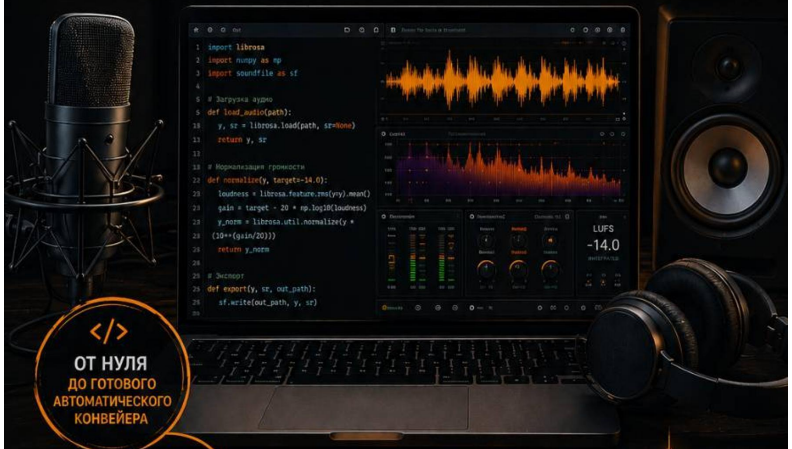
Александр Иванов

PYTHON ДЛЯ ТВОРЧЕСКИХ: ЗВУК НА ТВОЕЙ СТОРОНЕ

Практическое руководство по автоматизации обработки звука от нуля до собственного конвейера профессионального уровня



</>
ОТ НУЛЯ ДО ГОТОВОГО АВТОМАТИЧЕСКОГО КОНВЕЙЕРА

ЗАГРУЗКА И АНАЛИЗ → МОНТАЖ БЕЗ МЫШКИ → ВЫРАВНИВАНИЕ ГРОМКОСТИ → УДАЛЕНИЕ ШУМА → ЧИСТКА ПАУЗ И СЛОВ-ПАРАЗИТОВ → ИЗМЕНЕНИЕ ГОЛОСА → СВЕДЕНИЕ С МУЗЫКОЙ → МАСТЕРИНГ → ГОТОВЫЙ РЕЗУЛЬТАТ

ПИШЕМ СКРИПТЫ. ЭКОНОМИМ ЧАСЫ. ЗВУЧИМ ПРОФЕССИОНАЛЬНО.

Александр Иванов

Python для творческих. Звук на твоей стороне

https://litres.ru/74000411
SelfPub; 2026

Аннотация

Эта книга — не учебник по программированию. Это твой личный звукорежиссёр, который живёт внутри Python. Ты никогда не писал код? Отлично. Здесь ты начнёшь с нуля и уже через час сделаешь первую полезную вещь со своим звуком. Ты гуманитарий и боишься математики? Идеально. В этой книге нет ни одной формулы — только понятные аналогии, живой язык и рабочие скрипты, которые можно копировать и запускать сразу.

Ты узнаешь, как за минуту убрать шум, который сводил тебя с ума. Как выровнять громкость, чтобы твой голос звучал одинаково сочно и в наушниках, и в машине. Как находить и вырезать слова-паразиты, которые подрывают твой авторитет. Как менять голос для озвучки персонажей — от робота до белки. Как накладывать фоновую музыку, которая не заглушает речь, а поддерживает её. И наконец — как собрать всё это в один скрипт, который превращает сырую запись в готовый продукт, пока ты пьёшь кофе.

Девять глав. От первого скрипта до полного конвейера. Никакой магии — только Python.

Содержание

Python для творческих. Звук на твоей стороне

Предисловие

Почему я написал эту книгу

Я сидел в наушниках уже четвёртый час. Передо мной на экране была волновая форма моего голоса — неровная линия с провалами и пиками. Я в сто двадцатый раз выделял мышкой очередное «эээ» и нажимал Delete. Мой первый подкаст должен был выйти три дня назад. Но я всё ещё монтировал его. Не потому, что я перфекционист. А потому что я не знал, что эту работу можно автоматизировать.

Я был программистом. Я писал код каждый день. Я автоматизировал отчёты для бухгалтерии, рассылки для маркетинга, сбор данных для аналитики. Но когда дело касалось моего собственного контента, моего голоса, моего творчества, я почему-то вёл себя как человек, который никогда не слышал об автоматизации. Я сидел и вручную вырезал паузы. Вручную выравнивал громкость. Вручную чистил шум. Как будто Python не существовал. Как будто компьютер был просто дорогим проигрывателем, а не мощнейшим инструментом для работы со звуком.

В какой-то момент я остановился и спросил себя: почему?

Почему я автоматизирую скучные таблицы и не автоматизирую собственный подкаст? Ответ был простым и неприятным: я не знал как. Я умел программировать, но я не знал библиотек для работы со звуком. Я не знал, что звук можно представить в виде массива чисел. Я не знал, что шум можно вычесть математически. Я не знал, что громкость измеряется в LUFS и что под каждую платформу есть свой стандарт. Я был программистом, но в мире звука я был новичком.

Тогда я начал разбираться. Сначала медленно, методом проб и ошибок. Первые скрипты портили звук так, что слушать было невозможно. Голос после шумоподавления превращался в бульканье робота. После ускорения — в писк бурундука. После нормализации громкости становился то тише шёпота, то громче сирены. Но я продолжал. Потому что каждая удачная обработка экономила мне не минуты — часы. Час ручного монтажа превращался в тридцать секунд работы скрипта. Два часа выравнивания громкости — в одну команду. Полдня чистки шума — в запуск функции и ожидание в течение минуты.

Через полгода у меня был набор скриптов, которые делали всю черновую работу за меня. Я записывал подкаст, запускал конвейер, шёл пить кофе. Через десять минут в папке лежал готовый файл. Чистый звук, ровная громкость, никаких паразитов, музыка на фоне ровно там, где нужно, и ровно такой громкости, чтобы не мешать. Мои друзья-подкастеры спрашивали: «Слушай, а кто тебе звук сводит? До-

рого?» Я отвечал: «Python». Они не верили. Тогда я показал им скрипты. И они попросили меня научить их.

Эта книга выросла из тех самых объяснений. Из желания передать знание, которое превращает рутину в магию. Я писал её для людей, которые никогда не программировали. Для блогеров, подкастеров, видеомейкеров, преподавателей, которые записывают лекции — для всех, кто создаёт контент и устал тратить время на техническую обработку. Я писал её так, как объяснял бы другу за чашкой кофе: просто, с аналогиями, без заумных терминов, с шутками там, где это уместно.

Что внутри

В этой книге девять глав. Каждая глава — это один навык. Мы будем двигаться от простого к сложному, и каждая следующая глава опирается на предыдущую, но при этом самодостаточна. Если вам прямо сейчас нужно только убрать шум — открывайте Главу 4, там всё есть. Если нужно только выровнять громкость — Глава 3 к вашим услугам. Книгу можно читать последовательно, а можно использовать как справочник.

В первой главе мы познакомимся со звуком. Узнаем, что для компьютера звук — это просто список чисел, и научимся загружать эти числа в Python. Мы напишем первый скрипт, который расскажет о записи всё: длительность, громкость, количество тишины. Это фундамент.

Во второй главе мы освоим аудиомонтаж кодом. Научим-

ся отрезать лишнее от начала и конца, склеивать несколько файлов в один, добавлять паузы, выравнивать громкость фрагментов. Создадим скрипт, который собирает эпизод подкаста из заготовок.

В третьей главе мы разберёмся с громкостью. Узнаем, что такое LUFS и почему просто сделать громче — плохая идея. Научимся нормализовать громкость под стандарты YouTube, Spotify и подкастов. Освоим компрессию — эффект, который делает голос плотным и профессиональным.

В четвёртой главе мы объявим войну шуму. Освоим три метода шумоподавления: от простого гейта до продвинутого спектрального вычитания. Научимся записывать образец шума и использовать его для очистки всей записи.

В пятой главе мы займёмся чистотой речи. Напишем умный анализатор пауз и инструмент для их сокращения. Создадим детектор слов-паразитов, который находит «эээ», «нууу» и «как бы» по четырём признакам. Научимся либо удалять их автоматически, либо помечать для ручной проверки.

В шестой главе мы перейдём к творческой обработке. Научимся менять скорость речи без изменения высоты голоса и наоборот. Создадим голоса персонажей: робота, демона, белки, великана, инопланетянина. Освоим эффекты телефонного разговора и большого зала. Поймём, как комбинировать эффекты для создания разных голосов из одного.

В седьмой главе мы подружим голос с музыкой. Научимся

автоматически приглушать фоновую музыку, когда вы говорите, и плавно возвращать её в паузах. Освоим частотный дукинг — продвинутую технику, которая освобождает в музыке место для голоса.

В восьмой главе мы сделаем финальную полировку. Применим эквалайзер для формирования красивого частотного баланса. Используем многополосный компрессор для тонкой настройки динамики. Добавим аналоговую теплоту через насыщение. Проведём голос через полный конвейер мастеринга, который превращает хороший звук в студийный.

В девятой главе мы соберём всё вместе в один большой финальный конвейер. Напишем главный скрипт книги, который принимает сырую запись и через один запуск выдаёт полностью готовый продукт. Создадим систему конфигурационных файлов, чтобы не менять код для каждого нового проекта.

В конце книги — словарь терминов, справочник по библиотекам и шпаргалка по командам.

Как читать эту книгу

Книга построена по единому принципу. Каждая глава открывается разделом «О чём эта глава», где я рассказываю, что мы будем делать и зачем. Затем идёт раздел «В чём проблема» — описание ситуации из жизни, с которой вы наверняка сталкивались. Потом теория, объяснённая на пальцах, без формул и сложных терминов. Затем код — каждый скрипт с подробным разбором каждой строчки. После кода

— раздел «За кулисами», где для любопытных раскрывается внутренняя механика алгоритмов. Его можно пропустить без ущерба для практического результата. Затем «Лаборатория ошибок» — что может пойти не так и как это исправить. И наконец «Творческое задание» — идеи для самостоятельных экспериментов.

Для работы с книгой вам понадобится компьютер и желание. Всё остальное — Python, библиотеки, примеры — я дам. Код из книги можно копировать и вставлять. Он протестирован и работает. Вам не нужно быть программистом. Вам нужно быть человеком, который хочет, чтобы его контент звучал лучше.

Благодарности

Я благодарю всех создателей контента, которые делились своими проблемами со звуком и вдохновляли меня на поиск решений. Я благодарю сообщество разработчиков открытого программного обеспечения, создавших библиотеки librosa, numpy, noisereduce и десятки других инструментов, на которых построена эта книга. Я благодарю вас, читатель, за то, что взяли эту книгу в руки и решили инвестировать время в улучшение своего контента. Это решение окупится сторицей.

Поехали.

Введение. Почему Python, а не Audacity?

Для кого эта книга

Представьте: вы записали классное видео для YouTube. Свет выставлен идеально, сценарий выверен до секунды, шутки отрепетированы перед зеркалом. Вы нажимаете «экспорт» и с предвкушением ждете реакцию зрителей.

А потом открываете комментарии. И вместо обсуждения ваших гениальных мыслей видите: «Ничего не слышно», «Звук как из бочки», «Автор, купи нормальный микрофон».

Обидно? Еще как. Знакомо? Большинству контент-креаторов — да.

Эта книга для тех, кто хочет управлять звуком так же легко, как текстом или картинкой. Вы не должны становиться профессиональным звукорежиссером. Вы не обязаны тратить часы на изучение сложных программ вроде Audacity, Adobe Audition или Reaper. Ваша задача — создавать контент. А звук должен просто работать. И Python сделает его таким.

Почему именно Python, а не готовый софт с ползунками и кнопками? Давайте разберемся.

Три причины выбрать Python для работы со звуком

Причина первая: автоматизация.

Предположим, у вас есть подкаст из двадцати выпусков. В каждом выпуске нужно убрать фоновый шум от кондиционера, нормализовать громкость и вырезать неловкие паузы. В Audacity вы делаете это вручную: открываете файл, применяете эффект, ждете, сохраняете, открываете следующий. Двадцать раз.

На Python вы пишете скрипт один раз. И запускаете его для всех файлов одной командой. Пока Audacity еще открывается, ваш скрипт уже закончил работу и пошел пить кофе.

Причина вторая: контроль.

В графическом редакторе вы двигаете ползунок «Убрать шум» и надеетесь на лучшее. Вы не знаете, что именно происходит со звуком. Вам говорят: «Применить шумоподавление?» — вы нажимаете «Да». Результат иногда радует, иногда огорчает, но вы не понимаете почему.

Код не скрывает от вас ничего. Вы говорите: «Вот этот диапазон частот ослабить на 12 децибел». И он ослабляет именно этот диапазон. Вы говорите: «Найти участки тишины длиннее полсекунды и обрезать их до четверти секунды». И он делает именно это. Полный контроль. Никакой магии.

Причина третья: воспроизводимость.

Выпустили подкаст. Через неделю поняли, что в третьей минуте остался щелчок, который вы не заметили. В Audacity это означает: открыть проект, найти то место, исправить, снова экспортировать. Минут десять ручной работы.

В Python это означает: подправили одну строку в скрипте,

перезапустили — готово. Ваш процесс обработки звука становится документированным, повторяемым и неубиваемым. Вы всегда можете вернуться и улучшить результат, не начиная с нуля.

Что такое звук с точки зрения Python

Давайте договоримся о главном прямо сейчас. Когда вы слышите слово «код», у вас в голове, возможно, всплывает образ хакера в капюшоне, который стучит по клавишам под зеленые строчки, падающие сверху. Забудьте.

Код — это просто инструкция. Написанная на человеческом языке, просто немного более строгом, чем обычно.

Вот пример инструкции для человека:

«Возьми аудиофайл, найди в нем все места, где громкость ниже порога слышимости, и удали их».

А вот та же инструкция для Python:

```
python
import librosa
import soundfile as sf

y, sr = librosa.load('podcast.wav')
y_trimmed, _ = librosa.effects.trim(y, top_db=20)
sf.write('podcast_clean.wav', y_trimmed, sr)
```

Прямо сейчас это может выглядеть непонятно. Но к концу этой книги вы будете читать такой код как родной. Обещаю.

Звук для Python — это просто массив чисел. Представьте себе очень длинный ряд значений громкости, измеренных сорок четыре тысячи раз в секунду. Каждое число — это положение мембраны вашего динамика в конкретный момент времени. Положительное число — мембрана идет вперед, отрицательное — назад, ноль — покой.

Ваш голос, музыка, шум ветра, лай собаки за окном — все это превращается в столбики чисел. И Python умеет с этими столбиками работать быстрее, чем любой графический редактор.

Что вам понадобится

Никакого специального оборудования. Никаких платных программ. Только три вещи:

Первое: компьютер.

Подойдет любой — Windows, Mac, Linux. Python работает везде одинаково. Если ваш компьютер способен открыть браузер и не зависнуть, он справится с обработкой аудио.

Второе: Python версии 3.9 или новее.

Если у вас еще нет Python — не пугайтесь. Следующий раздел проведет вас через установку за десять минут. Это действительно просто: скачали установщик, нажали «Далее» несколько раз, готово.

Третье: желание автоматизировать рутину и вернуть себе время.

Оно у вас уже есть, раз вы открыли эту книгу.

Как устроена эта книга

В этой книге девять глав. Каждая глава — это одна конкретная задача, с которой сталкивается любой создатель контента. Никакой абстрактной теории в вакууме. Только реальные проблемы и их решения.

Каждая глава построена по одному и тому же принципу:

В чем проблема. Я описываю ситуацию из жизни. Возможно, вы узнаете себя.

Как мы будем ее решать. Коротко о том, какой инструмент применим и почему.

Код. Пошагово, с объяснением каждой строчки. Вы можете копировать его и запускать сразу.

Лаборатория ошибок. Что может пойти не так и как это исправить. Потому что идеальных запусков с первого раза не бывает ни у кого.

За кулисами. Для любопытных: как это работает внутри. Этот раздел можно пропустить, если хочется просто получить результат. Но если вы хотите понимать, что именно происходит со звуком, — welcome.

Творческий практикум. Что еще можно сделать с этим же инструментом, чтобы ваш контент стал еще круче.

Чек-лист. Краткое резюме главы. Пробежались глазами — вспомнили главное.

Книгу можно читать последовательно, от корки до корки,

а можно использовать как справочник: открыли нужную главу, скопировали код, решили проблему, закрыли. Оба подхода рабочие.

Установка Python за десять минут

Давайте сделаем это прямо сейчас, чтобы потом не отвлекаться. Выполните следующие шаги — и через десять минут Python будет готов к работе.

Шаг 1. Скачиваем Python.

Откройте официальный сайт python.org. Прямо на главной странице вы увидите большую желтую кнопку «Download Python 3.x» — где x заменяется на самую свежую версию. На момент написания книги это 3.12, но подойдет любая версия, начиная с 3.9. Смело нажимайте кнопку. Скачивается установщик.

Шаг 2. Запускаем установку.

Нашли скачанный файл, запустили. Внимание: на первом экране установщика есть важная галочка внизу — «Add Python to PATH». Обязательно поставьте ее. Если пропустите — Python установится, но командная строка не будет знать, где он лежит. Поставили галочку? Жмите «Install Now».

Подождите минуту. Готово.

Шаг 3. Проверяем.

Теперь проверим, что все работает. Откройте программу «Командная строка» на Windows (нажмите Win, начните вводить cmd, нажмите Enter) или «Терминал» на Mac и

Linux.

В открывшемся окне введите:

bash

python --version

И нажмите Enter. Если вы увидели строчку вроде Python 3.12.1 — поздравляю, Python с вами.

Если вы увидели ошибку — скорее всего, вы пропустили галочку «Add Python to PATH». Ничего страшного: переустановите Python и на этот раз обязательно поставьте галочку. Это самая частая проблема, она решается за две минуты.

Шаг 4. Устанавливаем библиотеки.

Сам по себе Python — это как пустая кухня. Плита есть, а продуктов нет. Нам нужно «закупить продукты» — библиотеки для работы со звуком.

В той же командной строке выполните три команды подряд, нажимая Enter после каждой:

bash

pip install librosa

pip install soundfile

pip install noisereduce

Пойдут строчки с процентами, скачиваниями, установками. Когда все закончится, командная строка снова будет ждать вашего ввода. Это значит — все установлено успешно.

Если команда pip не распознается — попробуйте pip3 вместо pip. Это случается на некоторых Mac и Linux. То есть вводите pip3 install librosa и так далее.

Шаг 5. Ваш первый скрипт.

Давайте удостоверимся, что все работает как надо. Создайте на рабочем столе папку audio_book. Внутри нее создайте текстовый файл с именем hello_sound.py.

Внимание: расширение должно быть именно .py, а не .txt. Windows по умолчанию скрывает расширения файлов, поэтому то, что вы видите как hello_sound, на самом деле может быть hello_sound.py.txt. Чтобы этого избежать, откройте проводник, перейдите на вкладку «Вид», поставьте галочку «Расширения имен файлов». Теперь вы видите все расширения и можете создавать .py файлы правильно.

Откройте hello_sound.py в любом текстовом редакторе. Можно в стандартном Блокноте, но лучше скачать бесплатный VS Code — он умеет подсвечивать код разными цветами, что сильно облегчает жизнь.

Вставьте в файл эти четыре строки:

python

import librosa

print("Библиотека librosa готова к работе!")

print("Звук на твоей стороне.")

Сохраните файл. Теперь вернитесь в командную строку. Нам нужно перейти в папку audio_book. Если она у вас на рабочем столе, команда будет такой:

bash

cd Desktop/audio_book

Нажмите Enter. Командная строка теперь находится внут-

ри вашей папки. Запустите скрипт:

bash

python hello_sound.py

Если вы увидели на экране:

text

Библиотека librosa готова к работе!

Звук на твоей стороне.

— то всё готово. Вы только что написали и запустили свой первый Python-скрипт. Добро пожаловать в мир, где звук вам подчиняется.

Пара слов перед стартом

Эта книга писалась для человека, который никогда не программировал. Или программировал когда-то давно и с тех пор забыл всё, кроме слова «переменная». Я буду объяснять каждую строчку кода, как если бы вы увидели ее впервые в жизни. Если вы уже знаете Python — отлично, читать будет еще быстрее. Но база заложена с расчетом на полного новичка.

Второе: не бойтесь ошибок. Ошибка в коде — это не провал. Это способ языка сказать: «Я тебя не понял, давай уточним». Программисты с двадцатилетним стажем до сих пор гуглят сообщения об ошибках ежедневно. Это нормальная часть работы, а не показатель некомпетентности.

Третье: в каждой главе вы будете получать готовый работающий инструмент. Берите его, применяйте к своему контенту, улучшайте, ломайте, чините. Код из этой книги —

ваш. Никаких лицензий, никаких ограничений.

Что вы будете уметь после этой книги

Давайте начистоту. Я не буду обещать, что после этой книги вы станете звукорежиссером уровня голливудских студий. Это было бы враньем. Но вот что вы будете уметь точно — конкретно и измеримо:

Вы сможете взять сырую запись своего голоса, сделанную в обычной комнате с обычным микрофоном, и превратить ее в чистый, сочный, профессионально звучащий трек. Не за час ручной работы. За тридцать секунд автозапуска скрипта.

Вы научитесь убирать фоновый шум. Тот самый гул холодильника, который почему-то становится слышен именно тогда, когда вы говорите что-то важное. Тот шум улицы, который пробивается даже сквозь закрытые окна. Тот шипящий фон дешевого микрофона, который выдает в вас любителя за километр.

Вы сможете находить и удалять паузы. Не вручную, прокручивая ползунок туда-сюда и мучительно вылавливая взглядом провалы на звуковой дорожке. А одной командой. Все паузы длиннее полусекунды — бах, и сокращены до комфортного размера. Ваш подкаст становится плотнее, энергичнее, профессиональнее.

Вы освоите нормализацию громкости. Ваши видео на YouTube перестанут звучать тише, чем реклама перед ними. Ваш подкаст будет одинаково хорошо слышен и в метро, и в тихой комнате. Зрителям больше не придется дергать пол-

зунок громкости туда-сюда — они будут просто слушать.

Вы сможете менять голос. Хотите озвучить персонажа с низким демоническим тембром? Сделать забавного робота для детского видео? Превратить свой голос в голос белки-летяги для комедийной вставки? Всё это будет делаться парой строк кода.

Вы соберете автоматический микс голоса и фоновой музыки. Чтобы музыка не забивала речь, а аккуратно поддерживала настроение. Чтобы голос всегда был на первом плане, а музыка уходила на второй, когда вы говорите, и возвращалась, когда вы замолкаете. Без ручной автоматизации. Без нервов.

Вы наложите финальный лоск — эквализацию и компрессию, которые превращают обычный голос в «радио-голос». Тот самый бархатный, глубокий, обволакивающий тембр, который заставляет людей дослушивать до конца и спрашивать: «А на чем ты записываешь?»

И самое главное: вы построите свою звуковую кухню. Набор скриптов, которые работают именно с вашим голосом, именно под ваши задачи. Не универсальные настройки из интернета, которые подходят «всем понемногу, но никому конкретно». А инструмент, заточенный под вас.

История автора: как я пришел к Python и звуку

Знаете, с чего начался мой путь в обработке звука? С позора. Самого настоящего, жгучего позора.

Я записал свой первый подкаст. Мне казалось, что это

шедевр. Тема была огненная, мысли — глубокие, шутки — смешные. Я залил выпуск на площадку и лег спать с чувством выполненного долга. Мне снились восторженные комментарии и графики растущих прослушиваний.

Проснулся я от первого комментария. Он гласил: «Слушал в машине. Ничего не понял. Сделай погромче, а?»

Второй был хуже: «Звук как из ведра. Автор, ты в колодце сидишь?»

Третий добил: «Бросил на второй минуте. Уши устали продираться сквозь шум».

Я был раздавлен. Я вложил душу в содержание. Но слушателю было плевать на содержание, потому что форма не пускала его внутрь. Звук поставил стену между мной и аудиторией. И я ничего не мог с этим сделать.

Вернее, мог. Я открыл Audacity. И провел в нем четыре часа. Четыре часа я двигал ползунки, применял эффекты, отменял, применял другие, слушал результат, сравнивал, снова отменял. В итоге я получил звук немного лучше, но все еще далекий от идеала. А главное — я не понимал, что именно я сделал. Я не мог повторить этот результат для следующего выпуска. Каждый раз начиналась та же битва с ползунками.

Тогда я подумал: «Я же программист. Почему я занимаюсь ручным трудом, который можно автоматизировать?»

Я открыл Python. Установил librosa. И написал свой первый скрипт для очистки звука. Он работал отвратительно.

Звук после обработки напоминал голос робота, говорящего из бочки с водой. Но я хотя бы понимал, что именно я сделал не так.

Через неделю экспериментов мой скрипт чистил звук лучше, чем я вручную в Audacity. Через месяц я мог обработать весь выпуск одной командой. Через три месяца друзья начали спрашивать: «Слушай, а что у тебя за микрофон? Звучит бомбически».

Микрофон был тот же самый, что и в первом позорном выпуске. Просто теперь между ним и слушателем стоял Python.

Эта книга — способ передать вам тот путь, который я прошел за месяцы, но в сжатом и понятном виде. Вы пройдете его за девять глав. И вам не придется набивать шишки, на которых набивал шишки я.

Ответы на страхи: «Я гуманитарий и боюсь кода»

Если вы прямо сейчас думаете: «Это все круто, но я гуманитарий. Я боюсь кода. Я не понимаю математику. Я не технарь. У меня не получится», — пожалуйста, прочитайте следующие четыре абзаца внимательно.

Первое. Код — это не магия, не математика и не инопланетный язык. Код — это инструкция, записанная в особой форме. Если вы можете объяснить другому человеку, как сделать бутерброд — пошагово, в правильной последовательности, с учетом всех деталей, — вы можете писать код. Потому что программирование — это просто умение рас-

кладывать задачу на шаги.

Второе. В этой книге нет математики. Серьезно. Ни одной формулы, которую нужно было бы запомнить. Ни одного математического доказательства. Все, что делает Python под капотом, я объясняю через метафоры и аналогии. Фурье-преобразование? Я расскажу вам, что это как разложить смузи на отдельные фрукты, из которых он сделан. Спектральное вычитание? Это как убрать шум дождя из записи разговора, если у вас есть отдельная запись шума дождя. Без формул. Без интегралов. Без боли.

Третье. Вам не нужно становиться программистом. Вы не будете писать код с нуля. Вы будете копировать готовые скрипты, подставляя в них название своего файла. И запускать. Всё. Постепенно вы начнете понимать, что именно происходит в коде, и сможете его менять под свои нужды. Но это произойдет естественно, без насилия над мозгом.

Четвертое. Я проверил эту методику на живых людях. Моя жена — филолог, она до знакомства со мной никогда не открывала командную строку. Сейчас она обрабатывает свои аудиозаписи для языковых курсов с помощью скриптов, которые я ей когда-то настроил. Через месяц она уже сама поправила в них пару параметров. Через три — написала свой первый мини-скрипт для обрезки тишины в начале и конце файла. Если смогла она — сможете и вы. Филолог победил код. Код не такой страшный, как кажется.

Как пользоваться книгой: три сценария

У людей разные цели и разный стиль обучения. Я спроектировал эту книгу так, чтобы она работала для трех типов читателей. Выберите свой сценарий.

Сценарий «Дай готовое».

Вы хотите просто решить проблему со звуком и не вдаваться в детали. Отлично. Откройте нужную главу. Найдите раздел «Код». Скопируйте скрипт целиком. Подставьте название своего файла. Запустите. Результат готов. Раздел «За кулисами» пропускайте без зазрения совести. Вы пришли за инструментом — берите инструмент. Никто не заставляет вас изучать устройство дрели, чтобы просверлить дырку в стене.

Сценарий «Хочу понимать».

Вы хотите не просто применять готовые решения, но и понимать, что именно происходит. Тогда читайте главу целиком. Раздел «За кулисами» для вас — основной. После каждой главы вы будете не просто уметь чинить звук, но и понимать, почему этот метод работает, а другой нет. Через несколько глав вы начнете соединять приемы в собственные цепочки обработки. Вы будете не просто пользователем, а мастером.

Сценарий «Хочу углубиться».

Вам мало того, что в книге. Вы хотите экспериментиро-

вать, ломать и чинить, создавать свои инструменты. Тогда ваша любимая часть — «Творческий практикум» в конце каждой главы. Там я даю идеи, которые выводят за пределы основного материала. Что будет, если применить шумоподавление дважды? А если сначала изменить голос, а потом убрать шум — результат изменится? А можно ли соединить два скрипта в один конвейер? Дерзайте. Код из этой книги — ваш полигон для экспериментов.

Независимо от выбранного сценария, помните главное: вы не обязаны проходить книгу от корки до корки. Глава 4 не требует прочтения Глав 1–3. Каждая глава самодостаточна. Захотелось прямо сейчас сделать радио-голос — открыли Главу 9 и сделали. Потом понадобилось убрать шум — вернулись к Главе 4. Книга — это набор инструментов, а не роман с сюжетом. Используйте ее как удобно.

Глава 1. Знакомимся со звуком через Python

«Твой голос заслуживает того, чтобы звучать профессионально. Даже если ты записываешь на кухне с воющим холодильником»

О чём эта глава

Вы когда-нибудь открывали аудиофайл в программе и видели перед собой волны, пики и провалы? Вы смотрите на них и, возможно, думаете: «Красиво, но что с этим делать?» В этой главе мы научимся не просто смотреть на звук, а понимать его. Мы загрузим аудиофайл в Python, разберём его на составные части и узнаем о нём то, что невозможно узнать ушами. Мы узнаем его точную длительность, громкость, количество тишины и речи. А главное — мы поймём, как компьютер представляет звук внутри себя. Это фундамент, на котором строится вся дальнейшая работа. Без этого фундамента нельзя двигаться дальше, как нельзя строить дом без знания того, из чего сделан фундамент. Мы начнём с самых основ, и я обещаю: если вы никогда в жизни не писали код, через час вы напишете свой первый работающий скрипт и он сделает что-то полезное.

Что такое звук для компьютера

Давайте начнём с простого вопроса. Вы слышите музыку,

голос, шум улицы. Это всё — звук. Но что такое звук для компьютера? Компьютер не имеет ушей. Он не слышит так, как слышим мы с вами. Для компьютера звук — это просто очень длинный список чисел. Представьте себе линейку, на которой вы отмечаете высоту волны в каждый момент времени. Или представьте, что вы фотографируете поверхность воды в озере много тысяч раз в секунду. Каждая фотография показывает вам одно число: насколько высоко поднялась вода в этой точке. Сложите все эти числа в ряд — и вы получите цифровое представление звука.

Этот ряд чисел называется аудиосигналом. Когда вы записываете голос на телефон, микрофон телефона колеблется от вашего голоса. Специальная электронная схема измеряет положение микрофона много тысяч раз в секунду и записывает каждое измерение в виде числа. Чем чаще происходят измерения, тем точнее запись. Стандартное значение — сорок четыре тысячи сто измерений в секунду для музыки и двадцать две тысячи пятьдесят для речи. Это называется частотой дискретизации. Звучит сложно, но суть простая: это просто количество чисел, которое компьютер сохраняет за одну секунду звука. Двадцать две тысячи чисел в секунду для речи — это более чем достаточно. Человеческий голос не забирается так высоко по частоте, как музыка, поэтому речь можно записывать с меньшей плотностью чисел, и качество не пострадает.

Почему именно такие числа, а не круглые двадцать тысяч

или пятьдесят тысяч? Тут есть интересная история. Существует правило, открытое инженерами Найквистом и Шенноном: чтобы точно записать звук определённой высоты, нужно делать измерения как минимум вдвое чаще, чем колеблется сам звук. Человеческое ухо слышит звуки примерно до двадцати тысяч колебаний в секунду. Значит, чтобы записать всё, что слышит человек, нужно не менее сорока тысяч измерений в секунду. На аудиодисках используется сорок четыре тысячи сто — это чуть больше сорока тысяч, взятое с запасом. А двадцать две тысячи пятьдесят — это ровно половина от дискового стандарта, чего вполне хватает для голоса. Голос редко поднимается выше восьми тысяч колебаний в секунду, поэтому двадцати двух тысяч измерений более чем достаточно. Такой выбор экономит место на диске и ускоряет обработку. Одна минута речи в таком качестве занимает около двух с половиной мегабайт — сущий пустяк для современного компьютера.

Теперь давайте представим сам список чисел. Каждое число в этом списке — это значение громкости в конкретный момент. Обычно эти числа лежат в диапазоне от минус единицы до плюс единицы. Минус один — это максимальная громкость в отрицательном направлении, когда мембрана динамика уходит назад. Плюс один — максимальная громкость вперёд. Ноль — это тишина, положение покоя. Реальный звук колеблется вокруг нуля: то в плюс, то в минус, то снова в плюс. Если вы посмотрите на эти числа, то увидите,

как они танцуют вокруг нуля. Громкий звук — это большие числа, близкие к единице или минус единице. Тихий звук — это маленькие числа, близкие к нулю. Тишина — это почти ноль. Шум — это много маленьких случайных чисел, которые немного отклоняются от нуля, но не уходят далеко.

Установка всего необходимого

Прежде чем мы начнём работать со звуком, нам нужно подготовить инструменты. Это как перед приготовлением обеда: сначала достаём кастрюли и продукты, потом готовим. Нам понадобится Python — это язык программирования, на котором мы будем писать наши инструкции для компьютера. И нам понадобятся несколько библиотек — это готовые наборы инструментов для работы со звуком, которые другие программисты уже написали за нас. Мы не будем изобретать велосипед, а воспользуемся их трудом.

Установка Python — это очень просто. Идите на сайт python.org. Прямо на главной странице вы увидите большую кнопку «Download Python» с номером версии. Смело нажимайте её. Скачается установщик. Запустите его. На первом экране установщика обязательно поставьте галочку внизу, которая называется «Add Python to PATH». Эта галочка говорит компьютеру: «Запомни, где лежит Python, чтобы я мог запускать его из любой папки». Если пропустите эту галочку, потом придётся немного повозиться с настройками. Ничего страшного, просто переустановите Python ещё раз и на этот раз галочку поставьте. Дальше нажимайте «Install Now»

и ждите минуту. Всё, Python на вашем компьютере.

Теперь проверим, что Python работает. Откройте программу «Командная строка», если у вас Windows — нажмите клавишу Windows, начните вводить слово «cmd», нажмите Enter. Если у вас Mac или Linux, откройте «Терминал». В появившемся чёрном окошке введите команду python --version и нажмите Enter. Если Python установился, вы увидите что-то вроде «Python 3.12.1» — номер версии. Если видите ошибку, скорее всего, вы не поставили ту самую галочку при установке. Переустановите Python, галочку поставьте, и всё получится.

Следующий шаг — установка библиотек. Библиотеки в Python — это как дополнительные кухонные приборы. Сам по себе Python умеет многое, но для работы со звуком ему нужны специальные инструменты. К счастью, установка библиотек делается одной командой. В той же командной строке введите по очереди три команды, нажимая Enter после каждой:

```
pip install librosa

pip install soundfile

pip install sounddevice
```

Каждая команда скачает из интернета нужную библиотеку и установит её. Вы увидите бегущие строчки с процентами — это нормально. Когда установка закончится, команд-

ная строка снова будет ждать вашего ввода. Это значит, всё прошло успешно.

Что делает каждая библиотека? librosa — это наш главный инструмент для работы со звуком. Она умеет загружать аудиофайлы, анализировать их, извлекать из них информацию. soundfile умеет сохранять обработанный звук обратно в файл. sounddevice умеет проигрывать звук прямо из кода, чтобы мы могли быстро проверить результат. Все три библиотеки бесплатны и созданы сообществом программистов со всего мира.

Теперь создайте на рабочем столе папку с любым названием, например audio_book. В этой папке мы будем хранить все наши скрипты и аудиофайлы. Порядок на компьютере — это важно. Когда все файлы лежат в одной папке, их легко найти и с ними удобно работать.

Ваш первый скрипт: загружаем аудио

Настал момент написать наш первый код. Не пугайтесь слова «код». Код — это просто инструкция для компьютера, записанная на особом языке. Если вы можете объяснить другому человеку, как сделать бутерброд, вы можете написать код. Это такое же пошаговое описание действий, только адресованное не человеку, а машине.

Откройте любой текстовый редактор. На Windows можно использовать Блокнот, но я рекомендую скачать бесплатную программу Visual Studio Code — она удобнее, потому что подсвечивает код разными цветами и помогает не ошибать-

ся. Создайте новый файл и сохраните его в нашей папке с именем first_script.py. Расширение .py говорит компьютеру, что это программа на Python. Это важно: если вы сохраните файл как .txt, Python не поймёт, что это код.

Теперь впишите в файл следующие строчки. Не обязательно понимать прямо сейчас каждую из них — мы разберём всё по шагам сразу после того, как запустим. Просто напечатайте их внимательно, соблюдая все отступы и знаки препинания. В Python регистр букв имеет значение: Print и print — это разные слова для компьютера. Пишите всё маленькими буквами, как в примере.

```
python
import librosa

y, sr = librosa.load('my_voice.wav')

print("Файл загружен!")
print(f"Частота измерений: {sr} раз в секунду")
print(f"Длина записи: {len(y) / sr:.2f} секунд")
print(f"Всего чисел в файле: {len(y)}")
```

Нажмите «Сохранить». Теперь вернитесь в командную строку. Нам нужно перейти в папку, где лежит наш скрипт. В командной строке введите команду cd Desktop/audio_book и нажмите Enter. cd означает «change directory», то есть «сменить папку». Теперь командная строка находится внутри нашей рабочей папки.

Прежде чем запускать скрипт, нам нужен аудиофайл. Если у вас уже есть запись голоса в формате WAV, скопируйте её в папку audio_book и переименуйте в my_voice.wav. Если нет — не беда. Откройте диктофон на телефоне, наговорите минуту любого текста — например, прочитайте вслух рецепт борща или опишите, что вы видите за окном. Перекиньте файл на компьютер, положите в папку audio_book и назовите my_voice.wav. Формат WAV обязателен: librosa лучше всего работает именно с WAV-файлами. Если ваш файл в другом формате, например MP3 или M4A, вы можете конвертировать его в WAV с помощью бесплатной программы Audacity или онлайн-конвертера. В следующих главах я покажу, как конвертировать форматы прямо в коде, а пока — пусть будет WAV.

Всё готово. В командной строке введите python first_script.py и нажмите Enter. Если всё сделано правильно, вы увидите на экране что-то вроде:

text

Файл загружен!

Частота измерений: 22050 раз в секунду

Длина записи: 58.34 секунд

Всего чисел в файле: 1286397

Поздравляю! Вы только что написали и запустили свою первую программу на Python. Она сделала полезное дело: загрузила аудиофайл и рассказала о нём главное. Теперь давайте разберём, что именно произошло.

Разбор кода: строка за строкой

import librosa — этой строкой мы говорим Python: «Достань с полки инструмент под названием librosa, он нам понадобится». В языке Python все дополнительные возможности хранятся в библиотеках. Чтобы воспользоваться библиотекой, её нужно сначала импортировать — то есть подключить. Это как включить кухонный комбайн в розетку перед использованием. Без этой строчки Python не будет знать, что мы вообще собираемся работать со звуком.

y, sr = librosa.load('my_voice.wav') — это главная строка скрипта. Она делает сразу несколько вещей. Во-первых, она ищет файл с именем my_voice.wav в той же папке, где лежит скрипт. Во-вторых, она считывает его содержимое и превращает в те самые списки чисел, о которых мы говорили. В-третьих, она возвращает нам два значения. Первое значение — это массив чисел y. Каждое число в этом массиве — одно измерение громкости. Второе значение — частота дискретизации sr, то есть сколько раз в секунду делались эти измерения. Мы сохраняем эти два значения в две переменные с короткими именами y и sr, чтобы потом было удобно к ним обращаться. Переменная — это просто коробочка с именем, в которую можно положить значение и потом достать его, когда понадобится.

print("Файл загружен!") — выводит сообщение на экран. Функция print — это способ Python поговорить с нами. Всё, что внутри скобок и кавычек, появится на экране. Это по-

лезно для того, чтобы понимать, что скрипт работает и дошёл до определённой точки.

print(f"Частота измерений: {sr} раз в секунду") — здесь мы используем особый вид строк, который называется f-строка. Буква f перед открывающей кавычкой говорит Python: «Внутри этой строки могут быть фигурные скобки, и всё, что в них, нужно заменить на значения переменных». Python видит фигурные скобки {sr}, смотрит, что в переменной sr лежит число 22050, подставляет это число в строку, и на экране мы видим «Частота измерений: 22050 раз в секунду». Удобно и читаемо.

print(f"Длина записи: {len(y) / sr:.2f} секунд") — здесь мы считаем длительность записи. len(y) — это функция, которая возвращает длину массива y, то есть общее количество чисел в нём. Делим это количество на частоту дискретизации sr и получаем длительность в секундах. Двоеточие и .2f в конце говорят Python: «Покажи результат с двумя знаками после запятой, не нужно больше». Если файл длится почти минуту, мы увидим «58.34 секунд», а не «58.3412345...» с кучей лишних цифр.

print(f"Всего чисел в файле: {len(y)}") — просто показываем общее количество измерений в файле. Для минутной записи это число будет больше миллиона. Представляете, сколько данных? Миллион точек только для того, чтобы описать одну минуту речи. А теперь представьте, что компьютер обрабатывает этот миллион точек за доли секунды.

Именно поэтому программирование — это мощный инструмент.

Продолжаем знакомство: измеряем громкость

Теперь, когда мы умеем загружать файл и узнавать его базовые характеристики, давайте научимся измерять громкость. Это пригодится нам во всех следующих главах. Дополните ваш скрипт новыми строками:

```
python
import numpy as np

max_volume = np.max(np.abs(y))
min_volume = np.min(y)
avg_volume = np.mean(np.abs(y))

print(f"Самая большая громкость: {max_volume:.4f}")
print(f"Самая маленькая громкость: {min_volume:.4f}")
print(f"Средняя громкость: {avg_volume:.4f}")
```

Сначала мы импортируем ещё одну библиотеку — numpy. Это библиотека для работы с массивами чисел. Она умеет делать вычисления над целыми массивами быстро и удобно. np — это короткое имя, которое мы даём библиотеке при импорте, чтобы не писать numpy каждый раз полностью. Сокращение np — общепринятое, все программисты его используют.

np.max(np.abs(y)) — здесь мы делаем три вещи за один

раз. Сначала np.abs(y) берёт весь массив y и превращает все отрицательные числа в положительные, не трогая уже положительные. Минус ноль пять становится плюс ноль пять. Плюс ноль три остаётся плюс ноль три. Это нужно потому, что громкость нас интересует без знака: неважно, вперёд идёт мембрана динамика или назад, она в обоих случаях создаёт звуковое давление. Затем np.max находит самое большое число в получившемся массиве. Это и есть пиковая громкость — самый громкий момент записи.

np.min(y) находит самое маленькое число в оригинальном массиве, без взятия модуля. Обычно это отрицательное число, близкое к минус единице. Интересно посмотреть на него и сравнить с максимумом: если они примерно равны по модулю, значит, запись симметрична, что хорошо. Если максимум сильно отличается от модуля минимума, возможно, с записью что-то не так — например, микрофон был смещён.

np.mean(np.abs(y)) считает среднюю абсолютную громкость. Мы снова берём модуль каждого числа, чтобы отрицательные и положительные не гасили друг друга, и затем вычисляем среднее арифметическое. Это число говорит нам, насколько запись громкая в целом, а не только в пиках. Две записи могут иметь одинаковую пиковую громкость, но совершенно разную среднюю. Первая может быть плотной и насыщенной, вторая — тихой с редкими громкими выкриками. Средняя громкость лучше отражает то, как человек воспринимает громкость на слух.

Слушаем аудио прямо из кода

Смотреть на числа интересно, но звук хочется ещё и слышать. Давайте научимся проигрывать аудиофайл прямо из Python-скрипта. Для этого мы используем библиотеку sounddevice, которую установили ранее. Добавьте в конец скрипта:

```
python
import sounddevice as sd

print("Сейчас вы услышите запись...")
sd.play(y, sr)
sd.wait()
print("Воспроизведение закончено.")
```

sd.play(y, sr) говорит компьютеру: «Возьми массив чисел y и отправь его в динамики, воспроизводя с частотой sr измерений в секунду». Компьютер начинает проигрывать звук и в этот же момент продолжает выполнять следующие строчки кода. Поэтому следующая строка критически важна: sd.wait() говорит компьютеру: «Стой здесь и жди, пока воспроизведение не закончится». Если убрать эту строку, скрипт запустит проигрывание и тут же закроется, музыка оборвётся через долю секунды. Мы этого не хотим, поэтому ждём.

Теперь у вас есть полноценный инструмент для первичного анализа аудиофайлов. Вы можете загрузить любой WAV-файл, узнать его длительность, частоту дискретизации, пи-

ковую и среднюю громкость, а затем прослушать его. Сохраните этот скрипт — мы будем использовать его на протяжении всей книги для быстрой проверки результатов.

Находим тишину в записи

Следующий важный навык — умение отличать речь от тишины. В любой записи, сделанной без подготовки, есть паузы: между предложениями, между словами, иногда просто потому что вы задумались или отвлеклись. Слушатель не обязан ждать, пока вы соберётесь с мыслями. Слушатель хочет получать информацию плотно и без проволочек. Поэтому нам нужно уметь находить тишину. Сначала — чтобы измерить её количество, а в следующих главах — чтобы её сократить.

Библиотека librosa содержит удобную функцию для поиска тишины. Она называется split и делает ровно то, что нам нужно: находит в аудио непрерывные участки, где есть значимый звук. Всё, что между этими участками — тишина. Давайте напишем скрипт для анализа тишины:

```
python
import librosa
import numpy as np

y, sr = librosa.load('my_voice.wav')

intervals = librosa.effects.split(y, top_db=30)
```

```
    total_speech = 0
    for start, end in intervals:
    duration = (end - start) / sr
    total_speech += duration

    total_length = len(y) / sr
    silence = total_length - total_speech

    print(f"Общая длина записи: {total_length:.2f} секунд")
    print(f"Длина речи: {total_speech:.2f} секунд")
    print(f"Длина тишины: {silence:.2f} секунд")
    print(f"Тишина занимает {silence / total_length * 100:.1f}
процентов записи")
```

Давайте разберём, что здесь происходит. librosa.effects.split(y, top_db=30) — это функция, которая сканирует весь массив y и ищет участки, где громкость достаточна, чтобы считать это речью. Параметр top_db=30 задаёт порог чувствительности. Число 30 означает, что всё, что тише самого громкого места в записи на 30 децибел, считается тишиной. Это довольно чувствительный порог, он хорошо работает для записей в тихой комнате. Если у вас шумно, можно уменьшить число до 20 или даже 15 — тогда детектор станет менее чувствительным и не будет принимать слабый шум за речь.

Функция возвращает список интервалов. Каждый интервал — это два числа: начало и конец участка речи, измерен-

ные в количестве измерений от начала файла. В цикле for start, end in intervals мы проходим по всем найденным речевым фрагментам. Для каждого вычисляем его длительность в секундах: (end - start) / sr. Разницу между концом и началом делим на частоту дискретизации и получаем секунды. Все эти длительности суммируем в переменную total_speech.

Затем общую длину файла в секундах вычисляем как len(y) / sr. Время тишины получаем вычитанием: всё, что не речь — тишина. Выводим результаты на экран.

Если вы запустите этот скрипт на необработанной записи, то, скорее всего, увидите, что от двадцати до сорока процентов времени занимает тишина. Это нормально для живого человека, но ненормально для качественного контента. Сорок процентов — это почти половина. Представьте: ваш слушатель тратит почти половину времени на то, чтобы слушать ничто. Это то, что мы будем исправлять в следующих главах.

Работа с разными форматами аудио

Мир аудиофайлов не ограничивается форматом WAV. Существуют десятки форматов: MP3, M4A, FLAC, OGG, WMA и многие другие. Каждый формат имеет свои особенности, преимущества и недостатки. Нам, как людям, которые работают со звуком через код, полезно понимать разницу между ними.

WAV — это формат без сжатия. Это значит, что каждое измерение громкости сохраняется как есть, без каких-либо упрощений и выбрасывания данных. Файл получается боль-

шим, но зато при каждом открытии и сохранении качество остаётся неизменным. WAV — идеальный формат для промежуточной работы. Все обработки мы будем делать именно в WAV, потому что это исключает накопление искажений. Представьте, что вы каждый раз сжимаете и разжимаете фотографию — после десяти таких операций от качества ничего не останется. Так и со звуком. Поэтому правило простое: работаем в WAV, а финальный результат можно сконвертировать в MP3 для публикации.

MP3 — это формат со сжатием и с потерями качества. Алгоритм MP3 анализирует звук и выбрасывает те частоты, которые человеческое ухо всё равно плохо слышит. За счёт этого размер файла уменьшается в десять и более раз по сравнению с WAV. Для финальной публикации это отлично: слушатель не заметит потери качества, а файл будет маленьким и быстрым для скачивания. Для промежуточной обработки MP3 не подходит, потому что при каждом сохранении потери накапливаются.

M4A — это формат, похожий на MP3, но с более эффективным сжатием. При том же размере файла качество M4A немного выше, чем у MP3. FLAC — это сжатие без потерь. Файл меньше, чем WAV, но больше, чем MP3, и при этом качество остаётся идеальным. Для архивного хранения FLAC — отличный выбор. Для нашей работы в Python — WAV вне конкуренции.

Если у вас есть файлы в других форматах, librosa может не

справиться с их загрузкой без дополнительных инструментов. На этот случай есть библиотека pydub, которая умеет работать практически со всеми форматами. Установите её командой pip install pydub, и вы сможете конвертировать что угодно в WAV одной строкой кода. Но для простоты в этой книге мы будем исходить из того, что все исходные файлы уже в формате WAV.

Что мы узнали о звуке

Давайте подведём промежуточный итог. Мы начали с нуля — и теперь у нас есть работающий инструмент для анализа аудиофайлов. Мы знаем, что звук для компьютера — это просто длинный список чисел, измеряющих громкость много тысяч раз в секунду. Мы умеем загружать этот список в Python, измерять его длину, находить пиковую и среднюю громкость, отличать речь от тишины. Мы понимаем, почему формат WAV лучше для работы, а MP3 — для публикации. Мы написали несколько скриптов, каждый из которых делает полезную работу.

Это может показаться скромным началом, но не недооценивайте его. Вы только что освоили фундамент, на котором стоит вся обработка звука. Всё, что мы будем делать дальше — вырезать куски, склеивать, убирать шум, выравнивать громкость, — строится на этом фундаменте. Зная, как устроен звук внутри компьютера, вы сможете управлять им с точностью, недоступной ни одному графическому редактору. В следующих главах мы начнём это управление, и я обещаю:

будет интересно.

Творческое задание

Прежде чем перейти к следующей главе, попробуйте применить полученные знания на практике. Проанализируйте несколько своих записей с помощью написанных скриптов. Найдите самую длинную и самую короткую запись. Найдите запись с самым большим процентом тишины. Найдите запись с самой высокой и самой низкой средней громкостью. Сравните цифры. Есть ли закономерности? Может быть, записи, сделанные утром, тише, чем вечерние? Или записи на определённую тему содержат больше пауз? Такое исследование собственного материала — первый шаг к тому, чтобы стать профессионалом. Вы начинаете видеть в звуке не просто шум, а структуру. И это видение останется с вами навсегда.

Также попробуйте записать один и тот же текст в разных условиях: в тихой комнате, при открытом окне, с работающим телевизором на фоне. Прогоните все три записи через скрипт анализа тишины и сравните результаты. Вы увидите, как шум влияет на способность детектора отличать речь от тишины. Это понимание пригодится нам в главе про шумоподавление.

И последнее: не бойтесь экспериментировать с кодом. Поменяйте параметр top_db с 30 на 20, на 40, на 10. Посмотрите, как меняется результат. Добавьте в вывод свои метрики — например, количество найденных речевых фрагмен-

тов. Чем больше вы играетесь с кодом, тем быстрее он становится для вас естественным языком, как родная речь. Код — это не магия, это навык. И как любой навык, он развивается практикой.

Чек-лист главы

Вы освоили первую главу, если: знаете, что звук внутри компьютера — это список чисел, измеряющих громкость; понимаете, что такое частота дискретизации и почему она равна 22050 для речи; умеете устанавливать Python и библиотеки через pip; можете загрузить WAV-файл в Python с помощью librosa.load; умеете измерять длительность записи, пиковую и среднюю громкость; можете проиграть аудио из кода через sounddevice; умеете находить участки речи и тишины с помощью librosa.effects.split; понимаете разницу между форматами WAV, MP3, FLAC и знаете, какой для чего использовать; написали и запустили свои первые Python-скрипты для анализа звука. В следующей главе мы перейдём от анализа к действию: научимся резать, склеивать и собирать аудиофайлы, как детали конструктора.

Глава 2. Учимся резать и склеивать звук

О чём эта глава

В предыдущей главе мы смотрели на звук и изучали его свойства. Мы были как учёные, которые исследуют образец под микроскопом. Теперь пришло время стать ремесленниками. Мы возьмём аудиофайлы в руки и начнём с ними работать: отрезать лишнее, соединять нужное, переставлять куски местами. Всё то, что обычно делается мышкой в программах вроде Audacity, мы будем делать кодом. Почему кодом, а не мышкой? Потому что код можно запустить повторно. Потому что код не устаёт и не ошибается от усталости. Потому что код может обработать сто файлов за время, которое вы тратите на ручную обработку одного. В этой главе мы напишем инструменты, которые превратят монтаж аудио из утомительной рутины в быстрое и приятное занятие. Мы научимся вырезать фрагменты из аудио, склеивать несколько файлов в один, добавлять паузы нужной длины, выравнивать громкость разных фрагментов и собирать полноценный эпизод подкаста из заготовок. К концу главы у вас будет скрипт, который делает всю черновую работу по монтажу за вас.

Как резать аудио в коде

Начнём с самого простого действия: отрезать кусок от на-

чала или конца записи. Представьте, что вы записали подкаст. Вы начали запись, проверили микрофон, сказали «раз-раз, меня слышно?», потом выпили воды, прокашлялись и только через три минуты начали говорить по делу. Эти три минуты не должны попасть в финальную версию. Более того, в конце записи вы, скорее всего, скажете что-то вроде «Всё, выключаю запись. Ой, она ещё пишет? Ну всё, пока». Эти последние десять секунд тоже нужно убрать. В графическом редакторе вы бы выделили ненужный фрагмент мышкой и нажали Delete. В коде мы сделаем то же самое, только вместо мышки у нас будут индексы массива.

Вспомним, что наш аудиофайл — это массив чисел y. Каждое число имеет свой порядковый номер, или индекс. Первое число имеет индекс ноль, второе — индекс один, сотое — индекс девяносто девять. В Python мы можем взять любую часть массива, указав начальный и конечный индекс в квадратных скобках через двоеточие. Это называется срез. Если мы хотим убрать первые три минуты, нам нужно выбросить все числа с нулевого индекса до индекса, соответствующего трём минутам. А этот индекс мы вычисляем умножением: три минуты — это сто восемьдесят секунд, умножаем на частоту дискретизации 22050, получаем около четырёх миллионов. Значит, нам нужны все числа начиная с индекса четыре миллиона и до конца массива.

Вот как это выглядит в коде:

python

```
import librosa
import soundfile as sf

y, sr = librosa.load('long_record.wav')

# Сколько секунд отрезать от начала
cut_seconds = 180
cut_samples = cut_seconds * sr

# Берём всё, начиная с cut_samples и до конца
y_trimmed = y[cut_samples:]

sf.write('record_trimmed.wav', y_trimmed, sr)
print(f"Было: {len(y) / sr:.1f} сек")
print(f"Стало: {len(y_trimmed) / sr:.1f} сек")
```

Запись y[cut_samples:] — это и есть срез. Квадратные скобки говорят Python, что мы хотим взять часть массива. Число до двоеточия — начальный индекс. Двоеточие без числа после него означает «и до самого конца». Всё, что было до индекса cut_samples, игнорируется. Обратите внимание: мы не удаляем данные из исходного массива y. Мы создаём новый массив y_trimmed, который содержит только нужную нам часть. Исходный массив остаётся нетронутым, и мы всегда можем к нему вернуться.

Теперь обратная задача: обрезать конец. Это делается похоже, только мы указываем конечный индекс, а начальный

оставляем пустым:

```
python
# Сколько секунд отрезать от конца
cut_end_seconds = 10
cut_end_samples = len(y) - (cut_end_seconds * sr)

# Берём всё от начала до cut_end_samples
y_trimmed = y[:cut_end_samples]
```

Здесь срез y[:cut_end_samples] означает «от начала до индекса cut_end_samples, не включая его». Мы вычисляем этот индекс как общую длину массива минус количество измерений в десяти секундах. Таким образом мы отрезаем ровно последние десять секунд.

Если нужно обрезать и начало, и конец одновременно, оба среза можно объединить в одну строчку:

```
python
y_trimmed = y[180 * sr : len(y) - 10 * sr]
```

Читается это так: «Возьми массив y от индекса, соответствующего ста восьмидесяти секундам, до индекса, который находится за десять секунд до конца». Всё просто и понятно.

Склеиваем несколько файлов

С разрезанием разобрались. Теперь обратная операция — склеивание. Предположим, у вас есть три файла: музыкальное вступление, записанный голос и музыкальная концовка. Их нужно объединить в один файл, который пойдёт на публикацию. В видеоредакторе это делается перетас-

киванием файлов на дорожку. В коде это делает функция np.concatenate из библиотеки numpy. Она берёт несколько массивов и соединяет их в один длинный массив, как будто склеивает ленту из кусков.

Вот базовый пример:

```
python
import librosa
import numpy as np
import soundfile as sf

# Загружаем три файла
intro, sr1 = librosa.load('intro.wav')
voice, sr2 = librosa.load('voice.wav')
outro, sr3 = librosa.load('outro.wav')

# Склеиваем
final = np.concatenate([intro, voice, outro])

# Сохраняем
sf.write('final.wav', final, sr1)
```

Однако здесь есть подводный камень: все три файла должны иметь одинаковую частоту дискретизации. Если вступление записано с частотой 44100, голос — с 22050, а концовка — с 48000, склеенный файл будет звучать неправильно. На стыках скорость воспроизведения будет меняться, и голос может стать то быстрее, то медленнее. Поэтому перед

склеиванием нужно убедиться, что все файлы имеют одинаковую частоту, и если нет — привести их к единому стандарту.

Вот улучшенная версия скрипта с проверкой частоты:

```
python
import librosa
import numpy as np
import soundfile as sf

# Загружаем и проверяем частоту
target_sr = 22050

intro, sr_i = librosa.load('intro.wav', sr=target_sr)
voice, sr_v = librosa.load('voice.wav', sr=target_sr)
outro, sr_o = librosa.load('outro.wav', sr=target_sr)

print(f"Интро: {len(intro) / target_sr:.1f} сек")
print(f"Голос: {len(voice) / target_sr:.1f} сек")
print(f"Аутро: {len(outro) / target_sr:.1f} сек")

# Склеиваем
final = np.concatenate([intro, voice, outro])
print(f"Итого: {len(final) / target_sr:.1f} сек")

sf.write('episode.wav', final, target_sr)
```

Параметр sr=target_sr в функции librosa.load говорит биб-

лиотеке: «Загрузи файл и пересчитай его так, как будто он был записан с частотой target_sr». Если исходный файл был записан с другой частотой, librosa автоматически подгонит его под нужную. Это очень удобно и экономит нам кучу ручной работы.

Добавляем паузы между фрагментами

Когда вы склеиваете вступление и голос вплотную, они могут звучать неестественно. Обычно между музыкальной заставкой и началом речи нужна небольшая пауза — полсекунды или секунда, чтобы слушатель переключился с музыки на голос. В коде пауза — это просто массив из нулей. Ноль в аудио означает абсолютную тишину. Чтобы создать секунду тишины, нам нужно создать массив из нулей длиной в частоту дискретизации. При частоте 22050 это будет 22050 нулей.

```
python
# Создаём паузу в одну секунду
silence_sec = 1.0
silence = np.zeros(int(target_sr * silence_sec))

# Склеиваем с паузами
final = np.concatenate([intro, silence, voice, silence, outro])
```

Функция np.zeros(n) создаёт массив из n нулей. Мы передаём ей количество сэмплов, которое равно длительности паузы в секундах, умноженной на частоту дискретизации. Этот массив нулей можно вставлять между любыми фраг-

ментами, и на слух это будет восприниматься как естественная пауза. Длину паузы можно менять по своему вкусу. Для энергичного контента паузы делают короче — четверть секунды или даже меньше. Для спокойного, вдумчивого — длиннее, до полутора секунд.

Выравниваем громкость фрагментов

Ещё одна важная задача при склеивании — выравнивание громкости. Музыкальная заставка часто бывает записана громче, чем голос. Если их склеить без обработки, слушателю придётся убавлять громкость на музыке и прибавлять на голосе. Это раздражает. Профессиональный монтаж подразумевает, что все части звучат на одном уровне.

Выравнивать громкость мы будем через пиковую нормализацию — метод, который мы кратко упоминали в Главе 1, а подробно разберём в Главе 3. Сейчас используем упрощённый вариант:

```
python
def normalize_volume(y, target_peak=0.9):
"""Подгоняет пиковую громкость к заданному уровню."""
current_peak = np.max(np.abs(y))
if current_peak == 0:
return y
gain = target_peak / current_peak
return y * gain

# Нормализуем все части
```

```
intro = normalize_volume(intro, 0.9)
voice = normalize_volume(voice, 0.9)
outro = normalize_volume(outro, 0.9)
```

После такого выравнивания все три части будут иметь одинаковый пиковый уровень, и слушателю не придётся тянуться к регулятору громкости.

Собираем готовый эпизод подкаста

Теперь давайте соберём всё вместе в один большой скрипт, который принимает набор файлов и выдаёт готовый смонтированный эпизод. Этот скрипт можно будет использовать для каждого выпуска подкаста, меняя только имена файлов.

```
python
import librosa
import numpy as np
import soundfile as sf


def build_podcast(intro_file, voice_file, outro_file, output_file,
trim_start=180, trim_end=10,
silence_between=1.0,
target_sr=22050,
target_peak=0.9):
"""
Собирает эпизод подкаста из трёх файлов.
trim_start - сколько секунд отрезать от начала голосовой
```

```
дорожки
    trim_end - сколько секунд отрезать от конца голосовой до-
рожки
    silence_between - длина паузы между частями в секундах
    """
    print("=== Сборка эпизода ===")

    # Загружаем
    print("Загружаю файлы...")
    intro, _ = librosa.load(intro_file, sr=target_sr)
    voice, _ = librosa.load(voice_file, sr=target_sr)
    outro, _ = librosa.load(outro_file, sr=target_sr)

    # Обрезаем голос
    start_sample = int(trim_start * target_sr)
    end_sample = len(voice) - int(trim_end * target_sr)
    voice = voice[start_sample:end_sample]
    print(f"Голос после обрезки: {len(voice) / target_sr:.1f}
сек")

    # Нормализуем громкость
    for arr in [intro, voice, outro]:
    peak = np.max(np.abs(arr))
    if peak > 0:
    arr *= target_peak / peak
```

```
# Создаём паузу
silence = np.zeros(int(target_sr * silence_between))

# Склеиваем
final = np.concatenate([intro, silence, voice, silence, outro])
print(f"Финальная длина: {len(final) / target_sr:.1f} сек")

# Сохраняем
sf.write(output_file, final, target_sr)
print(f"Сохранено: {output_file}")
print("Готово!")

# Запускаем
build_podcast(
intro_file='intro.wav',
voice_file='raw_voice.wav',
outro_file='outro.wav',
output_file='episode_ready.wav',
trim_start=180,
trim_end=10,
silence_between=1.0
)
```

Запустите этот скрипт, указав свои файлы. Если у вас нет готовых джинглов для интро и аутро, не беда — можете пока использовать любые музыкальные фрагменты или даже тишину. Главное — понять процесс. Когда появятся настоя-

щие джинглы, вы просто подставите их в скрипт, и всё будет работать.

Пакетная обработка: монтируем много файлов сразу

Одно из главных преимуществ кода перед ручным монтажом — возможность обрабатывать много файлов одной командой. Предположим, у вас в папке лежит десять записей, и каждую нужно обрезать с начала и с конца, а потом сохранить результат в другую папку. В Audacity это заняло бы час. В Python это делается одним скриптом и занимает минуту.

```
python
import os
import librosa
import soundfile as sf

input_folder = 'raw_recordings'
output_folder = 'trimmed_recordings'
```

Конец ознакомительного фрагмента.