любая нейросеть начинается не с нейронов, а с таблицы. Кто-то должен собрать примеры, записать их в файл, выкинуть повторы и опечатки – и только потом модель чему-то научится. В главе пройдем этот путь целиком: откуда берутся данные, как устроены файлы CSV и JSON, как программа забирает записи с сайта страница за страницей и как навести порядок в таблице, прежде чем считать. На отборе прошлых лет встречались как раз такие задачи: сколько страниц скачать у сервера, какие три книги самые популярные в жанре. Ответ в них – число или несколько шифров через запятую.

5.1. Данные, датасет и таблица

НА ПАЛЬЦАХ

записи в тетради

12 января мороз 18° снег 42 см
13 января мороз 24° снег 45 см
14 января мороз 12° снег 51 см



столбец = признак

день	мороз, °	снег, см
12 января	18	42
13 января	24	45
14 января	12	51

строка = объект (один день)

Рисунок 5.1 – Записи из тетради наблюдений превращаются в таблицу

Каждая запись в тетради – одна строка таблицы. Каждое поле записи – свой столбец.

Данные – это записанные наблюдения: числа, слова, отметки «да» или «нет». Термометр за окном показывает -18°C – пока ты просто смотришь, это не данные. Записал в тетрадь «12 января, мороз 18° , снег 42 см» – появились данные. Их можно сравнить с завтрашними, отправить другу, посчитать по ним среднее.

Если записи однотипные – про каждый день одно и то же, про каждого ученика одно и то же, – их удобно сложить в таблицу. Набор однотипных записей называют **набором данных**, или **датасетом** (dataset). В таблице датасета:

- **строка** – один **объект**: один день, один ученик, одна книга, один пассажир;
- **столбец** – одно свойство объекта: дата, мороз, высота снега. В анализе данных столбцы так и называют – **признаками**;
- **клетка** – одно значение: «снег 13 января – 45 см»;
- **заголовок** – самая верхняя строка с именами столбцов. Это не запись, объекта за ней нет.

Размер таблицы говорят так: «столько-то строк на столько-то столбцов». В образовательном курсе сферы собирают набор данных о пассажирах космического лайнера: 8693 строки и 14 столбцов. Сколько в нём значений? Строки умножаем на столбцы, как клетки в прямоугольнике тетрадного листа: $8693 \cdot 14 = 121\,702$ клетки. Больше ста тысяч – руками такую таблицу не заполнишь, её собирает программа.

В главе 1 мы делили столбцы на признаки, ответ (целевую переменную) и идентификаторы – это нужно, когда по таблице учат модель. Здесь вопрос другой: как таблицу получить и как привести её в порядок.

Запомни. Строка – объект, столбец – признак (свойство), клетка – одно значение. Строка заголовка – не запись. Число клеток = число строк · число столбцов.

5.2. Типы данных: число, текст, да/нет, дата

НА ПАЛЬЦАХ

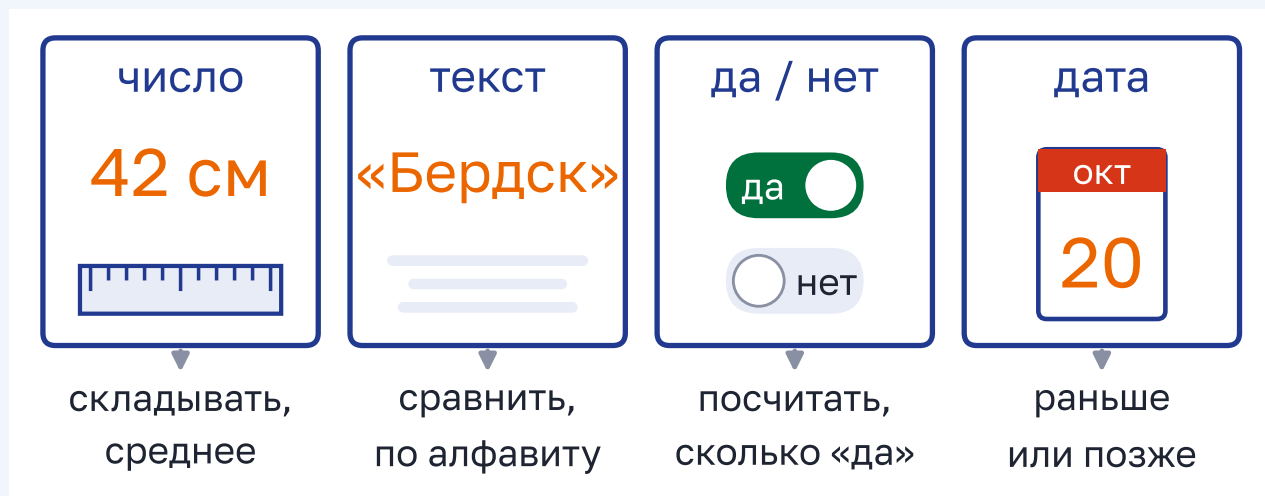


Рисунок 5.2 – Четыре типа данных и что с каждым можно делать

Тип значения решает, что с ним можно делать. Числа складывают, тексты сравнивают, «да» считают, даты ставят по порядку.

В одной таблице встречаются значения разных **типов**:

- **число** – высота снега 42 см, цена 350 руб., мороз $-12,5^{\circ}$. Бывает целым (42) и дробным ($-12,5$);
- **текст** (его ещё называют строкой) – «Бердск», «Незнайка на Луне», «фантастика»;
- **да/нет** – был ли снег, есть ли электронная версия книги. В файлах это пишут по-разному: «да/нет», 1 и 0, True и False;
- **дата** (и время) – 20.10.2026, 14:30.

Для анализа данных важнее другое деление – всего на две группы. **Количественные** данные – числа, которые что-то измеряют или считают: рост, цена, число страниц, мороз. С ними можно делать арифметику: сложить, найти среднее. **Категориальные** данные – значения-ярлыки: город, жанр, цвет, «да» или «нет». Их можно сравнить («одинаковые или разные?») и посчитать, сколько раз встречается каждый ярлык. Складывать их нельзя: «Бердск + Искитим» – бессмыслица.

Самая частая ловушка – цифры, которые не количество. Номер школы, номер телефона, почтовый индекс, шифр книги в библиотеке записаны цифрами, но это ярлыки, как номерок в

гардеробе. Средний номер школы по району посчитать можно, только он ничего не значит. Проверка простая: **есть ли смысл сложить эти значения или найти среднее?** Есть – количественные. Нет – категориальные, даже если там цифры.

У компьютера свои названия типов. В Python целое число – `int`, дробное – `float`, текст – `str`, да/нет – `bool` (значения `True` и `False`). Функция `type` показывает тип, а `int(...)` превращает текст из цифр в число:

```
print(type(42))           # <class 'int'> - целое число
print(type(-12.5))        # <class 'float'> - дробное число
print(type("Бердск"))     # <class 'str'> - текст
print(type(True))         # <class 'bool'> - да/нет
print("38" + "41")        # 3841 - два текста просто склеились
print(int("38") + int("41")) # 79 - а это уже сложение чисел
print(sorted(["9", "10", "100"])) # ['10', '100', '9'] - тексты встали по алфавиту
```

Последние три строки – главная засада. Когда программа читает таблицу из файла, числа часто приходят **текстом**. Текст «38» и число 38 для компьютера – разные вещи. Два текста при `+` склеиваются, а при сортировке встают по алфавиту: «1...» раньше «9...», как «арбуз» раньше «яблока». Поэтому перед подсчётами числа из файла переводят в числа.

С датами то же самое. Мы пишем 20.10.2026 – день, месяц, год. Если отсортировать такие даты как текст, компьютер сравнит сначала дни: «03.11» окажется раньше «15.09», хотя ноябрь позже сентября. Поэтому в данных даты часто пишут наоборот: **год-месяц-день**, 2026-10-20. Тогда сортировка по алфавиту совпадает с сортировкой по времени.

Запомни. Количественные данные можно складывать и усреднять, категориальные – только сравнивать и пересчитывать. Цифры в номере, шифре, индексе – это ярлык, а не количество. Числа из файла сначала переведи из текста в число.

Совет. Увидел в программе странный результат вроде «3841» вместо 79 или «10» раньше «9» – почти наверняка числа остались текстом. Оберни значение в `int(...)` или `float(...)`.

5.3. Форматы файлов: CSV и JSON

НА ПАЛЬЦАХ

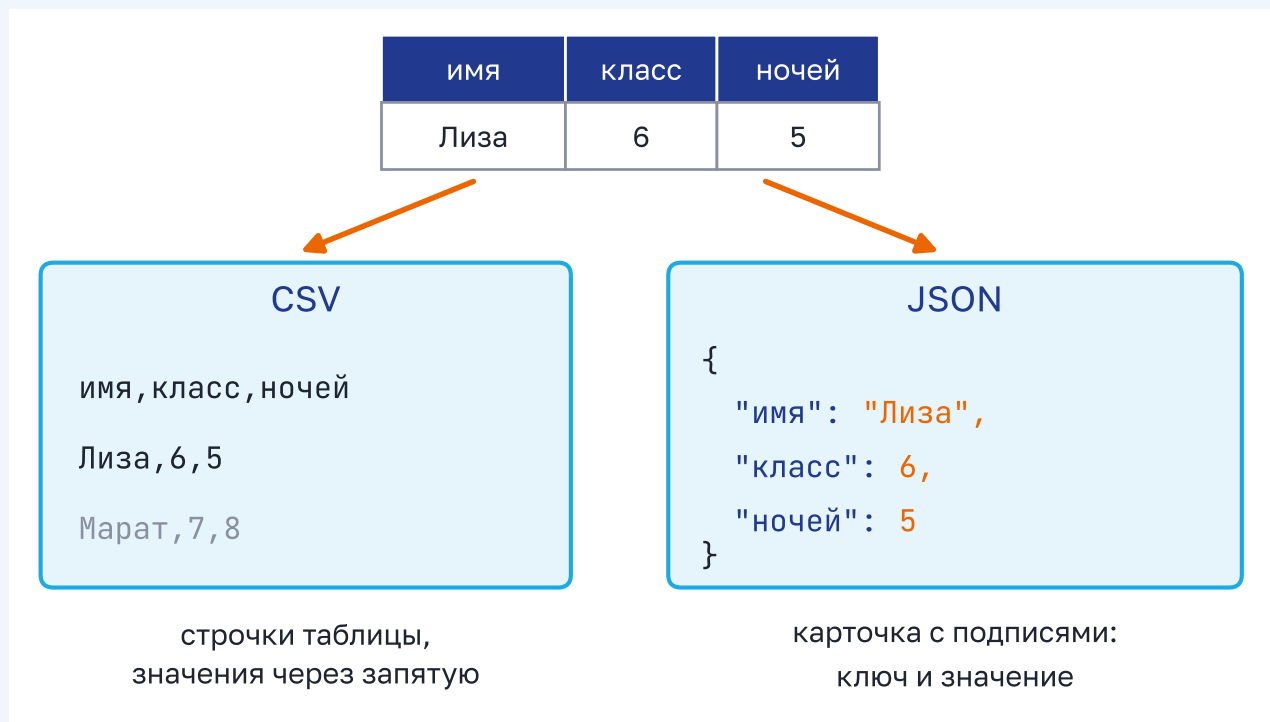


Рисунок 5.3 – Одна и та же запись в двух форматах

CSV записывает таблицу строками через запятую. **JSON** – как карточка, где у каждого значения есть подпись.

Таблицу на экране ты видишь с линиями и клетками. А в файле она хранится как обычный текст. **Формат** – это правило, по которому таблицу записывают текстом, чтобы другая программа поняла её так же. Для данных чаще всего встречаются два формата.

CSV (сокращение от английского «значения через запятую») – самый простой. Открой такой файл в Блокноте, и увидишь:

```
имя, класс, ночей  
Лиза, 6, 5  
Марат, 7, 8  
Вера, 5, 3
```

Первая строка – заголовок, имена столбцов. Каждая следующая – одна запись. Значения внутри строки разделены запятой. Отсюда удобно считать: записей в файле на одну меньше, чем строк (здесь 4 строки, значит, 3 записи), а запятых в каждой строке на одну меньше, чем столбцов (3 столбца – 2 запятые).

Разделителем бывает не только запятая. Русский Excel часто сохраняет CSV через точку с запятой: у нас запятая уже занята в дробных числах (12,5). А если запятая нужна внутри значения, его берут в кавычки: "Новосибирск, Академгородок".

Прочитать CSV в Python можно без всяких установок – модуль `csv` встроен:

```

import csv, io

text = """дата,снег_см
01.12,38
02.12,41
03.12,44"""

rows = list(csv.DictReader(io.StringIO(text)))
print(len(rows))           # 3 - записи, без строки заголовка
print(rows[0])             # {'дата': '01.12', 'снег_см': '38'}
total = 0
for r in rows:
    total = total + int(r["снег_см"])
print(total)               # 123

```

Что здесь происходит. Первая строка подключает два встроенных модуля. В переменную `text` мы положили маленький CSV прямо в программе. Тройные кавычки разрешают писать текст в несколько строк. `io.StringIO(text)` делает из строки как бы файл, а `csv.DictReader` читает его и превращает каждую запись в **словарь** – пары «имя столбца: значение». `list(...)` собирает все записи в список `rows`. Дальше цикл `for` идёт по записям и прибавляет к `total` высоту снега. Зачем `int(...)`? В словаре лежит текст `'38'`, а не число, и без перевода вместо сложения вышла бы склейка. Для настоящего файла вместо `io.StringIO(text)` пишут `open("sneg.csv", encoding="utf-8")`.

JSON устроен иначе. В нём у каждого значения есть подпись – **ключ**:

```

{
  "club": "Юный астроном",
  "members": [
    {"name": "Лиза", "class": 6, "nights": 5},
    {"name": "Марат", "class": 7, "nights": 8}
  ]
}

```

Правила JSON:

- фигурные скобки `{ }` – **объект**, та самая карточка: внутри пары `"ключ": значение` через запятую;
- квадратные скобки `[ ]` – **список**: значения по порядку через запятую;
- текст – в двойных кавычках, числа – без кавычек и с точкой вместо запятой (`-12.5`);
- да и нет пишут как `true` и `false`, пустое значение – `null`;
- внутри карточки может лежать список, а в списке – снова карточки. Так, как в примере выше: у кружка есть список участников, и про каждого – своя карточка.

Чтобы достать значение, идут по ключам и номерам, как по адресу: сначала ключ, потом номер в списке, потом снова ключ. **Номера в списке начинаются с нуля**: первая карточка – `[0]`, вторая – `[1]`.

```
import json

text = '{"club": "Юный астроном", "members": [{"name": "Лиза", "nights": 5}, {"name": "Марат", "nights": 8}]}'
```

```
data = json.loads(text)           # из текста JSON – в словари и списки Python
print(data["club"])               # Юный астроном
print(len(data["members"]))      # 2 – в списке две карточки
print(data["members"][1]["name"]) # Марат – [1] это ВТОРАЯ карточка
```

Когда использовать какой формат? CSV – для ровных таблиц, где у всех записей одинаковые столбцы: он короче. JSON – когда у данных есть вложенность (у кружка – список участников) и когда данные отдаёт сайт: почти все API, о которых следующий раздел, отвечают именно в JSON.

Запомни. CSV: первая строка – заголовок, записей на одну меньше, чем строк; запятых в строке на одну меньше, чем столбцов. JSON: {} – карточка с ключами, [] – список, счёт в списке с нуля.

5.4. Откуда берут данные и что такое API

НА ПАЛЬЦАХ

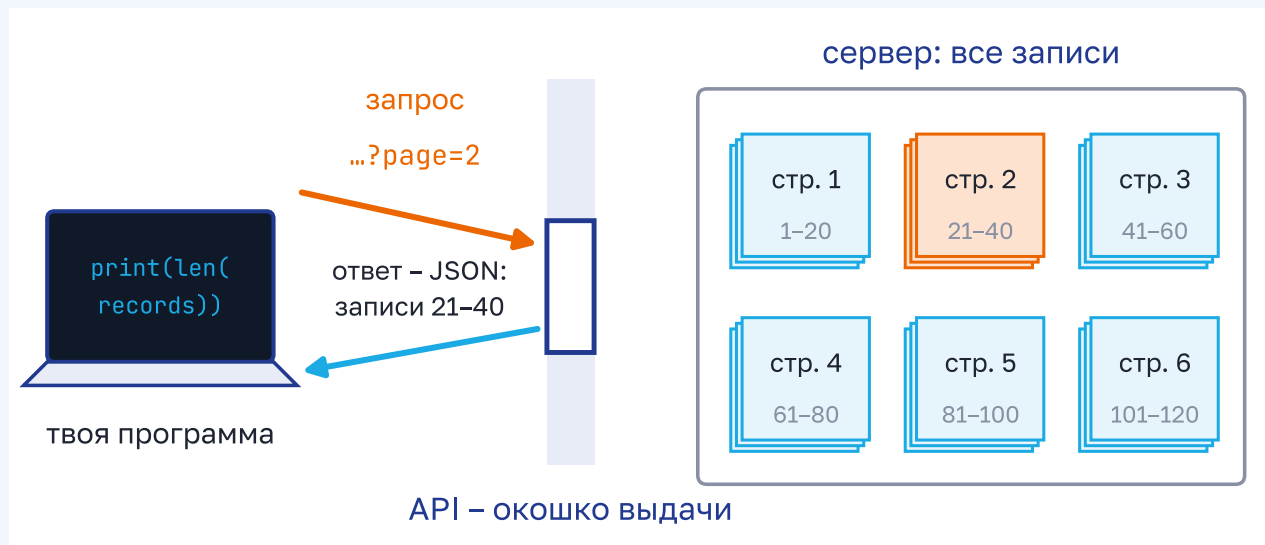


Рисунок 5.4 – API – окошко выдачи: программа просит страницу, сервер выдаёт порцию записей

Все записи хранятся у сервера. Программа не заходит на склад сама – она подаёт запрос в окошко и получает порцию.

Прежде чем что-то считать, данные надо где-то взять. Источников несколько, и у каждого своя работа:

Таблица 5.1

Источник	Что даёт	Пример
----------	----------	--------

Источник	Что даёт	Пример
Датчик	измеряет прямо сейчас и сколько угодно раз	термометр в теплице, шагомер в телефоне
Камера, спутник	снимки	снимок Обского моря из космоса, фото с телескопа
Человек	мнения и ответы, которые не измерит ни один прибор	опрос, анкета, голосование
Архив, открытые данные	то, что кто-то уже собрал за годы	погода за сто лет, численность населения
Сайт через API	свежие данные по запросу программы	каталог книг, расписание, курс валют

Открытые данные – наборы, которые научные институты, государственные службы и проекты выкладывают для всех. Бесплатно – не значит надёжно. Перед тем как их брать, выясни, кто их выложил и когда последний раз обновлял, как их собирали, подходят ли они к твоему вопросу, нет ли в них больших пропусков и разрешено ли их использовать. И в работе всегда указывай, откуда взял данные.

API (программный интерфейс) – это «окошко выдачи» сайта, только не для людей, а для программ. Человек открывает страницу и читает глазами. Программа отправляет **запрос** – обращение по особому адресу – и получает **ответ**: данные, чаще всего в формате JSON. Адрес запроса выглядит примерно так (это адрес-образец, настоящего API там нет):

```
https://example.com/api/books?genre=фантастика&page=2
```

До знака `?` – сам адрес, куда обращаемся. После него – **параметры**, уточнения к запросу в виде `имя=значение`. Параметры разделяют знаком `&`. Здесь их два: жанр – фантастика, страница – вторая.

Зачем страница? Сервер не отдаёт тысячи записей за один раз: это долго и тяжело. Он выдаёт их **постранично**, порциями одного размера. Такой приём называют пагинацией. Если на странице 20 записей, то страница 1 – записи с 1-й по 20-ю, страница 2 – с 21-й по 40-ю, страница 3 – с 41-й по 60-ю. Правило для любой страницы с номером n : она начинается с записи $(n - 1) \cdot 20 + 1$ и заканчивается записью $n \cdot 20$. Проверь на третьей: $2 \cdot 20 + 1 = 41$ и $3 \cdot 20 = 60$.

Сколько страниц нужно скачать? Словами: **всего записей делим на записи на одной странице и округляем вверх**. Вверх – потому что остаток тоже лежит на странице, пусть и неполной. На последней странице записей обычно меньше:

$$\text{на последней} = \text{всего} - (\text{страниц} - 1) \cdot \text{на странице}.$$

Пример. 90 записей по 20 на странице: $90 : 20 = 4,5$ – значит, 5 страниц. Четыре полные – это $4 \cdot 20 = 80$ записей, на пятой $90 - 80 = 10$.

Скачивать страницы по одной руками никто не станет: это делает цикл. В Python запросы отправляют библиотекой `requests` (в Colab она уже есть). Вот заготовка программы:

```

import requests

url = "https://example.com/api/meteors"      # адрес-образец: подставь адрес из задачи
records = []                                # сюда сложим записи со всех страниц
for page in range(1, 15):                    # страницы 1, 2, ..., 14
    res = requests.get(url, params={"page": page}).json()
    records.extend(res["records"])            # добавить записи этой страницы в конец
                                            # списка
print(len(records))

```

Разберём по строкам. `records = []` – пустой список, копилка. `range(1, 15)` даёт номера от 1 до 14: **последнее число в range не входит**. Поэтому для 14 страниц пишем 15.

`requests.get(...)` отправляет запрос, параметр `page` подставляется в адрес сам, а `.json()` превращает ответ в словари и списки Python. `extend` добавляет в копилку все записи страницы, а `len(records)` в конце показывает, сколько записей собрано.

Бывает, что сервер не говорит, сколько всего записей. Тогда страницы берут, пока не придёт пустая. Проверим этот способ без интернета: вместо сервера напишем функцию, которая ведёт себя так же. Эта программа запустится в любом онлайн- интерпретаторе:

```

VSEGO = 3380      # столько записей хранит «сервер»
NA_STRANICE = 250  # столько он отдаёт за один запрос

def get_page(page):
    """Притворяемся сервером: отдаём номера записей со страницы page."""
    first = (page - 1) * NA_STRANICE + 1
    last = min(page * NA_STRANICE, VSEGO)
    return list(range(first, last + 1))

records = []
page = 1
while True:      # повторяем, пока не встретим break
    chunk = get_page(page)
    if len(chunk) == 0:      # пустая страница – записи кончились
        break
    records.extend(chunk)
    page = page + 1
print("страниц с данными:", page - 1)      # 14
print("записей собрано:", len(records))     # 3380
print("на последней:", len(get_page(14)))  # 130

```

В курсе сферы такую программу собирают из блоков в среде **CodeChef** (codechef.ru, блоки «HTTP», «Pandas», «Списки», «Переменные»; Scratch – другой, не курсовой инструмент). Алгоритм при этом тот же, что в коде выше. А алгоритм везде один: пустая копилка – цикл по страницам – запрос – добавить в копилку – после цикла сохранить всё в файл.

Запомни. Страниц = всего : на странице, **с округлением вверх**. На последней = всего – (страниц – 1) · на странице. В цикле `range(1, N + 1)`: последнее число в `range` не входит.

Совет. Собрал данные – сверь итог: число записей в копилке должно совпасть с тем, сколько их на сервере. Не сошлось – ищи потерянную последнюю страницу или страницу, скачанную дважды.

5.5. Ошибки в данных и чистка

НА ПАЛЬЦАХ

имя	город	рост, см
Аня	Новосибирск	152
Боря	Бердск	
Вика	Новосибрск	148
Гоша	Искитим	1600
Даша	Бердск	150
Даша	Бердск	150

← пропуск: рост не записали

← опечатка в названии

← выброс: 16 метров?

← дубликат: та же строка ещё раз

Рисунок 5.5 – Четыре вида ошибок в одной маленькой таблице

Каждая такая ошибка тихо портит итог. Прежде чем считать, таблицу чистят.

У программистов есть поговорка: «мусор на входе – мусор на выходе». Модель, обученная на грязной таблице, выучит грязь. Среднее, посчитанное по таблице с ошибкой, сойдёт. Ошибки в данных бывают четырёх видов.

Пропуск – пустая клетка: значение не записали, датчик не ответил, человек пропустил вопрос анкеты. В файлах пропуск выглядит как пустое место, прочерк, `null`, `None` или `NaN`. Что делать: либо удалить строку, либо заполнить пропуск разумным значением – средним по столбцу, медианой или словом «неизвестно». Одно делать нельзя: молча поставить **ноль**. Снег 40 см, 50 см и пропуск: без пропуска среднее $(40 + 50) : 2 = 45$ см, а с нулём $(40 + 50 + 0) : 3 = 30$ см – будто снег растаял.

Дубликат – строка, повторённая дважды: ученик два раза нажал «Отправить», программа скачала одну страницу два раза. Лишнюю копию удаляют. Осторожно: две одинаковые покупки в магазине могут быть настоящими – человек купил два одинаковых пирожка. Если у записей есть номер (идентификатор), сверяй по нему.

Опечатки и разнобой – одно и то же записано по-разному: «Новосибирск», «новосибирск», «Новосибрск», «Новосибирск » с пробелом в конце. Человек видит один город, а компьютер – четыре разных: для него тексты равны, только если совпадает каждая буква, включая большие и маленькие и пробелы. Лечится так: убрать пробелы по краям, сделать все буквы маленькими и исправить опечатки по словарю.

```

goroda = ["Новосибирск", "новосибирск ", "Бердск", "Новосибрск", "бердск"]
ispravlit = {"новосибрск": "новосибирск"} # словарь опечаток: как написано → как пра-
вильно

chistye = []
for g in goroda:
    g = g.strip().lower() # strip – убрать пробелы по краям, lower – все буквы ма-
ленькие
    g = ispravlit.get(g, g) # есть в словаре опечаток – заменить, нет – оставить как
есть
    chistye.append(g)

print(chistye) # ['новосибирск', 'новосибирск', 'бердск', 'новоси-
бирск', 'бердск']
print(len(set(goroda)), len(set(chistye))) # 5 2 – было «пять городов», стало два

```

`set(...)` – это множество из главы 2: одинаковые значения в нём склеиваются. Поэтому `len(set(...))` – число **разных** значений.

Выброс – значение, резко не похожее на остальные: рост ученика 1600 см, мороз -95° в Новосибирске. Выброс бывает ошибкой: лишний ноль, перепутанные единицы (1,6 м записали в столбец «см»), сбой датчика. А бывает правдой: рекордный прыжок, редкий мороз. Здравый смысл помогает поставить границу: самый высокий человек, рост которого измеряли врачи, – американец Роберт Уодлоу, 2 м 72 см. Значит, 1600 см – точно ошибка. В задачах правило обычно дано: «значения больше 200 считать ошибкой». Выброс сильно тянет за собой среднее, а медиану почти не трогает (справочник, раздел «Среднее, медиана и другие»).

Порядок чистки такой: сначала убрать дубликаты и привести тексты к одному виду, потом разобраться с пропусками и выбросами, и только потом считать. Если сначала посчитать, а потом чистить, считать придётся заново.

В Colab и Jupyter для таблиц есть библиотека `pandas`: там таблица называется `DataFrame`, а чистка занимает по строчке на шаг. Та же маленькая таблица с картинками:

```

import pandas as pd

df = pd.DataFrame({"имя": ["Аня", "Боря", "Вика", "Гоша", "Даша", "Даша"],
                  "город": ["Новосибирск", "Бердск", "Новосибрск", "Искитим", "Бердск",
                           "Бердск"],
                  "рост": [152, None, 148, 1600, 150, 150]})

df = df.drop_duplicates() # убрать повторы строк: 6 → 5
df["город"] = df["город"].replace({"Новосибрск": "Новосибирск"}) # исправить опечатку
df = df.dropna() # убрать строки с пропусками: 5 → 4
df = df[df["рост"] < 250] # убрать выброс: 4 → 3
print(len(df), df["рост"].mean()) # 3 150.0

```

Запомни. Четыре беды: пропуск, дубликат, опечатка, выброс. Пропуск не превращай в ноль. Для компьютера «Бердск» и «бердск» – разные слова. Сначала чистка, потом подсчёт.

Совет. Перед подсчётом пробеги глазами по каждому столбцу: отсортируй его и посмотри на самые маленькие и самые большие значения. Выбросы и пустые клетки обычно собираются как раз по краям.

5.6. Фильтр, сортировка и подсчёт

НА ПАЛЬЦАХ

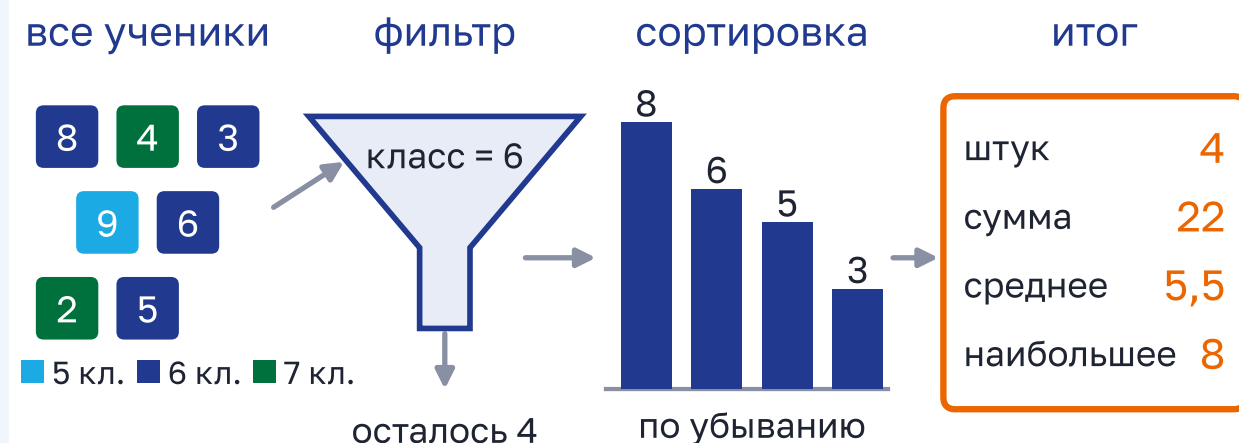


Рисунок 5.6 – Фильтр оставляет нужные строки, сортировка ставит их по порядку, а подсчёт даёт одно число

Из семи карточек фильтр «класс = 6» пропустил четыре. Их выстроили по убыванию и посчитали: сумма 22, среднее 5,5.

Почти любой вопрос к таблице собирается из трёх действий.

Фильтр оставляет только строки, для которых условие верно: «класс = 6», «снег больше 40 см», «жанр – фантастика». Условия можно соединять связками из главы 2: «фантастика и оценка не ниже 4,5». Остальные строки не удаляются навсегда – просто не участвуют в этом подсчёте.

Сортировка ставит строки по порядку одного столбца: по возрастанию (от меньшего к большему) или по убыванию. Тексты сортируются по алфавиту. «Три лучших», «топ-3» – это сортировка по убыванию и первые три строки. Если значения равны, нужно второе правило: «при равенстве – по алфавиту» или «по второму столбцу». Без него порядок одинаковых строк не определён.

Подсчёт сворачивает много строк в одно число. Основные подсчёты – те же, что в сервисах визуализации данных:

Таблица 5.2

Подсчёт	Что делает	Для ряда 8, 3, 6, 5
количество	сколько непустых	4
сумма	складывает значения	22

Подсчёт	Что делает	Для ряда 8, 3, 6, 5
среднее	сумма : количество	5,5
максимум, минимум	наибольшее и наименьшее	8 и 3
количество уникальных	сколько разных значений	4

Количество работает с любым столбцом, даже с текстовым. В таблице без пропусков оно равно числу строк, а у столбца с пропусками – меньше: пустые клетки не считаются. Сумма и среднее – только с количественными.

Часто подсчёт делают **по группам**: «сколько часов в сумме у каждого кружка», «средняя оценка в каждом жанре». Строки раскладывают на кучки по значению одного столбца и считают в каждой кучке отдельно. Это называют группировкой.

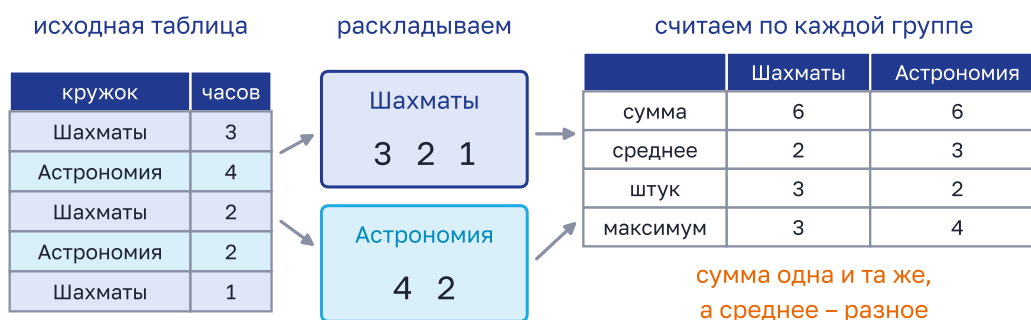


Рисунок 5.7 – Группировка: раскладываем строки по кружкам и считаем в каждой группе

Посмотри на рисунок внимательно: сумма часов у кружков одинаковая, а среднее – разное. В шахматы ходят трое, в астрономию двое. Вопрос «где больше часов всего» и вопрос «где больше часов у одного ученика» – разные вопросы, и ответы на них могут не совпасть. Читай условие до слова, которое говорит, что считать.

Вручную всё это делают так: отметить галочкой строки, прошедшие фильтр; выпиши нужные значения столбиком; посчитай; проверь, что количество выписанных чисел совпадает с числом галочек.

В Python – тем же порядком. Таблицу удобно хранить как список словарей: каждая запись – карточка, как в JSON.

```

ucheniki = [
    {"имя": "Аня", "класс": 6, "книг": 8},
    {"имя": "Боря", "класс": 7, "книг": 4},
    {"имя": "Вика", "класс": 6, "книг": 3},
    {"имя": "Гоша", "класс": 5, "книг": 9},
    {"имя": "Даша", "класс": 6, "книг": 6},
    {"имя": "Егор", "класс": 7, "книг": 2},
    {"имя": "Женя", "класс": 6, "книг": 5},
]

kolichество = 0
summa = 0
for u in ucheniki:
    if u["класс"] == 6:                # фильтр: только шестиклассники
        kolichество = kolichество + 1
        summa = summa + u["книг"]
print(kolichество, summa, summa / kolichество)    # 4 22 5.5

po_ubyvaniyu = sorted(ucheniki, key=lambda u: u["книг"], reverse=True)
for u in po_ubyvaniyu[:3]:            # [:3] – первые три
    print(u["имя"], u["книг"])        # Гоша 9, Аня 8, Даша 6

```

В `sorted` параметр `key` говорит, по какому столбцу сортировать: запись `lambda u: u["книг"]` читается как «для карточки `u` возьми число книг». `reverse=True` – по убыванию. Здесь сортировали всех учеников, без фильтра, поэтому первым стоит пятиклассник Гоша.

В `pandas` фильтр, сортировка и группировка пишутся ещё короче:

```

import pandas as pd

df = pd.DataFrame(ucheniki)
shestye = df[df["класс"] == 6]                # фильтр
print(len(shestye), shestye["книг"].sum(), shestye["книг"].mean())    # 4 22 5.5
print(df.sort_values("книг", ascending=False).head(3))    # три лидера по книгам
print(df.groupby("класс")["книг"].sum())                # по классам: 5 – 9, 6 – 22, 7 – 6
print(df["класс"].nunique())                            # 3 – разных классов

```

Запомни. Фильтр – оставить строки по условию, сортировка – расставить по столбцу, подсчёт – свернуть в число: количество, сумма, среднее, максимум, минимум, количество уникальных. «Сумма по группе» и «среднее по группе» – разные вопросы.

Совет. В Excel и похожих таблицах подсчёт по условию делают формулами: `СЧЁТЕСЛИ` – сколько строк подходит, `СУММЕСЛИ` – сумма по подходящим строкам, `СРЗНАЧЕСЛИ` – их среднее. Например, `=СЧЁТЕСЛИ(B2:B8; 6)` посчитает, сколько раз в клетках от B2 до B8 встречается 6. А кнопки «Сортировка» и «Фильтр» на панели сделают то же самое мышкой.

5.7. Разбираем задачи

Задача 5.1 Откуда взять данные

разминка · источники данных, датчик, снимок, архив, опрос

Шестиклассники из Кольцово готовят четыре проекта. Для каждого проекта выбери подходящий источник данных.

1. Каждые 10 минут записывать температуру и влажность воздуха в школьной теплице.
2. Узнать, в какие годы за последние полвека первый снег в Новосибирске выпал раньше 1 октября.
3. Выяснить, в какие кружки хотели бы ходить шестиклассники школы.
4. Посмотреть, какая часть Обского водохранилища покрыта льдом 1 декабря.

Источники: **А** – датчик; **Б** – снимки со спутника; **В** – архив метеонаблюдений; **Г** – опрос (анкета). Ответ запиши в виде «1А, 2...».

Дано: четыре проекта, четыре источника.

Найти: пары «проект – источник».

Решение

Шаг 1. Три вопроса к каждому проекту. Данные уже кто-то собрал или их надо измерить? Нужны сейчас или за прошлые годы? Их знают люди или показывают приборы?

Шаг 2. Проект 1. Температура и влажность прямо сейчас, и не один раз, а каждые 10 минут, днём и ночью. Человек так не просидит. Нужен прибор, который меряет сам, – датчик. 1А.

Шаг 3. Проект 2. Нужны прошлые 50 лет. Сегодня их не измеришь никаким прибором: можно только найти записи, которые метеостанции вели всё это время. Это архив. 2В.

Шаг 4. Проект 3. Кто в какой кружок хочет, не покажет ни один прибор – это знают только сами ребята. Надо их спросить – это опрос. 3Г.

Шаг 5. Проект 4. Водоохранилище огромное, лёд надо увидеть сразу целиком, сверху. Это снимок со спутника. 4Б.

Все буквы в деле? А, Б, В, Г – каждая досталась своему проекту, ни одна не осталась лишней и не повторилась.

 **Ловушка в проекте 2.** Рука тянется выбрать датчик: «там же про погоду». Датчик меряет только с той минуты, как его поставили. Про погоду 1980 года он ничего не знает. Прошлые – это всегда архив.

Ответ: 1А, 2В, 3Г, 4Б.

Своя задача по программе сферы, модуль 3

Реши сам

5.1. Сопоставь задачу и источник данных. Задачи: 1) сколько шагов в день делает ученик; 2) какие фильмы больше нравятся семиклассникам школы; 3) как менялось население Новосибирска за последние сто лет; 4) программа должна каждые 5 минут получать с сайта свежее расписание электричек. Источники: А – датчик в фитнес-браслете; Б – опрос; В – архив открытых данных статистики; Г – API сайта.

5.2. Пятиклассники из Бердска готовят четыре проекта. Подбери каждому источник данных: 1) как менялась высота снега в Бердске каждую зиму за последние 30 лет; 2) сколько раз за учебный день открывается дверь в школьную столовую; 3) какие настольные игры хотели бы купить в класс сами ребята; 4) какая часть огромного поля за селом уже зазеленела весной – смотреть надо на всё поле сразу, сверху. Источники: А – снимки со спутника; Б – опрос; В – датчик на двери; Г – архив метеонаблюдений.

Задача 5.2 Каталог школьной библиотеки

разминка · типы данных, количественные и категориальные

Вот две строки из каталога школьной библиотеки. Столбцов у каталога семь, поэтому таблицу повернули боком: имена столбцов идут сверху вниз, а каждая книга – это колонка.

Таблица 5.3

Столбец каталога	Первая книга	Вторая книга
Шифр	1045	1046
Название	Незнайка на Луне	Денискины рассказы
Автор	Н. Носов	В. Драгунский
Страниц	336	160
Цена, руб.	420	350
Есть электронная версия	да	нет
Дата поступления	12.09.2025	03.10.2025

а) Определи тип каждого столбца: число, текст, да/нет или дата. б) Для каких столбцов имеет смысл сложить значения или найти среднее? Укажи, сколько таких столбцов.

Дано: семь столбцов каталога.

Найти: тип каждого столбца; количественные столбцы.

Решение

Шаг 1. Типы по виду значения. Цифрами записаны «Шифр», «Страниц», «Цена». Словами – «Название» и «Автор», это текст. «Есть электронная версия» принимает только два значения – да/нет. «Дата поступления» – дата.

Шаг 2. Проверка «есть ли смысл сложить». Страницы: средняя толщина книги в каталоге – разумный вопрос. Цена: сколько стоят все книги вместе – тоже. Это количественные столбцы.

Шаг 3. Шифр. Сложим шифры: $1045 + 1046 = 2091$. Что это? Ничего. Средний шифр 1045,5 – книги с таким шифром нет. Шифр – ярлык, как номерок в гардеробе, просто записан цифрами. Категориальный.

Шаг 4. Остальные. Складывать названия, авторов, «да» с «нет» или даты бессмысленно. Даты можно сравнивать – какая книга пришла раньше, – но не складывать.

Примета на будущее. У обоих количественных столбцов есть единица измерения – страницы и рубли. У шифра её нет.

 **Типичная ошибка.** Записать «Шифр» в количественные, потому что там цифры. Цифры – не повод. Решает вопрос «имеет ли смысл средний шифр?», а он не имеет.

Ответ: а) число – «Шифр», «Страниц», «Цена»; текст – «Название», «Автор»; да/нет – «Есть электронная версия»; дата – «Дата поступления». б) Два столбца: «Страниц» и «Цена».

Своя задача по программе сферы, модуль 3

Реши сам

5.3. В наборе данных о погоде есть столбцы: «Дата», «Температура, °C», «Был снег» (да/нет), «Направление ветра» (С, Ю, З, В), «Скорость ветра, м/с», «Номер станции». Сколько из них количественных? Назови их.

5.4. В столбце «Дата» записаны даты наблюдений: 15.09.2026, 03.11.2026, 20.10.2026, 28.09.2026. Программа отсортировала их командой `sorted`, как обычный текст. Какая дата окажется первой в списке? Какая из этих дат на самом деле самая ранняя?

Задача 5.3 Ответ метеостанции

формат JSON, ключи и номера в списке, среднее

У школьной метеостанции в Академгородке есть API. Программа отправила запрос и получила ответ, который сохранила в переменную `res`:

```
{
  "station": "Академгородок",
  "page": 1,
  "records": [
    {"date": "2026-01-10", "temp": -18.5, "snow": 42, "wind": 3},
    {"date": "2026-01-11", "temp": -24.0, "snow": 45, "wind": 1},
    {"date": "2026-01-12", "temp": -12.5, "snow": 51, "wind": 6},
    {"date": "2026-01-13", "temp": -9.0, "snow": 50, "wind": 4}
  ]
}
```

Здесь `temp` – температура в °C, `snow` – высота снега в сантиметрах, `wind` – скорость ветра в м/с. а) Сколько записей о погоде в ответе? б) Что выведет команда `print(res["records"][2])`

`["snow"]`) ? в) Найди среднюю температуру за эти дни. Ответ дай в °C с точностью до десятых.

Дано: ответ API в формате JSON.

Найти: число записей; значение по адресу `["records"][2]["snow"]`; среднюю температуру.

Решение

Шаг 1. Где лежат записи. Сверху – карточка с тремя ключами: `station`, `page`, `records`. По ключу `records` лежит список в квадратных скобках. В нём четыре карточки в фигурных скобках – по одной на день. Записей 4.

Шаг 2. Идём по адресу. `res["records"]` – это список. `[2]` – карточка номер 2. Номера начинаются с нуля: `[0]` – 10 января, `[1]` – 11 января, `[2]` – 12 января. У этой карточки берём ключ `snow` – там 51.

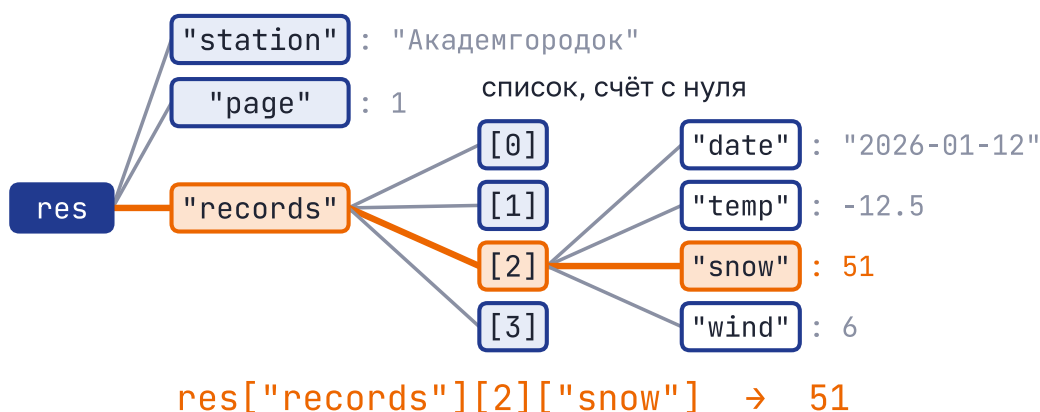


Рисунок 5.8 – Путь к значению: ключ – номер в списке – снова ключ

Шаг 3. Сумма температур. Все четыре числа отрицательные. Сложить отрицательные числа – всё равно что сложить морозы и поставить минус: $18,5 + 24 + 12,5 + 9 = 64$. Сумма равна -64 °C.

Шаг 4. Среднее. Делим на количество дней: $-64:4 = -16,0$ °C.

Проверка кодом.

```
import json

text = """{"station": "Академгородок", "page": 1, "records": [
  {"date": "2026-01-10", "temp": -18.5, "snow": 42, "wind": 3},
  {"date": "2026-01-11", "temp": -24.0, "snow": 45, "wind": 1},
  {"date": "2026-01-12", "temp": -12.5, "snow": 51, "wind": 6},
  {"date": "2026-01-13", "temp": -9.0, "snow": 50, "wind": 4}]}"""
res = json.loads(text)
print(len(res["records"]))          # 4
print(res["records"][2]["snow"])    # 51
temps = [r["temp"] for r in res["records"]] # список всех температур
print(sum(temps) / len(temps))      # -16.0
```

Строка с `temps` собирает из каждой карточки `r` значение по ключу `temp` – получается список из четырёх чисел.

А на глаз? Среднее не может выскочить за края: -16°C лежит между самым сильным морозом (-24) и самым слабым (-9). Для января под Новосибирском – обычная погода.

❗ Где теряют баллы. На вопрос б) отвечают «45», посчитав [2] второй записью. В Python счёт в списке идёт с нуля, и [2] – это третья карточка. А в пункте в) теряют минус и пишут «16,0»: средний мороз 16 градусов – это -16°C .

Ответ: а) 4; б) 51; в) $-16,0^{\circ}\text{C}$.

Своя задача по программе сферы, модуль 3

Реши сам

5.5. Что выведет программа?

```
data = {"club": "Юный астроном",
        "members": [{"name": "Лиза", "class": 6, "nights": 5},
                     {"name": "Марат", "class": 7, "nights": 8},
                     {"name": "Вера", "class": 5, "nights": 3}]}
print(data["members"][1]["name"])
print(len(data["members"]))
```

5.6. Файл `pogoda.csv` открыли в Блокноте. В нём 31 строка, первая выглядит так:
дата, температура, осадки, ветер . Сколько в файле записей о погоде? Сколько запятых в каждой строке?

Задача 5.4 Сколько страниц скачать

API, постраничная выдача, округление вверх, range

Каталог наблюдений метеоров обсерватории отдаёт данные через API постранично: по 250 записей на странице, страницы нумеруются с 1 (параметр `page`). Всего в каталоге 3380 записей.

а) Сколько страниц нужно скачать, чтобы собрать все записи? б) Сколько записей окажется на последней странице? в) Программа перебирает страницы циклом `for page in range(1, ...)` . Какое число нужно поставить вместо многоточия?

Дано: всего 3380 записей; на странице 250; страницы с номера 1.

Найти: число страниц; записей на последней; второе число в `range` .

Решение

Шаг 1. Делим. $3380 : 250 = 13,52$. Значит, 13 страниц заполнены целиком, и ещё что-то осталось.

Шаг 2. Округляем вверх. 13 полных страниц – это $13 \cdot 250 = 3250$ записей. До 3380 не хватает, остаток лежит на 14-й странице. Страниц **14**.

Шаг 3. Последняя страница. $3380 - 13 \cdot 250 = 3380 - 3250 = 130$ записей. Проверка: $3250 + 130 = 3380$ – все записи на месте.



Рисунок 5.9 – Постраничная выдача: 13 полных страниц по 250 и неполная 14-я

Шаг 4. Граница цикла. `range(1, N)` выдаёт числа от 1 до $N - 1$ – последнее число не входит. Нам нужны страницы с 1-й по 14-ю, значит, $N = 14 + 1 = 15$: `range(1, 15)`.

Проверка кодом. Сервер заменим функцией, которая отдаёт по 250 номеров записей:

```
import math

VSEGO, NA_STRANICE = 3380, 250
stranic = math.ceil(VSEGO / NA_STRANICE)           # ceil – округление вверх
print(stranic)                                     # 14
print(VSEGO - (stranic - 1) * NA_STRANICE)         # 130

def get_page(page):
    first = (page - 1) * NA_STRANICE + 1
    last = min(page * NA_STRANICE, VSEGO)
    return list(range(first, last + 1))

records = []
for page in range(1, 15):
    records.extend(get_page(page))
print(len(records))                                # 3380 – собрали всё
```

Прикинем. 14 страниц по 250 вместили бы 3500 записей – чуть больше, чем есть. А 13 страниц – 3250, меньше, чем есть. Значит, 14 – в самый раз.

⚠ **Типичная ошибка.** Округлять «по правилам», до ближайшего. Здесь это случайно сработало: $13,52 \rightarrow 14$. Но при 3310 записях $3310 : 250 = 13,24$, по правилам выйдет 13 страниц, и 60 записей с 14-й страницы потеряются. Страницы округляют **только вверх**. И с циклом легко промахнуться на единицу: написать `range(1, 14)` – и последняя страница останется на сервере.

Ответ: а) 14 страниц; б) 130 записей; в) 15.

По мотивам задачи отборочного этапа НТО Junior 2023/24 «Сбор данных»

Реши сам

5.7. API отдаёт по 400 записей на странице, всего записей 5270. Сколько страниц нужно скачать и сколько записей будет на последней?

5.8. Программа отправляет запрос `https://example.com/api/meteors?year=2025&month=8&page=3`.
а) Сколько параметров в этом запросе? б) На каждой странице 50 записей, страницы нумеруются с 1. С какой по какую запись вернёт этот запрос?

5.9.* В каталоге 2350 записей, API отдаёт их по 200 на странице, страницы нумеруются с 1. Программист написал цикл `for page in range(1, 12):` и скачал все страницы, которые этот цикл перебирает. Сколько записей он соберёт и сколько потеряет?

Задача 5.5 Чистим журнал снега

пропуски, дубликаты, выбросы, среднее

Школьная метеостанция записывает высоту снега каждое утро. За первые дни декабря в журнале вот что:

Таблица 5.4

Дата	Снег, см
01.12	38
02.12	41
03.12	
04.12	44
04.12	44
05.12	470
06.12	47
07.12	46
08.12	52
09.12	50

Почисти журнал по правилам: 1) повторяющиеся строки оставить по одной; 2) строки с пропуском удалить; 3) значения больше 200 см – ошибка прибора, такие строки удалить. Найди среднюю высоту снега по оставшимся строкам. Ответ дай в сантиметрах, округли до десятых.

Дано: 10 записей; три правила чистки.

Найти: среднюю высоту снега после чистки, см.

Решение

Шаг 1. Дубликаты. Запись «04.12 – 44» встречается два раза. Оставляем одну. Было 10 записей, стало 9.

Шаг 2. Пропуски. 3 декабря клетка пустая – удаляем строку. Осталось 8.

Шаг 3. Выбросы. 470 см больше 200 – удаляем. Скорее всего, наблюдатель приписал лишний ноль, а было 47. Но правило велит удалить, и мы не гадаем. Осталось 7 записей: 38, 41, 44, 47, 46, 52, 50.

Шаг 4. Сумма. $38 + 41 + 44 + 47 + 46 + 52 + 50 = 318$ см.

Шаг 5. Среднее. $318 : 7 = 45,428...$ Округляем до десятых: 45,4 см.

Проверка кодом.

```
zhurnal = [("01.12", 38), ("02.12", 41), ("03.12", None), ("04.12", 44), ("04.12", 44),  
           ("05.12", 470), ("06.12", 47), ("07.12", 46), ("08.12", 52), ("09.12", 50)]  
  
bez_povtorov = []  
for zapis in zhurnal:  
    if zapis not in bez_povtorov:           # такой строки ещё не было – берём  
        bez_povtorov.append(zapis)  
  
chistye = []  
for data, sneg in bez_povtorov:  
    if sneg is not None and sneg ≤ 200:    # нет пропуска и нет выброса  
        chistye.append(sneg)  
  
print(len(chistye), sum(chistye), round(sum(chistye) / len(chistye), 1))  # 7 318 45.4
```

Пустую клетку в Python записывают словом `None` – «ничего». Условие `sneg is not None` пропускает только строки, где значение есть.

Прикидка. Чистые значения лежат между 38 и 52 см, и 45,4 – примерно посередине. Снег за неделю подрос на полтора десятка сантиметров – обычное начало сибирской зимы.

❗ **Коварство этой задачи** в том, что каждое пропущенное правило даёт свой неверный ответ, и все они выглядят правдоподобно. Забыл убрать повтор – 45,3 см. Поставил вместо пропуска ноль – 39,8 см. Оставил выброс – 98,5 см. Вот почему чистку делают по списку, отмечая каждое правило.

Ответ: 45,4 см.

Своя задача по программе сферы, модули 3 и 4

Реши сам

5.10. Ученик засекал, сколько минут уходит у него на одну задачу: 12, 15, 11, 14, 13 и 95 (над последней он отвлёкся на обед и не выключил секундомер). На сколько минут уменьшится среднее время, если убрать выброс 95? Округли до десятых.

5.11.* В таблице ответов на анкету 40 строк. Четыре из них – точные копии других строк: кто-то нажал «Отправить» дважды. Ещё в 5 строках не заполнено поле «Класс». Подвох: один ученик с пустым «Классом» отправил анкету два раза, поэтому среди этих пяти строк есть и его ответ, и копия ответа. Сколько строк останется, если сначала удалить копии, а потом строки с пропусками?

5.12. В таблице 2450 строк с записями (строку заголовка не считаем) и 12 столбцов. Сколько в ней клеток со значениями? Сколько клеток останется, если удалить 3 столбца-идентификатора и 50 строк-дубликатов?

Задача 5.6 Фантастика месяца

фильтр, сортировка, топ-3, пересечение

Библиотекарь хочет выставить на полку «Фантастика месяца» три фантастические книги. Вот данные из электронного каталога («Прочитали» – сколько читателей брали книгу за год, «Оценка» – средняя оценка читателей):

Таблица 5.5

Шифр	Название	Жанр	Оценка	Прочитали
101	Незнайка на Луне	сказка	4,7	58
102	Приключения Электроника	фантастика	4,6	64
103	Сто лет тому вперёд	фантастика	4,8	71
104	Тимур и его команда	повесть	4,3	40
105	Голова профессора Доуэля	фантастика	4,5	33
106	Денискины рассказы	рассказы	4,9	90
107	Малыш и Карлсон	сказка	4,8	77
108	Понедельник начинается в субботу	фантастика	4,7	52
109	Остров сокровищ	приключения	4,6	66
110	Человек-амфибия	фантастика	4,4	68

а) Выбери три фантастические книги, которые читали чаще всего. Запиши их шифры по убыванию числа прочтений. б) Какие три фантастические книги получили самую высокую оценку? Шифры – по убыванию оценки. в) Какие книги попали в обе тройки?

Дано: таблица из 10 книг.

Найти: две тройки шифров и их общие книги.

Решение

Шаг 1. Фильтр. Оставляем только строки с жанром «фантастика». Это 102, 103, 105, 108, 110 – пять книг. Остальные пять в этой задаче не участвуют.

Шаг 2. Сортировка по прочтениям. Выписываем у пяти книг столбец «Прочитали» и ставим по убыванию: 71 (103), 68 (110), 64 (102), 52 (108), 33 (105). Первые три: 103, 110, 102.

Шаг 3. Сортировка по оценке. Оценки: 4,8 (103), 4,7 (108), 4,6 (102), 4,5 (105), 4,4 (110). Первые три: 103, 108, 102.

Шаг 4. Общие. В обеих тройках есть 103 и 102. Книга 110 популярна, но оценка у неё самая низкая среди фантастики. У книги 108 наоборот.

Проверка кодом (нужна библиотека `pandas` – в Colab она уже есть):

```
import pandas as pd

knigi = pd.DataFrame({
    "шифр": [101, 102, 103, 104, 105, 106, 107, 108, 109, 110],
    "жанр": ["сказка", "фантастика", "фантастика", "повесть", "фантастика",
            "рассказы", "сказка", "фантастика", "приключения", "фантастика"],
    "оценка": [4.7, 4.6, 4.8, 4.3, 4.5, 4.9, 4.8, 4.7, 4.6, 4.4],
    "прочитали": [58, 64, 71, 40, 33, 90, 77, 52, 66, 68]})

fant = knigi[knigi["жанр"] == "фантастика"] # фильтр
top_chit = fant.sort_values("прочитали", ascending=False).head(3)["шифр"].tolist()
top_ocen = fant.sort_values("оценка", ascending=False).head(3)["шифр"].tolist()
print(top_chit) # [103, 110, 102]
print(top_ocen) # [103, 108, 102]
print(set(top_chit) & set(top_ocen)) # {102, 103} – пересечение, как в главе 2
```

Сверимся с условием. Все шесть шифров в ответе – из списка фантастики, и в каждой тройке значения идут по убыванию.

❗ **Самая обидная ошибка** – забыть про фильтр и отсортировать всю таблицу: получится 106, 107, 103 – самые читаемые книги вообще, но две из них не фантастика. Если на отборе ответ не принят, первым делом проверь, применил ли ты все условия из задания: жанр, возраст, год.

Ответ: а) 103, 110, 102; б) 103, 108, 102; в) 103 и 102.

По мотивам задачи отборочного этапа НТО Junior 2023/24 «Лучшие книги»

Реши сам

5.13. Итоги тренировки по решению задач:

Таблица 5.6

Имя	Класс	Решено задач
Катя	6	7
Лёва	5	3
Маша	6	4
Нина	7	9
Олег	6	5
Паша	6	8
Рита	5	6
Саша	7	2

Сколько задач в среднем решил один шестиклассник? Сколько шестиклассников решили больше 5 задач?

5.14. Шесть команд запускали водяные ракеты. Высота полёта: А – 34 м, Б – 41 м, В – 29 м, Г – 41 м, Д – 38 м, Е – 27 м. Команды расставили по убыванию высоты, а при равной высоте – по алфавиту. Какая команда оказалась на третьем месте?

Задача 5.7* Кружки: сначала порядок, потом подсчёт

приведение текста к одному виду, группировка, сумма и среднее по группе

Школа собрала через онлайн-форму записи в кружки. Название кружка каждый вписывал сам:

Таблица 5.7

Кружок	Часов в неделю
Робототехника	4
робототехника	2
Шахматы	3
Роботехника	4
шахматы	2
Астрономия	3
Шахматы	2
астрономия	2
Робототехника	3

Кружок	Часов в неделю
Астрономия	4
ШАХМАТЫ	1
Астрономия	3

а) Программа построила по этой таблице, ничего не исправляя, столбчатую диаграмму «сумма часов по кружкам». Сколько столбиков на ней будет и какой кружок выйдет в лидеры? б) Приведи названия к одному виду («Робототехника» – опечатка в слове «Робототехника»). Какой кружок набирает больше всего часов в сумме и сколько? в) Сколько в среднем часов в неделю у одного записавшегося в этот кружок?

Дано: 12 записей, названия кружков написаны вразнобой.

Найти: число столбиков и лидера до чистки; лидера и его сумму после чистки; среднее у лидера.

Решение

Шаг 1. Как видит таблицу программа. Для компьютера два текста одинаковые, только если совпадает каждая буква, в том числе большая или маленькая. Выписываем разные написания: «Робототехника», «робототехника», «Роботехника», «Шахматы», «шахматы», «ШАХМАТЫ», «Астрономия», «астрономия». Их 8 – на диаграмме будет 8 столбиков.

Шаг 2. Лидер до чистки. Считаем сумму часов для каждого написания отдельно:

Таблица 5.8

Написание	Сколько записей	Сумма часов
Робототехника	2	$4 + 3 = 7$
робототехника	1	2
Роботехника	1	4
Шахматы	2	$3 + 2 = 5$
шахматы	1	2
ШАХМАТЫ	1	1
Астрономия	3	$3 + 4 + 3 = 10$
астрономия	1	2

Самый высокий столбик – «Астрономия», 10 часов.

Шаг 3. Наводим порядок. Все буквы делаем маленькими, опечатку «роботехника» заменяем на «робототехника». Остаются три кружка.

Шаг 4. Считаем по группам.

- робототехника – 4 записи: $4 + 2 + 4 + 3 = 13$ ч;
- шахматы – 4 записи: $3 + 2 + 2 + 1 = 8$ ч;
- астрономия – 4 записи: $3 + 2 + 4 + 3 = 12$ ч.

Лидер – робототехника, 13 часов. До чистки он даже не был виден: его часы рассыпались по трём разным столбикам.

Шаг 5. Среднее у лидера. В робототехнику записались 4 человека: $13 : 4 = 3,25$ ч в неделю.

Проверка. Всего часов в таблице: $4 + 2 + 3 + 4 + 2 + 3 + 2 + 2 + 3 + 4 + 1 + 3 = 33$. По группам: $13 + 8 + 12 = 33$ – ни одна строка не потерялась. И строк по группам $4 + 4 + 4 = 12$, сколько и было.

Проверка кодом.

```
zapisi = [("Робототехника", 4), ("робототехника", 2), ("Шахматы", 3), ("Роботехника", 4),
          ("шахматы", 2), ("Астрономия", 3), ("Шахматы", 2), ("астрономия", 2),
          ("Робототехника", 3), ("Астрономия", 4), ("ШАХМАТЫ", 1), ("Астрономия", 3)]

syroe = {} # сумма часов по написанию «как есть»
for kruzhek, chasy in zapisi:
    syroe[kruzhek] = syroe.get(kruzhek, 0) + chasy
print(len(syroe), max(syroe, key=syroe.get)) # 8 Астрономия

ispravit = {"робототехника": "робототехника"}
summa, kolvo = {}, {}
for kruzhek, chasy in zapisi:
    k = kruzhek.strip().lower()
    k = ispravit.get(k, k)
    summa[k] = summa.get(k, 0) + chasy
    kolvo[k] = kolvo.get(k, 0) + 1
print(summa) # {'робототехника': 13, 'шахматы': 8, 'астро-
номия': 12}
lider = max(summa, key=summa.get)
print(lider, summa[lider], summa[lider] / kolvo[lider]) # робототехника 13 3.25
```

Словарь `summa` работает как ряд копилки, подписанных названиями кружков. `summa.get(k, 0)` – «сколько уже лежит в копилке `k`, а если её ещё нет – ноль». `max(summa, key=summa.get)` находит ключ с самым большим значением.

Откуда такое среднее? Людей в каждом кружке поровну, по четыре, поэтому лидер по сумме часов – тот, где ходят дольше. У робототехники записи от 2 до 4 часов, и 3,25 ч попадает между ними.

⚠ Типичная ошибка. Поверить диаграмме, построенной по грязной таблице, и ответить «Астрономия». Прежде чем читать любую диаграмму или сумму, посмотри на список разных значений в столбце: если одно слово встречается в трёх написаниях, сначала чистка. Ещё одна ловушка – посчитать количество записей вместо суммы часов: у всех трёх кружков по 4 записи, и лидера так не найти.

Ответ: а) 8 столбиков, лидер «Астрономия» (10 ч); б) робототехника, 13 ч; в) 3,25 ч.

По мотивам задачи отборочного этапа НТО Junior 2023/24 «Описательная статистика и анализ возрастных ограничений»

Реши сам

5.15.* Продажи на школьной ярмарке записывали от руки:

Таблица 5.9

Товар	Продано, шт.
Чай	12
Пирожок	15
чай	8
Блины	9
блины	6
ЧАЙ	5
Пирожок	7
Блины	10
пирожок	4
Чай	3

а) Какой товар выйдет в лидеры по сумме проданных штук, если считать по таблице как есть, не исправляя написание? Сколько штук у него? б) Какой товар на самом деле продали больше всего и сколько?

5.16.* Классы сдавали макулатуру, и каждый вписывал свой класс в форму сам:

Таблица 5.10

Класс	Сдано, кг
6A	12
6a	8
7Б	15
6 A	10
7б	9
5B	14
7Б	11
5в	16
6A	7
5B	13
6a	9

а) Сколько разных классов насчитает программа, если ничего не исправлять? б) Приведи записи к одному виду. Какой класс сдал больше всего макулатуры в сумме и сколько? в) У какого класса больше всего килограммов в среднем на одну сдачу? Округли до десятых.

Графики и анализ данных



Таблица на сто строк молчит, пока её не превратишь в картинку. Один взгляд на график – и видно, где максимум, куда всё растёт и есть ли вообще то, что ты ищешь. В этой главе мы учимся выбирать график под вопрос, читать его без ошибок, считать среднее и медиану, проверять догадки и замечать графики, которые обманывают. На отборе такие задачи выглядят так: дан график или таблица, нужно ввести число, выбрать верные выводы из списка или сопоставить ситуацию и вид диаграммы.

6.1. Четыре главных графика и когда какой

НА ПАЛЬЦАХ

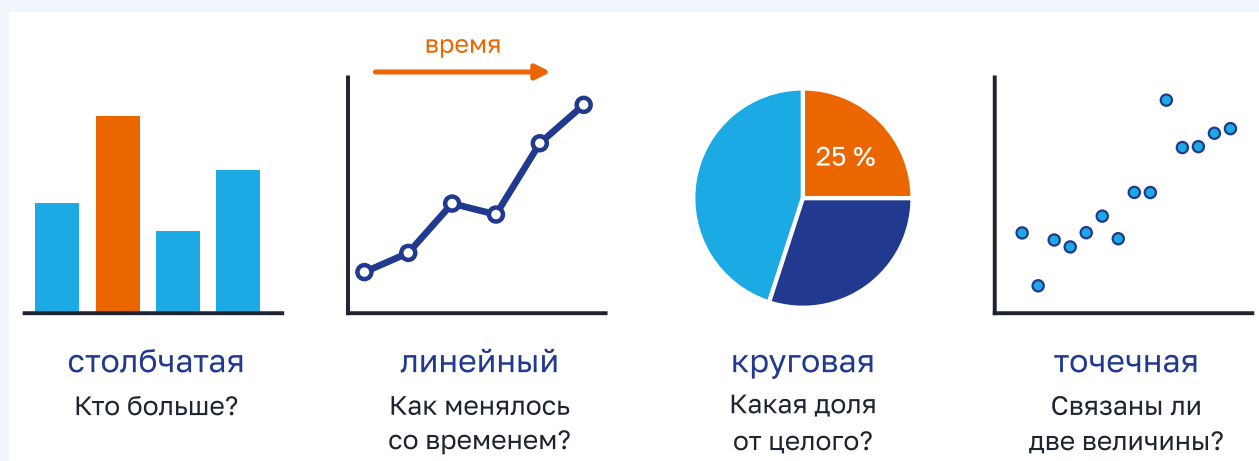


Рисунок 6.1 – Четыре вида графиков – и вопрос, на который отвечает каждый

Сначала реши, какой у тебя вопрос. Вид графика выбирают под вопрос, а не под настроение.

Визуализация – это превращение чисел в картинку: столбики, линии, точки, дольки. Слова «график» и «диаграмма» в жизни значат почти одно и то же, путать их не страшно. Видов графиков десятки, но почти любую школьную задачу закрывают четыре.

Столбчатая диаграмма сравнивает величины между собой. Сколько макулатуры собрал каждый класс, сколько книг прочитал каждый друг, в каком кружке больше всего ребят. Каждому предмету сравнения – свой столбик, и чем выше столбик, тем больше число. Порядок столбиков не важен, их можно даже отсортировать по высоте.

Линейный график показывает, как величина **меняется со временем**. По горизонтали – дни, месяцы, годы, по вертикали – то, что мерили: температуру, число подписчиков, рост тыквы на даче. Соседние точки соединяют линией, потому что между ними время шло без перерыва. Линия вверх – величина растёт, вниз – падает, чем круче линия – тем быстрее.

Круговая диаграмма показывает, **какую долю от целого** занимает каждая часть. Весь круг – это все 100 %: все ученики школы, весь бюджет, все дни месяца. Долька в четверть круга – 25 %. Круговая годится, только когда части вместе и правда составляют одно целое и не пересекаются. Если в опросе можно было отметить несколько ответов, проценты в сумме выйдут больше 100, и круга из них не сложить – тогда рисуют столбики.

Точечная диаграмма (её ещё называют диаграммой рассеяния) показывает, **связаны ли две величины**. Каждая точка – один объект: ученик, день, книга. По горизонтали откладывают одну его величину, по вертикали – другую. Если точки вытягиваются в полосу, идущую вверх, – чем больше одно, тем больше и другое. Если полоса идёт вниз – чем больше одно, тем меньше другое. Если точки рассыпаны кашей, связи не видно.

Таблица 6.1

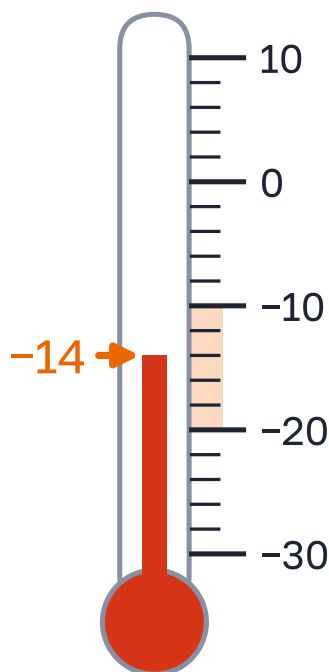
Вопрос	Какой график	Пример
Кто больше, кто меньше?	столбчатая	макулатура по классам
Как менялось со временем?	линейный	температура по месяцам
Какая доля от целого?	круговая	как ученики добираются до школы
Связаны ли две величины?	точечная	рост и размер обуви у 30 ребят

Запомни. Сравнить – столбики. Время – линия. Доли одного целого – круг. Две величины у каждого объекта – точки.

Совет. Проверка для круговой диаграммы: сложи все проценты. Вышло ровно 100 % – круг можно рисовать. Больше 100 % – ответы пересекались, и нужны столбики.

6.2. Как читать график: оси, единицы, масштаб

НА ПАЛЬЦАХ



1. Между подписями -20 и -10 разница 10 градусов
2. Между ними 5 делений
3. Одно деление – $10 : 5 = 2$ градуса
4. Столбик на 3 деления выше -20 :
 $-20 + 3 \cdot 2 = -14$ градусов

Рисунок 6.2 – Градусник за окном и цена деления

Подписана не каждая чёточка, а только каждая пятая. Сколько градусов в одном промежутке между чёточками, считаем сами: разницу подписей делим на число промежутков.

Прежде чем смотреть на линии и столбики, прочитай подписи. Это занимает десять секунд и спасает от половины ошибок. Порядок всегда один.

1. **Название.** Про что график? Где и когда собраны данные?
2. **Горизонтальная ось** (её называют осью X). Что на ней: месяцы, дни, классы, температура?
3. **Вертикальная ось** (ось Y). Что на ней и **в каких единицах**: градусы, минуты, тысячи просмотров, проценты. «Просмотры, тысяч» и число 30 означают 30 000 просмотров, а не 30.
4. **Масштаб.** С какого числа начинается ось и сколько стоит одно деление.

Промежуток между соседними чёточками называют **делением**, а сколько он стоит – **ценой деления**. Находят её так же, как на градуснике с рисунка: возьми две соседние подписи, вычти меньшую из большей и раздели на число делений между ними. На градуснике подписи -20 и -10 , между ними 5 делений: $10 : 5 = 2$ градуса. Если на оси подписаны 0, 20, 40, а между подписями по 4 деления, то цена деления $20 : 4 = 5$.

Теперь настоящий график. Ниже – средняя температура в Новосибирске по месяцам. Данные условные, округлены до целых, но похожи на то, что показывают метеостанции.

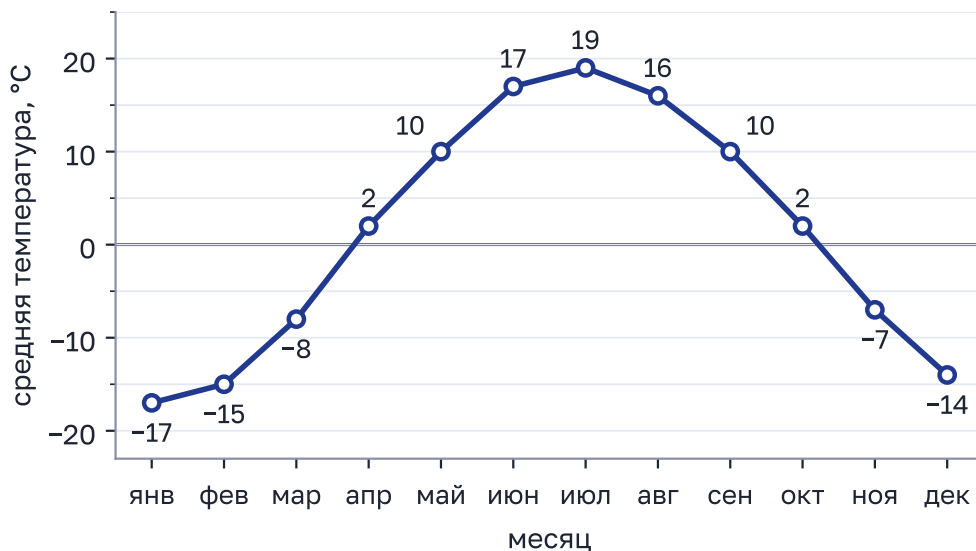


Рисунок 6.3 – Средняя температура воздуха в Новосибирске по месяцам (данные условные). Над каждой точкой подписано её значение

Как найти по графику значение: найди нужный месяц на горизонтальной оси, поднимись (или опустишься) до точки, от точки иди влево к вертикальной оси. Для июля это 19 °С, для ноября –7 °С. Отсюда многое видно без единого вычисления:

- самый тёплый месяц – июль, это самая высокая точка;
- ниже нуля средняя температура держится пять месяцев: январь, февраль, март, ноябрь и декабрь;
- осенью линия идёт вниз, и чем она круче, тем быстрее холодает.

Насколько изменилась температура, считают так: «стало минус было». С октября по ноябрь было 2, стало –7: $-7 - 2 = -9$. Минус в ответе значит «стало холоднее», на 9 градусов. С отрицательным «было» аккуратнее. С февраля по март было –15, стало –8: $-8 - (-15) = -8 + 15 = 7$, потеплело на 7 градусов. Вычесть отрицательное число – всё равно что прибавить его без минуса. Проверь по числовой прямой: от –15 до –8 ровно 7 шагов.

Графики удобно строить в Python. Библиотека `matplotlib` в Colab и Jupyter уже установлена, в простом онлайн-интерпретаторе её может не быть.

```
import matplotlib.pyplot as plt

mesyacy = ["январь", "фев", "мар", "апр", "май", "июнь",
           "июль", "авг", "сен", "окт", "ноя", "дек"]
temp = [-17, -15, -8, 2, 10, 17, 19, 16, 10, 2, -7, -14]

plt.plot(mesyacy, temp, marker="o")      # линия с кружками в точках
plt.axhline(0, color="gray")           # линия нуля
plt.title("Новосибирск: средняя температура (данные условные)")
plt.xlabel("месяц")
plt.ylabel("температура, °C")
plt.show()
```

Построчно. Первая строка подключает библиотеку графиков и даёт ей короткое имя `plt`. Дальше два **списка** – наборы значений в квадратных скобках: подписи месяцев и темпера-

туры, по одной на месяц, в том же порядке. `plt.plot` рисует линию: первым идёт то, что по горизонтали, вторым – то, что по вертикали; `marker="o"` ставит кружок в каждой точке. `plt.axhline(0)` проводит горизонтальную линию на нуле. Три следующие строки пишут название и подписи осей, а `plt.show()` показывает картинку. Для столбиков вместо `plt.plot` пишут `plt.bar`, для точек – `plt.scatter`, для круга – `plt.pie`.

Запомни. Сначала подписи, потом картинка: что по осям, в каких единицах, с какого числа начинается ось. Цена деления – разница соседних подписей, делённая на число промежутков между ними.

Совет. Если на оси написано «тыс.», «млн» или «%», запиши это рядом с ответом ещё до вычислений. Ответ «30» вместо «30 000» – одна из самых обидных потерь баллов.

6.3. Числа-сводки: среднее, медиана, минимум, максимум, размах

НА ПАЛЬЦАХ

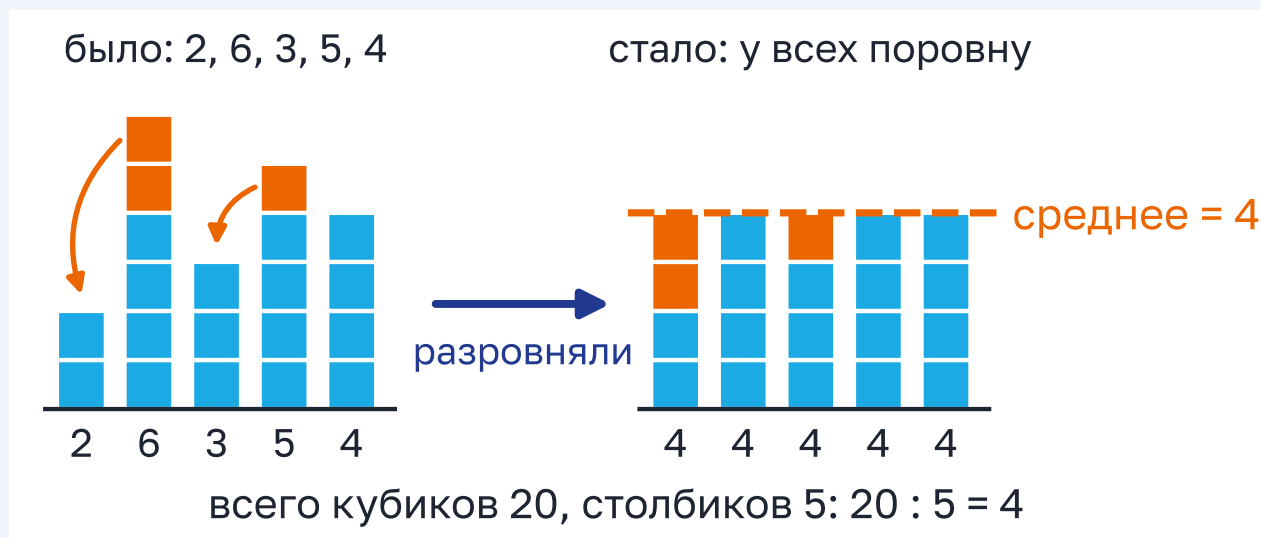


Рисунок 6.4 – Среднее – это высота, которая получится, если разровнять все столбики

Возьми лишние кубики с высоких столбиков и отдай низким. Когда все сравнялись, их высота и есть среднее.

Когда чисел много, их хочется описать одним-двумя числами. В этой книге такие числа мы зовём **сводками**, в учебнике статистики – числовыми характеристиками. Главных пять.

- **Минимум** – самое маленькое значение, **максимум** – самое большое.
- **Размах** – насколько далеко разбежались значения: максимум минус минимум.
- **Среднее арифметическое**, или просто среднее, – сумма всех значений, делённая на их количество. Это та самая высота разровненных столбиков:
 $(2 + 6 + 3 + 5 + 4) : 5 = 20 : 5 = 4$.

- **Медиана** – число, которое стоит посередине, если выстроить все значения по возрастанию. Половина значений не больше медианы, половина – не меньше.

Разберём на примере. Семь ребят решали одну задачу и записали, сколько минут у каждого ушло: 9, 4, 12, 30, 6, 10, 7.

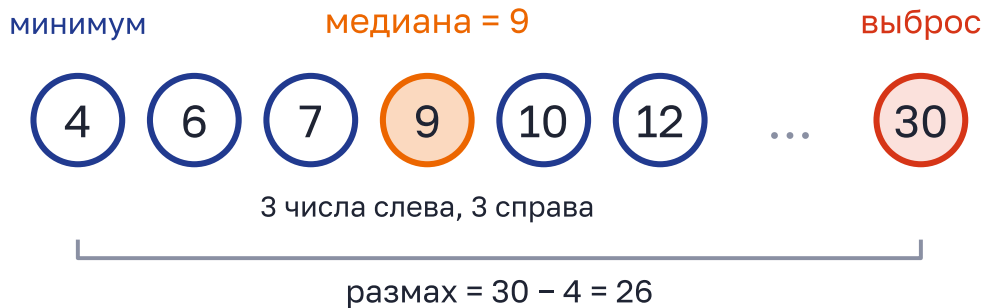


Рисунок 6.5 – Те же семь чисел по порядку. Медиана – в середине, выброс – справа

Шаг 1. Выстраиваем по возрастанию: 4, 6, 7, 9, 10, 12, 30. Без этого шага медиану не найти – это первое, что забывают.

Шаг 2. Минимум и максимум: 4 и 30 мин. **Размах:** $30 - 4 = 26$ мин.

Шаг 3. Среднее. Сумма $4 + 6 + 7 + 9 + 10 + 12 + 30 = 78$, чисел 7, среднее $78 : 7 = 11,1$ мин (округлили до десятых).

Шаг 4. Медиана. Чисел 7 – нечётное количество, посередине стоит четвёртое: слева три числа, справа три. Медиана – 9 мин.

Если чисел **чётное** количество, посередине оказываются два числа. Тогда медиана – их среднее. Для 4, 6, 7, 9, 10, 12 середина – между 7 и 9, медиана $(7 + 9) : 2 = 8$.

А теперь сравни. Среднее 11,1 мин больше, чем у пятерых ребят из семи. Кто-то один задумался на 30 минут (может, отвлёкся на телефон) – и утащил среднее вверх. Такое значение, резко не похожее на остальные, называют **выбросом**. Выброс сильно тянет за собой среднее и почти не трогает медиану. Убери 30 из ряда – среднее упадёт до $48 : 6 = 8$ мин, а медиана сдвинется совсем чуть-чуть: с 9 до $(7 + 9) : 2 = 8$. Поэтому, когда в данных есть выбросы, честнее смотреть на медиану. Так поступают, например, когда рассказывают о зарплатах: пара миллионеров в посёлке сильно поднимает среднюю зарплату, а медиану почти не меняет.

Посчитаем всё это программой. Здесь хватит самого простого онлайн-интерпретатора Python.

```

minuty = [4, 6, 7, 9, 10, 12, 30]

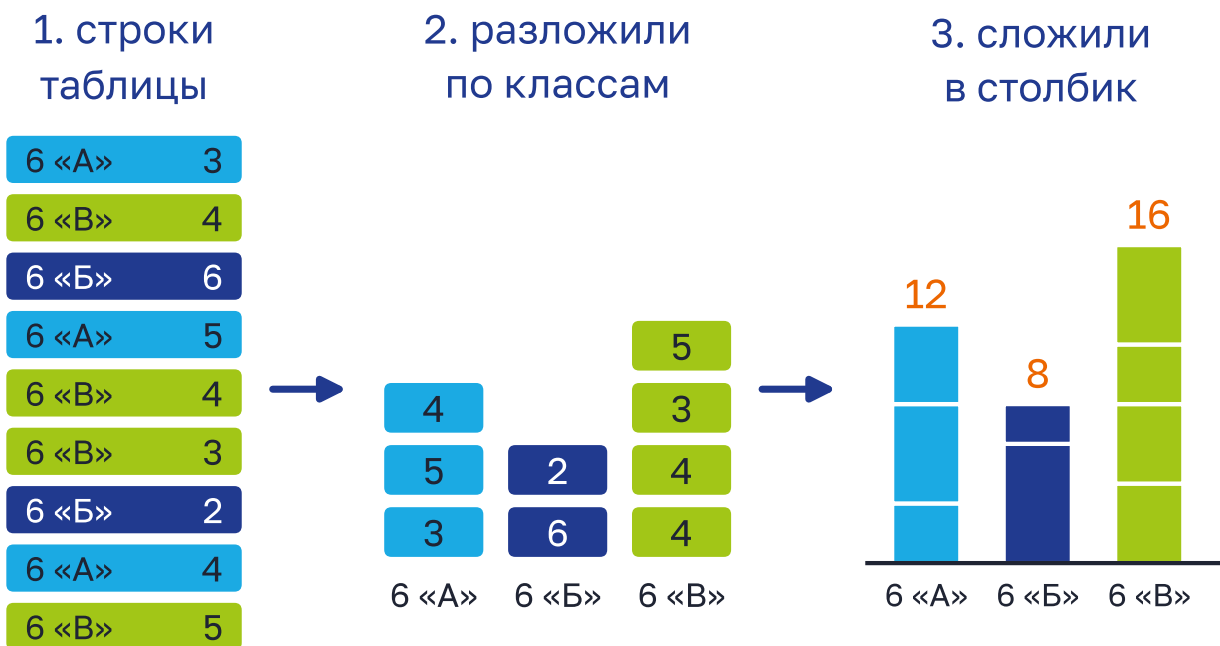
srednee = sum(minuty) / len(minuty)
po_poryadku = sorted(minuty)
n = len(po_poryadku)
if n % 2 == 1:
    mediana = po_poryadku[n // 2]
else:
    mediana = (po_poryadku[n // 2 - 1] + po_poryadku[n // 2]) / 2

print(round(srednee, 1))          # 11.1
print(mediana)                   # 9
print(max(minuty) - min(minuty)) # 26

```

`sum` складывает все числа списка, `len` считает, сколько их. `sorted` выстраивает числа по возрастанию. `n % 2` – остаток от деления на 2: если он равен 1, чисел нечётное количество. `if ... else` читается как «если ..., иначе ...»: строка под `if` выполнится при нечётном количестве, строка под `else` – при чётном. `n // 2` – деление без остатка: для $n = 7$ получится 3. Номер 3 в Python – это **четвёртое** число, потому что счёт номеров идёт с нуля. При чётном количестве берём два средних числа и находим их среднее. `max` и `min` дают максимум и минимум, `round(..., 1)` округляет до десятых, `print` выводит результат на экран. После `#` идёт комментарий – Python его не читает, это заметка для человека.

НА ПАЛЬЦАХ



агрегация «сумма»: всё, что в кучке, ставим друг на друга

Рисунок 6.6 – Как из таблицы получаются столбики: разложили строки по кучкам, каждую кучку превратили в одно число

В таблице одна строка – один ученик. На диаграмме один столбик – целый класс. Между ними стоит правило, по которому кучка чисел становится одним числом.

Сводки по группам. В разделе 5.6 мы уже раскладывали строки таблицы на кучки и считали в каждой сумму, среднее, максимум. В сервисах для графиков такой подсчёт по группам называют **агрегацией**. Когда по таблице строят столбчатую диаграмму, выбирают три вещи: какое поле пойдёт по горизонтали (например, класс), какое по вертикали (прочитанные книги) и какую агрегацию применить. Высота столбика – результат агрегации для своей группы. На рисунке выбрана «сумма», и выше всех 6 «В». Возьми «среднее» – и все три столбика станут одинаковыми: $12:3 = 4$, $8:2 = 4$, $16:4 = 4$. Данные те же, а картинка совсем другая. Как выглядят эти настройки в Датавиз-конструкторе, покажут на курсе сферы, а смысл у них тот же.

Запомни. Медиана – середина **упорядоченного** ряда; при чётном количестве – среднее двух средних. Выброс тянет среднее и почти не трогает медиану. Сумма, среднее, максимум по группам – разные вопросы и разные ответы.

Совет. Проверь среднее на разумность: оно всегда лежит между минимумом и максимумом. Вышло среднее больше максимума – ошибка в сумме или в количестве.

6.4. Гипотеза: проверяем догадку графиком

НА ПАЛЬЦАХ

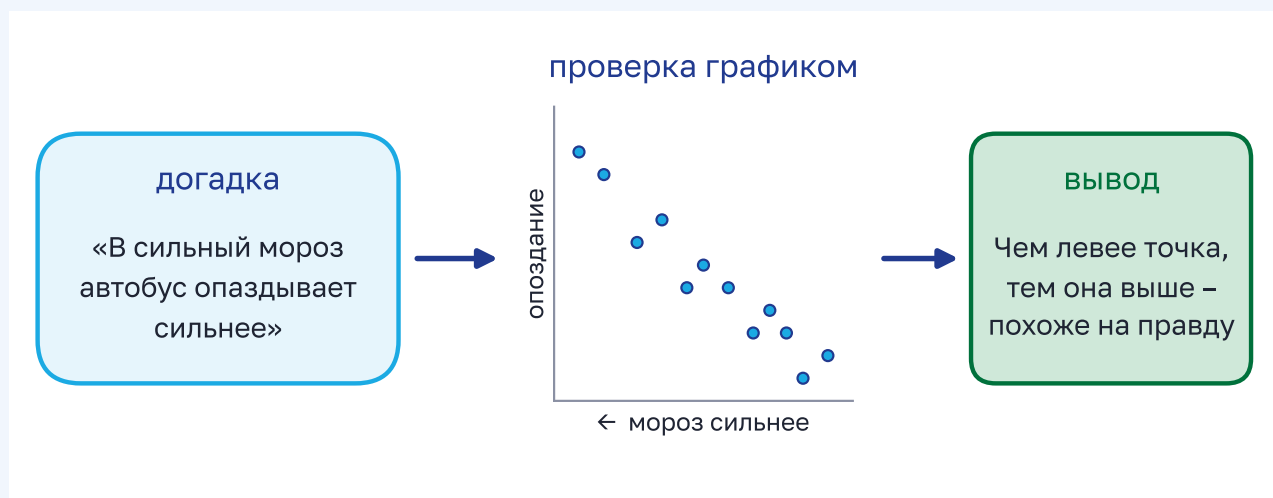


Рисунок 6.7 – Догадка, проверка, вывод

Догадку не принимают на веру и не отбрасывают на глаз: строят по данным график и смотрят, согласен ли он с ней.

Гипотеза – предположение, которое можно проверить данными. «Зимой в библиотеке берут больше книг, чем летом». «Чем дольше ребята сидят вечером в телефоне, тем позже засыпают». Хорошая гипотеза говорит конкретно, **что с чем** связано, и её можно проверить таблицей. «Погода влияет на жизнь» – не гипотеза: непонятно, что мерить.

Сначала решают, какие две величины сравнивать. Потом собирают таблицу, где у каждого объекта (дня, ученика) есть обе величины. По ней строят подходящий график: связь двух величин – точечная диаграмма, сравнение групп – столбчатая, изменение во времени – линейный. И уже глядя на график, честно говорят: подтверждает он догадку, опровергает или по этим данным сказать нельзя.

Пример. Школьник из Бердска две зимы подряд записывал утреннюю температуру и на сколько минут опоздал школьный автобус. Гипотеза: в сильный мороз автобус опаздывает сильнее.

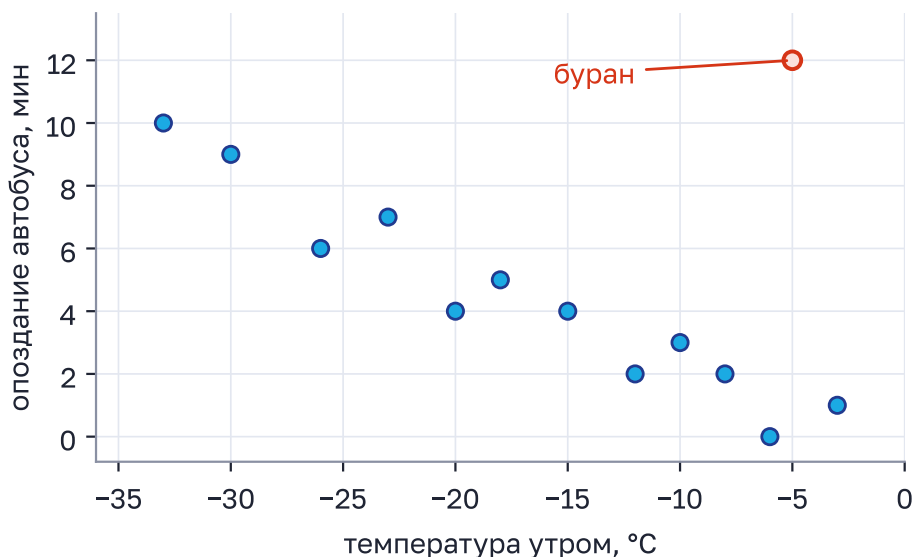


Рисунок 6.8 – Каждая точка – одно утро (данные условные). Красная точка – утро, когда был буря

Точки вытянулись полосой из левого верхнего угла в правый нижний. Левее – значит холоднее, выше – значит дольше опоздание. В утра от -25°C и холоднее автобус опаздывал на 6–10 минут, в утра от -10°C и теплее (кроме одного) – на 0–3 минуты. Гипотеза подтверждается. Одна точка выбивается: -5°C и 12 минут. Это утро бурана – дорогу замело. Выброс не отменяет вывод, но подсказывает, что на опоздание влияет не только мороз.

Но радоваться рано: у такого вывода есть два слабых места.

- **Мало данных – слабый вывод.** По трём точкам можно «увидеть» что угодно. Чем больше точек, тем больше доверия.
- **Связь – ещё не причина.** Летом на Обском море в дни, когда продают больше мороженого, больше людей и обгорают на солнце. Мороженое тут ни при чём: обе величины растут в жаркие солнечные дни. График показывает, что две величины меняются вместе, но не говорит, какая из них «виновата». В нашем примере причина правдоподобна – в мороз хуже заводятся моторы и скользко, – но это знание о жизни, а не о графике.

Запомни. Гипотеза – конкретная догадка, которую можно проверить таблицей. Точки полосой вверх или вниз – связь есть, кашей – не видно. Связь не доказывает, что одно вызывает другое.

6.5. Обманчивые графики

НА ПАЛЬЦАХ

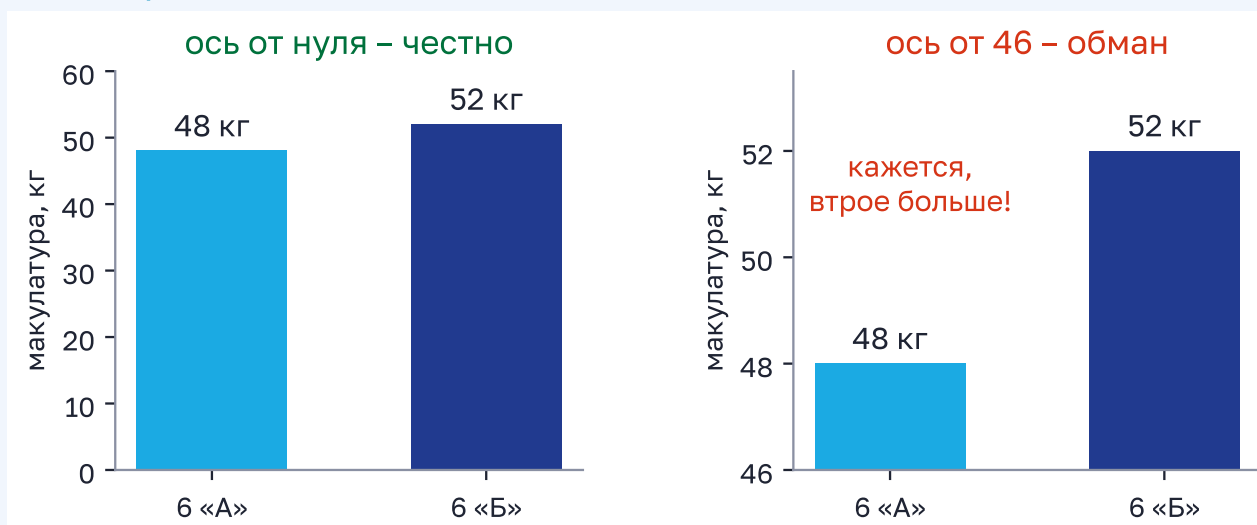


Рисунок 6.9 – Одни и те же числа, два графика

Слева ось начинается с нуля, и видно: классы собрали почти поровну. Справа низ оси отрезали – и второй столбик выглядит втрое выше первого.

График можно нарисовать так, что все числа на нём верные, а впечатление – ложное. Иногда это делают нечаянно, иногда – нарочно, в рекламе. Приёмы у таких графиков одни и те же.

Обрезанная ось. Вертикальная ось столбчатой диаграммы начинается не с нуля, а с числа поближе к данным. Тогда столбики сравнивают только «верхушки». На рисунке 48 и 52 кг, ось от 46: видимые столбики – $48 - 46 = 2$ и $52 - 46 = 6$ делений. Кажется, что второй класс собрал в $6 : 2 = 3$ раза больше. А по числам – в $52 : 48 = 1,08$ раза, то есть всего на 8 % больше. У линейного графика ось часто начинают не с нуля, и обмана тут нет: там важна форма линии, а не высота. А вот столбики без нуля внизу почти всегда вводят в заблуждение.

Неровный шаг. На оси времени подписаны 2020, 2021, 2022, 2025 – между последними двумя прошло три года, а расстояние такое же. Линия после 2022 года будто взлетает круче, чем на самом деле.

Нет единиц или подписи оси. «Выросло до 40» – сорок чего? Процентов, тысяч, штук? Без подписи график ничего не доказывает.

Круг, который не круг. Круговая диаграмма, где доли в сумме дают 130 %, или трёхмерный наклонённый круг, где передняя долька кажется больше, чем она есть.

Выбранный кусочек. Показали только те месяцы, где линия идёт вверх, а остальные «забыли».

Запомни. Перед сравнением столбиков посмотри, с какого числа начинается ось. Сравнивай сами числа, а не высоту картинki: во сколько раз – делением, на сколько процентов – от того значения, с которым сравниваешь.

6.6. Дашборд и вычислительное эссе

НА ПАЛЬЦАХ



Рисунок 6.10 – Приборная панель в машине и дашборд данных устроены одинаково

Водитель не листает отчёт – он одним взглядом видит скорость и бензин. Дашборд делает то же с данными.

Дашборд (от английского dashboard – приборная панель) – один экран, на котором собраны главные числа и несколько графиков про одно и то же. Сверху обычно стоят **плитки** с главными числами: среднее, максимум, итог. Ниже – графики, каждый отвечает на свой вопрос. Зачем всё на одном экране? Чтобы сравнивать. Посмотрел на круг – ясных дней больше всего. Перевёл взгляд на линию – а ясные дни, оказывается, самые холодные.

Метеостанция кружка, 1–10 марта (данные условные)

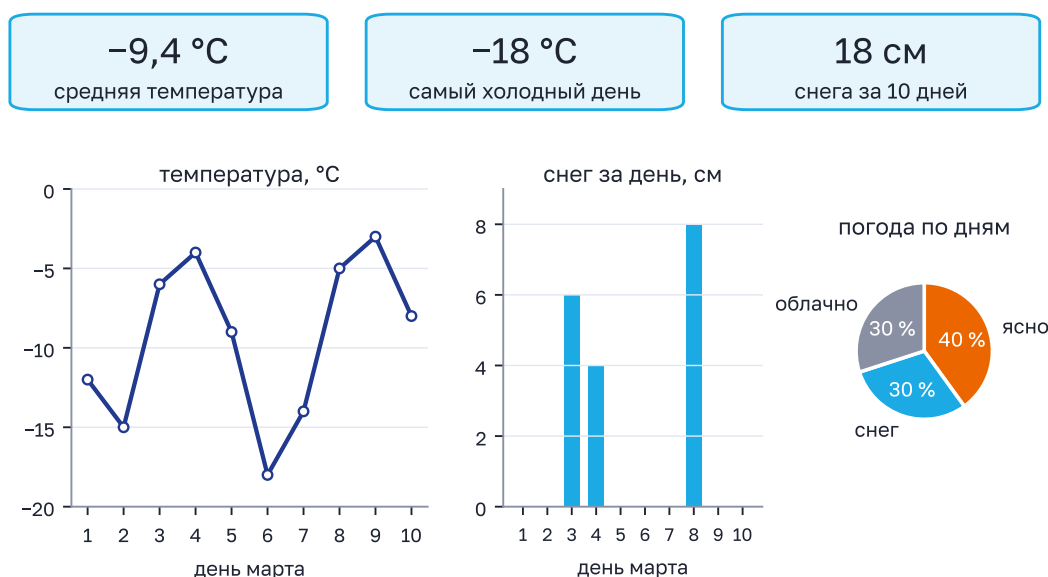


Рисунок 6.11 – Дашборд метеостанции школьного кружка за 1–10 марта. Плитки с главными числами, линия температуры, столбики снега и круг погоды по дням

Читают дашборд сверху вниз. Заголовок скажет, про что он, где и за какое время. Плитки – ответы на самые частые вопросы. Каждый график сначала прочитай отдельно, по правилам

раздела 6.2, а потом сопоставь их между собой: ось времени у них общая (здесь – дни марта), и один и тот же день можно найти на каждом.

Вычислительное эссе – это короткое исследование, где рассуждение идёт вперемешку с таблицами и графиками, а в конце стоит вывод словами. В программе сферы эссе и дашборд делают в Датавиз-конструкторе: это последовательность диаграмм и текстовых пояснений (текст пишут в формате Markdown), кода там нет. Для сравнения: тетради Colab и Jupyter устроены похоже, только там между текстом и графиком есть ещё и код. Проверять себя можно одним вопросом: если закрыть весь текст и оставить только график, подтвердит ли он твой вывод?

Хорошее эссе отвечает на пять вопросов:

1. **Что проверяем?** Вопрос или гипотеза одной фразой.
2. **На каких данных?** Откуда, сколько строк, за какое время, условные они или настоящие.
3. **Что показывает график?** Какой вид выбран и почему.
4. **Какой вывод?** Словами и **с числами**: «в 4 ясных дня средняя температура $-14,75^{\circ}\text{C}$, в 3 снежных -5°C ».
5. **Какие оговорки?** Мало данных, выбросы, связь не равна причине.

Вот пример вывода по графику опозданий автобуса из раздела 6.4:

Мы проверяли, опаздывает ли школьный автобус сильнее в мороз. Данные – 13 утр двух зим (условные). Мы выбрали точечную диаграмму: у каждого утра две величины, температура и опоздание. Самые морозные утра (-25°C и холоднее) дали опоздания 6–10 минут, утра от -10°C и теплее – не больше 3 минут. Гипотеза подтверждается. Исключение одно: бурян при -5°C и опоздание 12 минут, значит, на опоздания влияет не только мороз. Утр всего 13, вывод стоит проверить ещё на одной зиме.

Текст между кодом в тетрадях Colab и Jupyter пишут на **Markdown** – это простая разметка текста значками: строка с **#** в начале становится заголовком, слово в ****звёздочках**** – жирным, строки с **-** в начале – списком. Такой текст легко читать даже без оформления, в обычном блокноте.

Опаздывает ли автобус в мороз

****Гипотеза:**** в сильный мороз автобус опаздывает сильнее.

- данные: 13 утр, две зимы (условные)
- график: точечная диаграмма «температура – опоздание»

Запомни. Дашборд – главное на одном экране: плитки с числами и несколько графиков. Эссе – вопрос, данные, график, вывод с числами и оговорки. Вывод без чисел – это мнение, а не вывод.

Совет. Задачи «выбери все верные выводы» решай по одному утверждению за раз. Для каждого найди на графике или в таблице число, которое его подтверждает или опровергает, и поставь рядом «да» или «нет». Слова «потому что», «из-за», «всегда» – повод насторожиться: график редко доказывает причину.

6.7. Разбираем задачи

Задача 6.1 Год в Новосибирске

разминка · чтение линейного графика, размах, изменение

По графику средней температуры в Новосибирске (раздел 6.2) ответь на вопросы. а) Какой месяц самый холодный и какая в нём средняя температура? б) Сколько месяцев в году средняя температура выше нуля? в) Найди размах средних температур за год. г) Между какими соседними месяцами средняя температура выросла сильнее всего и на сколько градусов?

Дано: линейный график, 12 точек с подписанными значениями.

Найти: самый холодный месяц; число месяцев выше нуля; размах; самый большой рост между соседями.

Решение

Шаг 1. Самый холодный месяц. Ищем самую нижнюю точку. Это январь, -17°C .

Шаг 2. Выше нуля. Считаем точки над линией нуля: апрель (2), май (10), июнь (17), июль (19), август (16), сентябрь (10), октябрь (2). Всего 7 месяцев.

Шаг 3. Размах. Максимум – июль, 19°C , минимум – январь, -17°C . Размах $19 - (-17) = 19 + 17 = 36$ градусов.

Шаг 4. Самый большой рост. Рост – это «следующий минус предыдущий». Смотрим весну, где линия идёт вверх: январь $\rightarrow$ февраль: $-15 - (-17) = 2$; февраль $\rightarrow$ март: $-8 - (-15) = 7$; март $\rightarrow$ апрель: $2 - (-8) = 10$; апрель $\rightarrow$ май: $10 - 2 = 8$; май $\rightarrow$ июнь: $17 - 10 = 7$; июнь $\rightarrow$ июль: $19 - 17 = 2$. Самый большой – 10 градусов, с марта по апрель. На графике это самый крутой подъём линии.

В Новосибирске и правда холоднее всего в январе, а в апреле сходит снег – так что резкий скачок весной не удивляет.

 **Типичная ошибка.** Потерять минус у -17 и посчитать размах как $19 - 17 = 2$. Размах не может быть меньше разницы между любыми двумя точками, а июль и январь явно отличаются больше чем на 2 градуса. Ещё путают «самый большой рост» с самой высокой точкой (июль). Высокая точка – не то же самое, что крутой подъём.

Ответ: а) январь, -17°C ; б) 7 месяцев; в) 36 $^{\circ}\text{C}$; г) с марта по апрель, на 10 $^{\circ}\text{C}$.

Своя задача по программе сферы, модуль 4

Реши сам

6.1. По графику температуры в Новосибирске (раздел 6.2): а) сколько месяцев в году средняя температура ниже -10°C ? б) Какой месяц теплее: май или сентябрь?

6.2. На вертикальной оси подписаны 0, 50, 100, 150, а между соседними подписями – по 5 одинаковых промежутков. Сколько стоит одно деление? Столбик кончается на третьем делении выше подписи 100. Какое число он показывает?

Задача 6.2 Какой график нужен

разминка · выбор вида графика, сопоставление

Сопоставь каждую ситуацию с видом графика, который подходит лучше всего. Один вид может подойти к нескольким ситуациям.

Ситуации:

1. Как менялся вес тыквы на даче каждую неделю лета.
2. Какую часть учеников 6 классов занимает каждый кружок, если каждый ученик ходит ровно в один кружок.
3. Сколько килограммов макулатуры собрал каждый из пяти классов.
4. Есть ли связь между временем в телефоне вечером и временем засыпания у 30 учеников.
5. Опрос «Какие предметы тебе нравятся? Можно отметить несколько»: сколько процентов учеников отметили каждый предмет.

Виды: А – столбчатая, Б – линейный, В – круговая, Г – точечная.

Дано: 5 ситуаций, 4 вида графиков.

Найти: пары «ситуация – вид».

Решение

Шаг 1. Для каждой ситуации задаём вопрос из таблицы раздела 6.1.

Шаг 2. Ситуация 1. Одна величина (вес тыквы) меняется со временем (недели). Это линейный график – **Б**.

Шаг 3. Ситуация 2. Все ученики – одно целое, каждый попал ровно в одну долю, доли в сумме дают 100 %. Круговая диаграмма – **В**.

Шаг 4. Ситуация 3. Сравниваем пять классов между собой – «кто больше». Столбчатая – **А**.

Шаг 5. Ситуация 4. У каждого ученика две величины: время в телефоне и время засыпания. Вопрос «связаны ли» – точечная диаграмма, **Г**. Каждый ученик станет одной точкой.

Шаг 6. Ситуация 5. Вроде бы проценты и доли – тянет нарисовать круг. Но ответы можно было отмечать по нескольку, поэтому проценты в сумме выйдут больше 100. Один и тот же ученик попадёт в несколько долек сразу, и целого круга из них не сложить. Сравниваем предметы между собой – столбчатая, **А**.

Столбчатая понадобилась дважды – условие это прямо разрешает.

! **Самая соблазнительная ошибка** – в ситуации 5 выбрать круговую: «там же проценты». Проценты ещё не значат «доли одного целого». Круговая годится, только когда каждый объект попадает ровно в одну дольку.

Ответ: 1 – Б, 2 – В, 3 – А, 4 – Г, 5 – А.

Своя задача по программе сферы, модуль 4

Реши сам

6.3. Сопоставь ситуацию и вид графика (А – столбчатая, Б – линейный, В – круговая, Г – точечная). 1) Как менялось число подписчиков канала кружка по месяцам. 2) Какую долю учеников школы привозит автобус, привозят родители, приходят пешком (каждый назвал один способ). 3) Рост и масса 25 учеников: есть ли связь. 4) Сколько книг прочитал каждый из пяти друзей.

6.4. Два опроса в 6 классах. Первый: «Где ты провёл больше всего времени летом?» (один ответ) – на даче 45 %, в лагере 30 %, в городе 25 %. Второй: «Где ты бывал летом? Можно отметить несколько» – на даче 60 %, в лагере 35 %, в городе 70 %. Для какого опроса можно нарисовать круговую диаграмму? Какой график нужен для другого?

Задача 6.3 Какой жанр популярнее

агрегация по группам, сумма, среднее, максимум

Школьный клуб выложил в свой онлайн-кинотеатр 10 мультфильмов и через месяц выгрузил таблицу просмотров.

Таблица 6.2

Код	Жанр	Возраст	Просмотры
M1	комедия	0+	120
M2	комедия	6+	90
M3	комедия	6+	60
M4	комедия	0+	70
M5	приключения	6+	150
M6	приключения	12+	110
M7	приключения	0+	40
M8	фантастика	12+	160
M9	фантастика	6+	50
M10	фантастика	12+	60

Аналитик строит столбчатую диаграмму: по горизонтали – жанр, по вертикали – просмотры. Какой жанр окажется самым высоким столбиком, если выбрать агрегацию а) «сумма»; б) «среднее»; в) «максимум»? г) Сколько всего просмотров у мультфильмов с меткой 6+?

Дано: таблица, 10 строк, 3 жанра.

Найти: победителя по сумме, по среднему и по максимуму; сумму просмотров для 6+.

Решение

Шаг 1. Раскладываем строки по жанрам. Комедия: 120, 90, 60, 70 (4 мультфильма). Приключения: 150, 110, 40 (3). Фантастика: 160, 50, 60 (3).

Шаг 2. Сумма. Комедия: $120 + 90 + 60 + 70 = 340$. Приключения: $150 + 110 + 40 = 300$. Фантастика: $160 + 50 + 60 = 270$. Больше всех – комедия.

Шаг 3. Среднее. Сумму группы делим на число мультфильмов в ней. Комедия: $340 : 4 = 85$. Приключения: $300 : 3 = 100$. Фантастика: $270 : 3 = 90$. Больше всех – приключения.

Шаг 4. Максимум. Самое большое число в каждой группе: комедия – 120, приключения – 150, фантастика – 160. Больше всех – фантастика.

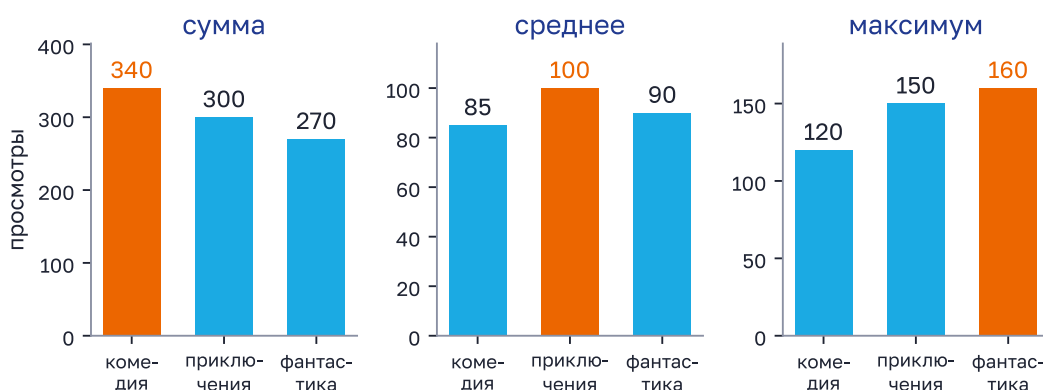


Рисунок 6.12 – Одна и та же таблица, три агрегации – три разных победителя (оранжевый столбик)

Шаг 5. Метка 6+. Выбираем строки, где в столбце «Возраст» стоит 6+: М2 (90), М3 (60), М5 (150), М9 (50). Сумма $90 + 60 + 150 + 50 = 350$.

Шаг 6. Проверка программой (в Colab, где есть библиотека `pandas` для таблиц).

```
import pandas as pd

df = pd.DataFrame({
    "жанр": ["комедия", "комедия", "комедия", "комедия",
            "приключения", "приключения", "приключения",
            "фантастика", "фантастика", "фантастика"],
    "просмотры": [120, 90, 60, 70, 150, 110, 40, 160, 50, 60],
})
print(df.groupby("жанр")["просмотры"].agg(["sum", "mean", "max", "count"]))
```

`pd.DataFrame` собирает таблицу из столбцов. `groupby("жанр")` раскладывает строки по жанрам, `["просмотры"]` выбирает столбец, а `agg` считает для каждой группы сразу четыре агрегации: сумму, среднее, максимум и количество. Программа печатает таблицу: комедия – 340, 85, 120, 4; приключения – 300, 100, 150, 3; фантастика – 270, 90, 160, 3.

Почему победители разные? Комедий больше всех – четыре, поэтому они и выигрывают по сумме: побеждает количество. По среднему комедии проигрывают – каждую в отдельности смотрят не так уж много. А в «максимуме» фантастику вытягивает один хит на 160 просмотров.

 **Типичная ошибка.** Посчитать среднее, поделив сумму на 10 (на все мультфильмы), а не на число мультфильмов своего жанра. Тогда у комедии выйдет 34, у приключений 30, у фантастики 27, и победитель по «среднему» совпадёт с победителем по сумме. А в пункте г) легко пропустить строку – пересчитай: мультфильмов с меткой 6+ должно быть четыре.

Ответ: а) комедия (340); б) приключения (100); в) фантастика (160); г) 350 просмотров.

По мотивам задачи отборочного этапа НТО Junior 2023/24 «Описательная статистика и анализ возрастных ограничений»

Реши сам

6.5. Таблица результатов олимпиады кружков: «Роботы» – участники набрали 7, 9 и 5 баллов; «ИИ» – 8 и 10; «Космос» – 6, 6, 6 и 9. Строим столбчатую диаграмму «кружок – баллы». Какой кружок будет самым высоким столбиком при агрегации а) «сумма»; б) «среднее»; в) «количество»?

6.6.* В 6 «А» 20 учеников, в среднем каждый прочитал за лето 8 книг. В 6 «Б» 30 учеников, в среднем по 3 книги. Сколько книг в среднем прочитал один ученик, если считать оба класса вместе?

Задача 6.4 Лучшие ролики планетария

точечная диаграмма, топ по двум признакам, проценты

Планетарий выложил 10 роликов о космосе. Аналитик построил точечную диаграмму: по горизонтали – сколько тысяч раз ролик посмотрели, по вертикали – какой процент зрителей досмотрел его до конца. Каждая точка подписана кодом ролика.

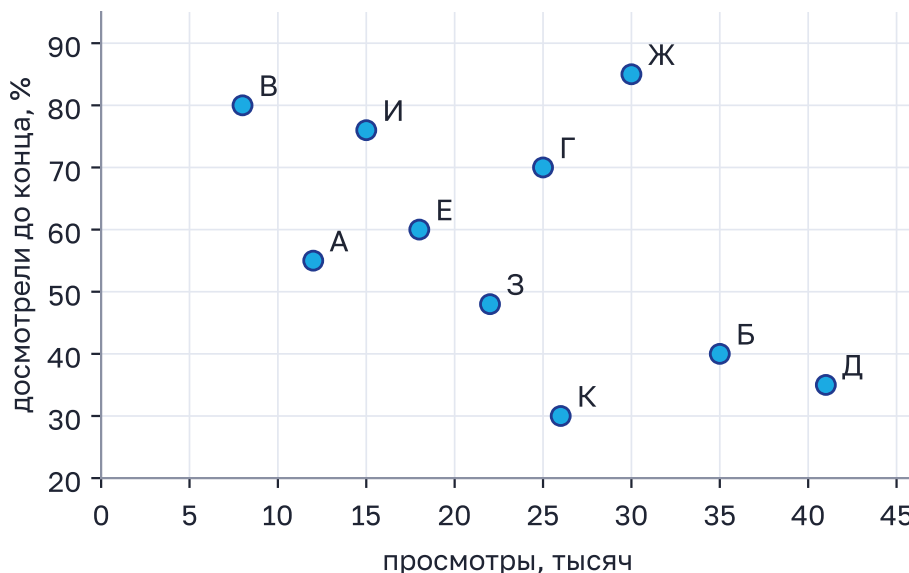


Рисунок 6.13 – Ролики планетария: просмотры и досмотр до конца (данные условные)

- а) Запиши коды трёх роликов с наибольшим числом просмотров, от большего к меньшему. б) Запиши коды трёх роликов с наибольшим процентом досмотра, от большего к меньшему. в) Какой ролик попал в обе тройки? г) Подсказка у точек показала: у ролика Д – 41 тыс. просмотров и 35 % досмотра, у ролика Ж – 30 тыс. и 85 %. Сколько человек досмотрели до конца ролик Д и сколько – ролик Ж?

Дано: точечная диаграмма, 10 точек; для Д и Ж – точные значения.

Найти: две тройки кодов; общий код; число досмотревших Д и Ж.

Решение

Шаг 1. Тройка по просмотрам. Больше просмотров – правее. Идём справа налево: Д (около 41 тыс.), Б (35 тыс.), Ж (30 тыс.). Следующая точка, К, – уже левее, около 26 тыс.

Шаг 2. Тройка по досмотру. Больше процент – выше. Идём сверху вниз: Ж (85 %), В (80 %), И (около 76 %). Следующая, Г, – 70 %, заметно ниже.

Шаг 3. Общий ролик. В первой тройке Д, Б, Ж, во второй Ж, В, И. Общий – Ж. Это правая верхняя точка облака: ролик и смотрят много, и досматривают.

Шаг 4. Досмотревшие. Сначала переводим единицы: 41 тыс. = 41 000 человек, 30 тыс. = 30 000. Процент от числа – умножаем на процент и делим на 100. Ролик Д:
 $41\,000 \cdot 35 : 100 = 14\,350$ человек. Ролик Ж: $30\,000 \cdot 85 : 100 = 25\,500$ человек.

Выходит, самый просматриваемый ролик Д досмотрели почти вдвое меньше людей, чем Ж: его часто открывают, но бросают. Для планетария это ценный вывод – только по числу просмотров его не увидеть.

❗ **Ошибка номер один** – перепутать оси: искать «больше просмотров» по высоте точки. Тогда в первую тройку попадут Ж, В, И, и обе тройки совпадут. Всегда сначала читай подписи осей. Бывает и так: в пункте г) выходит «14,35» и «25,5» – забыли, что на оси тысячи, а спрашивали людей.

Ответ: а) Д, Б, Ж; б) Ж, В, И; в) Ж; г) ролик Д – 14 350 человек, ролик Ж – 25 500 человек.

Реши сам

6.7. В школьном приложении шесть игр. Оценка и число скачиваний: Г1 – 4,8 и 1200; Г2 – 4,1 и 3000; Г3 – 4,9 и 2500; Г4 – 3,5 и 3500; Г5 – 4,6 и 800; Г6 – 4,7 и 2800. Какая игра попадает и в тройку лучших по оценке, и в тройку лучших по скачиванием?

6.8. Круговая диаграмма показывает, какой кружок выбрали 240 учеников (каждый – один). Доля «Кванториум» занимает угол 90° . а) Сколько учеников выбрали «Кванториум»? б) Доля «Спорт» – 30 % учеников. Какой у неё угол? Весь круг – 360° .

Задача 6.5 Реклама прогноза погоды

обрезанная ось, во сколько раз, проценты, верные выводы

Два приложения прогнозируют погоду с помощью ИИ. За 100 дней прогноз А оказался верным в 92 днях, прогноз Б – в 98. Разработчики Б нарисовали рекламу.

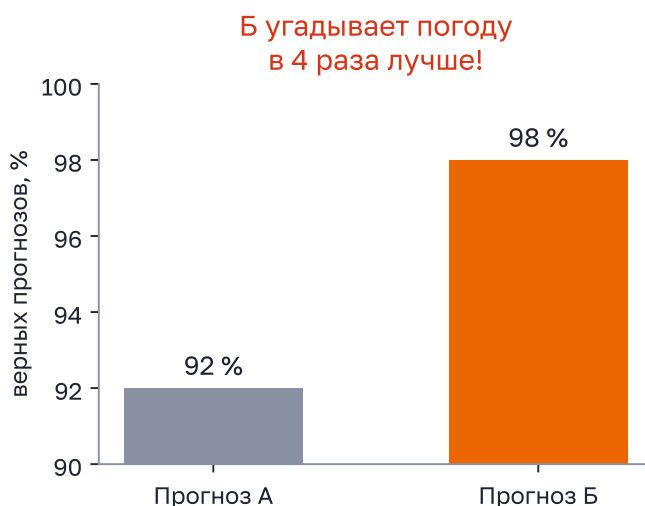


Рисунок 6.14 – Рекламный график приложения Б

а) На сколько процентов прогноз Б верен чаще, чем прогноз А? Округли до десятых. б) Выбери все верные утверждения.

1. На рисунке столбик Б в 4 раза выше столбика А.
2. Приложение Б угадывает погоду в 4 раза чаще, чем А.
3. Из 100 дней Б угадало на 6 дней больше, чем А.
4. Приложение Б ошибается в 4 раза реже, чем А.
5. Вертикальная ось на рисунке начинается с нуля.

Дано: А – 92 верных дня из 100, Б – 98 из 100; ось рисунка от 90 до 100.

Найти: на сколько процентов Б верен чаще; верные утверждения.

Решение

Шаг 1. Проценты. «На сколько процентов больше» считаем от того, с чем сравниваем, – от А. Разница $98 - 92 = 6$ дней. $6:92 \cdot 100 = 6,52 \approx 6,5 \%$.

Шаг 2. Утверждение 1. Ось начинается с 90. Видимая высота столбика А – $92 - 90 = 2$ деления, Б – $98 - 90 = 8$ делений. $8:2 = 4$. На рисунке – правда в 4 раза. Верно.

Шаг 3. Утверждение 2. Угадывает в $98:92 = 1,07$ раза чаще, а не в 4. Неверно – этот обман и сделан обрезанной осью.

Шаг 4. Утверждение 3. $98 - 92 = 6$ дней. Верно.

Шаг 5. Утверждение 4. Ошибок у А $100 - 92 = 8$ дней, у Б $100 - 98 = 2$ дня. $8:2 = 4$ – ошибается в 4 раза реже. Верно! Любопытно: «в 4 раза» в рекламе взялось из неправильного рисунка, а про ошибки оно случайно правда.

Шаг 6. Утверждение 5. Нижняя подпись оси – 90. Неверно.

Можно было и не считать. Оба приложения угадывают больше 9 дней из 10, поэтому «в 4 раза чаще» невозможно в принципе: в 4 раза больше 92 дней – это 368 дней из 100.

 **Типичная ошибка.** Отбросить утверждение 4 заодно с утверждением 2: «раз про 4 раза – значит, обман». Каждое утверждение проверяй своим числом. Угадывать и ошибаться – разные величины, и отношение у них разное. И в пункте а) не дели на 98: выйдет 6,1 %, а сравнивали с А.

Ответ: а) на 6,5 %; б) верны 1, 3 и 4.

Своя задача по программе сферы, модуль 4

Реши сам

6.9. 6 «А» собрал 150 кг макулатуры, 6 «Б» – 160 кг. На столбчатой диаграмме вертикальная ось начинается со 140. а) Во сколько раз столбик 6 «Б» выглядит выше столбика 6 «А»? б) Во сколько раз 6 «Б» собрал больше на самом деле? Округли до сотых.

6.10. Число посетителей планетария по месяцам (условно): январь – 1200, февраль – 1500, март – 2100, апрель – 2600, май – 1900, июнь – 1400. а) Между какими соседними месяцами посетителей прибавилось больше всего? б) На сколько процентов уменьшилось число посетителей с апреля по май? Округли до целых.

Задача 6.6* Кто больше читает

среднее и медиана, выброс, проверка гипотезы

Библиотекарь проверяет гипотезу: «6 «А» читает больше, чем 6 «Б»». В каждом классе по 9 учеников, каждый назвал, сколько книг прочитал за лето.

6 «А»: 4, 2, 5, 40, 3, 6, 4, 3, 5.

6 «Б»: 7, 5, 8, 6, 9, 7, 6, 8, 7.

а) Найди среднее и медиану для каждого класса. б) Подтверждает ли гипотезу среднее? А медиана? в) Сколько учеников 6 «А» прочитали больше среднего по своему классу? г) Что честнее сказать о гипотезе и почему?

Дано: два ряда по 9 чисел.

Найти: средние и медианы; вывод о гипотезе.

Решение

Шаг 1. Упорядочиваем. 6 «А»: 2, 3, 3, 4, 4, 5, 5, 6, 40. 6 «Б»: 5, 6, 6, 7, 7, 7, 8, 8, 9.

Шаг 2. Средние. Сумма 6 «А»: $2 + 3 + 3 + 4 + 4 + 5 + 5 + 6 + 40 = 72$, среднее $72 : 9 = 8$ книг. Сумма 6 «Б»: $5 + 6 + 6 + 7 + 7 + 7 + 8 + 8 + 9 = 63$, среднее $63 : 9 = 7$ книг.

Шаг 3. Медианы. В каждом ряду 9 чисел, медиана – пятое по порядку (четыре слева, четыре справа). 6 «А»: 4 книги. 6 «Б»: 7 книг.

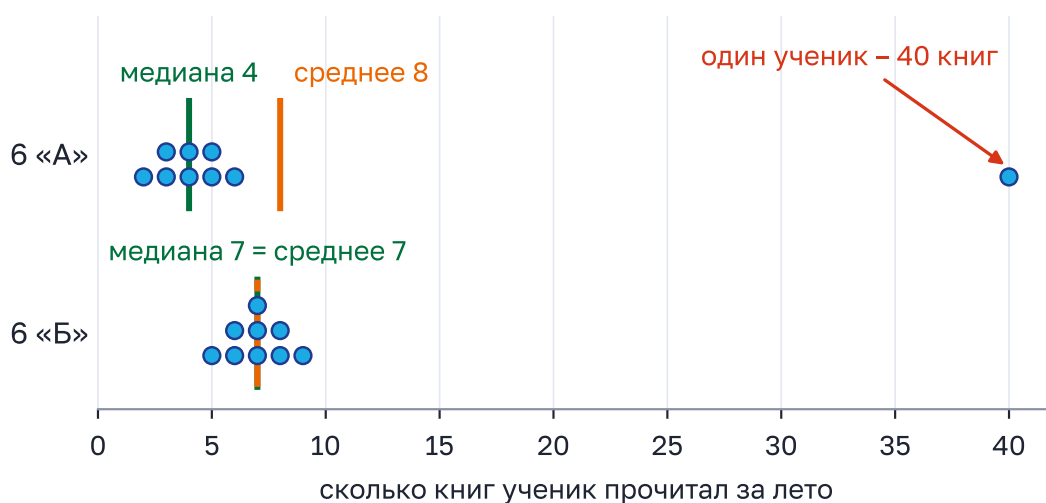


Рисунок 6.15 – Каждая точка – ученик. Зелёная линия – медиана, оранжевая – среднее

Шаг 4. Что говорят сводки. По среднему 6 «А» впереди: 8 больше 7 – гипотеза как будто подтверждается. По медиане впереди 6 «Б»: 7 больше 4 – гипотеза опровергается.

Шаг 5. Больше среднего в 6 «А». Среднее 8. Больше 8 книг прочитал только один ученик – тот, у кого 40. Остальные восемь прочитали от 2 до 6.

Шаг 6. Вывод. В 6 «А» есть выброс – один ученик с 40 книгами. Он один поднял среднее класса вдвое: без него сумма $72 - 40 = 32$, среднее $32 : 8 = 4$ книги. Обычный ученик 6 «А» прочитал около 4 книг, обычный ученик 6 «Б» – около 7. Честный вывод: гипотеза **не подтверждается**, больше читает 6 «Б», а в 6 «А» есть один очень увлечённый читатель.

Шаг 7. Проверка программой. Модуль `statistics` входит в Python, ставить ничего не нужно.

```

from statistics import mean, median

klass_a = [2, 3, 3, 4, 4, 5, 5, 6, 40]
klass_b = [5, 6, 6, 7, 7, 7, 8, 8, 9]

print(mean(klass_a), median(klass_a))    # 8 4
print(mean(klass_b), median(klass_b))    # 7 7

bolshe = 0
for k in klass_a:
    if k > mean(klass_a):
        bolshe = bolshe + 1
print(bolshe)                           # 1

```

`mean` – среднее, `median` – медиана, сортировать для них вручную не нужно. Цикл `for` перебирает учеников 6 «А», и каждый раз, когда число книг больше среднего, счётчик `bolshe` увеличивается на 1.

Примета выброса: среднее вдвое больше медианы, а выше среднего всего один человек из девяти. У 6 «Б» среднее и медиана совпали: числа там ровные, без выбросов.

❗ **Самая частая ошибка** – взять медиану из неупорядоченного ряда. Пятое число в исходной записи 6 «А» – это 3, а в 6 «Б» – 9, и вывод получится случайным. Обидно и другое: сказать «гипотеза подтверждается, среднее же больше» и не заметить, что одно число из девяти решило всё.

Ответ: а) 6 «А»: среднее 8, медиана 4; 6 «Б»: среднее 7, медиана 7; б) среднее подтверждает, медиана – нет; в) один ученик; г) гипотеза не подтверждается: среднее 6 «А» раздуто выбросом (40 книг), по медиане больше читает 6 «Б».

Своя задача по программе сферы, модуль 4

Реши сам

6.11. Пять недель подряд на даче мерили высоту снега, см: 12, 20, 26, 18, 39. Найди среднее, медиану и размах.

6.12. Восемь ребят записали, сколько минут идут до школы: 25, 7, 12, 40, 15, 10, 18, 9. Найди медиану и среднее.

6.13.* Средний балл пятерых участников команды – 12. В команду пришёл шестой, и средний балл команды стал 13. Сколько баллов у шестого участника?

Задача 6.7* Дашборд метеостанции

дашборд, среднее по группам, верные выводы, связь и причина

Ребята из кружка поставили во дворе школы метеостанцию и собрали дашборд за 1–10 марта (рисунок в разделе 6.6). Температура по дням, °C: –12, –15, –6, –4, –9, –18, –14, –5, –3, –8. Погода: 1, 2, 6, 7 марта – ясно; 3, 4, 8 марта – снег; 5, 9, 10 марта – облачно без снега. Выбери все верные утверждения.

1. Самый холодный день – 6 марта.
2. Больше всего снега за один день выпало 8 марта – 8 см.
3. Ясных дней вдвое больше, чем снежных.
4. В снежные дни средняя температура больше чем на 9 градусов выше, чем в ясные.
5. Средняя температура за 10 дней ниже –10 °C.
6. Снег идёт потому, что становится теплее.

Дано: дашборд; температура и погода по 10 дням.

Найти: номера верных утверждений.

Решение

Шаг 1. Утверждение 1. Самая нижняя точка на линии температуры – день 6, –18 °C. То же число – на плитке «самый холодный день». Верно.

Шаг 2. Утверждение 2. Самый высокий столбик снега – день 8, он доходит до 8 см. Верно.

Шаг 3. Утверждение 3. По кругу: ясно – 40 %, снег – 30 %. Из 10 дней это 4 и 3. Вдвое больше было бы 6 и 3. Неверно.

Шаг 4. Утверждение 4. Считаем среднее для каждой группы дней. Ясные (1, 2, 6, 7 марта):
 $(-12 - 15 - 18 - 14) : 4 = -59 : 4 = -14,75$ °C. Снежные (3, 4, 8 марта):
 $(-6 - 4 - 5) : 3 = -15 : 3 = -5$ °C. Разница: $-5 - (-14,75) = 9,75$ градуса. Больше 9 – верно.

Шаг 5. Утверждение 5. Сумма всех десяти температур:
 $-12 - 15 - 6 - 4 - 9 - 18 - 14 - 5 - 3 - 8 = -94$. Среднее $-94 : 10 = -9,4$ °C – это есть и на плитке. –9,4 **выше** –10: на градуснике оно ближе к нулю. Неверно.

Шаг 6. Утверждение 6. Дашборд показывает, что снег и относительно тёплые дни совпадают. Но «потому что» – это про причину, а график показывает только связь. Может быть, всё наоборот, а может, у обоих общая причина. Вообще-то зимой облака работают как одеяло: в ясную погоду тепло уходит в небо, и становится холоднее, а снег падает из облаков. Но это знание не из дашборда. Утверждение из данных не следует – неверно.

Шаг 7. Вывод словами. «За 1–10 марта средняя температура была –9,4 °C. Самыми холодными были ясные дни (в среднем –14,75 °C), в снежные дни было теплее – в среднем –5 °C. Снега выпало 18 см за три дня. Данных всего 10 дней, вывод о связи снега и тепла стоит проверить на всём марте».

Сверимся с плитками. На них те же числа, что у нас: –9,4 °C, –18 °C, 6 + 4 + 8 = 18 см.

⚠ Типичная ошибка. В утверждении 5 сравнить числа без минуса: 9,4 меньше 10, значит, будто бы –9,4 «ниже» –10. Среди отрицательных чисел **ниже** (холоднее) то, у которого без минуса число больше: –10 холоднее –9,4. А с утверждением 6 соглашаются, потому что на графике «всё видно». График видит только «вместе», а не «потому что».

Ответ: верны 1, 2 и 4.

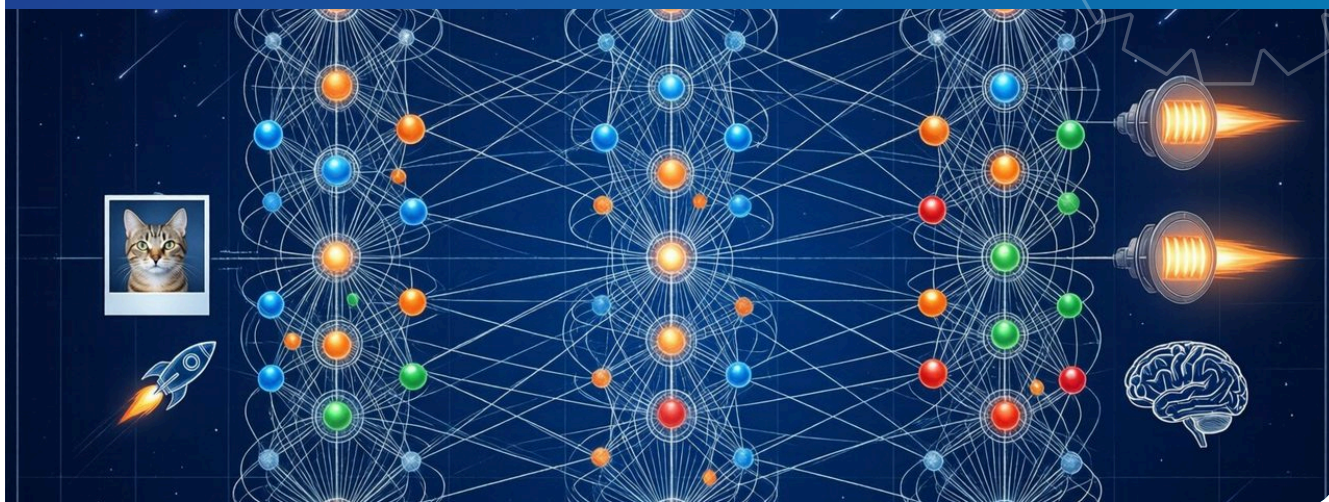
Своя задача по программе сферы, модуль 4

Реши сам

6.14. Гипотеза: «Чем больше часов в неделю ученик тренируется, тем быстрее он пробегает 60 м». Данные шести ребят (часы в неделю – время бега): 1 ч – 11,2 с; 2 ч – 10,8 с; 3 ч – 10,5 с; 4 ч – 10,1 с; 5 ч – 9,9 с; 6 ч – 9,6 с. Какой вывод верен? А) Гипотеза подтверждается. Б) Гипотеза опровергается: время растёт. В) По этим данным ничего сказать нельзя.

6.15.* Пожарная часть целый год записывала про каждый пожар, сколько машин приехало его тушить и какой ущерб (в рублях) он нанёс. На точечной диаграмме точки идут полосой вверх: чем больше машин, тем больше ущерб. Выбери все верные выводы. 1) Пожарные машины увеличивают ущерб. 2) Обе величины больше, когда больше сам пожар. 3) Чтобы уменьшить ущерб, надо присылать меньше машин. 4) График показывает, что величины меняются вместе, но не показывает, что одна вызывает другую.

Нейросети своими руками



В главе 1 ты познакомился с одним нейроном: входы умножаются на веса, складываются и сравниваются с порогом. Теперь соберём из нейронов сеть, посчитаем её ответ вручную и посмотрим, как она учится, – что такое эпоха, скорость обучения и почему сеть иногда «зубрит». В конце главы – сервисы, где нейросеть собирают мышкой: Teachable Machine, Machine Learning for Kids и конструктор GraphNet, с которыми работают на сфере. В задачах будем считать выход маленькой сети и связи между слоями, искать по таблице точности, где сеть переобучилась, и подбирать сеть под задачу.

7.1. Из нейронов – сеть: слои и связи

НА ПАЛЬЦАХ

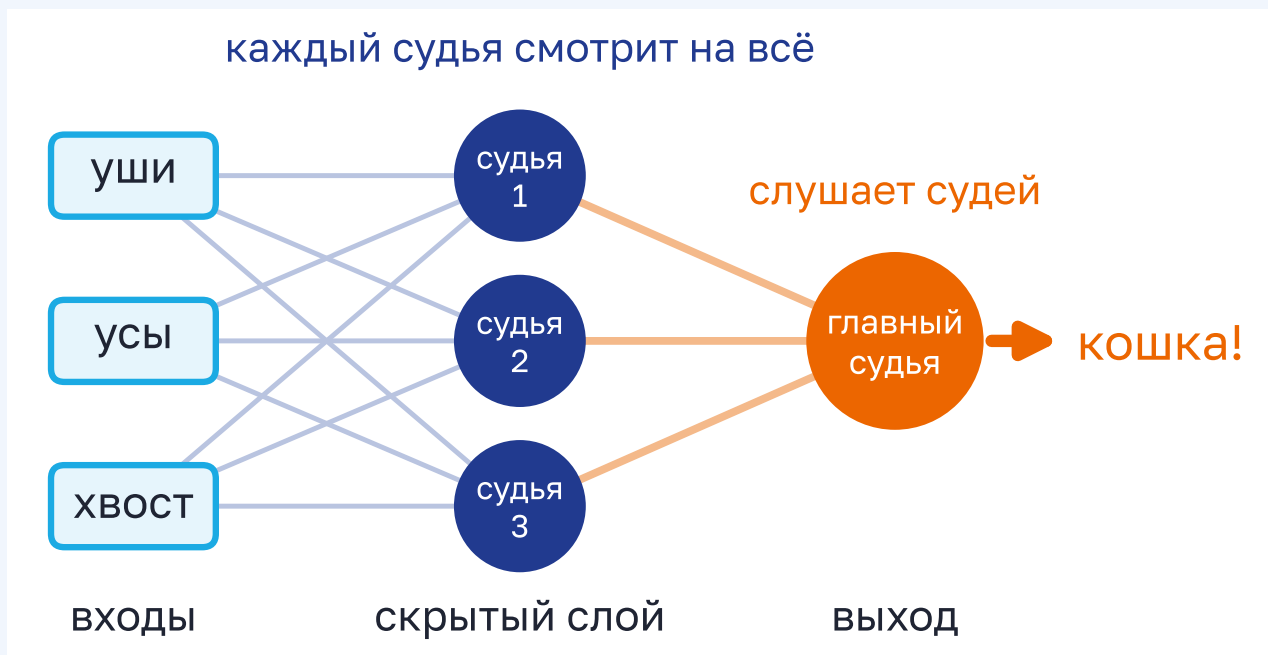


Рисунок 7.1 – Нейросеть похожа на жюри конкурса

Каждый судья скрытого слоя смотрит на все признаки сразу и ставит свою оценку. Главный судья слушает судей, у каждого – свой вес. Его решение и есть ответ сети.

Один нейрон умеет немного: сложить входы с весами и сравнить с порогом. Решить, открывать ли купол телескопа, он может, а узнать кошку на фото – уже нет. Поэтому нейроны соединяют в **сеть**: ответ одного нейрона становится входом для других.

Нейроны в сети стоят **слоями** – столбиками. Напомним, какие бывают слои (подробнее – в главе 1):

- **входной слой** – сюда подают признаки объекта, по одному числу на вход;
- **скрытые слои** – нейроны внутри сети, их может быть один, два, десять;
- **выходной слой** – отсюда берут ответ.

Чаще всего каждый нейрон одного слоя соединён с **каждым** нейроном следующего. Такие слои называют **полносвязными**. Каждая связь – это одна стрелочка со своим весом. Сеть описывают коротко, числами через тире: «3–4–2» значит 3 входа, 4 нейрона в скрытом слое и 2 выхода.

Сколько связей между слоями

Сосчитаем связи в сети 3–4–2. От каждого из 3 входов идёт по стрелке к каждому из 4 скрытых нейронов: это $3 \cdot 4 = 12$ связей. От каждого из 4 скрытых нейронов – к каждому из 2 выходов: $4 \cdot 2 = 8$ связей. Всего $12 + 8 = 20$.

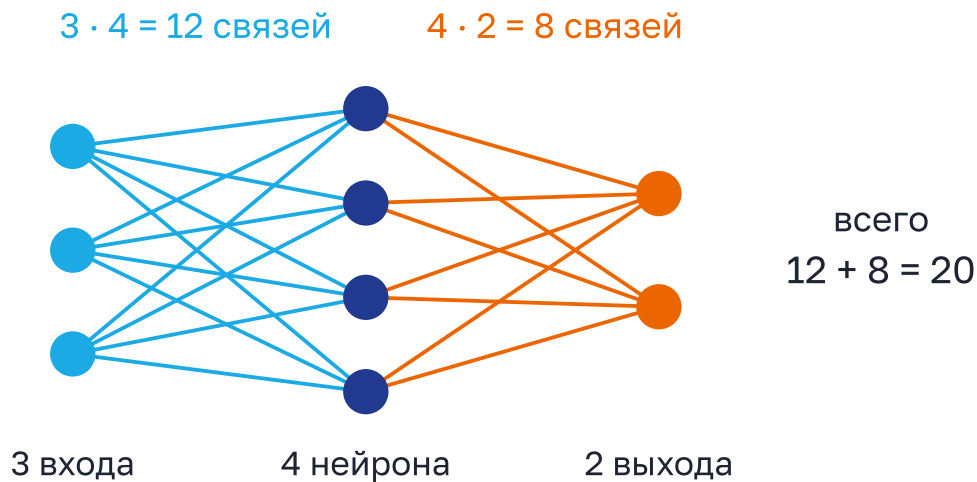


Рисунок 7.2 – Сеть 3–4–2. Между соседними слоями столько связей, сколько получится, если перемножить число нейронов в них

Выходит, **число связей между двумя соседними слоями равно произведению числа нейронов в них**. Для всей сети складываем связи между всеми парами соседних слоёв:

$$N = n_1 \cdot n_2 + n_2 \cdot n_3 + \dots,$$

где n_1 – число входов, n_2 – число нейронов в следующем слое и так далее, N – число связей. Перемножаем только **соседние** слои: входы с выходами в сети 3–4–2 напрямую не соединены.

Кроме весов связей у каждого нейрона скрытых и выходного слоя есть ещё одно число – **смещение**. Это добавка, которую нейрон прибавляет к сумме всегда, даже когда все входы равны нулю. По сути, это порог из главы 1, только с обратным знаком: порог 5 – всё равно что смещение -5 . У входов смещения нет: они ничего не считают, а просто передают числа. Веса и смещения вместе называют **параметрами** сети: именно их она подбирает при обучении. В сети 3–4–2 смещений $4 + 2 = 6$, а параметров $20 + 6 = 26$.

Считать связи удобно программой. Список `sloi` хранит число нейронов в каждом слое. Запись `sloi[1:]` – тот же список без первого числа, то есть `[4, 2]`. `zip` из главы 1 складывает два списка в пары, и получаются пары соседних слоёв: (3, 4), потом (4, 2).

```
sloi = [3, 4, 2]                                # сеть 3-4-2

svyazi = 0
for a, b in zip(sloi, sloi[1:]):                 # пары соседних слоёв
    svyazi += a * b
smeshcheniya = sum(sloi[1:])                     # у всех слоёв, кроме входного
print(svyazi, smeshcheniya, svyazi + smeshcheniya) # 20 6 26
```

Какую сеть выбрать под задачу

Число входов и выходов сеть не выбирает – его диктует задача.

- **Входов** столько, сколько признаков. Для таблицы – столько, сколько столбцов-признаков. Для картинки каждый пиксель (точка картинки) – отдельный признак: чёрно-белая картинка 8×8 пикселей даёт $8 \cdot 8 = 64$ входов.

- **Выходов** при классификации столько, сколько классов. Каждый выход показывает, насколько сеть уверена «в своём» классе, и ответом считают выход с самым большим числом. Для трёх видов облаков – 3 выхода. Для ответа «да или нет» хватает одного выхода: близко к 1 – «да», близко к 0 – «нет».
- При **регрессии**, когда сеть предсказывает число (температуру, цену, время), выход один, и его число берут как есть, без ступеньки.

А вот сколько скрытых слоёв и нейронов в них, решает человек. Такие настройки, которые выбирают до обучения, называют **гиперпараметрами**: число слоёв и нейронов, число эпох, скорость обучения (о двух последних – в разделе 7.3). Веса же сеть подбирает сама. Мало скрытых нейронов – сеть не уловит сложную закономерность. Слишком много – учится долго и легко начинает зубрить примеры (раздел 7.4).

Запомни. Связи между соседними слоями = произведение числа нейронов в них; связи сети – сумма по всем парам соседних слоёв. Смещения есть у каждого нейрона, кроме входов. Входов – сколько признаков, выходов – сколько классов, для числа – один выход.

Совет. Прежде чем считать, выпиши слои в строчку: «64 – 16 – 10». Потом под каждым тире – произведение соседей: $64 \cdot 16$, $16 \cdot 10$. Так ты не перемножишь случайно первый слой с последним.

7.2. Считаем выход сети

НА ПАЛЬЦАХ

Шаг 1: скрытый слой



Шаг 2: выход

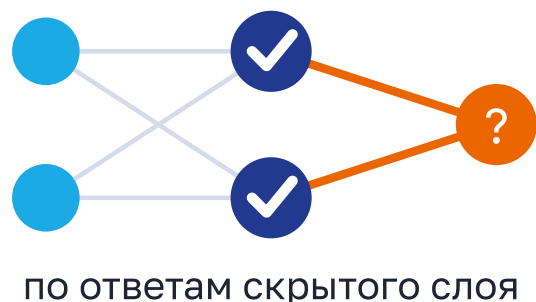


Рисунок 7.3 – Сеть считает по очереди, слой за слоем

Пока скрытый слой не досчитал, выходу нечего складывать. Сначала находим ответы всех нейронов скрытого слоя, потом по ним – ответ выхода.

Каждый нейрон сети умножает каждый свой вход на вес его связи и всё складывает. Потом прибавляет своё смещение – получается сумма S . И наконец пропускает сумму через **функцию активации** – правило, которое превращает сумму в ответ нейрона.

Буквами для нейрона с двумя входами:

$$S = x_1 \cdot w_1 + x_2 \cdot w_2 + b,$$

где x_1, x_2 – входы, w_1, w_2 – веса, b – смещение.

Функций активации в задачах три:

- **ступенька** из главы 1: $S \geq 0$ – ответ 1, иначе 0 (если в задаче дан порог T , сравнивают S с порогом);
- **ReLU** (читается «рэ-лу»): отрицательную сумму заменяет нулём, а остальное пропускает как есть. $-3 \rightarrow 0, 0 \rightarrow 0, 2,5 \rightarrow 2,5$. Проще всего запомнить так: «минус – в ноль»;
- **без изменений** (её ещё называют линейной): ответ равен сумме. Так делают у выхода, когда сеть предсказывает число.

ReLU – самая частая функция в скрытых слоях настоящих сетей. Если в задаче про неё ничего не сказано – читай условие ещё раз: какая функция у какого слоя, всегда написано.

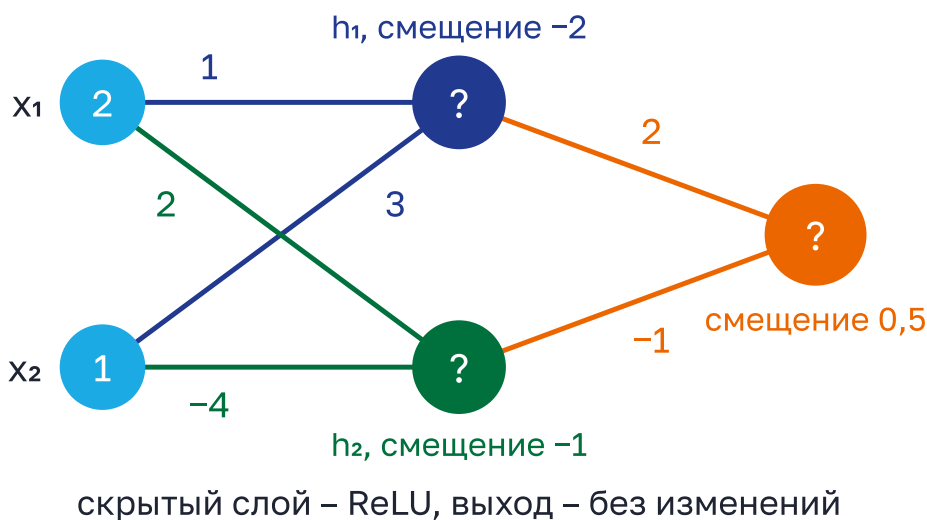


Рисунок 7.4 – Сеть 2–2–1. Цвет связи – цвет нейрона, в который она входит; число на связи – её вес

Посчитаем сеть с рисунка. На входах $x_1 = 2, x_2 = 1$. В скрытом слое ReLU, у выхода – без изменений.

Нейрон h_1 . Веса связей, которые в него входят: от x_1 – 1, от x_2 – 3. Смещение – 2.

$S = 2 \cdot 1 + 1 \cdot 3 + (-2) = 2 + 3 - 2 = 3$. Сумма положительная, ReLU пропускает её: $h_1 = 3$.

Нейрон h_2 . Веса: от x_1 – 2, от x_2 – (–4). Смещение – 1.

$S = 2 \cdot 2 + 1 \cdot (-4) + (-1) = 4 - 4 - 1 = -1$. Сумма отрицательная, ReLU превращает её в ноль: $h_2 = 0$.

Выход. Его входы – ответы скрытого слоя: $h_1 = 3$ и $h_2 = 0$. Веса 2 и (–1), смещение 0,5.

$S = 3 \cdot 2 + 0 \cdot (-1) + 0,5 = 6 + 0 + 0,5 = 6,5$. Функция «без изменений» – ответ сети **6,5**.

Проверим программой. Каждый нейрон – одна строчка: сумма произведений плюс смещение. `max(0, s)` – вся ReLU в одной записи: из числа и нуля берём большее, поэтому минус превращается в ноль.

```
def relu(s):
    return max(0, s)      # «минус - в ноль»

x1, x2 = 2, 1
h1 = relu(x1 * 1 + x2 * 3 + (-2))
h2 = relu(x1 * 2 + x2 * (-4) + (-1))
y = h1 * 2 + h2 * (-1) + 0.5      # у выхода - без изменений
print(h1, h2, y)                  # 3 0 6.5
```

Что было бы, если забыть ReLU? Тогда $h_2 = -1$ и выход $3 \cdot 2 + (-1) \cdot (-1) + 0,5 = 7,5$. Ответ другой, хотя вся арифметика верна. На этом баллы теряют чаще всего.

Запомни. Порядок всегда один: сумма «вход · вес» по всем входам, плюс смещение, потом функция активации. Считаем слой за слоем: входами выходного нейрона служат **от-веты** скрытых нейронов, а не исходные x .

Совет. Выписывай каждую связь отдельным слагаемым и отрицательные числа – в скобках: « $2 \cdot 2 + 1 \cdot (-4) + (-1)$ ». Рядом с каждым скрытым нейроном сразу пиши его ответ после ReLU, обведи его – и дальше бери только обведённые числа.

7.3. Как сеть учится: ошибка, веса, эпохи

НА ПАЛЬЦАХ

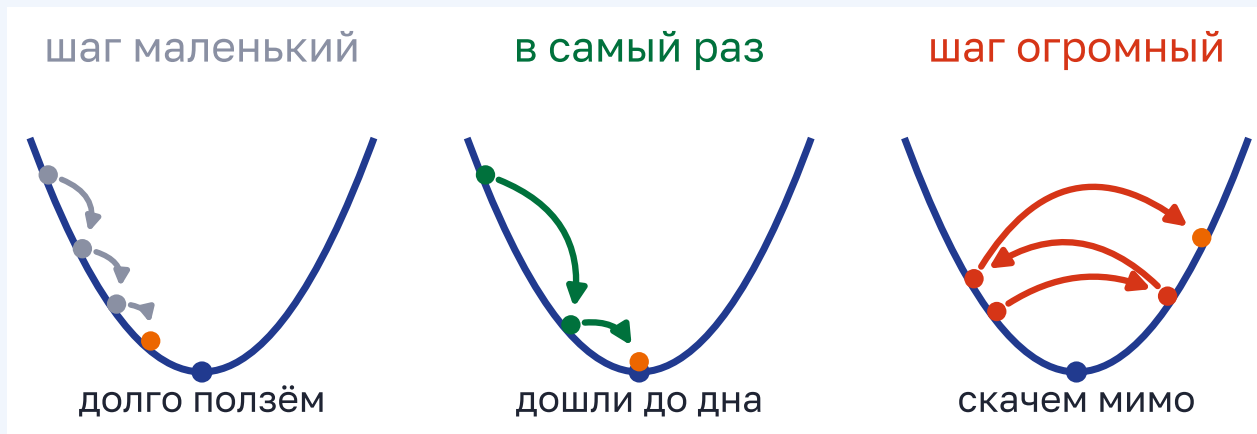


Рисунок 7.5 – Ищем дно ямы, шагая вниз. Высота – ошибка сети, шаг – поправка веса

Маленькими шажками до дна идти долго. Слишком большими – перепрыгиваешь его и скачешь со склона на склон.

Обученная сеть – это сеть с хорошими весами. Сначала веса ставят наугад, и сеть отвечает неважно. Потом ей по одному показывают обучающие примеры и каждый раз чуть-чуть подкручивают веса, чтобы ответ стал ближе к правильному. Никто не подбирает веса вручную – это делает программа, но по понятному правилу.

Разберём его на самой маленькой «сети» – одном нейроне с одним входом, без смещения и без функции активации. Его ответ $y = w \cdot x$. Это **линейная регрессия** – предсказание числа по прямой. С неё начинают и в конструкторе GraphNet (раздел 7.5).

Пример. Дачник поливает грядки и хочет, чтобы модель подсказывала, сколько вёдер воды нужно. Вход x – число грядок, ответ – вёдра. Обучающий пример один: 2 грядки – 6 вёдер. (Ты уже видишь, что на грядку нужно 3 ведра, но модель этого не знает.) Начальный вес $w = 1$.

Ошибка – разность между правильным ответом и ответом модели:

$$E = y_{\text{верный}} - y_{\text{модели}}.$$

Модель ответила $2 \cdot 1 = 2$ ведра, а нужно 6. Ошибка $E = 6 - 2 = 4$. Она положительная – модель ответила мало, вес надо увеличить. Была бы отрицательной – уменьшить.

Правило поправки. Словами: **новый вес равен старому весу плюс скорость обучения, умноженная на ошибку и на вход.** Буквами:

$$w_{\text{нов}} = w + \eta \cdot E \cdot x,$$

где η (греческая буква «эта») – **скорость обучения**, небольшое число, которое выбирает человек. Возьмём $\eta = 0,1$. Поправка $0,1 \cdot 4 \cdot 2 = 0,8$, новый вес $1 + 0,8 = 1,8$.

Почему в правиле стоит вход x ? Чем больше был вход, тем сильнее этот вес повлиял на ответ – тем сильнее его и поправляем. А если вход был 0, вес в ответе вообще не участвовал, и трогать его незачем: $\eta \cdot E \cdot 0 = 0$.

Повторим поправку несколько раз на том же примере:

Таблица 7.1

Шаг	Вес w	Ответ $2w$	Ошибка $E = 6 - 2w$	Поправка $0,1 \cdot E \cdot 2$	Новый вес
1	1	2	4	0,8	1,8
2	1,8	3,6	2,4	0,48	2,28
3	2,28	4,56	1,44	0,288	2,568
4	2,568	5,136	0,864	0,1728	2,7408

Вес ползёт к 3, ошибка с каждым шагом тает: 4; 2,4; 1,44; 0,864. Ровно к 3 вес будет подбираться бесконечно долго, но уже через десяток шагов ошибка станет меньше 0,1 ведра, а этого хватает. Настоящая сеть делает то же самое сразу со всеми своими весами – с тысячами и миллионами.

```

w = 1          # начальный вес
eta = 0.1      # скорость обучения
x, y = 2, 6    # пример: 2 грядки – 6 вёдер

for shag in range(1, 5):
    E = y - w * x          # ошибка
    w = w + eta * E * x    # поправка веса
    print(shag, round(E, 4), round(w, 4))
# 1 4 1.8
# 2 2.4 2.28
# 3 1.44 2.568
# 4 0.864 2.7408

```

Эпоха

Обычно примеров не один, а сотни. Сеть проходит их по очереди и после каждого поправляет веса. **Эпоха** – один полный проход по всем обучающим примерам. «Обучили за 20 эпох» – значит, каждый пример сеть увидела 20 раз. Если в обучающей выборке 500 примеров и поправка идёт после каждого, за 20 эпох будет $500 \cdot 20 = 10\,000$ поправок.

Одной эпохи почти всегда мало: веса за один проход не успевают подобраться, как в нашей таблице после первого шага. Сколько эпох нужно, заранее не знает никто, и число эпох – тоже гиперпараметр.

Скорость обучения

Скорость обучения η решает, насколько большой шаг делает сеть при каждой поправке. На графике – наш пример с грядками при трёх разных скоростях.

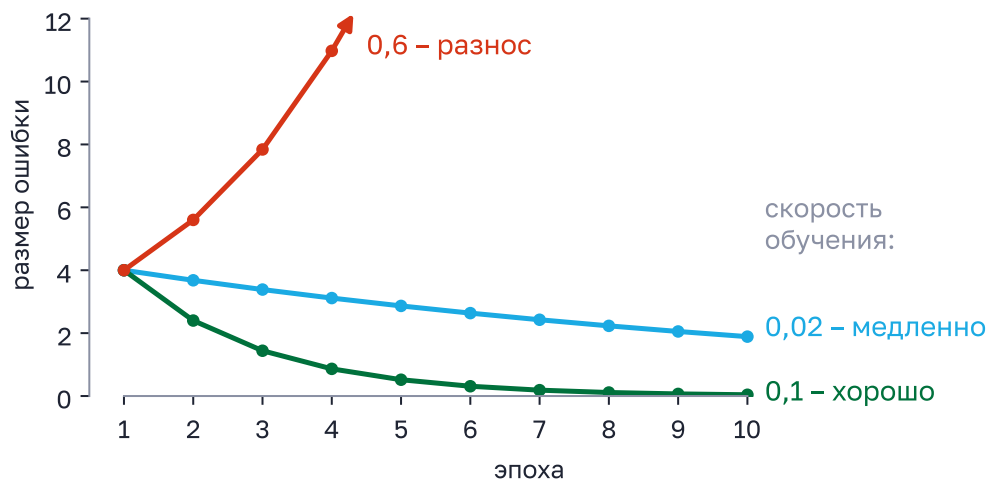


Рисунок 7.6 – Размер ошибки (без знака) от эпохи к эпохе при трёх скоростях обучения. Пример с грядками

- $\eta = 0,02$ – шаги крошечные. Через 10 эпох ошибка всё ещё почти 2 ведра. Учится, но очень медленно.
- $\eta = 0,1$ – ошибка быстро тает до нуля.
- $\eta = 0,6$ – шаг огромный. Вес с 1 прыгает сразу на $1 + 0,6 \cdot 4 \cdot 2 = 5,8$ – далеко за 3. На следующем шаге модель пытается вернуться и снова перелетает, только ещё дальше. Ошибка скачет с плюса на минус и растёт: 4; –5,6; 7,84; –10,98... (минус – модель ответила больше, чем нужно). Сеть «разносит», и ничему она уже не научится.

Отсюда правило: если ошибка от эпохи к эпохе растёт или скачет вверх-вниз, скорость обучения слишком большая – уменьши её. Если ошибка падает, но еле-еле, – можно немного увеличить скорость или добавить эпох.

Для любопытных. В сервисах поправку делают не после каждого примера, а после небольшой **пачки** примеров: ошибки по пачке усредняют и делают один шаг. Размер пачки по-английски – batch size. В Teachable Machine в дополнительных настройках (Advanced) три числа: Epochs – эпохи, Batch Size – размер пачки, Learning Rate – скорость обучения. По умолчанию там 50 эпох, пачка 16 и скорость 0,001.

Запомни. Ошибка = верный ответ – ответ модели. Новый вес = вес + скорость · ошибка · вход. Эпоха – один проход по всем обучающим примерам. Большая скорость – ошибка скачет и растёт, маленькая – учимся медленно.

Совет. Считая поправку вручную, записывай строчку таблицы целиком: вес, ответ, ошибка, поправка, новый вес. Знак ошибки проверяй по смыслу: модель ответила мало – ошибка положительная, вес растёт.

7.4. Переобучение: выучил наизусть, а новое не узнаёт

НА ПАЛЬЦАХ

выучил ответы наизусть

понял, как решать

10 из 10

9 из 10

8 из 10

4 из 10

дома – обучающие
примеры

контрольная –
новые задачи

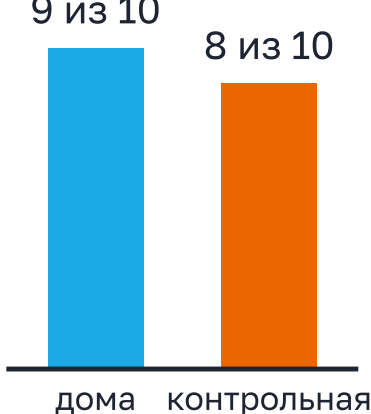
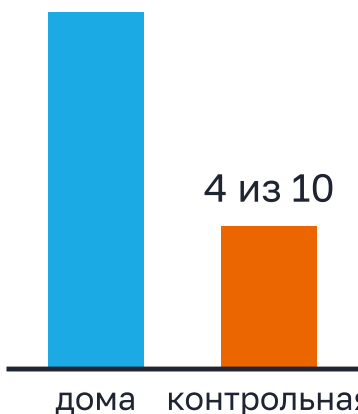


Рисунок 7.7 – Двое готовились к контрольной по одним и тем же десяти задачам

Первый выучил ответы наизусть: дома 10 из 10, а на контрольной с другими числами – 4. Второй понял, как решать, и на новых задачах почти не потерял. Сеть тоже умеет «зубрить».

Вспомни из главы 1: модель учат на обучающей выборке, а проверяют на проверочной – на примерах, которых она не видела. Если следить за точностью на обеих выборках после каждой эпохи, получится два графика.

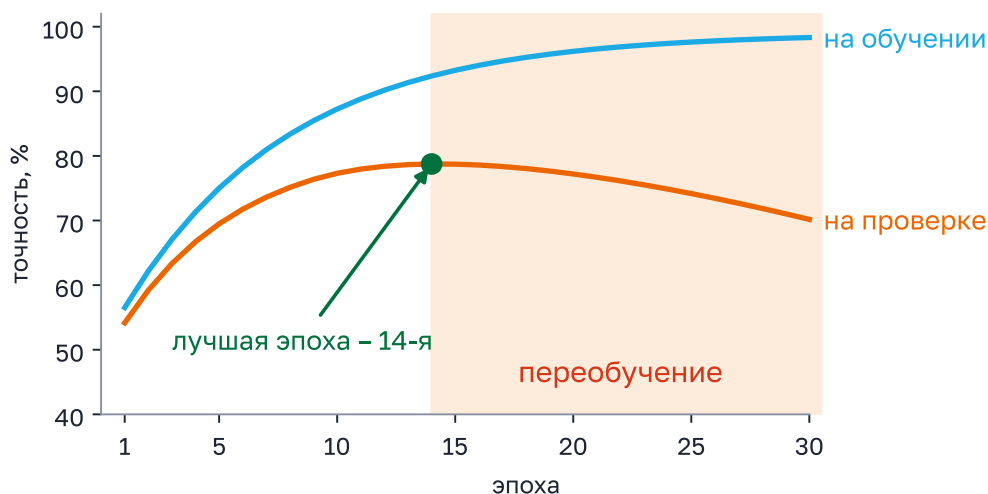


Рисунок 7.8 – Точность на обучающей и на проверочной выборке по эпохам

Сначала обе линии растут: сеть находит настоящие закономерности, и они работают и на новых примерах. Потом голубая линия ползёт дальше вверх, почти к 100 %, а оранжевая останавливается и начинает падать. Сеть перестала учиться общему и начала запоминать мелочи именно этих примеров: пятнышко на одной фотографии, шум на одной записи. На новых данных таких мелочей нет – и точность там падает. Так выглядит **переобучение**. На нашем графике оно начинается после 14-й эпохи.

Главный признак переобучения – **большой разрыв**: на обучении точность высокая, на проверке заметно ниже, и разрыв растёт с каждой эпохой. Бывает и обратная беда – **недообучение**: низкая точность и там, и там. Сеть ещё не выучила даже обучающие примеры: мало эпох, слишком маленькая скорость или слишком простая сеть.

Таблица 7.2

На обучении	На проверке	Что с сетью
высокая	тоже высокая, чуть ниже	всё хорошо
высокая	заметно ниже	переобучилась
низкая	низкая	недообучилась

Что делать, если сеть переобучилась:

- **остановиться вовремя** – взять веса с той эпохи, где точность на проверке была лучшей (у нас – 14-я). Этот приём называют ранней остановкой;
- **добавить примеров**, и разных: сто одинаковых снимков в одной комнате научат меньше, чем тридцать в разных местах;
- **упростить сеть** – меньше скрытых нейронов. Сети с тысячами весов на полсотни примеров слишком легко запомнить каждый пример наизусть.

Разница двух точностей считается простым вычитанием, но будь внимателен со словами. Было 99 % и 72 %, разница – 27 **процентных пунктов**: так говорят, когда из процентов вычи-

тают проценты. В задачах часто пишут проще: «на сколько процентов точность на обучении выше» – и ждут то же число 27.

Запомни. Переобучение – на обучении точность растёт, а на проверке падает. Лучшая эпоха – та, где **выше всего точность на проверке**, а не на обучении. Низкая точность на обеих выборках – недообучение.

Совет. В таблице точности по эпохам смотри только на «проверку» и найди там максимум. «Обучение» используй, лишь чтобы увидеть разрыв. Самая частая ошибка – выбрать эпоху, где точность на обучении 99 %.

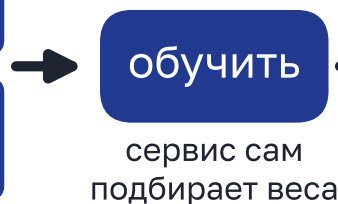
7.5. Нейросеть без программирования

НА ПАЛЬЦАХ

1. Собрал примеры



2. Обучил



3. Проверил

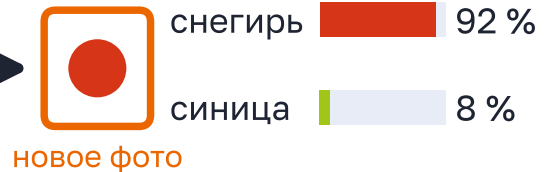


Рисунок 7.9 – Три шага в любом low-code сервисе

Разложил примеры по коробкам-классам, нажал кнопку – сервис сам подобрал веса. Показал новое фото – сеть ответила процентами: насколько она уверена в каждом классе.

Low-code (читается «лоу-код», по-английски «мало кода») – сервисы, где модель собирают мышкой, почти без программирования. Внутри у них те же слои, веса и эпохи, что в этой главе, просто считает всё программа.

Teachable Machine (teachablemachine.withgoogle.com) – сервис в браузере. Есть проекты для картинок, для звуков и для поз человека. Работа идёт так:

1. Выбираешь проект, для картинок – «Standard image model». Создаёшь **классы** – те самые коробки с подписями. Сначала они называются «Class 1», «Class 2», их можно переименовать и добавить новые.
2. В каждый класс добавляешь **примеры**: снимаешь с веб-камеры или загружаешь файлы (Upload). В звуковом проекте сервис дополнительно просит записать класс «фоновый шум» (Background Noise), чтобы сеть отличала тишину от звуков. Примеров звука нужно не меньше 20 на класс.

3. Нажимаешь «Train Model» – «обучить модель». В дополнительных настройках можно поменять число эпох, размер пачки и скорость обучения (раздел 7.3).
4. **Проверяешь** в окне Preview: показываешь камере новый предмет, и сеть рисует полоски с процентами для каждого класса. Проценты в сумме дают 100, ответ – класс с самой длинной полоской.
5. Если модель устраивает, её можно выгрузить (Export Model) и вставить в свою программу.

Machine Learning for Kids (machinelearningforkids.co.uk) умеет учить модели распознавать текст, картинки, числа и звуки. В проекте три шага: «Train» – собрать примеры, «Learn & Test» – обучить модель и проверить её, «Make» – вставить её в свою программу на Scratch. Scratch там свой, встроенный в сайт: блоки для модели ищи в нём, на scratch.mit.edu их нет. Перед началом работы сайт может попросить подтвердить, что ты не робот; не пустил – попробуй ещё раз или с другого интернета. Автор сервиса – Дейл Лейн, тот самый, что написал книгу «Машинное обучение для детей» из списка литературы программы сферы.

Оба сервиса зарубежные и могут открываться не везде. Если не открылся – не беда: порядок работы у всех таких сервисов один, кнопки только называются по-разному.

GraphNet – конструктор нейросетей, который используют на сфере. Он ставится на компьютер, а сеть в нём собирают из **блоков**, соединяя их стрелками. Первые блоки загружают данные, дальше настраивают слои, функции активации и гиперпараметры. Обученная сеть делает прогноз на проверочных данных, программа сравнивает его с настоящими ответами и считает метрики точности. Начинают там с архитектуры для линейной регрессии: это нейрон $y = w \cdot x$ из раздела 7.3, только со смещением – $y = w \cdot x + b$.

Сколько примеров нужно

Точного числа нет. Для первых опытов с картинками обычно хватает нескольких десятков примеров на класс, лучше – сотни. Важнее количества **разнообразие**: снимай предмет при разном свете, на разном фоне, с разных сторон, а голос записывай у разных людей. Сеть выучит только то, что ты ей показал. Показал кружку только сбоку – сверху она её не узнает.

И проверяй модель на **новых** примерах, которых не было в обучении. Проверка на тех же фото, что ушли в классы, ничего не скажет: их сеть могла просто запомнить.

Почему классы нельзя сильно перекашивать

Допустим, в классе «синица» 180 фото, а в классе «снегирь» – 20. Сеть быстро замечает, что «синица» почти всегда правильный ответ. Ленивая модель, которая про любую птицу говорит «синица», угадает 180 фото из 200: $180/200 \cdot 100 = 90\%$. Точность выглядит прилично, а снегиря такая модель не узнает ни одного. Это та же ловушка редких случаев, что в главе 1.

Такой перекош называют **дисбалансом классов**, а выборку, где классы сопоставимы, – **сбалансированной**. Поровну быть не обязано: 500 и 500 – хороший баланс, 900 и 100 – сильный дисбаланс. Главное, чтобы примеров каждого класса было достаточно и они были примерно одного порядка. Так же, если в таблице из 100 записей какой-то класс встречается всего 5 раз, модель почти не увидит его. Если снегирей мало, добери снимков снегирей. Если добрать негде, убери часть лишних синиц: 20 и 20 лучше, чем 180 и 20, хотя примеров и меньше. В

курсе вред дисбаланса называют «переобучением из-за дисбаланса»; не путай с переобучением из раздела 7.4: там модель хороша на обучении и плоха на проверке.

Ловушка фона. Сеть не знает, на что ты хотел, чтобы она смотрела. Если все снегири сняты зимой на снегу, а все синицы – летом на зелёной ветке, сеть легко научится отличать снег от листвы. На проверке с теми же условиями точность будет отличной, а синицу на снегу она назовёт снегирём. Лечится разнообразием: у каждого класса должен быть разный фон.

Запомни. Low-code: собрал примеры по классам → обучил → проверил на новых примерах. Классы – сопоставимо, без сильного перекоса, примеры – разные. Ответ сети – класс с наибольшей уверенностью.

Совет. Перед обучением посчитай примеры в каждом классе и посмотри, чем, кроме нужного признака, классы отличаются друг от друга: фоном, светом, размером картинки. Всё, чем они отличаются, сеть может принять за подсказку.

7.6. Разбираем задачи

Задача 7.1 Сколько связей в сети

разминка · полносвязные слои, число связей

Сеть по 6 признакам поездки (расстояние, погода, день недели и так далее) советует, чем добраться из Новосибирска до Академгородка: автобусом, электричкой или на машине. В сети один скрытый слой из 4 нейронов, все соседние слои полносвязные. Сколько связей (весов) в этой сети? Смещения не считай.

Дано: сеть 6–4–3, слои полносвязные.

Найти: N – число связей.

Решение

Шаг 1. Выписываем слои. 6 признаков – 6 входов. Скрытый слой – 4 нейрона. Способов доехать 3 – значит, 3 выхода, по одному на класс. Сеть 6–4–3.

Шаг 2. Связи между входами и скрытым слоем. Каждый из 6 входов соединён с каждым из 4 нейронов: $6 \cdot 4 = 24$ связи.

Шаг 3. Связи между скрытым слоем и выходами. $4 \cdot 3 = 12$ связей.

Шаг 4. Всего. $N = 24 + 12 = 36$.

Прикинем. Нейронов в сети всего $6 + 4 + 3 = 13$, а связей почти втрое больше: каждый нейрон тянет стрелки сразу к нескольким соседям. Если бы спросили параметры вместе со смещениями, добавилось бы $4 + 3 = 7$ смещений – вышло бы 43.

❗ **Типичная ошибка.** Перемножить все слои подряд: $6 \cdot 4 \cdot 3 = 72$. Слои соединены парами, поэтому произведения соседей складывают. Бывает и так: посчитали нейроны ($6 + 4 + 3 = 13$), а спрашивали связи.

Ответ: 36.

Своя задача по программе сферы, модуль 5

Реши сам

7.1. Сеть 3–5–2, все соседние слои полносвязные. Сколько в ней связей (без смещений)?

7.2. Сеть 8–6–6–1 (два скрытых слоя по 6 нейронов), слои полносвязные. а) Сколько в ней связей? б) Сколько всего параметров, если к связям добавить смещения? Ответ – два числа через точку с запятой.

7.3.* а) Сеть 10–?–4 с одним скрытым слоем, слои полносвязные, всего 196 связей. Сколько нейронов в скрытом слое? б) В сети 10–?–?–4 два скрытых слоя с одинаковым числом нейронов, всего 312 связей. Сколько нейронов в каждом скрытом слое? (Подсказка: подставляй по очереди 10, 11, 12... и считай связи, пока не получится 312.)

Задача 7.2 Сколько литров налить в грядку

выход нейросети, смещение, ReLU

Нейросеть на даче подсказывает, сколько литров воды налить в грядку. Входы: x_1 – сколько дней не было дождя, x_2 – температура днём, делённая на 10 (так сети удобнее: числа на входах одного размера). В скрытом слое два нейрона с функцией ReLU, у выхода – без изменений.

Таблица 7.3

Нейрон	Вес от x_1	Вес от x_2	Смещение
h_1	2	1	–3
h_2	–1	2	–4

Выход: вес от h_1 равен 2, вес от h_2 равен 4, смещение 1. Сколько литров посоветует сеть, если дождя не было 3 дня, а днём было 25 °C?

Дано: $x_1 = 3$; $x_2 = 25/10 = 2,5$; веса и смещения – в таблице; скрытый слой – ReLU.

Найти: y – ответ сети, л.

Решение

Шаг 1. Готовим входы. $x_1 = 3$. Температуру по условию делим на 10: $x_2 = 25/10 = 2,5$.

Шаг 2. Нейрон h_1 . $S = 3 \cdot 2 + 2,5 \cdot 1 + (-3) = 6 + 2,5 - 3 = 5,5$. Сумма положительная, ReLU её не меняет: $h_1 = 5,5$.

Шаг 3. Нейрон h_2 . $S = 3 \cdot (-1) + 2,5 \cdot 2 + (-4) = -3 + 5 - 4 = -2$. Сумма отрицательная, ReLU даёт ноль: $h_2 = 0$.

Шаг 4. Выход. Входы выходного нейрона – ответы скрытого слоя, 5,5 и 0:
 $y = 5,5 \cdot 2 + 0 \cdot 4 + 1 = 11 + 0 + 1 = 12$. Функция «без изменений» – ответ 12 л.

Проверка кодом.

```
relu = lambda s: max(0, s)    # короткая запись функции: «минус – в ноль»  
  
x1, x2 = 3, 25 / 10  
h1 = relu(x1 * 2 + x2 * 1 - 3)  
h2 = relu(x1 * (-1) + x2 * 2 - 4)  
y = h1 * 2 + h2 * 4 + 1  
print(h1, h2, y)            # 5.5 0 12.0
```

Прикинь по жизни: 12 литров – чуть больше ведра на грядку после трёх сухих жарких дней. Дачник примерно столько и нальёт.

❗ **Где теряют баллы.** Забыли ReLU и оставили $h_2 = -2$ – выход $5,5 \cdot 2 + (-2) \cdot 4 + 1 = 4$ л, в три раза меньше верного. И ещё: подали на вход 25 вместо 2,5, хотя в условии прямо сказано, что температуру делят на 10.

Ответ: 12 л.

Своя задача по программе сферы, модуль 5

Реши сам

7.4. Суммы трёх нейронов скрытого слоя (уже со смещением) равны -4; 0; 2,5. Функция активации – ReLU. Запиши ответы нейронов через точку с запятой.

7.5. Сеть 2–2–1. Входы $x_1 = 4$, $x_2 = 2$. Нейрон h_1 : веса 1 (от x_1) и -1 (от x_2), смещение 0. Нейрон h_2 : веса -2 и 1, смещение 3. В скрытом слое ReLU. Выход: вес от h_1 равен 3, от h_2 равен 5, смещение -1, функция без изменений. Найди ответ сети.

Задача 7.3 Когда остановить обучение

переобучение, точность на обучении и проверке

Школьники из Академгородка учат сеть различать на снимках неба три вида облаков. Каждые 5 эпох они записывали точность на обучающей и на проверочной выборке:

Таблица 7.4

Эпоха	Точность на обучении, %	Точность на проверке, %
5	64	61
10	73	70
15	81	77
20	87	82

Эпоха	Точность на обучении, %	Точность на проверке, %
25	92	83
30	96	80
35	98	76
40	99	72

- а) Веса после какой эпохи лучше сохранить, чтобы сеть хорошо работала на новых снимках?
 б) На сколько процентов точность на обучении выше точности на проверке после 40-й эпохи?
 Ответ запиши двумя числами через точку с запятой.

Дано: таблица точности по эпохам.

Найти: лучшую эпоху; разрыв точностей на 40-й эпохе.

Решение

Шаг 1. Какой столбец важен. Работать сеть будет на новых снимках, а «новые» – это проверочная выборка. Значит, лучшую эпоху ищем по столбцу «на проверке».

Шаг 2. Ищем максимум на проверке. 61, 70, 77, 82, **83**, 80, 76, 72. Самое большое число – 83 %, эпоха 25.

Шаг 3. Что после. После 25-й эпохи точность на обучении растёт (92 → 96 → 98 → 99), а на проверке падает (83 → 80 → 76 → 72). Это переобучение: сеть начала запоминать снимки обучающей выборки.

Шаг 4. Разрыв на 40-й эпохе. $99 - 72 = 27$ процентных пунктов. Для сравнения, на 25-й эпохе разрыв был $92 - 83 = 9$ – втрое меньше.

Проверка кодом.

```
epochi = [5, 10, 15, 20, 25, 30, 35, 40]
obuch = [64, 73, 81, 87, 92, 96, 98, 99]
prov = [61, 70, 77, 82, 83, 80, 76, 72]

i = prov.index(max(prov))      # номер места, где проверка лучше всего
print(epochi[i], max(prov))    # 25 83
print(obuch[-1] - prov[-1])    # 27
```

83 % для трёх видов облаков – совсем неплохо: если тыкать наугад, выйдет около 33 %. А вот 99 % на обучении при 72 % на проверке – это уже зубрёжка.

⚠ Типичная ошибка. Выбрать эпоху 40, потому что там «99 % – самая высокая точность». Это точность на примерах, которые сеть уже видела. Про новые снимки говорит только правый столбец.

Ответ: 25; 27.

Своя задача по программе сферы, модули 1 и 5

Реши сам

7.6. Точность модели, которая узнаёт по голосу, кто из четырёх ребят говорит:

Таблица 7.5

Эпоха	На обучении, %	На проверке, %
10	55	53
20	68	65
30	78	74
40	85	79
50	90	78
60	94	75
70	97	71
80	99	68

а) Веса после какой эпохи лучше сохранить? б) На сколько процентов точность на обучении выше, чем на проверке, после 80-й эпохи? Ответ – два числа через точку с запятой.

7.7. Верно (В) или неверно (Н)? 1) Эпоха – один проход по всем обучающим примерам. 2) Чем больше скорость обучения, тем лучше: сеть обучится и быстрее, и точнее. 3) Если точность на обучении 99 %, а на проверке 60 %, сеть переобучилась. 4) У сети, которая относит объект к одному из 4 классов, обычно 4 выхода. 5) Между полносвязными слоями из 5 и 3 нейронов 8 связей. Ответ – пять букв подряд.

Задача 7.4 Подбери сеть под задачу

архитектура сети, число входов и выходов

Есть три задачи:

1. По 5 признакам погоды (температура, давление, ветер, влажность, облачность) предсказать температуру завтра в градусах.
2. По чёрно-белой картинке 10 × 10 пикселей узнать, что на ней нарисовано: круг, крест или треугольник.
3. По тем же 5 признакам погоды определить, что будет завтра: снег, дождь или без осадков.

И четыре сети:

- А) 5–8–1, выход без изменений;
- Б) 100–16–3;
- В) 5–8–3;
- Г) 3–16–100.

Подбери к каждой задаче сеть. Ответ запиши тремя буквами для задач 1, 2, 3.

Дано: три задачи и четыре сети.

Найти: сеть для каждой задачи.

Решение

Шаг 1. Задача 1. Признаков 5 – входов 5. Ответ – число (температура), это регрессия: один выход, число берут как есть. Подходит **А** (5–8–1, выход без изменений).

Шаг 2. Задача 2. Каждый пиксель – вход: $10 \cdot 10 = 100$ входов. Классов три – выходов 3. Это **Б** (100–16–3).

Шаг 3. Задача 3. Входов снова 5, но ответ теперь – один из трёх классов: 3 выхода. Это **В** (5–8–3).

Шаг 4. Лишняя сеть. Г (3–16–100) не подходит никому: у неё 3 входа и 100 выходов – как будто картинку перепутали местами с ответом.

Заметь: у задач 1 и 3 входы одинаковые, а сети разные. Входы задают признаки, а выходы – вопрос, который мы задаём сети.

 **Типичная ошибка.** Взять для картинки 10 входов («картинка 10 на 10») вместо $10 \cdot 10 = 100$. Ещё легко выбрать для задачи 3 сеть А: погода та же, но ответ теперь не число, а класс из трёх.

Ответ: АБВ.

Своя задача по программе сферы, модули 1 и 5

Реши сам

7.8. Сеть узнаёт по чёрно-белой картинке 12×12 пикселей одну из 5 букв. В ней один скрытый слой из 20 нейронов, слои полносвязные. Сколько у сети входов, выходов и связей? Ответ – три числа через точку с запятой.

7.9. Есть три задачи: 1) по 4 признакам яблока (диаметр, цвет в баллах, твёрдость, сколько дней лежит на складе) предсказать его массу в граммах; 2) по тем же 4 признакам определить, какой из 5 сортов перед нами; 3) по чёрно-белой картинке 8×8 пикселей узнать, какая цифра на ней нарисована: 0 или 1. И четыре сети: А) 64–12–2; Б) 4–10–5; В) 4–10–1, выход без изменений; Г) 8–12–2. Подбери к каждой задаче сеть. Ответ – три буквы для задач 1, 2, 3.

Задача 7.5 Синицы и снегири у кормушки

баланс классов, ленивая модель, low-code

Ваня всю зиму фотографировал птиц у кормушки на даче и собрал в Teachable Machine два класса: «синица» – 180 фото и «снегирь» – 20 фото. Модель на всех фото отвечает «синица».

а) Какой будет точность такой модели в процентах, если проверять её на фото с тем же соотношением птиц? б) Сколько фото снегирей надо добавить, чтобы классы стали поровну? Ответ запиши двумя числами через точку с запятой.

Дано: синиц 180, снегирей 20; модель всегда отвечает «синица».

Найти: точность, %; сколько добавить снегирей.

Решение

Шаг 1. Сколько всего фото. $180 + 20 = 200$.

Шаг 2. Сколько раз модель права. Она всегда говорит «синица» и права ровно тогда, когда на фото синица: 180 раз из 200.

Шаг 3. Точность. $180/200 \cdot 100 = 0,9 \cdot 100 = 90 \%$.

Шаг 4. Сколько добавить. Чтобы снегирей стало столько же, сколько синиц: $180 - 20 = 160$ фото.

Проверка кодом.

```
sinicy, snegiri = 180, 20
print(sinicy / (sinicy + snegiri) * 100)    # 90.0
print(sinicy - snegiri)                    # 160
```

Вот где подвох. 90 % звучит хорошо, а ни одного снегиря модель не узнала. Виновата тут не сеть, а данные: классы сильно разные по размеру.

❗ **Типичная ошибка.** Поверить цифре 90 % и решить, что модель хорошая. Когда классы сильно неравные, смотри на редкий: сколько снегирей нашла модель? Ноль. А в пункте б) часто пишут 180 – но спрашивали, сколько **добавить**, а не сколько должно стать.

Ответ: 90; 160.

Своя задача по программе сферы, модуль 9

Реши сам

7.10. В проекте Teachable Machine три класса: «кошка» – 150 фото, «собака» – 90, «хомяк» – 60. а) Сколько фото всего нужно добавить в меньшие классы, чтобы во всех трёх стало по 150? б) Модель на любое фото отвечает «кошка». Какой будет её точность в процентах на этих 300 фото? Ответ – два числа через точку с запятой.

7.11. Настя обучила модель отличать лис от собак. Все фото лис она нашла зимние, на снегу, а все фото собак – летние, на траве. На проверочных фото, выбранных так же, точность 97 %, а на фото собаки в сугробе модель уверенно отвечает «лиса». Какая причина вероятнее всего? А) мало эпох; Б) модель научилась отличать снег от травы, а не лису от собаки; В) слишком маленькая скорость обучения; Г) слишком много классов. Ответ – одна буква.

7.12. Расставь шаги работы в Teachable Machine по порядку: А) проверить модель на новых снимках, которых не было в примерах; Б) создать классы и назвать их; В) нажать «Train Model» и дождаться конца обучения; Г) снять или загрузить примеры для каждого класса; Д) выгрузить готовую модель, чтобы вставить её в свою программу. Ответ – пять букв подряд.

Задача 7.6* Модель для школьного автобуса

обучение, поправка веса, эпохи, скорость обучения, линейная регрессия

Модель предсказывает, сколько минут школьный автобус едет по маршруту: $t = w \cdot s$, где s – длина маршрута в километрах, w – вес. Обучающие примеры: 2 км – 6 мин, 4 км – 12 мин. Начальный вес $w = 1$, скорость обучения $\eta = 0,05$. После каждого примера вес поправляют по правилу $w_{\text{нов}} = w + \eta \cdot E \cdot s$, где E – ошибка (верное время минус ответ модели). В каждой эпохе примеры идут по порядку: сначала 2 км, потом 4 км. Каким станет вес после двух эпох? Округли до сотых. Калькулятор можно.

Дано: примеры (2; 6) и (4; 12); $w = 1$; $\eta = 0,05$; 2 эпохи.

Найти: w после 2 эпох.

Решение

Шаг 1. Сколько поправок. Эпоха – проход по всем примерам, их два. За две эпохи – $2 \cdot 2 = 4$ поправки. Знаков после запятой будет много, поэтому считай в столбик или на калькуляторе и после каждой поправки записывай строчку: ответ, ошибка, поправка, новый вес. Одна описка в середине – и все следующие строчки поедут.

Шаг 2. Эпоха 1, пример 2 км. Ответ модели $1 \cdot 2 = 2$ мин. Ошибка $E = 6 - 2 = 4$. Поправка $0,05 \cdot 4 \cdot 2 = 0,4$. Новый вес $1 + 0,4 = 1,4$.

Шаг 3. Эпоха 1, пример 4 км. Берём уже новый вес 1,4. Ответ $1,4 \cdot 4 = 5,6$ мин. Ошибка $12 - 5,6 = 6,4$. Поправка $0,05 \cdot 6,4 \cdot 4 = 1,28$. Вес $1,4 + 1,28 = 2,68$.

Шаг 4. Эпоха 2, пример 2 км. Ответ $2,68 \cdot 2 = 5,36$. Ошибка $6 - 5,36 = 0,64$. Поправка $0,05 \cdot 0,64 \cdot 2 = 0,064$. Вес $2,68 + 0,064 = 2,744$.

Шаг 5. Эпоха 2, пример 4 км. Ответ $2,744 \cdot 4 = 10,976$. Ошибка $12 - 10,976 = 1,024$. Поправка $0,05 \cdot 1,024 \cdot 4 = 0,2048$. Вес $2,744 + 0,2048 = 2,9488 \approx 2,95$.

Проверка кодом.

```
w, eta = 1, 0.05
primery = [(2, 6), (4, 12)]      # (км, минуты)

for epocha in range(1, 3):
    for s, t in primery:
        E = t - w * s
        w = w + eta * E * s
        print(epocha, s, round(w, 4))
print(round(w, 2))
# 1 2 1.4
# 1 4 2.68
# 2 2 2.744
# 2 4 2.9488
# 2.95
```

Вес подобрался почти к 3 – и правильно: в обоих примерах автобус тратит 3 минуты на километр. Это 20 км/ч, для школьного автобуса с остановками обычное дело.

А если взять скорость обучения побольше, $\eta = 0,2$? Уже первый пример даёт вес $1 + 0,2 \cdot 4 \cdot 2 = 2,6$, а второй – $2,6 + 0,2 \cdot (12 - 10,4) \cdot 4 = 3,88$. Вес проскочил 3 и дальше

будет качаться вокруг него. Так на числах выглядит слишком большой шаг.

❗ **Главная ловушка.** Считать оба примера одной эпохи со старым весом 1: поправки 0,4 и 1,6, после первой эпохи вес 3, и ответ 3,00 – красивый, но неверный. Вес меняется **после каждого** примера, и следующий пример считают уже с новым весом. А кто-то делает всего две поправки: «две эпохи – два шага».

Ответ: 2,95.

Своя задача по программе сферы, модуль 5

Реши сам

7.13. В обучающей выборке 300 примеров. Сеть обучали 20 эпох и поправляла веса после каждого примера. Сколько раз она поправляла веса?

7.14. Модель $y = w \cdot x$, вес $w = 2$, скорость обучения $\eta = 0,1$. Обучающий пример: $x = 3$, верный ответ $y = 12$. Сделай одну поправку по правилу $w_{\text{нов}} = w + \eta \cdot E \cdot x$. Каким станет вес?

7.15.* Модель $y = w \cdot x$ учат на двух примерах: $x = 2, y = 6$ и $x = 5, y = 15$ – в каждой эпохе по порядку. Начальный вес $w = 1$, скорость обучения $\eta = 0,05$, вес поправляют после каждого примера по правилу $w_{\text{нов}} = w + \eta \cdot E \cdot x$. Каким станет вес после двух эпох? Будь внимателен: во второй эпохе ошибка может оказаться отрицательной.

Задача 7.7* Два выключателя в коридоре

выход нейросети, ступенька, зачем скрытый слой

В длинном коридоре лампу можно включать и выключать двумя выключателями – у одного конца и у другого. Лампой управляет маленькая нейросеть. Входы A и B : 1 – выключатель поднят, 0 – опущен. В скрытом слое два нейрона-ступеньки, у выхода тоже ступенька. Все нейроны отвечают 1, если сумма не меньше порога T , иначе 0.

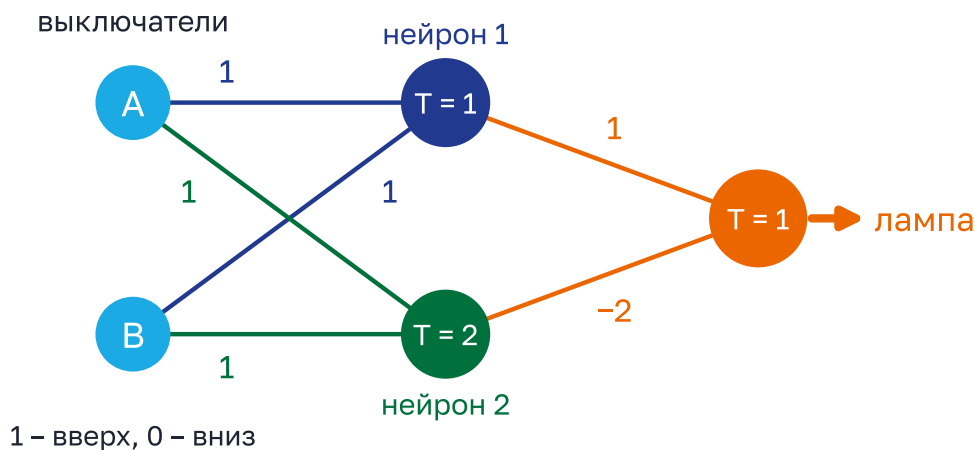


Рисунок 7.10 – Сеть для лампы. Числа на связях – веса, в кружках – пороги

- Нейрон 1: веса от A и от B равны 1, порог $T = 1$.
- Нейрон 2: веса от A и от B равны 1, порог $T = 2$.
- Выход: вес от нейрона 1 равен 1, вес от нейрона 2 равен -2 , порог $T = 1$. Выход 1 – лампа горит.

Горит ли лампа при положениях выключателей $(A; B) = (0; 0), (0; 1), (1; 0), (1; 1)$? Ответ запиши четырьмя цифрами подряд в этом порядке.

Дано: сеть 2–2–1, ступеньки с порогами 1, 2 и 1; веса на рисунке.

Найти: ответ сети для четырёх наборов входов.

Решение

Шаг 1. Что делает нейрон 1. Сумма $A + B$, порог 1. Он отвечает 1, если поднят **хотя бы один** выключатель.

Шаг 2. Что делает нейрон 2. Та же сумма $A + B$, порог 2. Он отвечает 1, только если подняты **оба**.

Шаг 3. Считаем все четыре случая. Выход складывает $1 \cdot h_1 + (-2) \cdot h_2$ и сравнивает с порогом 1.

Таблица 7.6

$(A; B)$	$A + B$	h_1	h_2	Сумма выхода	Лампа
(0; 0)	0	0	0	$0 + 0 = 0$	0
(0; 1)	1	1	0	$1 + 0 = 1$	1
(1; 0)	1	1	0	$1 + 0 = 1$	1
(1; 1)	2	1	1	$1 + (-2) = -1$	0

Шаг 4. Что получилось. Лампа горит, когда выключатели стоят **по-разному**. Щёлкнул любым – положение меняется, и лампа переключается. Так устроены выключатели в любом длинном коридоре или на лестнице.

Проверка кодом.

```
def stupenka(s, T):
    return 1 if s ≥ T else 0

for A, B in [(0, 0), (0, 1), (1, 0), (1, 1)]:
    h1 = stupenka(A + B, 1)
    h2 = stupenka(A + B, 2)
    lampa = stupenka(1 * h1 + (-2) * h2, 1)
    print(A, B, lampa)

# 0 0 0
# 0 1 1
# 1 0 1
# 1 1 0
```

Почему так вышло. Выход «сердится» на нейрон 2 (вес -2): когда подняты оба выключателя, он гасит лампу, хотя нейрон 1 за то, чтобы гореть. Так сеть из трёх нейронов собирает правило «ровно один из двух».

❗ **Типичная ошибка.** Подать на выход исходные A и B вместо ответов нейронов 1 и 2. А кто потеряет минус у веса -2 , получит при $(1; 1)$ сумму $1 + 2 = 3$, горящую лампу и ответ 0111.

Ответ: 0110.

Своя задача по программе сферы, модули 1 и 5

Для любопытных. Одному нейрону-ступеньке такое правило не под силу – вот почему нужен скрытый слой. Пусть веса w_1 , w_2 и порог T . При $(0; 0)$ лампа не горит – значит, $0 < T$. При $(1; 0)$ горит – $w_1 \geq T$; при $(0; 1)$ – $w_2 \geq T$. Сложим: $w_1 + w_2 \geq 2T$, а раз T больше нуля, то $2T > T$, и при $(1; 1)$ сумма $w_1 + w_2$ тоже не меньше порога – лампа загорится. А должна погаснуть. Противоречие: ни одного подходящего набора весов нет.

Реши сам

7.16. Сеть 2–2–1 из нейронов-ступенек (ответ 1, если сумма не меньше порога T , иначе 0). Нейрон 1: вес от A равен 2, от B равен -1 , порог $T = 1$. Нейрон 2: вес от A равен -1 , от B равен 2, порог $T = 1$. Выход: веса от нейронов 1 и 2 равны 1, порог $T = 1$. Что ответит сеть на входы $(A; B) = (0; 0), (0; 1), (1; 0), (1; 1)$? Ответ – четыре цифры подряд в этом порядке.

7.17.* Игрушечным светофором управляют две кнопки A и B (1 – нажата, 0 – нет), зелёный горит, когда сеть отвечает 1. Сеть 2–2–1 из ступенек (ответ 1, если сумма не меньше порога T). Нейрон 1: веса от A и B равны 1, порог $T = 1$. Нейрон 2: веса от A и B равны 1, порог $T = 2$. Выход: вес от нейрона 1 равен -1 , от нейрона 2 равен 2, порог $T = 0$. Что ответит сеть на входы $(A; B) = (0; 0), (0; 1), (1; 0), (1; 1)$? Ответ – четыре цифры подряд в этом порядке.

Python, диалоги и промпты



Эта глава – про разговор с компьютером. Сначала на языке Python: переменные, условия, циклы, списки и строки. Типичная олимпиадная задача по программированию – короткий код и вопрос «что он напечатает?» или «где здесь ошибка?». Потом посмотрим, как устроен разговор чат-бота: его рисуют схемой из реплик и стрелок и считают по ней ветки. А в конце научимся просить нейросеть так, чтобы она поняла с первого раза, и проверять код, который она за нас написала.

8.1. Первая программа: переменные, ввод и вывод

НА ПАЛЬЦАХ

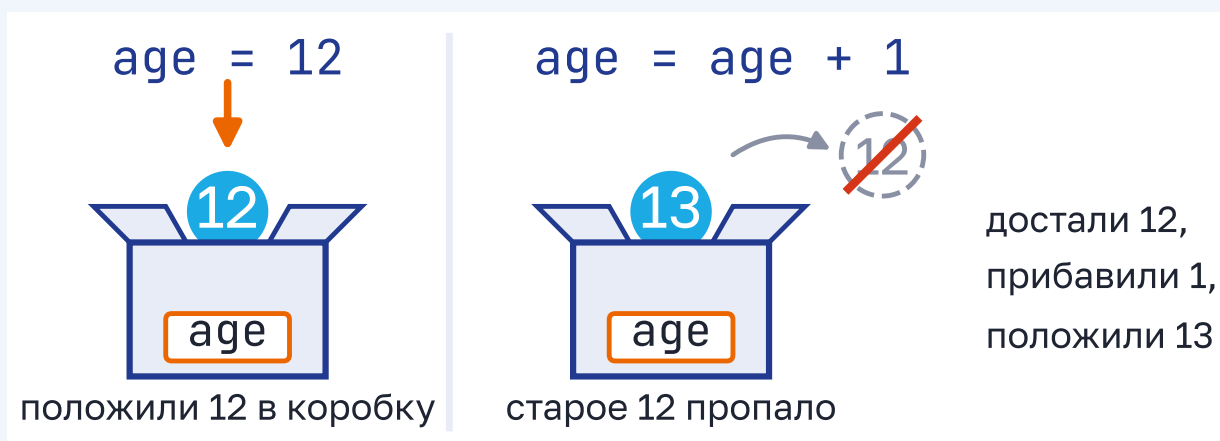


Рисунок 8.1 – Переменная – коробка с наклейкой-именем

Знак `=` кладёт значение в коробку. Положили новое – старое пропало, в коробке всегда лежит что-то одно.

Программа – список команд для компьютера. Python выполняет их по очереди: сверху вниз, строчку за строчкой. Писать и запускать код можно в любом онлайн-интерпретаторе Python 3 (набери в поиске «Python онлайн») или в Google Colab. **Интерпретатор** – программа, которая читает твой код и сразу его выполняет. Ставить ничего не нужно.

Вот первая программа. Набери её сам – так запоминается лучше, чем глазами.

```
name = input("Как тебя зовут? ")
print("Привет,", name)
age = int(input("Сколько тебе лет? "))
print("Через 5 лет тебе будет", age + 5)
```

Запускаем, отвечаем на вопросы «Аня» и «12» – на экране получится разговор:

```
Как тебя зовут? Аня
Привет, Аня
Сколько тебе лет? 12
Через 5 лет тебе будет 17
```

Разберём каждую строку.

- **Строка 1.** Команда `input("Как тебя зовут? ")` показывает вопрос и ждёт, пока человек напечатает ответ и нажмёт Enter. Напечатанное попадает в **переменную** `name` – ячейку памяти с именем. Знак `=` читается «положи в»: положи ответ в коробку `name`. Это не равенство из математики, а команда.
- **Строка 2.** `print` выводит на экран то, что стоит в скобках. Через запятую можно перечислить несколько вещей – Python поставит между ними пробел. Текст в кавычках печатается как есть. А `name` без кавычек – имя коробки, и печатается то, что в ней лежит: Аня.
- **Строка 3.** Здесь две команды одна в другой. `input` всегда отдаёт **текст**, даже если набрали цифры: «12» для него – две буквы-цифры, а не число. `int(...)` превращает текст «12» в число 12. Скобки выполняются изнутри наружу: сначала `input`, потом `int`, потом результат кладём в `age`.
- **Строка 4.** Python сначала считает `age + 5`, то есть $12 + 5 = 17$, и только потом печатает.

Имя переменной пишут латинскими буквами, цифрами и знаком `_`, без пробелов, и начинать его с цифры нельзя. Большие и маленькие буквы различаются: `Age` и `age` – две разные коробки. В этой книге имена – русские слова латиницей: `sneg` (снег), `dni` (дни). Так сразу видно, что в коробке.

У каждого значения есть **тип** – какого оно рода. Главных типов четыре:

Таблица 8.1

Тип	Что это	Как записать	Пример
<code>int</code>	целое число	без кавычек	<code>12</code> , <code>-30</code>
<code>float</code>	дробное число	через точку	<code>2.5</code> , <code>-0.7</code>
<code>str</code>	строка, то есть текст	в кавычках	<code>"Аня"</code> , <code>"12"</code>

Тип	Что это	Как записать	Пример
bool	«да» или «нет»	True или False	результат сравнения <code>5 > 3</code> – это True

От типа зависит, что делает знак `+`. Числа он складывает: `12 + 5` даёт 17. Строки он склеивает: `"12" + "5"` даёт `"125"`. Строку можно умножить на число – она повторится: `"ха" * 3` даёт `"хахаха"`. А вот сложить строку с числом нельзя: на `"5" + 3` Python остановится с ошибкой `TypeError` («ошибка типа»). Превратить одно в другое помогают `int("5")` – в целое число, `float("2.5")` – в дробное, `str(5)` – в строку.

Арифметика в Python почти школьная, только знаки другие:

Таблица 8.2

Запись	Что делает	Пример	Результат
<code>+ - *</code>	плюс, минус, умножить	<code>4 * 3</code>	12
<code>/</code>	деление (дробное)	<code>7 / 2</code>	3.5
<code>//</code>	сколько раз помещается	<code>17 // 5</code>	3
<code>%</code>	остаток	<code>17 % 5</code>	2
<code>**</code>	степень	<code>2 ** 5</code>	32

Обычное деление `/` всегда даёт дробное число, даже когда делится нацело: `6 / 2` – это 3.0. А `//` отвечает на вопрос, сколько раз делитель целиком помещается в числе.

Пример из жизни: 17 конфет раскладываем в пакеты по 5. Полных пакетов `17 // 5 = 3`, в руке остаётся `17 % 5 = 2` конфеты. Дробную часть Python всегда пишет через точку: 3.5, а не 3,5.

В задачах «что напечатает программа» всё решает одно умение – проследить, что лежит в каждой коробке после каждой строки. Такая запись по шагам называется **трассировкой**. Возьмём программу:

```
a = 5
b = a * 2
a = a + b
print(a, b)
```

Таблица 8.3

Строка	Код	Считаем	a	b
1	<code>a = 5</code>	–	5	–
2	<code>b = a * 2</code>	<code>5 * 2 = 10</code>	5	10
3	<code>a = a + b</code>	<code>5 + 10 = 15</code>	15	10
4	<code>print(a, b)</code>	печать	15	10

На экране будет `15 10`. В строке 3 справа от `=` стоит старое значение `a` (5), а в коробку ложится новое (15). Старое пять пропадает навсегда.

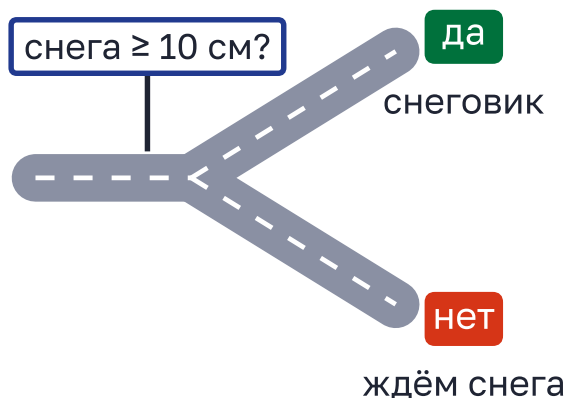
Запомни. = кладёт значение в переменную, новое вытесняет старое. Правая часть считается целиком, а уже потом результат кладётся в коробку слева. `input` всегда отдаёт текст – для чисел оборачивай его в `int(...)`.

Совет. Трассируй на бумаге: столбец на каждую переменную, строка на каждую команду. Новое значение пиши ниже, старое зачёркивай. На задачах «что напечатает программа» этот приём спасает чаще любых других.

8.2. Условия и циклы

НА ПАЛЬЦАХ

if – развилка



идём только по одной дороге

for, while – круги



тот же круг – пока не скажут «хватит»

Рисунок 8.2 – Условие – развилка, цикл – круги по стадиону

На развилке идёшь по одной дороге из двух. На стадионе бежишь один и тот же круг, пока тренер не скажет «хватит».

Без условий программа всегда делает одно и то же. **Условие** позволяет ей выбирать: если выпало много снега – одно, если мало – другое.

```
sneg = int(input("Сколько снега выпало, см? "))
if sneg ≥ 30:
    print("Берём лыжи")
elif sneg ≥ 10:
    print("Лепим снеговика")
else:
    print("Ждём снега")
```

`if` значит «если». За ним – условие, которое бывает верным или неверным, и **двоеточие**. Команды, которые выполняются при верном условии, пишут с **отступом** в 4 пробела. Отступ

здесь не для красоты: по нему Python понимает, какие строки «внутри» условия. `elif` – «иначе если»: это условие проверяется, только если все предыдущие оказались неверными. `else` – «иначе»: работает, когда не подошло ничего.

Выполняется ровно одна ветка. Введём 45: условие `45 ≥ 30` верно – печатается «Берём лыжи», и на этом всё. То, что 45 больше и 10 тоже, уже не важно: до `elif` очередь не дошла. Введём 10: первое условие неверно, второе `10 ≥ 10` верно – «Лепим снеговика». Введём 3 – «Ждём снега».

Знаки сравнения:

Таблица 8.4

Python	Значит	Python	Значит
<code>></code>	больше	<code><</code>	меньше
<code>≥</code>	больше или равно	<code>≤</code>	меньше или равно
<code>=</code>	равно (два знака!)	<code>≠</code>	не равно

Условия можно соединять: `and` – «и» (верно, когда верны оба), `or` – «или» (хватит одного), `not` – «не». Условие «снега от 10 до 29 см» записывают так: `sneg ≥ 10 and sneg < 30`.

Цикл повторяет одни и те же команды много раз. Цикл `for` проходит по готовому ряду значений:

```
for i in range(1, 6):  
    print(i, "круг")
```

Программа напечатает пять строк: «1 круг», «2 круг» и так до «5 круг». `range(1, 6)` – это ряд чисел 1, 2, 3, 4, 5: **последнее число в него не входит**. Переменная `i` на каждом круге принимает следующее значение. Ещё два вида: `range(5)` – это 0, 1, 2, 3, 4 (с нуля, пять чисел), а `range(2, 11, 2)` – 2, 4, 6, 8, 10 (третье число – шаг).

Самый частый приём с циклом – **копилка**: заводим переменную с нулём и на каждом круге что-то в неё добавляем.

```
sneg = [5, 0, 12, 3, 8]      # сколько сантиметров снега выпало за 5 дней  
vsego = 0  
for s in sneg:  
    vsego = vsego + s  
print(vsego)                # 28
```

Здесь `for` идёт прямо по списку чисел (о списках – в следующем разделе): `s` по очереди становится 5, 0, 12, 3, 8. Трассировка: `vsego` было 0, потом `0 + 5 = 5`, `5 + 0 = 5`, `5 + 12 = 17`, `17 + 3 = 20`, `20 + 8 = 28`. Если в копилку каждый раз добавлять не число, а единицу – получится **счётчик**: сколько раз что-то случилось.

Когда заранее не знаешь, сколько будет кругов, берут цикл `while` – «пока». Он крутится, пока условие верно:

```
sugrob = 20
den = 0
while sugrob < 50:
    sugrob = sugrob + 8
    den = den + 1
print(den, sugrob)           # 4 52
```

Сугроб в 20 см каждый день подрастает на 8 см. Через сколько дней он станет не меньше 50 см? Перед каждым кругом Python проверяет условие: $20 < 50$ – круг, сугроб 28; $28 < 50$ – круг, 36; $36 < 50$ – круг, 44; $44 < 50$ – круг, 52. Теперь $52 < 50$ неверно – цикл закончен. Напечатано 4 52. Если условие не станет неверным никогда, цикл будет крутиться бесконечно – программу придётся остановить кнопкой.

Если ты собирал программы в Scratch (в курсе сферы блоки собирают в CodeChef, а Scratch – дополнительный материал кружка), многое уже знакомо:

Таблица 8.5

Блок Scratch	Python
«спросить ... и ждать», «ответ»	<code>input()</code>
«сказать ...»	<code>print()</code>
«если ..., то», «иначе»	<code>if ...: , else:</code>
«повторить ... раз»	<code>for i in range(...):</code>
«повторять пока не ...»	<code>while ...:</code> – но условие наоборот!

Последняя строка – ловушка. Scratch крутит «повторять пока не» до тех пор, пока условие **неверно**. Python крутит `while`, пока условие **верно**. Блок «повторять пока не сугроб > 49» – это `while sugrob < 50`.

Запомни. После `if`, `elif`, `else`, `for`, `while` – двоеточие, а строки внутри – с отступом. Из цепочки `if – elif – else` выполняется только первая подходящая ветка. `range(1, 6)` – это 1, 2, 3, 4, 5, без шести.

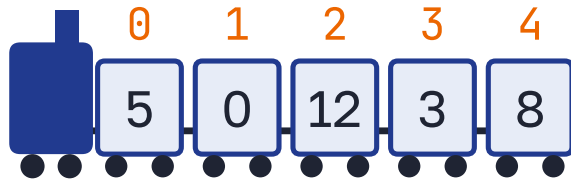
Совет. В задачах с циклом выпиши таблицу «круг – значение переменной – что в копилке». Проверь первый круг и последний: на границах ($>$ или $\geq$, `range(1, 6)` или `range(1, 7)`) ошибаются чаще всего.

8.3. Списки, словари и строки

НА ПАЛЬЦАХ

список: ищем по номеру

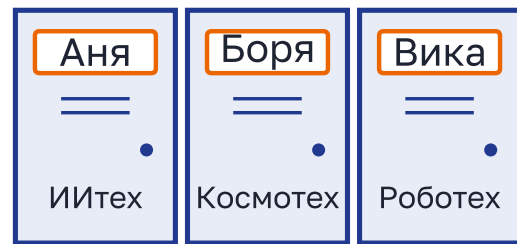
номер:



`sneg[2] → 12`

номер 2 – это третий вагон

словарь: ищем по имени



`kruzhki["Боря"] → Космотех`

на дверце – ключ, внутри – значение

Рисунок 8.3 – Список – вагоны с номерами, словарь – шкафчики с именами

В списке ищешь по номеру вагона, и считать начинают с нуля. В словаре ищешь по подписи на дверце.

Список – много значений в одной переменной, по порядку. Его пишут в квадратных скобках через запятую. У каждого элемента есть **номер** (индекс) – и начинается он с **нуля**.

```
sneg = [5, 0, 12, 3, 8]
print(sneg[0])      # 5 – первый элемент
print(sneg[2])      # 12 – третий, а не второй!
print(sneg[-1])     # 8 – последний
print(len(sneg))    # 5 – сколько элементов
sneg.append(7)       # добавить 7 в конец
print(sneg)         # [5, 0, 12, 3, 8, 7]
print(sum(sneg), max(sneg), min(sneg)) # 35 12 0
print(12 in sneg)   # True – есть ли в списке 12
```

Почему с нуля? Номер показывает, сколько элементов надо пропустить от начала. Чтобы взять первый, пропускать не нужно ничего – номер 0. Отрицательные номера считают с конца: -1 – последний, -2 – предпоследний.

Иногда нужен сам номер, а не только значение: например, чтобы сравнить два списка элемент за элементом. Тогда цикл идёт по номерам: `for i in range(len(sneg)):` – `i` пробежит 0, 1, 2, ... до последнего номера. Длина списка 6, а номера – от 0 до 5, и `range(6)` выдаёт как раз их.

Строка в Python похожа на список букв: у неё есть длина, номера букв (тоже с нуля) и кусочки.

```

gorod = "Новосибирск"
print(len(gorod))           # 11 - букв в слове
print(gorod[0], gorod[-1]) # Н к - первая и последняя
print(gorod[:4])            # Ново - первые 4 буквы
print(gorod.lower())        # новосибирск - все буквы маленькие
print(gorod.count("и"))     # 2 - сколько раз встречается «и»
print("сиб" in gorod)      # True - есть ли такой кусочек
fraza = "Снег идёт весь день"
print(fraza.split())       # ['Снег', 'идёт', 'весь', 'день']

```

`split()` режет строку по пробелам и отдаёт список слов – так программа начинает «читать» текст. По строке, как по списку, можно пройти циклом – буква за буквой. Посчитаем гласные:

```

slovo = "Кванториум"
glasnye = "аеёиоуыэюя"
k = 0
for буква in slovo.lower():
    if буква in glasnye:
        k = k + 1
print(k)                       # 4

```

Гласные в слове – а, о, и, у, их 4. Для «Кванториума» `lower()` ничего не меняет. А возьми слово «Объ»: без `lower()` программа не узнала бы большую «О» – в строке `glasnye` только маленькие буквы – и насчитала бы 0.

Словарь хранит пары «ключ – значение». Ключ – подпись на дверце шкафчика, значение – то, что внутри. Пишут в фигурных скобках: ключ, двоеточие, значение.

```

kruzhki = {"Аня": "ИИтех", "Боря": "Космотех"}
print(kruzhki["Аня"])          # ИИтех - достаём по ключу
kruzhki["Вика"] = "Роботех"    # новая пара
kruzhki["Боря"] = "ИИтех"      # у Бори значение заменилось
print(kruzhki)                 # {'Аня': 'ИИтех', 'Боря': 'ИИтех', 'Вика': 'Роботех'}
print("Гоша" in kruzhki)       # False - in проверяет ключи
print(kruzhki.get("Гоша", "нет такого")) # нет такого

```

Ключи в словаре не повторяются: если положить значение по ключу, который уже есть, старое заменится. Обращение `kruzhki["Гоша"]` к несуществующему ключу остановит программу с ошибкой `KeyError`. Безопасный способ – `get`: вторым значением в скобках пишут, что выдать, если ключа нет.

Со словарём удобно считать, какое слово сколько раз встретилось. Этим пользуются чат-боты и нейросети, которые работают с текстом.

```

tekst = "снег снег лыжи горка снег лыжи"
schet = {}
for slovo in tekst.split():
    if slovo in schet:
        schet[slovo] = schet[slovo] + 1 # слово уже было - прибавляем 1
    else:
        schet[slovo] = 1                 # встретили впервые
print(schet)                            # {'снег': 3, 'лыжи': 2, 'горка': 1}

```

Запомни. Номера в списке и в строке начинаются с 0, последний элемент – `[-1]`, а номер последнего равен длине минус один. Словарь ищет по ключу, ключи не повторяются. "Снег" и "снег" для Python – разные слова.

Совет. Прежде чем считать слова, приведи текст к одному виду: `lower()` – все буквы маленькие, `replace(".", "")` – убрать точки (так же для запятых). Иначе «Робот» в начале предложения и «робот» в середине посчитаются отдельно.

8.4. Диалог – это граф

НА ПАЛЬЦАХ

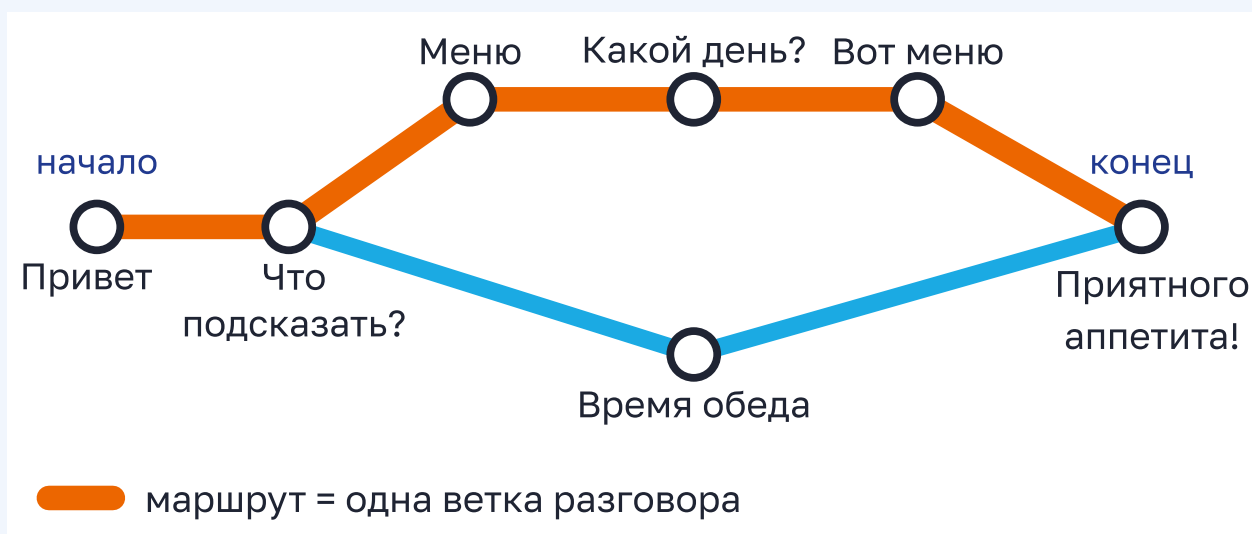


Рисунок 8.4 – Диалог чат-бота как схема метро

Станции – реплики бота, перегоны – переходы. Одна поездка от начала до конца – одна ветка разговора.

Чат-бот – программа, которая ведёт разговор с человеком. Перед тем как его программировать, разговор проектируют: какие реплики говорит бот и куда разговор идёт дальше после каждого ответа человека. Удобнее всего нарисовать это **графом** – схемой из кружков и стрелок.

- **Узел** (вершина) графа – одна реплика бота. Узлы нумеруют.
- **Стрелка** (ребро) – переход: после этой реплики разговор может пойти в эту сторону. На стрелке подписывают, что выбрал человек.
- **Начало** – реплика, с которой бот начинает. **Конец** – реплика, после которой разговор окончен.
- **Ветка** – путь от начала до конца по стрелкам. Одна ветка – один возможный разговор.
- **Длина ветки** – сколько реплик в ней. Считаем кружки, а не стрелки.

Вот бот школьной столовой. Его сценарий можно записать списком: номер реплики, что говорит бот и куда разговор идёт дальше.

- **1.** «Привет! Подсказать меню или время обеда?» – «меню» → 2, «время» → 4.
- **2.** «На какой день недели?» → 3.
- **3.** «Вот меню на этот день.» → 5.
- **4.** «Обед для 5–7 классов – в 12:15.» → 5.
- **5.** «Приятного аппетита!» – конец.

А можно нарисовать – это тот же сценарий, только нагляднее:

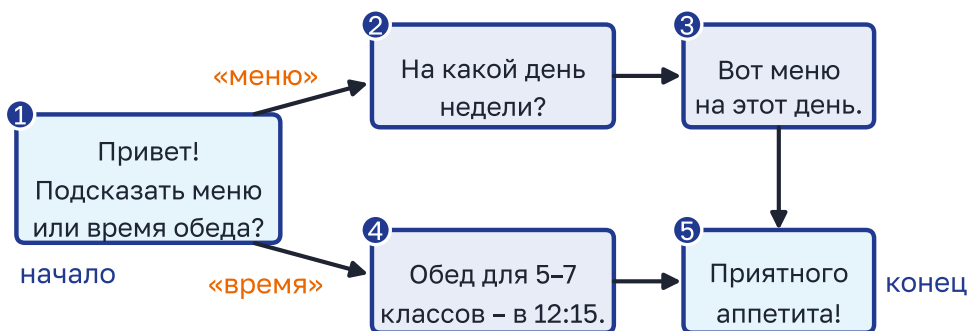


Рисунок 8.5 – Граф бота столовой. Узлы – реплики, стрелки – переходы, на стрелках – выбор человека

Ветки здесь две: 1 – 2 – 3 – 5 (4 реплики) и 1 – 4 – 5 (3 реплики). Обе заканчиваются одной и той же репликой 5 – ветки в графе могут сходиться. У дерева из главы 3 ветки только расходятся, а в сценарии бота они часто снова встречаются: «Приятного аппетита!» и «Пока!» говорят всем.

Как найти ветки в большом графе? Иди от начала и на каждой развилке выписывай столько цепочек, сколько стрелок выходит из узла. Каждую цепочку води до конца. Когда ветки ещё и сходятся посередине, выписывать их долго. Есть приём быстрее: **подпиши у каждого узла, сколькими путями в него можно прийти**. У начала пишем 1. У любого другого узла – сумму чисел у тех узлов, из которых в него ведут стрелки. Число у конца – это и есть количество веток. Так мы решим задачу 8.6.

Как бот понимает, куда идти. Если человек нажимает кнопки, всё просто: одна кнопка – одна стрелка. Но люди пишут как хотят: «что сегодня на обед?», «покажи меню», «какой суп?». Всё это одно **намерение** – по-английски **интент** (intent): человек хочет узнать меню. Разработчик задаёт боту интенты и к каждому – десяток **тренировочных фраз**, а бот учится относить новую фразу к одному из интентов. Это задача классификации – та самая, из главы 1.

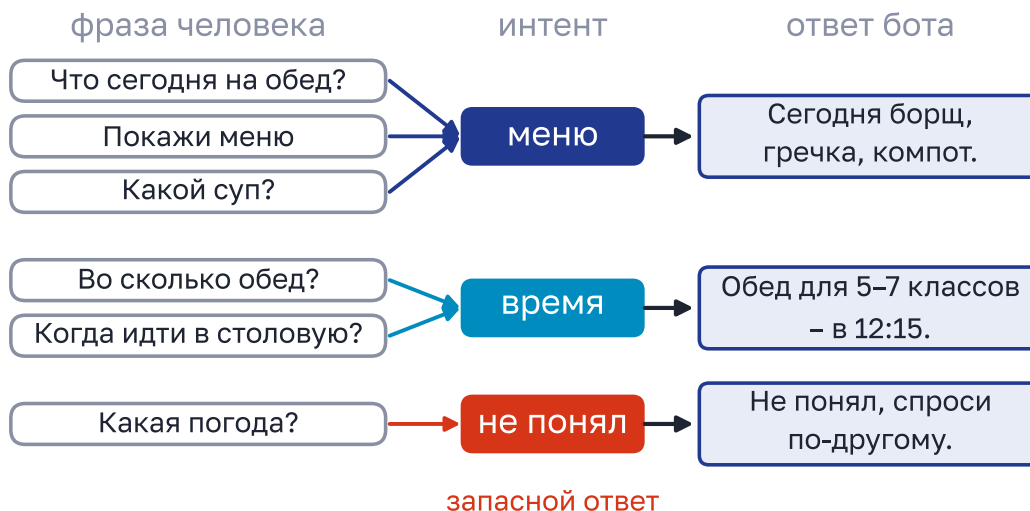


Рисунок 8.6 – Разные фразы – одно намерение. Бот определяет интент и по нему выбирает следующую реплику

Ещё два слова из мира чат-ботов. **Параметр** – деталь, которую бот вытаскивает из фразы: в «посоветуй детектив» интент – «посоветовать книгу», а параметр «жанр» – детектив. **Контекст** – память бота о том, о чём сейчас разговор. Если бот только что спросил «какой жанр любишь?», то ответ из одного слова «детектив» он поймёт правильно. Без контекста слово «детектив» ему ни о чём не скажет.

В задачах НТО прошлых лет ботов собирали в сервисе Dialogflow. Там контекст – это метка-пропуск: первый интент её выдаёт («выходной контекст»), а второй срабатывает, только если метка есть («входной контекст»). И всегда нужен **запасной ответ** на случай, когда бот не понял ни одного интента: «Не понял, спроси по-другому». В Dialogflow для этого есть готовый интент Default Fallback Intent.

Настоящие боты учатся на примерах. А самого простого бота можно написать на Python за два десятка строк: он ищет в фразе ключевые слова.

```

klyuchi = {
    "меню": ["меню", "обед", "суп", "есть"],
    "время": ["когда", "во сколько", "время"],
    "привет": ["привет", "здравствуй"],
}

otvety = {
    "меню": "Сегодня борщ, гречка с котлетой и компот.",
    "время": "Обед для 5-7 классов – в 12:15.",
    "привет": "Привет! Спроси про меню или время обеда.",
}

def uznat_intent(fraza):
    fraza = fraza.lower()
    for intent, slova in klyuchi.items(): # по очереди перебираем интенды
        for slovo in slova:
            if slovo in fraza: # нашли ключевое слово – интент готов
                return intent
    return "не понял"

for fraza in ["Привет!", "Что сегодня на обед?", "Во сколько идти в столовую?", "Какая по-
года?"]:
    intent = uznat_intent(fraza)
    print(fraza, "→", intent, "→", otvety.get(intent, "Не понял, спроси по-другому."))

```

def создаёт **функцию** – маленькую программу с именем, которую можно вызывать много раз. `uznat_intent` получает фразу и делает все буквы маленькими. Потом перебирает словарь `klyuchi: items()` выдаёт пары «ключ – значение», и на каждом круге в `intent` попадает имя интента, а в `slova` – его список слов. Внутренний цикл проверяет слова по одному. Как только в фразе нашёлся нужный кусочек, `return` сразу возвращает имя интента, и функция заканчивает работу. Не нашлось ничего – возвращается «не понял». Программа напечатает:

```

Привет! → привет → Привет! Спроси про меню или время обеда.
Что сегодня на обед? → меню → Сегодня борщ, гречка с котлетой и компот.
Во сколько идти в столовую? → время → Обед для 5-7 классов – в 12:15.
Какая погода? → не понял → Не понял, спроси по-другому.

```

Спроси этого бота «Когда будет суп?» – он ответит меню, хотя спрашивали про время. Интент «меню» проверяется первым, а слово «суп» в фразе есть. Бот видит слова, а не смысл, и порядок проверки для него важен.

Запомни. Узел – реплика, стрелка – переход, ветка – путь от начала до конца. Длина ветки – число реплик в ней. Число веток = число у конца, если подписать у каждого узла сумму чисел узлов, из которых в него входят стрелки. Интент – намерение человека, которое бот узнаёт по фразе.

Совет. Перед подсчётом прочитай, как в задаче определена длина: реплики или переходы. Переходов всегда на один меньше, чем реплик. Выписывай каждую ветку полностью, номерами через дефис, и зачёркивай на рисунке пройденные стрелки.

8.5. Промпт: как попросить нейросеть

НА ПАЛЬЦАХ

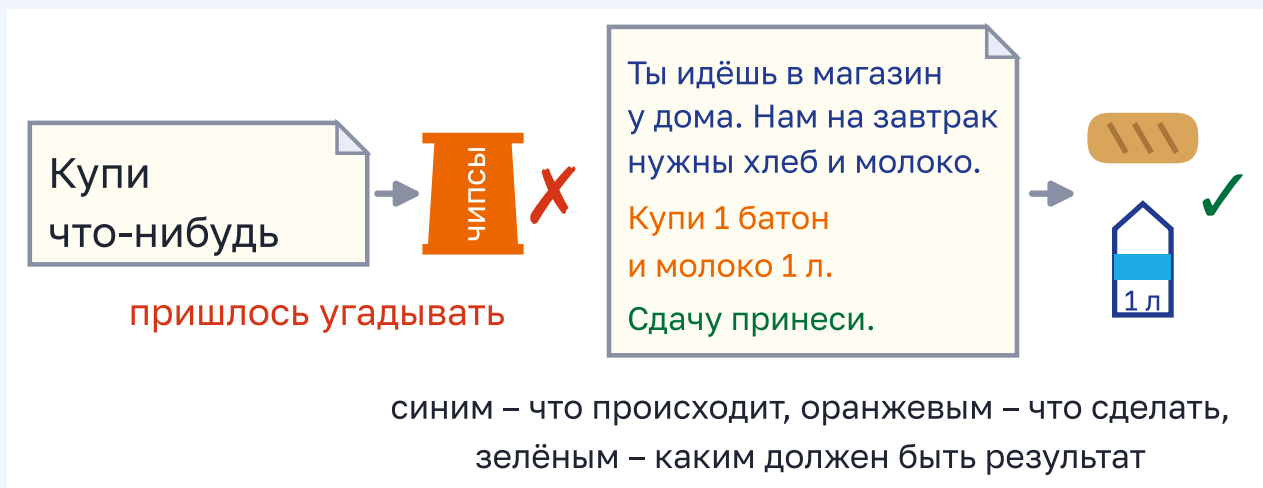


Рисунок 8.7 – Две записки брату, который идёт в магазин

«Купи что-нибудь» – и он принесёт чипсы. Чем точнее записка, тем меньше приходится угадывать.

Промпт – запрос к нейросети словами. Нейросетевые чаты вроде GigaChat, Алисы или DeepSeek (с ними работают в модуле 8 программы сферы) отвечают на любой вопрос. Но они не знают, кто ты, зачем спрашиваешь и какой ответ тебе нужен. Всё, что ты не написал, нейросеть додумает сама – и не обязательно так, как хотел ты.

В программе сферы хороший промпт собирают из пяти частей:

- **Контекст и роль** – кем нейросеть должна быть и что она должна знать о тебе и о ситуации. Пример: «Ты – учитель информатики, который объясняет шестиклассникам. Я только начал учить Python.» Роль – это «кем быть», контекст – «для кого и зачем»; в курсе их держат в одной части.
- **Цель** – что именно сделать. Пример: «Напиши программу на Python, которая считает, сколько всего снега выпало за эти дни и сколько в среднем за день.»
- **Исходные данные и детали** – с чем работать: сами данные и важные подробности о них. Пример: «Я веду дневник снега: каждый вечер записываю, сколько сантиметров выпало. Числа – в одну строку через пробел.»
- **Инструменты и ограничения** – чем можно пользоваться и чего нельзя. Пример: «Без дополнительных библиотек, только `input`, `split` и `sum`.»
- **Формат результата** – в каком виде нужен ответ. Пример: «Дай только код, а под ним – по одной строке пояснения к каждой строке кода.»

Сравни с проммптом «напиши код про снег». Нейросеть не знает, что считать: сумму, среднее, самый снежный день или прогноз погоды. Не знает, в каком виде у тебя числа. Может подключить библиотеку, которой у тебя нет, и объяснить словами, которых ты не знаешь. Ответ будет, но, скорее всего, не тот.

Таблица 8.6

	Плохой промпт	Хороший промпт
Текст	«напиши код про снег»	все пять частей из списка выше
Что нейросети приходится угадывать	что считать, в каком виде данные, для кого объяснение, нужен ли код целиком	почти ничего
Что получаем	что-то «про снег», часто длинное и не то	короткий код, который можно проверить на своих числах

Как понять, что ответ хороший? Проверь три вещи. **Учтено ли всё, что ты написал:** и сумма, и среднее, и без лишних библиотек. **Логичен ли ответ:** нет ли в нём противоречий и слов, которые ни к чему не относятся. **Работает ли он:** код – запусти, числа – пересчитай, факты – найди в надёжном источнике. Если что-то не так, не начинай заново, а уточни: «в твоём коде среднее считается неверно, если в какой-то день было 0 см, исправь». Разговор с нейросетью – это тоже диалог, и она помнит, что было в нём раньше.

Нейросеть может ошибаться – и очень уверенно. Языковая модель подбирает самое правдоподобное продолжение текста (как в задаче 3.7 из главы 3), а не ищет правду в справочнике. Поэтому она может выдумать книгу, которой нет, перепутать дату или сослаться на функцию Python, которой не существует, – и всё это гладким, убедительным языком. Такую выдумку называют **галлюцинацией** нейросети. На тот же вопрос в другой раз она может ответить по-другому.

Правила безопасности, которые не обсуждаются:

- не пиши в чат своё полное имя, адрес, телефон, номер школы и класса, пароли, не загружай фото документов;
- важные факты (даты, цифры, медицинские и денежные советы) проверяй в надёжных источниках или у взрослых;
- пользуйся ИИ-чатами вместе со взрослым – у сервисов есть свои правила о возрасте пользователей.

Запомни. Промпт = контекст и роль + цель + исходные данные + ограничения + формат. Чего не написал – нейросеть додумает сама. Любой ответ нейросети – черновик, пока ты его не проверил.

Совет. Попробуй сам на одном вопросе: сначала короткий промпт в одну строку, потом полный из пяти частей. Сравни ответы: какой короче, понятнее, какой выполнил всё, что ты просил.

8.6. Вайбкодинг: код от нейросети и его проверка

НА ПАЛЬЦАХ



Рисунок 8.8 – Круг вайбкодинга: попросил – получил – проверил – поправил

Нейросеть пишет код быстро, но проверяешь его ты. Запускать в работу можно только то, что прошло проверку.

Вайбкодинг – способ писать программы, когда код по твоему описанию словами пишет нейросеть, а ты проверяешь и поправляешь. Порядок работы такой же, как в программе сферы: сформулировать промпт → получить ответ → проверить и исправить → и только потом запускать программу в дело.

Чтобы проверить код, мало на него посмотреть. Его надо запустить на **тесте** – таком примере, где ответ ты заранее знаешь сам. Для дневника снега подойдут три дня: 4, 0 и 11 см. Сумма $4 + 0 + 11 = 15$ см, среднее $15 : 3 = 5$ см. Код должен выдать то же самое. Допустим, нейросеть ответила так:

```
dni = input().split()
summa = 0
for d in dni:
    summa = summa + d
print(summa / len(dni))
```

Вводим 4 0 11 – и программа падает:

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Сообщение означает: «не умею складывать целое число (int) со строкой (str)». `split()` дал список строк `['4', '0', '11']`, а строку к числу не прибавить. Исправляем строку 4: `summa =`

`summa + int(d)` . Запускаем снова – получаем `5.0` . Это наши 5 см, тест пройден.

Ошибки в коде бывают трёх видов, от безобидных к опасным:

Таблица 8.7

Вид	Как выглядит	Пример
Программа не запускается	Python сразу пишет <code>SyntaxError</code> и строку	забыто двоеточие после <code>if</code>
Программа падает на ходу	<code>TypeError</code> , <code>NameError</code> , <code>KeyError</code>	строку сложили с числом, опечатка в имени переменной
Программа работает, но считает не то	ошибок нет, ответ неверный	цикл пропустил первый элемент, <code>></code> вместо <code>≥</code>

Третий вид самый коварный: на экране аккуратное число, и хочется ему поверить. Ловит такие ошибки только тест. Хорошие тесты – маленькие и с подвохом: пустой ввод, один день, одинаковые числа, ноль, отрицательные числа (температура зимой).

Запомни. Код от нейросети проверяют запуском на тестах, где ответ известен заранее. Падение программы – это подсказка, где искать. Самая опасная ошибка – когда программа работает и выдаёт правдоподобное неверное число.

Совет. Попроси нейросеть объяснить код построчно – и сверь объяснение с тем, что код делает на самом деле. Если в коде есть строка, которую ты не можешь объяснить, не запускай его в дело, пока не разберёшься.

8.7. Разбираем задачи

Задача 8.1 Сколько снеговиков

разминка · переменные, ввод, строки и числа

Программа спрашивает, сколько снеговиков слепили вчера и сколько сегодня:

```
vchera = input()
segodnya = input()
print(vchera + segodnya)
print(int(vchera) + int(segodnya))
vsego = int(vchera) * 2
vchera = vsego - int(segodnya)
print(vchera, vsego)
```

Пользователь ввёл сначала `7` , потом `5` . Запиши, что появится на экране: три строки.

Дано: код из 7 строк; ввод `7` , затем `5` .

Найти: три строки вывода.

Решение

Шаг 1. Что лежит в переменных после ввода. `input` отдаёт текст. В `vchera` лежит строка "7", в `segodnya` – строка "5". Не числа!

Шаг 2. Строка 3. "7" + "5" – это склейка строк: "75". Первая строка вывода – 75.

Шаг 3. Строка 4. Здесь обе строки превращены в числа: $7 + 5 = 12$. Вторая строка вывода – 12.

Шаг 4. Строки 5–6, трассировка.

Таблица 8.8

Строка	Считаем	vchera	segodnya	vsego
5	$7 \cdot 2 = 14$	"7"	"5"	14
6	$14 - 5 = 9$	9	"5"	14

В строке 6 в `vchera` кладут новое значение – теперь это число 9, а строка "7" пропала.

Шаг 5. Строка 7. Печатаем `vchera` и `vsego` через пробел: 9 14.

Прикинем на глаз: 12 снеговиков за два дня – обычная работа для двора. А 75 – явно не сумма, а две цифры, приставленные друг к другу: так + обходится со строками.

 **Типичная ошибка.** Написать в первой строке 12: не заметить, что `input` отдаёт текст. А в последней строке легко напечатать 7 14, забыв, что строка 6 заменила значение `vchera`.

Ответ: 75, 12, 9 14.

Своя задача по программе сферы, модуль 3

Реши сам

8.1. Что напечатает программа? Запиши два числа через пробел.

```
x = 4
y = x * 3
x = y - x
print(x, y)
```

8.2. Что напечатает каждая строка?

```
print("3" + "4")
print(3 + 4)
print("ла" * 3)
```

8.3. На экскурсию в планетарий едут 58 школьников. В один микроавтобус садится 12 человек. Что напечатает `print(58 // 12, 58 % 12)`? Сколько микроавтобусов нужно заказать, чтобы поехали все?

Задача 8.2 Снежные дни

цикл `for`, условие, счётчик и копилка

Метеостанция в Академгородке 8 дней подряд записывала, сколько сантиметров снега выпало за день. Программа ищет снежные дни:

```
sneg = [3, 0, 12, 7, 0, 15, 5, 9]
dni = 0
vsego = 0
for s in neg:
    if s > 5:
        dni = dni + 1
        vsego = vsego + s
print(dni, vsego)
```

Что напечатает программа? Запиши два числа через пробел.

Дано: список из 8 чисел; условие `s > 5`.

Найти: значения `dni` и `vsego` в конце.

Решение

Шаг 1. Что делает программа. Цикл по очереди берёт каждое число списка. Если оно больше 5, программа прибавляет 1 к счётчику `dni` и само число – к копилке `vsego`. Обе строки внутри `if` имеют одинаковый отступ, значит, выполняются вместе и только при верном условии.

Шаг 2. Трассировка.

Таблица 8.9

Круг	s	s > 5 ?	dni	vsego
1	3	нет	0	0
2	0	нет	0	0
3	12	да	1	12
4	7	да	2	19
5	0	нет	2	19
6	15	да	3	34
7	5	нет	3	34
8	9	да	4	43

Шаг 3. Вывод. После цикла `dni = 4`, `vsego = 43`. Печать через пробел: `4 43`.

Без таблицы то же видно сразу: больше 5 см выпало в четыре дня – 12, 7, 15 и 9 см, вместе $12 + 7 + 15 + 9 = 43$. Почти полметра снега за четыре дня – для обычной недели многовато, но в сильный снегопад так и бывает.

❗ **Чаще всего** день с 5 см тоже записывают в снежные и получают `5 48`. Условие `s > 5` строгое: 5 не больше 5. Было бы `s ≥ 5` – тогда да.

Ответ: 4 43 .

Своя задача по программе сферы, модуль 3

Реши сам

8.4. В школьной игре «Снежный квест» за баллы дают значки. Что напечатает программа, если ввести 73 ?

```
bally = int(input())
if bally ≥ 85:
    print("золотой значок")
elif bally ≥ 70:
    print("серебряный значок")
elif bally ≥ 50:
    print("бронзовый значок")
else:
    print("значок участника")
```

8.5. Что напечатает программа? Запиши одно число.

```
k = 0
for i in range(1, 6):
    k = k + i * 2
print(k)
```

8.6. В копилке 150 рублей. Каждую неделю туда кладут 40 рублей. Что напечатает программа? Запиши два числа через пробел.

```
kopilka = 150
nedeli = 0
while kopilka < 400:
    kopilka = kopilka + 40
    nedeli = nedeli + 1
print(nedeli, kopilka)
```

Задача 8.3 Отзыв о роботе

строки, lower, replace, split, словарь-счётчик

Программа разбирает отзыв о школьном роботе-пылесосе:

```

otzyv = "Робот быстрый. Робот умный, но робот шумный"
slova = otzyv.lower().replace(".", "").replace(",", "").split()
schet = {}
for w in slova:
    if w in schet:
        schet[w] = schet[w] + 1
    else:
        schet[w] = 1
print(len(slova), len(schet), schet["робот"])

```

Что она напечатает? Запиши три числа через пробел.

Дано: строка отзыва; цепочка `lower`, `replace`, `split`; словарь-счётчик.

Найти: длину списка `slova`, число ключей в `schet`, значение `schet["робот"]`.

Решение

Шаг 1. Строка 2 по шагам, слева направо. `lower()` делает все буквы маленькими: «робот быстрый. робот умный, но робот шумный». Первый `replace` убирает точку, второй – запятую: «робот быстрый робот умный но робот шумный». `split()` режет по пробелам:

```
['робот', 'быстрый', 'робот', 'умный', 'но', 'робот', 'шумный']
```

В списке 7 слов: `len(slova) = 7`.

Шаг 2. Словарь-счётчик. Идём по словам. «робот» – впервые, ставим 1. «быстрый» – 1. «робот» – уже есть, стало 2. «умный» – 1. «но» – 1. «робот» – 3. «шумный» – 1. Получилось: `{'робот': 3, 'быстрый': 1, 'умный': 1, 'но': 1, 'шумный': 1}`.

Шаг 3. Ответ по частям. Разных слов (ключей) – 5. «робот» встретилось 3 раза. Печать: `7 5 3`.

Сложи все значения словаря: $3 + 1 + 1 + 1 + 1 = 7$ – столько же, сколько слов в списке. Каждое слово попало в счётчик один раз, ни одно не потерялось.

⚠ Типичная ошибка. Забыть про `lower()` и решить, что «Робот» и «робот» – разные слова. Тогда выйдет `7 6 1`: в словаре отдельно «Робот» (2 раза) и «робот» (1 раз). Тот же `7 6 1` напечатает и сама программа, если убрать из неё `lower()`. Красной надписи не будет – просто неверные числа.

Ответ: 7 5 3.

Своя задача по программе сферы, модули 3 и 8

Реши сам

8.7. Что напечатает программа? Две строки.

```

a = [10, 20, 30, 40, 50]
print(a[1], a[-1], len(a))
a.append(60)
print(sum(a))

```

8.8. Что напечатает программа? Запиши четыре значения через пробел.

```
s = "Академгородок"
print(len(s), s[0], s[-2], s.count("o"))
```

8.9. Возьми бота столовой из раздела 8.4 (функция `uznat_intent` со словарём `klyuchi`). Человек пишет: «Во вторник автобус в шесть?» Какой интент выберет бот: «меню», «время», «привет» или «не понял»? Почему?

Задача 8.4 Бот-огородник

граф диалога, ветки, длина ветки

Для дачников сделали чат-бота с советами, что посадить. Сценарий разговора нарисовали графом: узлы – реплики бота, стрелки – переходы. Разговор всегда начинается с реплики 1 и заканчивается репликой 18.

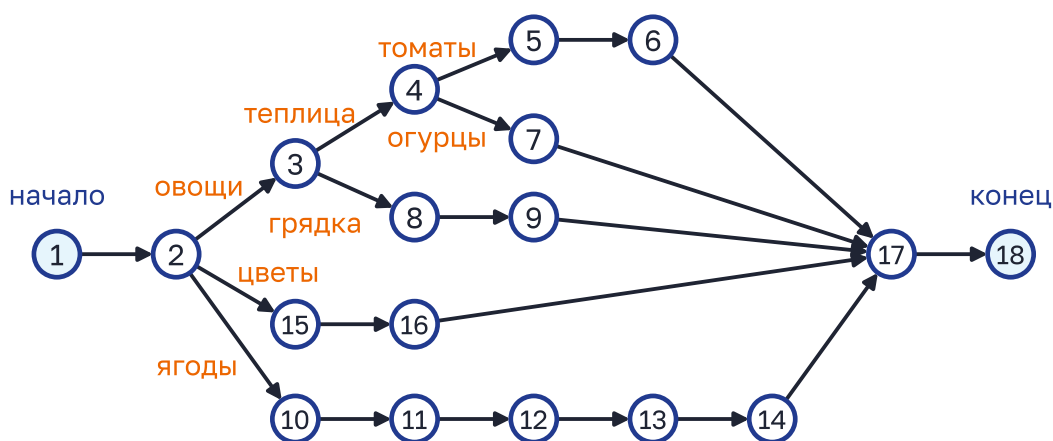


Рисунок 8.9 – Сценарный граф бота-огородника

Сколько веток у диалога? Какая длина у самой длинной и у самой короткой ветки? Длина ветки – число реплик в ней, включая первую и последнюю. Запиши три числа через пробел.

Дано: граф из 18 узлов; начало – 1, конец – 18.

Найти: число веток; наибольшую и наименьшую длину ветки.

Решение

Шаг 1. Идём от начала. Из 1 одна стрелка – в 2. Из 2 выходят три: «овощи» (3), «ягоды» (10), «цветы» (15). Значит, разговор делится минимум на три части. Больше всего развилкок у «овощей», с них и начнём.

Шаг 2. Ветка «овощи». Из 3 две стрелки: «теплица» (4) и «грядка» (8). Из 4 ещё две: «томаты» (5) и «огурцы» (7). Получаем три цепочки: через 5, через 7 и через 8.

Шаг 3. Выписываем все ветки до конца и считаем в каждой реплики:

1. 1-2-3-4-5-6-17-18 – 8 реплик;

2. 1-2-3-4-7-17-18 – 7 реплик;
3. 1-2-3-8-9-17-18 – 7 реплик;
4. 1-2-10-11-12-13-14-17-18 – 9 реплик;
5. 1-2-15-16-17-18 – 6 реплик.

Шаг 4. Считаем. Веток 5. Самая длинная – ветка «ягоды», 9 реплик. Самая короткая – «цветы», 6 реплик.

Пересчитаем по развилкам. «Теплица» делится на томаты и огурцы – 2 ветки, «грядка» – 1, значит, у «овощей» $2 + 1 = 3$ ветки. У «ягод» и «цветов» по одной. Всего $3 + 1 + 1 = 5$, как и в списке. Реплики 17 и 18 есть в каждой ветке: прощается бот со всеми одинаково.

 **Типичная ошибка.** Считать не реплики, а стрелки: тогда выйдет «5 8 5» – на единицу меньше. А кто не заметил развилку в узле 4, насчитает 4 ветки.

Ответ: 5 9 6 .

По мотивам задачи отборочного этапа НТО Junior 2023/24 «Диалоги»

Реши сам

8.10. Сценарий бота-библиотекаря задан стрелками: $1 \rightarrow 2, 1 \rightarrow 3; 2 \rightarrow 4, 2 \rightarrow 5; 3 \rightarrow 6; 4 \rightarrow 7; 5 \rightarrow 8; 6 \rightarrow 9; 7 \rightarrow 10; 8 \rightarrow 9; 9 \rightarrow 10$. Начало – реплика 1, конец – реплика 10. Сколько веток у диалога, какая длина самой длинной и самой короткой ветки (длина – число реплик)? Запиши три числа через пробел.

8.11. Сценарий бота «Кинозал» задан стрелками: $1 \rightarrow 2, 1 \rightarrow 3; 2 \rightarrow 4, 2 \rightarrow 5, 2 \rightarrow 6; 3 \rightarrow 7; 4 \rightarrow 8; 5 \rightarrow 8; 6 \rightarrow 10; 7 \rightarrow 9; 8 \rightarrow 9; 9 \rightarrow 10$. Начало – реплика 1, конец – реплика 10. В этой задаче длина ветки – число **переходов** (стрелок), а не реплик. Сколько веток у диалога, какая длина самой длинной и самой короткой ветки? Запиши три числа через пробел.

Задача 8.5 Собери промпт

структура промпта, проверка ответа нейросети

Шестиклассник Миша хочет, чтобы нейросеть объяснила ему цикл. Он написал пять кусочков промпта:

- **А.** «Вот мой список оценок: 5, 4, 5, 3, 5. Нужно посчитать, сколько в нём пятёрок.»
- **Б.** «Ответ – не длиннее 15 строк, в конце дай 2 задачи для проверки с ответами.»
- **В.** «Ты – терпеливый учитель информатики. Я учусь в 6 классе и только начал учить Python.»
- **Г.** «Объясни, как работает цикл for в Python.»
- **Д.** «Пользуйся только циклом for и обычными переменными, без готовых функций вроде count.»

а) Сопоставь кусочки с пятью частями промпта: контекст и роль, цель, исходные данные и детали, инструменты и ограничения, формат результата.

- б) Миша отправил все пять кусочков. Нейросеть ответила на три экрана, но хорошо и понятно. Какой кусочек она не выполнила? Запиши букву.
- в) Кусочек, который отвечает на вопрос «что сделать», – какой? Без какого кусочка промпт вообще нельзя отправлять?

Дано: 5 кусочков промпта; 5 частей структуры.

Найти: соответствие; невыполненный кусочек; обязательный кусочек.

Решение

Шаг 1. Ищем контекст и роль. Кусочек В начинается с «Ты – ...» (это роль) и рассказывает, кто спрашивает и что уже знает (это контекст). Обе половинки – одна часть.

Шаг 2. Ищем цель. Цель – «что сделать», в ней обычно есть глагол-просьба: «объясни», «напиши», «посчитай». Это кусочек Г.

Шаг 3. Ищем исходные данные. Данные – то, с чем нейросеть будет работать: конкретный список оценок. Это кусочек А.

Шаг 4. Ищем ограничения. Ограничения говорят, чем можно пользоваться, а чем нет: «только цикл for, без count». Это кусочек Д.

Шаг 5. Формат. Остался Б: он отвечает на вопрос «в каком виде ответ» – длина и что добавить в конце.

Шаг 6. Что не выполнено. Три экрана – это явно больше 15 строк. Нарушен формат, кусочек Б.

Шаг 7. Без чего нельзя. Без остальных частей ответ будет хуже, но будет. Без цели нейросеть просто не узнает, что от неё хотят. Обязательный кусочек – Г.

 **Легко принять А за цель:** там тоже есть «нужно посчитать». Но цель – что сделать нейросети (объяснить цикл), а в А Миша даёт материал, с которым она будет работать. Это исходные данные.

Ответ: а) А – исходные данные и детали, Б – формат результата, В – контекст и роль, Г – цель, Д – инструменты и ограничения; б) Б; в) Г.

Своя задача по программе сферы, модуль 8

Реши сам

8.12. Дима написал промпт: «Ты – опытный тренер по лыжам. Составь мне план тренировок на неделю. Сделай таблицу: день, что делать, сколько минут». Нейросеть предложила по две тренировки в день по 90 минут – а Диме 11 лет, и он встал на лыжи месяц назад. Какой части промпта не хватило: контекста и роли, цели, исходных данных и деталей, инструментов и ограничений или формата результата?

8.13. Шестиклассница Оля просит нейросеть помочь с каникулами и пишет пять кусочков промпта:

- **А.** «Сделай список из 5 пунктов, в каждом – не больше одной строки.»
- **Б.** «Придумай, чем заняться на зимних каникулах дома.»
- **В.** «Ты – вожатый в детском лагере. Мне 12 лет, на каникулах я буду дома с младшей сестрой, ей 6 лет.»
- **Г.** «Дома есть конструктор, краски и три настольные игры.»
- **Д.** «Ничего не покупать и не выходить из дома.»

а) Сопоставь кусочки с частями промпта: контекст и роль, цель, исходные данные и детали, инструменты и ограничения, формат результата.

б) Нейросеть ответила длинным рассказом на целую страницу. Какой кусочек она не выполнила? Запиши букву.

Задача 8.6* Бот «Выходной»

граф диалога, число путей, сходящиеся ветки

Бот помогает спланировать субботу. Его сценарий – граф из 10 реплик, начало – реплика 1, конец – реплика 10. Ветки здесь не только расходятся, но и сходятся посередине: например, в реплику 6 можно прийти и из 2, и из 3.

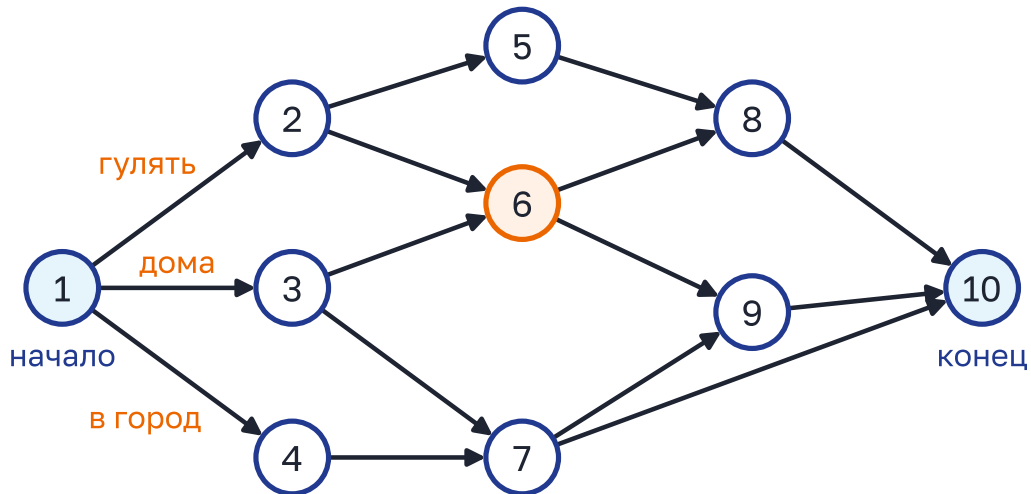


Рисунок 8.10 – Сценарный граф бота «Выходной». Все стрелки ведут слева направо

а) Сколько разных разговоров (веток от 1 до 10) может получиться? б) Сколько из них проходят через реплику 6? Запиши два числа через пробел.

Дано: стрелки: $1 \rightarrow 2, 3, 4$; $2 \rightarrow 5, 6$; $3 \rightarrow 6, 7$; $4 \rightarrow 7$; $5 \rightarrow 8$; $6 \rightarrow 8, 9$; $7 \rightarrow 9, 10$; $8 \rightarrow 10$; $9 \rightarrow 10$.

Найти: число веток от 1 до 10; число веток через 6.

Решение

Шаг 1. Почему не перемножить. Из разных узлов выходит разное число стрелок, и ветки сливаются. Правило умножения из главы 3 здесь не работает. Выписывать все ветки подряд

можно, но легко одну потерять или посчитать дважды.

Шаг 2. Подписываем узлы. У начала пишем 1. Дальше идём слева направо: у каждого узла – сумма чисел у узлов, **из которых** в него ведут стрелки. Смотрим, откуда стрелки входят в узел, а не куда выходят.

Таблица 8.10

Узел	Откуда входят стрелки	Число путей
1	– (начало)	1
2	1	1
3	1	1
4	1	1
5	2	1
6	2, 3	$1 + 1 = 2$
7	3, 4	$1 + 1 = 2$
8	5, 6	$1 + 2 = 3$
9	6, 7	$2 + 2 = 4$
10	7, 8, 9	$2 + 3 + 4 = 9$

Всего веток – число у конца: **9**.

Шаг 3. Ветки через 6. Каждая такая ветка состоит из двух кусков: от 1 до 6 и от 6 до 10. До 6 можно дойти 2 путями (это число у узла 6). От 6 до 10 – тоже 2 пути: 6-8-10 и 6-9-10. Любой первый кусок подходит к любому второму, поэтому перемножаем: $2 \cdot 2 = 4$.

Шаг 4. Проверяем программой. Сценарий запишем словарём: ключ – номер реплики, значение – список, куда из неё ведут стрелки. Номера узлов идут так, что стрелки ведут только от меньшего к большему, поэтому можно пройти узлы по порядку и «раздать» каждое число вперёд по стрелкам.

```
perehody = {1: [2, 3, 4], 2: [5, 6], 3: [6, 7], 4: [7], 5: [8],
            6: [8, 9], 7: [9, 10], 8: [10], 9: [10], 10: []}
puti = {}
for u in range(1, 11):
    puti[u] = 0
puti[1] = 1                                # в начало – один путь
for u in range(1, 11):                    # узлы по порядку, слева направо
    for v in perehody[u]:                  # по каждой стрелке из u
        puti[v] = puti[v] + puti[u]
print(puti)
print("всего веток:", puti[10])           # всего веток: 9
```

Программа печатает те же числа, что в таблице: {1: 1, 2: 1, 3: 1, 4: 1, 5: 1, 6: 2, 7: 2, 8: 3, 9: 4, 10: 9}.

Для недоверчивых – все ветки подряд: 1-2-5-8-10, 1-2-6-8-10, 1-2-6-9-10, 1-3-6-8-10, 1-3-6-9-10, 1-3-7-9-10, 1-3-7-10, 1-4-7-9-10, 1-4-7-10. Их девять, и через 6 идут четыре, со второй по пяту.

❗ **Типичная ошибка.** Сложить стрелки, выходящие из узлов, или перемножить развилки: $3 \cdot 2 \cdot 2 \cdot \dots$. Ещё одна частая ошибка – при подписи узла брать числа у узлов, **куда** ведут стрелки, а не **откуда**. В пункте б) ошибаются, складывая $2 + 2 = 4$ – здесь случайно вышло то же число, но правильное действие – умножение: первый кусок **и** второй кусок.

Ответ: 9 4 .

Своя задача по программе сферы, модуль 10

Реши сам

8.14.* Сценарий бота «Экскурсия» задан стрелками: $1 \rightarrow 2, 1 \rightarrow 3; 2 \rightarrow 4, 2 \rightarrow 5; 3 \rightarrow 5, 3 \rightarrow 6; 4 \rightarrow 7; 5 \rightarrow 7, 5 \rightarrow 8; 6 \rightarrow 8; 7 \rightarrow 9; 8 \rightarrow 9$. Начало – 1, конец – 9. Сколько разных веток от 1 до 9? Сколько из них проходят через реплику 5? Запиши два числа через пробел.

8.15.* Сценарий бота «Кружки» задан стрелками: $1 \rightarrow 2, 1 \rightarrow 3, 1 \rightarrow 4; 2 \rightarrow 5; 3 \rightarrow 5, 3 \rightarrow 6; 4 \rightarrow 6; 5 \rightarrow 7, 5 \rightarrow 8; 6 \rightarrow 7; 7 \rightarrow 8$. Начало – реплика 1, конец – реплика 8. а) Сколько разных веток от 1 до 8? б) Сколько из них **не** проходят через реплику 7? Запиши два числа через пробел.

Задача 8.7* Нейросеть проверяет нейросеть

вайбкодинг, поиск ошибки, точность классификации

Нейросеть разметила 12 отзывов о школьной столовой: «+» – отзыв хороший, «-» – плохой. Учитель разметил те же отзывы сам. Чтобы узнать точность нейросети, Катя попросила другую нейросеть: «Ты – программист на Python. У меня два списка одинаковой длины: правильные метки и метки нейросети. Напиши программу, которая печатает точность в процентах».

Ответ:

```
uchitel = ["+", "-", "+", "+", "-", "+", "-", "-", "+", "+", "-", "+"]
model   = ["+", "-", "+", "-", "-", "+", "-", "+", "+", "+", "-", "-"]
sovpalo = 0
for i in range(1, len(uchitel)):
    if uchitel[i] == model[i]:
        sovpalo = sovpalo + 1
print(sovpalo / len(uchitel) * 100)
```

а) Какое число напечатает программа? б) Какова настоящая точность нейросети? Округли до целого процента. в) Какую строку и как нужно исправить?

Дано: 12 пар меток; код из 7 строк.

Найти: вывод программы; настоящую точность, %; исправление.

Решение

Шаг 1. Настоящая точность – сначала руками. Сравниваем списки по парам, номера с 0. Выпишем их друг под другом:

номер:	0	1	2	3	4	5	6	7	8	9	10	11
учитель:	+	-	+	+	-	+	-	-	+	+	-	+
нейросеть:	+	-	+	-	-	+	-	+	+	+	-	-
совпало?	да	да	да	нет	да	да	да	нет	да	да	да	нет

Не совпали номера 3, 7 и 11 – три пары. Совпало $12 - 3 = 9$. Точность – доля верных ответов: $\frac{9}{12} \cdot 100\% = 75\%$.

Шаг 2. Что делает код. Цикл идёт по `range(1, len(uchitel))`, то есть `range(1, 12)`: номера 1, 2, ..., 11. Номер 0 **пропущен!** Пара номер 0 совпадает («+» и «+»), но программа её не видит и насчитает `sovpaLo = 8`. Делит она при этом на все 12: $\frac{8}{12} \cdot 100 = 66,666\ldots$ Python напечатает `66.66666666666666`.

Шаг 3. Исправление. В строке 4 цикл должен начинаться с нуля: `for i in range(len(uchitel)):` (или `range(0, len(uchitel))`). После исправления программа печатает `75.0` – как у нас руками.

Похоже ли на правду? 75 % – это ошибка в каждом четвёртом отзыве: 3 из 12. Для коротких отзывов, где бывает ирония, вполне. А 66,7 % с ручным подсчётом не совпало – на этом программа и попала.

❗ **Самое опасное здесь – поверить программе:** она не упала, число аккуратное. Это третий, самый коварный вид ошибок из раздела 8.6. Заметь: если бы пара номер 0 **не** совпадала, программа выдала бы правильный ответ, и ошибка спряталась бы. Поэтому тест хорошо составлять так, чтобы первый и последний элементы что-то меняли в ответе.

Ответ: а) `66.66666666666666` (то есть 66,7 %); б) 75 %; в) строка 4: `for i in range(len(uchitel)):` – цикл должен начинаться с 0.

Своя задача по программе сферы, модуль 8

Реши сам

8.16. Нейросеть разложила 25 вопросов читателей по трём интендам. Проверка показала: в 4 случаях она выбрала не тот интенд. Какова точность нейросети в процентах?

8.17.* Нейросеть написала программу, которая ищет самую высокую температуру за неделю:

```
temp = [-12, -7, -15, -9, -21, -8, -10]
maks = 0
for t in temp:
    if t > maks:
        maks = t
print(maks)
```

а) Что напечатает программа? б) Какой ответ правильный? в) Какую строку нужно исправить и как, чтобы программа работала для любого списка?

Ответы и решения

Сначала реши задачу сам и только потом сверяйся. Не сошлось – найди в решении шаг, на котором твой путь разошёлся с нашим.

К главе 1. Что такое искусственный интеллект

1.1. Ответ: 2132.

Подсказка: в пункте г) число людей – величина, а не номер класса.

1.2. Ответ: 246.

Подсказка: про каждое устройство спроси: оно выполняет правило, которое записал человек, или научилось на примерах?

1.3. Ответ: удалить «Фамилия», «Номер в журнале», «Прошёл в финал»; тип – классификация.

1.4. Ответ: ГБДАВ.

1.5. Ответ: а) 70 %; б) 2 раза.

Подсказка: здесь спрашивают про ложные тревоги, а не про пропуски: день, когда снег был, а модель его не обещала, в пункт б) не входит.

1.6. Ответ: а) 97 %; б) 12.

Подсказка: «ленивая» модель права про каждый неопасный астероид и ошибается про каждый опасный.

1.7. Ответ: у модели А средняя ошибка 1,8 °С, у модели Б – 1,2 °С; лучше модель Б.

Подсказка: ошибки складывай без знака минус, иначе промах вверх и промах вниз дадут ноль.

1.8. Ответ: 100.

Подсказка: мороз –30 °С сильнее, чем –25 °С, так что во второй день вход «мороз» равен 1.

1.9. Ответ: $S = 2$, выход 1.

Подсказка: входы здесь не 0 и 1, а дробные числа; сумма вышла ровно на пороге – по условию это выход 1.

1.10. Ответ: 300.

1.11. Ответ: 36.

Подсказка: спрашивают число ошибок, а не верных ответов: доля ошибок $100 - 85 = 15$ %.

1.12. Ответ: 86 %.

1.13. Ответ: ВННВН.

Подсказка: в утверждении 5 вспомни, зачем нужна проверочная выборка.

1.14. Ответ: 4 значения, от 20 до 23.

1.15. Ответ: $w_2 = 3$; на входы (7; 1) выход 1.

Подсказка: здесь неизвестен вес, а не порог: из каждого примера получи неравенство для w_2 .

К главе 2. Множества и логика

2.1. Ответ: 5 учеников.

Подсказка: задача «наоборот»: сначала найди, сколько учеников любят хотя бы один предмет, а потом – пересечение.

2.2. Ответ: 7; 17.

Подсказка: общая часть не больше меньшей группы, а хотя бы в одну секцию ходят не больше 30 человек – весь класс.

2.3. Ответ: 10.

2.4. Ответ: $n(A \cap B) = 3, n(A \cup B) = 13$.

Подсказка: общие – числа, которые делятся и на 2, и на 3.

2.5. Ответ: 3 мультфильма.

2.6. Ответ: 10.

Подсказка: сначала найди общую часть, потом вычти её из B .

2.7. Ответ: 2700 страниц.

Подсказка: неизвестно одно из множеств, а пересечение дано – вырази n (Обское море) из формулы; пересечение здесь прибавляют.

2.8. Ответ: БАГВ.

2.9. Ответ: 1345.

2.10. Ответ: 0.

2.11. Ответ: 2 школьника.

2.12. Ответ: 22 записи.

Подсказка: из каждой пары вычти центр – записи с тремя метками входят во все три пары.

2.13. Ответ: 4.

Подсказка: раздели перебор на два случая: $A = 0$ и $A = 1$.

2.14. Ответ: 3.

Подсказка: «НЕ (X чётное)» – это « X нечётное».

К главе 3. Комбинаторика и вероятность

3.1. Ответ: 1000.

Подсказка: цифры на колёсиках могут повторяться – код 555 тоже годится.

3.2. Ответ: 24.

3.3. Ответ: 20.

3.4. Ответ: 27.

3.5. Ответ: 720.

3.6. Ответ: 18.

Подсказка: посчитай все порядки и вычти те, где Петя первый.

3.7. Ответ: 336.

Подсказка: считай по местам, но остановись на третьем – медалей три, а команд восемь.

3.8. Ответ: 21.

Подсказка: партия Ани с Борей и партия Бори с Аней – одна и та же.

3.9. Ответ: 33 %.

Подсказка: «больше 4» – это 5 и 6, сама четвёрка не подходит.

3.10. Ответ: 60 %.

3.11. Ответ: 17 %.

3.12. Ответ: 87,5 %.

Подсказка: проще через противоположное событие – «ни одного орла» бывает только одним способом.

3.13. Ответ: около 650 раз.

Подсказка: сначала найди частоту в опыте, потом умножь её на число новых бросков.

3.14. Ответ: около 30 ошибок.

Подсказка: спрашивают про ошибки, а не про верные ответы.

3.15. Ответ: лиса – 10 %, модель назовёт «кошка».

Подсказка: вероятности всех классов вместе дают 100 %.

3.16. Ответ: «вечером дома», 32 %.

3.17. Ответ: 48 %.

Подсказка: «отлично» встречается в трёх фразах – найди вероятность каждой и сложи.

К главе 4. Закономерности, последовательности, уравнения

4.1. Ответ: 1.

Подсказка: ряд убывает, но разности разные – проверь отношения: каждый раз делим на 3.

4.2. Ответ: 63.

Подсказка: разности 2, 4, 8, 16 сами растут не на одно и то же, а удваиваются.

4.3. Ответ: на 8-й день.

Подсказка: это ряд с умножением на 3; выпиши просмотры по дням, начиная с 5 в первый день.

4.4. Ответ: 121 клетка.

4.5. Ответ: на 15-й странице.

4.6. Ответ: 41-м.

Подсказка: обратная задача – сначала число прыжков $(250 - 10) : 6$, а номер на единицу больше.

4.7. Ответ: $x = 10$.

4.8. Ответ: 10 минут.

4.9. Ответ: 1 – Б, 2 – Г, 3 – А, 4 – В.

Подсказка: «удвоенная сумма» – сначала сложить, потом удвоить, поэтому нужны скобки; проверь выражения на $x = 5$.

4.10. Ответ: 27.

Подсказка: сначала оба умножения, потом вычитание.

4.11. Ответ: через 10 минут.

Подсказка: b здесь дано сразу – это температура в начале, 20°C .

4.12. Ответ: 21 апреля.

Подсказка: с 1 по 6 апреля прошло 5 дней, а не 6; в ответе – число месяца, а не число дней таяния.

4.13. Ответ: а) $y = 2x - 10$; б) 44 л; в) +3 л (модель занизила).

4.14. Ответ: а) $y = -3x + 76$; б) на 12-й; в) -2 с (модель завысила).

Подсказка: время круга уменьшается, поэтому k отрицательный; вопрос б) – это уравнение $-3x + 76 = 40$.

4.15. Ответ: 1,25 мин.

4.16. Ответ: а) А – 0,5 мин, Б – 2 мин; б) модель А; в) 50 мин.

Подсказка: если сложить ошибки со знаком, у обеих моделей выйдет по 0,5 мин – знак отбрасывают до сложения.

4.17. Ответ: а) 15 деталей; б) в мастерской А, на 70 ₽.

4.18. Ответ: 25 минут.

Подсказка: у тарифа Б старт бесплатный, его цена – просто 10 х.

К главе 5. Данные: собираем и наводим порядок

5.1. Ответ: 1А, 2Б, 3В, 4Г.

5.2. Ответ: 1Г, 2В, 3Б, 4А.

5.3. Ответ: 2 – «Температура» и «Скорость ветра».

5.4. Ответ: первой окажется 03.11.2026, а самая ранняя – 15.09.2026.

Подсказка: текст сравнивается знак за знаком слева, а первым в такой дате стоит день, а не год.

5.5. Ответ: сначала Марат, потом 3.

5.6. Ответ: 30 записей, по 3 запятые.

Подсказка: первая строка файла – заголовок, а не запись.

5.7. Ответ: 14 страниц, на последней 70 записей.

5.8. Ответ: а) 3 параметра; б) с 101-й по 150-ю запись.

Подсказка: страница n начинается с записи $(n - 1) \cdot 50 + 1$ и кончается записью $n \cdot 50$.

5.9. Ответ: соберёт 2200, потеряет 150.

Подсказка: `range(1, 12)` перебирает страницы только с 1-й по 11-ю, а их нужно 12.

5.10. Ответ: на 13,7 мин.

Подсказка: найди два средних – с выбросом и без него – и вычти одно из другого.

5.11. Ответ: 32 строки.

Подсказка: копия строки с пропуском уходит вместе с остальными копиями – не вычитай её второй раз.

5.12. Ответ: 29 400 и 21 600.

Подсказка: сначала вычти удалённые строки и столбцы, потом умножай.

5.13. Ответ: в среднем 6 задач; больше 5 задач решили двое.

Подсказка: «больше 5» – значит 6 и выше, ровно 5 не подходит.

5.14. Ответ: команда Д.

Подсказка: две команды с равной высотой занимают два места подряд, а не одно.

5.15. Ответ: а) «Пирожок», 22 шт.; б) чай, 28 шт.

5.16. Ответ: а) 7; б) 6А, 46 кг; в) 5В, 14,3 кг.

Подсказка: в «6 А» пробел стоит посередине, и strip его не уберёт – пробелы надо убрать все. Лидер по сумме и лидер по среднему здесь разные.

К главе 6. Графики и анализ данных

6.1. Ответ: а) 3 месяца; б) одинаково, в обоих 10 °С.

Подсказка: март (–8) и ноябрь (–7) не подходят: они теплее –10.

6.2. Ответ: 10; 130.

6.3. Ответ: 1 – Б, 2 – В, 3 – Г, 4 – А.

6.4. Ответ: круговая – для первого опроса; для второго – столбчатая.

Подсказка: сложи проценты: в первом опросе вышло 100 %, во втором 165 % – ответы пересекались.

6.5. Ответ: а) «Космос»; б) «ИИ»; в) «Космос».

6.6. Ответ: 5 книг.

Подсказка: сначала найди, сколько книг прочитал каждый класс целиком. Ответ 5,5 – ловушка: классы разного размера.

6.7. Ответ: Г6.

6.8. Ответ: а) 60 учеников; б) 108°.

Подсказка: угол 90° – это четверть круга (360°), а 30 % – доля от 360°.

6.9. Ответ: а) в 2 раза; б) в 1,07 раза.

6.10. Ответ: а) между февралём и мартом (+600); б) на 27 %.

Подсказка: проценты считай от апреля – от того, что было до изменения.

6.11. Ответ: среднее 23 см, медиана 20 см, размах 27 см.

6.12. Ответ: медиана 13,5 мин, среднее 17 мин.

Подсказка: чисел восемь – чётное количество, медиана равна среднему двух средних чисел.

6.13. Ответ: 18 баллов.

Подсказка: иди от сумм: сумма = среднее · количество, было $5 \cdot 12$, стало $6 \cdot 13$.

6.14. Ответ: А.

Подсказка: время бега уменьшается – значит, бегают быстрее.

6.15. Ответ: 2 и 4.

К главе 7. Нейросети своими руками

7.1. Ответ: 25.

7.2. Ответ: 90; 103.

Подсказка: связи считай для каждой пары соседних слоёв – их тут три; смещения есть у всех нейронов, кроме входов.

7.3. Ответ: а) 14; б) 12.

Подсказка: обозначь число нейронов скрытого слоя буквой h : связей $10 \cdot h + h \cdot 4$; во втором пункте не забудь связи между двумя скрытыми слоями, $h \cdot h$.

7.4. Ответ: 0; 0; 2,5.

Подсказка: ноль не отрицательный – ReLU оставляет его нулём.

7.5. Ответ: 5.

7.6. Ответ: 40; 31.

7.7. Ответ: BHBVN.

Подсказка: в пункте 5 связи перемножают, а не складывают.

7.8. Ответ: 144; 5; 2980.

Подсказка: каждый пиксель – отдельный вход, поэтому входов $12 \cdot 12$, а не $12 + 12$.

7.9. Ответ: ВБА.

Подсказка: картинка 8×8 – это 64 входа, а не 8.

7.10. Ответ: 150; 50.

Подсказка: в а) сложи, сколько не хватает каждому из двух меньших классов.

7.11. Ответ: Б.

Подсказка: найди, чем все фото лис отличаются от всех фото собак, кроме самих зверей.

7.12. Ответ: БГВАД.

Подсказка: проверять можно только обученную модель, а выгружать – только проверенную.

7.13. Ответ: 6000.

Подсказка: за одну эпоху поправок столько же, сколько примеров.

7.14. Ответ: 3,8.

7.15. Ответ: 2,92.

Подсказка: во второй эпохе ответ модели уже больше верного: ошибка отрицательная, и вес уменьшается.

7.16. Ответ: 0111.

7.17. Ответ: 1001.

Подсказка: порог выхода 0 – ступенька отвечает 1 уже при сумме 0.

К главе 8. Python, диалоги и промпты

8.1. Ответ: 8 12 .

8.2. Ответ: 34 , 7 , лалала .

8.3. Ответ: программа напечатает 4 10 ; заказать нужно 5 микроавтобусов.

Подсказка: // – сколько автобусов заполнятся целиком, % – сколько человек осталось; их тоже надо везти, поэтому округляем вверх.

8.4. Ответ: серебряный значок .

Подсказка: из цепочки if – elif – else выполняется только первая верная ветка – дальше проверки не идут.

8.5. Ответ: 30 .

8.6. Ответ: 7 430 .

Подсказка: while проверяет условие перед кругом: при 390 условие ещё верно, и цикл делает ещё один круг.

8.7. Ответ: 20 50 5 и 210 .

Подсказка: номера в списке начинаются с нуля, поэтому a[1] – второй элемент.

8.8. Ответ: 13 A o 3 .

Подсказка: count("o") ищет только букву «о»: «А» и «а» в начале слова – другая буква.

8.9. Ответ: «меню».

Подсказка: slovo in fraza ищет кусочек, а не целое слово, – ключевое «есть» спряталось внутри «шесть».

8.10. Ответ: 3 6 5 .

8.11. Ответ: 4 5 3 .

Подсказка: здесь длина – переходы, их в каждой ветке на один меньше, чем реплик.

8.12. Ответ: контекста (часть «контекст и роль»).

Подсказка: роль («тренер»), цель и формат в промпте есть, а о самом Диме – ни слова: ни возраста, ни опыта. Это и есть контекст – что нейросеть должна знать о тебе. Запрет «не больше 30 минут в день» тоже помог бы, но нейросеть ошиблась потому, что вообще не знала, для кого план.

8.13. Ответ: а) А – формат результата, Б – цель, В – контекст и роль, Г – исходные данные и детали, Д – инструменты и ограничения; б) А.

8.14. Ответ: 6 4 .

8.15. Ответ: 6 2 .

Подсказка: сначала посчитай ветки через 7 (пути до 7 умножить на пути от 7 до конца) и вычти их из всех веток.

8.16. Ответ: 84 %.

Подсказка: в условии дано число ошибок, а точность – доля верных ответов.

8.17. Ответ: а) 0 ; б) -7; в) строка 2: `maxs = temp[0]` .

Подсказка: все температуры ниже нуля – посмотри, с какого числа начинается `maxs` .

Всё, что может понадобиться под рукой во время решения: формулы и правила из глав, мини-шпаргалка по Python, словарь слов из мира ИИ и все «Запомни» из глав на нескольких страницах.

Формулы и правила

Если в условии задачи на отборе правило записано по-своему, решай по условию.

Множества и логика

$n(A)$ – число элементов в множестве A . Знак $\cup$ читается «или» (хотя бы в одном из множеств), знак $\cap$ – «и» (в обоих сразу).

Хотя бы в одном из двух (формула включений и исключений):

$$n(A \cup B) = n(A) + n(B) - n(A \cap B).$$

На робототехнику ходят 18 человек, на шахматы 15, на оба кружка 7. Хотя бы в один кружок ходят $18 + 15 - 7 = 26$.

В обоих сразу, если известно, сколько хотя бы в одном:

$$n(A \cap B) = n(A) + n(B) - n(A \cup B).$$

Книг про космос 12, про роботов 9, хотя бы про что-то одно из двух – 17. Про то и другое сразу: $12 + 9 - 17 = 4$.

Ни в одном: из всех вычтешь тех, кто хотя бы в одном. В классе 30 человек, значит, ни в один кружок не ходят $30 - 26 = 4$.

Хотя бы в одном из трёх:

$$\begin{aligned} n(A \cup B \cup C) = & \\ & n(A) + n(B) + n(C) \\ & - n(A \cap B) - n(A \cap C) - n(B \cap C) \\ & + n(A \cap B \cap C). \end{aligned}$$

Общую часть всех трёх сначала трижды прибавили, потом трижды вычли – поэтому в конце прибавляем её ещё раз.

Логические связи: «истина» – 1, «ложь» – 0.

Таблица С.1

A	B	$A \text{ и } B$	$A \text{ или } B$	НЕ A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0

A	B	$A \text{ и } B$	$A \text{ или } B$	НЕ A
1	1	1	1	0

«И» истинно, только когда верны оба. «ИЛИ» ложно, только когда неверны оба. В Python это `and`, `or`, `not`.

Комбинаторика

Таблица С.2

Правило	Формула	Пример
Правило умножения: выбираем по шагам, на каждом шаге свой выбор	$n_1 \cdot n_2 \cdot n_3 \cdot \dots$	3 каши, 2 напитка, 4 булочки: $3 \cdot 2 \cdot 4 = 24$ разных завтрака
Правило сложения: выбираем что-то одно из разных групп	$n_1 + n_2$	5 комедий или 3 мультфильма: $5 + 3 = 8$ вариантов на вечер
Перестановки: сколькими способами выстроить n предметов в ряд	$n! = 1 \cdot 2 \cdot \dots \cdot n$	4 робота в очередь на зарядку: $4! = 24$
Выбор по порядку: k мест из n участников	$n \cdot (n - 1) \cdot \dots$ (k множителей)	1, 2 и 3 место из 10 команд: $10 \cdot 9 \cdot 8 = 720$
Пары, где порядок не важен	$\frac{n \cdot (n - 1)}{2}$	5 человек пожали друг другу руки: $\frac{5 \cdot 4}{2} = 10$ рукопожатий

Вероятность

$$P(A) = \frac{m}{n},$$

где m – число подходящих исходов, n – число всех равновозможных исходов.

Таблица С.3

Правило	Формула	Пример
Вероятность всегда от 0 до 1	$0 \leq P \leq 1$	0 – не бывает никогда, 1 – бывает всегда
Выпадет чётное на кубике	$\frac{3}{6} = \frac{1}{2}$	0,5 = 50%
Событие не случится	$P(\text{не } A) = 1 - P(A)$	если $P = 0,3$, то «не случится» = 0,7
Оба независимых события сразу	$P(A) \cdot P(B)$	две монеты, обе орлом: $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$, то есть 25 %

Дробь в проценты: умножь на 100. $\frac{1}{4} = 0,25 = 25\%$.

Последовательности, уравнения, проценты

Арифметическая прогрессия – каждый раз прибавляем одно и то же число d :

$$a_n = a_1 + (n - 1) \cdot d.$$

Ряд 5, 8, 11, ...: $d = 3$, десятый член $a_{10} = 5 + 9 \cdot 3 = 32$.

Сумма первых n членов арифметической прогрессии:

$$S_n = \frac{(a_1 + a_n) \cdot n}{2}.$$

$$5 + 8 + \dots + 32 = \frac{(5 + 32) \cdot 10}{2} = 185.$$

Геометрическая прогрессия – каждый раз умножаем на одно и то же число q :

$$b_n = b_1 \cdot q^{n-1}.$$

Ряд 3, 6, 12, ...: $q = 2$, шестой член $b_6 = 3 \cdot 2^5 = 96$.

Линейное уравнение:

$$a \cdot x + b = c \Rightarrow x = \frac{c - b}{a}.$$

$$3x + 7 = 22: x = \frac{22 - 7}{3} = 5. \text{ Проверка: } 3 \cdot 5 + 7 = 22.$$

Проценты. p процентов от числа A : $A \cdot \frac{p}{100}$. 15 % от 600: $600 \cdot 0,15 = 90$. Какую часть A составляет от B : $\frac{A}{B} \cdot 100\%$. 12 из 48: $\frac{12}{48} \cdot 100\% = 25\%$.

Среднее, медиана и другие

Таблица С.4

Что	Как найти
Среднее арифметическое	сложить все значения и разделить на их количество
Медиана	упорядочить по возрастанию и взять число посередине; если чисел чётное количество – среднее двух средних
Мода	значение, которое встречается чаще всех
Размах	наибольшее минус наименьшее

Пример. Ряд 3, 7, 8, 12, 40. Среднее: $(3 + 7 + 8 + 12 + 40) : 5 = 70 : 5 = 14$. Медиана: 8. Размах: $40 - 3 = 37$. Для ряда 3, 7, 8, 12 медиана $(7 + 8) : 2 = 7,5$.

Совет. Одно странное значение (выброс) сильно тянет среднее за собой, а медиану почти не трогает. В примере выше среднее 14 больше четырёх чисел из пяти – всё из-за числа 40. Если в данных есть выбросы, смотри на медиану.

Насколько хороша модель

Точность (accuracy) – для задач классификации, где ответ «кошка» или «собака»:

$$\text{точность} = \frac{\text{число верных ответов}}{\text{число всех примеров}} \cdot 100\%.$$

Модель верно узнала 45 картинок из 50: $\frac{45}{50} = 0,9 = 90\%$.

Ошибка прогноза – для регрессии, где модель предсказывает число. Для каждого примера берём разницу между прогнозом и настоящим значением без знака минус. **Средняя абсолютная ошибка** (MAE) – среднее этих разниц. **Средний квадрат ошибки** (MSE) – среднее их квадратов.

Таблица С.5

Настоящее значение	Прогноз	Разница без знака	Квадрат разницы
10	11	1	1
12	12	0	0
15	13	2	4

$MAE = (1 + 0 + 2) : 3 = 1$, $MSE = (1 + 0 + 4) : 3 \approx 1,67$. У точности лучше та модель, у которой число **больше**, у ошибки – та, у которой **меньше**.

Обучающая и тестовая выборки. Данные делят на две части: на одной модель учится, на другой её проверяют. Часто делят в пропорции 80 на 20: из 200 строк 160 идут на обучение, 40 – на проверку.

Как составить промпт

В программе сферы хороший запрос к нейросети состоит из пяти частей:

Таблица С.6

Часть	Что написать	Пример
Контекст и роль	кем быть нейросети и что она должна знать о тебе	«Ты – учитель информатики для шестиклассников. Я готовлюсь к олимпиаде.»
Цель	что именно сделать	«Объясни, что такое медиана, и дай 3 задачи.»
Исходные данные и детали	с чем работать, важные подробности	«Вот мои числа: 4, 9, 2, 7, 5.»
Инструменты и ограничения	чем можно пользоваться, чего нельзя	«Без формул с буквами, только счёт по порядку.»
Формат результата	в каком виде нужен ответ	«Коротко, до 10 строк, ответы к задачам – в конце.»

Мини-шпаргалка Python

Всё ниже работает в любом онлайн-интерпретаторе Python 3. После `#` идёт комментарий: Python его не читает.

Вывод и переменные

```

name = "Аня"           # строка – в кавычках
age = 12                # целое число
print("Привет,", name) # Привет, Аня
print(f"{name} через год будет {age + 1}") # Аня через год будет 13

```

Арифметика

```

print(7 / 2)    # 3.5 – обычное деление
print(7 // 2)   # 3 – целая часть от деления
print(7 % 2)    # 1 – остаток от деления
print(2 ** 10)  # 1024 – степень
print(round(3.14159, 2)) # 3.14 – округлить до сотых

```

Условие

```

t = -25
if t < -30:
    print("сиди дома")
elif t < 0:
    print("мороз, шапку!")
else:
    print("тепло")

```

Циклы

```

for i in range(1, 4): # i = 1, 2, 3 (до 4, но не включая 4)
    print(i)

n = 1
while n < 100:        # повторяем, пока условие верно
    n = n * 2
print(n)              # 128

```

Списки

```

marks = [5, 4, 5, 3, 5]
marks.append(4)           # добавить в конец
print(len(marks))         # 6 – сколько элементов
print(sum(marks), max(marks), min(marks)) # 26 5 3
print(sum(marks) / len(marks)) # среднее
print(sorted(marks))      # [3, 4, 4, 5, 5, 5]
print(marks[0], marks[-1]) # 5 4 – первый и последний
print(marks.count(5))     # 3 – сколько раз встречается 5

```

Строки

```

s = "Нейросеть учится на данных"
print(len(s))           # 26 – длина вместе с пробелами
print(s.lower())        # все буквы маленькие
print(s.split())        # ['Нейросеть', 'учится', 'на', 'данных']
print("данных" in s)    # True – есть ли такое слово

```

Словарь и функция

```

pets = {"кошка": 12, "собака": 9} # ключ: значение
pets["хотьяк"] = 3
for k, v in pets.items():
    print(k, v)

def accuracy(right, total):
    return right / total * 100

print(accuracy(45, 50)) # 90.0

```

Таблицы и графики (библиотеки `pandas` и `matplotlib` в Colab уже есть, в простом онлайн-интерпретаторе их может не быть)

```

import pandas as pd
import matplotlib.pyplot as plt

df = pd.DataFrame({"месяц": ["январь", "фев", "мар"], "снег_см": [45, 60, 38]})
# или из файла: df = pd.read_csv("data.csv")
print(df.shape) # (3, 2) - строк и столбцов
print(df["снег_см"].mean()) # среднее по столбцу
print(df["снег_см"].median()) # медиана
print(df[df["снег_см"] > 40]) # только строки, где снега больше 40 см

plt.bar(df["месяц"], df["снег_см"]) # столбчатая диаграмма
plt.title("Высота снега")
plt.show()

```

Совет. Пять ошибок, на которых спотыкаются чаще всего: `=` кладёт значение в переменную, `a ==` сравнивает; `range(1, 10)` доходит только до 9; номера в списке начинаются с 0; после `if`, `for`, `while` и `def` нужно двоеточие и отступ в 4 пробела; дробные числа пишутся через точку: 2.5.

Словарик

Таблица С.7

Слово	Что значит
Алгоритм	точная последовательность шагов, которая приводит к результату
API	«окошко» сайта для программ: программа отправляет запрос по адресу и получает данные, часто в формате JSON
Вайбкодинг	способ писать программы, когда код по твоему описанию словами пишет нейросеть, а ты проверяешь и поправляешь
Вес	число при входе нейрона: насколько этот вход важен. При обучении веса подбираются сами

Слово	Что значит
Выброс	значение, резко не похожее на остальные: рост ученика 5 метров, возраст 150 лет
Выборка обучающая и тестовая	на обучающей модель учится, на тестовой её проверяют. Тестовые примеры модель до проверки не видит
Галлюцинация нейросети	уверенный, гладкий, но выдуманный ответ. Поэтому ответы нейросети всегда проверяют
Гиперпараметр	настройка, которую человек выбирает до обучения: число эпох, скорость обучения, число слоёв и нейронов
Гипотеза	предположение, которое проверяют по данным: «зимой в библиотеке берут больше книг»
Дашборд	один экран, на котором собраны несколько графиков и главные числа
Данные категориальные	значения-названия: цвет, порода, город. Их сравнивают, но не складывают
Данные количественные	значения-числа: рост, цена, температура. С ними можно считать
Датасет (набор данных)	таблица с примерами, на которых учат и проверяют модель
Искусственный интеллект (ИИ)	программы, которые решают задачи, где человеку пришлось бы подумать: узнать предмет на фото, понять вопрос, сделать прогноз
Классификация	задача «к какому классу относится»: кошка или собака, спам или не спам
Кластеризация	задача «разбей на группы похожих», когда заранее неизвестно, какие будут группы
Малокодовая система (low-code)	инструмент, где модель или программу собирают мышкой из блоков, почти без кода
Машинное обучение	способ сделать ИИ: правила программе не пишут, а показывают много примеров, и она сама находит закономерность
Метрика	число, по которому человек оценивает модель: точность, средняя ошибка
Модель	то, что получилось после обучения: программа, которая по входным данным выдаёт ответ
Нейрон (искусственный)	маленький вычислитель: умножает каждый вход на свой вес, складывает и пропускает через функцию активации

Слово	Что значит
Нейросеть	много нейронов, соединённых слоями: входной слой получает данные, скрытые слои их обрабатывают, выходной выдаёт ответ
Переобучение	модель «вызубрила» обучающие примеры и плохо работает на новых
Признак	то, по чему модель делает вывод; в таблице это столбец: рост, вес, цвет
Промпт	запрос к нейросети словами
Пропуск	пустая клетка в таблице: значение не записали
Регрессия	задача «предскажи число»: температура завтра, цена квартиры
Скорость обучения	гиперпараметр: насколько сильно меняются веса за один шаг обучения
Функция активации	правило, по которому нейрон решает, что передать дальше
Функция потерь	число, которое показывает, насколько модель ошибается. Обучение – это уменьшение потерь
Целевая переменная	то, что модель должна угадать: порода, цена, «спам или нет»
Чат-бот	программа, которая ведёт диалог с человеком
Эпоха	один проход обучения по всей обучающей выборке
CSV	таблица, записанная как текст: одна строка – одна запись, значения через запятую или точку с запятой
JSON	формат данных в фигурных скобках, пары «ключ: значение». Так отвечают многие API

Шпаргалка: всё «Запомни» из глав

Глава 1. Что такое искусственный интеллект

Запомни. ИИ – программы для задач, где человеку нужен интеллект. Машинное обучение – та часть ИИ, которая учится на примерах, а нейросети – часть машинного обучения. Если программа только выполняет правила, записанные человеком (звонок, калькулятор, светофор по таймеру), – это не ИИ.

Запомни. Строка таблицы – объект. Столбцы, по которым угадываем, – признаки, а столбец, который надо угадать, – ответ (целевая переменная). Номера и фамилии, которые лишь называют строку, – не признаки: перед обучением их удаляют вместе с ответом.

Запомни. Готовые группы и ответы в примерах – классификация. Число – регрессия. Ответов нет, ищем группы сами – кластеризация. Первым делом смотри, **что должно получиться на выходе**.

Запомни. Обучающая выборка – учимся, проверочная – сдаём экзамен. Проверочные примеры модель при обучении не видит. Точность, которой можно верить, – только на проверочной выборке.

Запомни. Нейрон: умножь каждый вход на его вес, сложи, сравни сумму с порогом. $S \geq T$ – выход 1, иначе 0. Отрицательный вес работает «против». Обучение – это подбор весов.

Запомни. Точность = верных : всего · 100 %. Доля ошибок = 100 % – точность. Считаем на проверочной выборке. Для регрессии – средняя ошибка: насколько в среднем промахнулись, без учёта знака.

Глава 2. Множества и логика

Запомни. Множество – набор без повторов, порядок в нём не важен. $n(A)$ – сколько в нём элементов. Подмножество – часть, все элементы которой есть в большом множестве.

Запомни. $A \cap B$ – «и то, и другое», $A \cup B$ – «хотя бы одно», « A , но не B » – круг A без общей части, «не A » – всё за пределами круга A . Одно понятие может занимать и один кусочек картинки, и несколько.

Запомни. Хотя бы в одном = $n(A) + n(B) - n(A \cap B)$. Ни в одном = все минус хотя бы в одном. Для трёх кругов заполняй кусочки от центра наружу, а в конце проверь сумму формулой.

Запомни. И – истина, только если истинны оба. ИЛИ – ложь, только если ложны оба. НЕ – наоборот. Порядок: скобки, НЕ, И, ИЛИ. Для k высказываний в таблице истинности 2^k строк: 4, 8, 16...

Запомни. & – И – пересечение: найдётся меньше. | – ИЛИ – объединение: найдётся больше. Пересечение по трём числам: $n(A \cap B) = n(A) + n(B) - n(A \cup B)$.

Глава 3. Комбинаторика и вероятность

Запомни. Выбор по шагам – перемножаем, сколько вариантов на каждом шаге: $a \cdot b \cdot c$. Так можно, только если число вариантов на шаге не зависит от того, что выбрали раньше.

Запомни. «И» – берём по варианту на каждом шаге – умножаем. «Или» – берём одно из нескольких групп – складываем. Если у групп есть общие элементы, общее вычитаем один раз.

Запомни. Расставляем по местам: на каждое следующее место вариантов на один меньше. Все n предметов в ряд – $n! = n \cdot (n - 1) \cdot \dots \cdot 1$. Пары, где порядок не важен, – $\frac{n \cdot (n - 1)}{2}$.

Запомни. $P = \frac{\text{подходящие}}{\text{все}}$, и все исходы должны быть равновероятными. $0 \leq P \leq 1$. «Не случится» – $1 - P$. У двух монет 4 исхода, у двух кубиков – 36: каждую монету и каждый кубик считай отдельно.

Запомни. Вероятность – сколько должно выходить в среднем, частота – сколько вышло в нашем опыте. На малом числе опытов они могут сильно различаться, на большом – сближаются. Ожидаемое число случаев = $P \cdot N$.

Запомни. Проценты модели – её оценка, а не факт про один снимок. Честность модели проверяют на многих ответах: из тех, где она «уверена на 80 %», верными должны оказаться около 80 из 100. Вероятность фразы из нескольких слов – произведение вероятностей вдоль её ветки дерева.

Глава 4. Закономерности, последовательности, уравнения

Запомни. Порядок поиска: разности, отношения, разности разностей, знакомые ряды, два ряда вперемешку. Правило, которое подходит к первым двум числам, но не подходит к третьему, – не правило.

Запомни. $a_n = a_1 + (n - 1) \cdot d$: прыжков на один меньше, чем номер. Номер по числу – это (число – первый член) : шаг, и к результату прибавить 1. Не делится нацело – такого числа в ряду нет. В геометрической прогрессии вместо прибавления – умножение: $b_n = b_1 \cdot q^{n-1}$.

Запомни. Переменная – место для числа. Подставил – посчитал в порядке: скобки, умножение и деление, сложение и вычитание. «На ...» – это плюс или минус, «в ... раз» – умножение или деление.

Запомни. Что делаешь с одной частью уравнения, делай и с другой. Действия с x снимаем в обратном порядке. Корень всегда проверяй подстановкой.

Запомни. $y = kx + b$: k – прибавка на каждую единицу x (со знаком), b – значение при $x = 0$. $k = (y_2 - y_1) : (x_2 - x_1)$. Прогноз – подставить x . «Когда станет...» – решить уравнение.

Запомни. Ошибка = настоящее – прогноз: плюс – модель занизила, минус – завысила. Две модели сравнивают по средней ошибке **без знака** – у кого она меньше, та и лучше. Одна странная точка – не повод гнуть прямую, а далеко за краем облака прямой не верь.

Глава 5. Данные: собираем и наводим порядок

Запомни. Строка – объект, столбец – признак (свойство), клетка – одно значение. Строка заголовка – не запись. Число клеток = число строк · число столбцов.

Запомни. Количественные данные можно складывать и усреднять, категориальные – только сравнивать и пересчитывать. Цифры в номере, шифре, индексе – это ярлык, а не количество. Числа из файла сначала переведи из текста в число.

Запомни. CSV: первая строка – заголовок, записей на одну меньше, чем строк; запятых в строке на одну меньше, чем столбцов. JSON: {} – карточка с ключами, [] – список, счёт в списке с нуля.

Запомни. Страниц = всего : на странице, **с округлением вверх**. На последней = всего – (страниц – 1) · на странице. В цикле `range(1, N + 1)` : последнее число в `range` не входит.

Запомни. Четыре беды: пропуск, дубликат, опечатка, выброс. Пропуск не превращай в ноль. Для компьютера «Бердск» и «бердск» – разные слова. Сначала чистка, потом подсчёт.

Запомни. Фильтр – оставить строки по условию, сортировка – расставить по столбцу, подсчёт – свернуть в число: количество, сумма, среднее, максимум, минимум, количество уникальных. «Сумма по группе» и «среднее по группе» – разные вопросы.

Глава 6. Графики и анализ данных

Запомни. Сравнить – столбики. Время – линия. Доли одного целого – круг. Две величины у каждого объекта – точки.

Запомни. Сначала подписи, потом картинка: что по осям, в каких единицах, с какого числа начинается ось. Цена деления – разница соседних подписей, делённая на число промежутков между ними.

Запомни. Медиана – середина **упорядоченного** ряда; при чётном количестве – среднее двух средних. Выброс тянет среднее и почти не трогает медиану. Сумма, среднее, максимум по группам – разные вопросы и разные ответы.

Запомни. Гипотеза – конкретная догадка, которую можно проверить таблицей. Точки полосой вверх или вниз – связь есть, кашей – не видно. Связь не доказывает, что одно вызывает другое.

Запомни. Перед сравнением столбиков посмотри, с какого числа начинается ось. Сравнивай сами числа, а не высоту картинки: во сколько раз – делением, на сколько процентов – от того значения, с которым сравниваешь.

Запомни. Дашборд – главное на одном экране: плитки с числами и несколько графиков. Эссе – вопрос, данные, график, вывод с числами и оговорки. Вывод без чисел – это мнение, а не вывод.

Глава 7. Нейросети своими руками

Запомни. Связи между соседними слоями = произведение числа нейронов в них; связи сети – сумма по всем парам соседних слоёв. Смещения есть у каждого нейрона, кроме входов. Входов – сколько признаков, выходов – сколько классов, для числа – один выход.

Запомни. Порядок всегда один: сумма «вход · вес» по всем входам, плюс смещение, потом функция активации. Считаем слой за слоем: входами выходного нейрона служат **ответы** скрытых нейронов, а не исходные x .

Запомни. Ошибка = верный ответ – ответ модели. Новый вес = вес + скорость · ошибка · вход. Эпоха – один проход по всем обучающим примерам. Большая скорость – ошибка скачет и растёт, маленькая – учимся медленно.

Запомни. Переобучение – на обучении точность растёт, а на проверке падает. Лучшая эпоха – та, где **выше всего точность на проверке**, а не на обучении. Низкая точность на обеих выборках – недообучение.

Запомни. Low-code: собрал примеры по классам → обучил → проверил на новых примерах. Классы – сопоставимо, без сильного перекоса, примеры – разные. Ответ сети – класс с наибольшей уверенностью.

Глава 8. Python, диалоги и промпты

Запомни. `=` кладёт значение в переменную, новое вытесняет старое. Правая часть считается целиком, а уже потом результат кладётся в коробку слева. `input` всегда отдаёт текст – для чисел оборачивай его в `int(...)`.

Запомни. После `if`, `elif`, `else`, `for`, `while` – двоеточие, а строки внутри – с отступом. Из цепочки `if` – `elif` – `else` выполняется только первая подходящая ветка. `range(1, 6)` – это 1, 2, 3, 4, 5, без шести.

Запомни. Номера в списке и в строке начинаются с 0, последний элемент – `[-1]`, а номер последнего равен длине минус один. Словарь ищет по ключу, ключи не повторяются. "Снег" и "снег" для Python – разные слова.

Запомни. Узел – реплика, стрелка – переход, ветка – путь от начала до конца. Длина ветки – число реплик в ней. Число веток = число у конца, если подписать у каждого узла сумму чисел узлов, из которых в него входят стрелки. Интент – намерение человека, которое бот узнаёт по фразе.

Запомни. Промпт = контекст и роль + цель + исходные данные + ограничения + формат. Чего не написал – нейросеть додумает сама. Любой ответ нейросети – черновик, пока ты его не проверил.

Запомни. Код от нейросети проверяют запуском на тестах, где ответ известен заранее. Падение программы – это подсказка, где искать. Самая опасная ошибка – когда программа работает и выдаёт правдоподобное неверное число.

Где что найти

Таблица С.8

Что	Где
Правила олимпиады, сферы, сроки	junior.ntcontest.ru
Личный кабинет участника	my.ntcontest.ru
Курс сферы «ИИтех» (за тест курса +5 баллов к отбору)	academy.junior.online , вход через «Талант»
Платформа отбора	orbita.education (заходить кнопкой из личного кабинета)
Памятки для Новосибирской области	nto.kvantorium54.ru
GraphNet – как установить	teletype.in/@yupest/graphnet
Датавиз-конструктор – как установить	teletype.in/@irk.digitalwave/dataviz-install
Python и таблицы в браузере	colab.research.google.com (запускать код – с аккаунтом Google)
Модель на своих картинках и звуках без кода	Teachable Machine, Machine Learning for Kids
CodeChef – блочная программа сбора данных (инструмент курса)	codechef.ru
Scratch – дополнительно, не инструмент курса	scratch.mit.edu

Что	Где
ИИ-чаты из программы сферы	GigaChat, Алиса – вместе со взрослым

Об авторе-составителе



Артём Осиновский

Методист и разработчик детского технопарка «Кванториум» (Новосибирск). Придумывает образовательные программы и сам делает для них цифровые инструменты: платформу смены «Миссия на Луну» для фестиваля «Технопром-2026», сайт с памятками НТО Юниоры для Новосибирской области. Эту книгу собрал для школьников области, чтобы к отбору можно было подготовиться самостоятельно – даже без кружка рядом с

домом.

Нашли ошибку – пишите на osinovsky.aa@kvantorium54.ru (рабочая почта): сфера, версия книги и страница. По другим вопросам и проектам – личная: artem081033@gmail.com.