Посвящается всем детям,
которые любят творить,
строить и исследовать
новые миры.

Оглавление

Как пользоваться этой книгой

Добро пожаловать! Как вы, наверное, уже догадались, эта книга познакомит вас с языком программирования Python, но вместе с тем, что гораздо важнее, она научит вас мыслить как программист.

Программисты умеют решать любые, самые сложные проблемы: они делят задачу на небольшие части, а затем пошагово разбираются с каждым таким кусочком. Этот навык пригодится вам не только при работе с кодом, но и в жизни!

В начале книги мы поговорим о том, что такое программирование вообще, а затем перейдем к изучению азов языка Python — и узнаем, как загрузить его себе на компьютер. Но прежде чем скачивать что-либо из интернета, обязательно посоветуйтесь со старшими, которым доверяете: с родителями, учителями или опекунами!

Главы с третьей по девятую — сплошное развлечение: практические примеры и уроки. Вы познакомитесь с кодированием и узнаете, как использовать его принципы для создания программ и игр. Каждая глава содержит три упражнения: первое попроще, второе посложнее, а третье — самое замысловатое. Все они нужны для того, чтобы вы усвоили изложенный в главе материал и потренировались в написании программ. К каждому упражнению прилагается полный код его решения, чтобы вы могли себя проверить.

Наконец, в десятой главе мы задействуем все полученные знания и приобретенные навыки, чтобы создать игру, в которую вы сможете сыграть на своем компьютере.

Когда вы полностью проработаете материал книги, почитайте раздел **«Код — вперед!»** на с. 199: там вы найдете много полезных советов, которые помогут вам улучшить навыки программирования.

Ну а если что-то пойдет не так, загляните в **Руководство по ловле багов** на с. 200. Кроме того, на с. 204 находится **Словарик**, где объясняются термины, используемые в программировании.

Начнем!

1

Добро пожаловать, или Программирование открывает двери

Вспомните, чем вы занимались последние двадцать четыре часа, что делали на компьютере, на планшете или мобильном телефоне. Может быть, фотографировали на смартфон или играли в видеоигры? Или общались в чате с друзьями, скачивали домашнее задание, смеялись над видео с забавными котиками или выясняли, сколько глаз у осьминога?

Компьютеры сейчас везде — по той причине, что у них очень, очень хорошо получается считать, причем чрезвычайно быстро и без ошибок.

И мы, люди, то есть разумные существа, приспособили компьютеры к тому, чтобы решать различные задачи. Мы нашли способ оформлять расчеты в виде кода, и теперь с его помощью программисты создают полезные программы. Например: как поддерживать связь с друзьями? Надо изобрести социальные сети. Как смотреть любимые фильмы, не тратя кучу денег

на диски и не забивая ими все полки? Нужно потоковое видео. Чем развлечься, когда скучно? Без видеоигр не обойтись.

Все эти задачи решаются с помощью компьютерных программ — полезных, интересных и забавных, — которые могут сделать нашу жизнь лучше. И здесь мы ограничены только собственным воображением.

Если вы раньше никогда не писали программы, эта книга — идеальный старт. Благодаря интересным примерам и упражнениям вы научитесь пользоваться инструментами для создания любых игр, приложений или сайтов — каких только пожелаете.

Что такое программы?

Программа — это фрагмент кода, предназначенный для решения задачи, которую ставят перед собой люди. Задача может быть сложной, например, «Как вывести спутник в космос?», или забавной, вроде «Как пририсовать участникам видеочата собачьи уши?»

Написать программу — это все равно что объяснить пришельцу, как добраться до школы. Ведь пришелец никогда не был на Земле и не знает ни что такое школа, ни что такое автобус, ни что такое перемена или домашнее задание. Растолковывать придется буквально все!

Для начала надо будет во всех деталях объяснить пришельцу, как ему дойти до автобусной остановки, например: «Тебе надо выйти из дома в 8:15 утра. Иди по подъездной дорожке, а потом сверни направо. Иди пять минут, остановись возле дуба и жди большую-большую желтую машину».

Предположим, ваш пришелец зашел в автобус. Но он не знает, какое место ему следует выбрать. Какие инструкции вы ему дадите? Можно, например, велеть ему сесть на сиденье, «где сидит меньше двух человек». Или сказать так: «Всегда садись у окна, если там есть свободное место». Или: «Иди в самый конец автобуса и садись там!»

Чтобы написать компьютерную программу — или дать инструкции пришельцу, — вы разбиваете большую задачу на задачи поменьше, и уже для решения каждой такой маленькой

подзадачи создаете набор конкретных пошаговых инструкций. Получается, что все решения, включая те, что вы принимаете автоматически, не задумываясь, – например, где сесть в автобусе, – придется обдумать и объяснить.

И если то, что у вас выйдет, перевести на язык программирования, результатом станет именно программа!

Программы могут решать самые разные задачи, но есть несколько вещей, которые присутствуют в любом коде:

- ввод данных;
- вывод результатов;
- вычисление;
- вычисление в зависимости от условия;
- повторное выполнение.

Давайте разберемся с ними

ВВОД ДАННЫХ

Предположим, вы хотите купить себе мороженое. Но прежде чем мороженщик займется приготовлением лакомства, он задаст вам несколько вопросов, например: «Маленькая, средняя или большая порция?», «В рожке или в стаканчике?», «Какое мороженое – клубничное, фисташковое, ванильное?»

Вводные данные – это информация, которая нужна вашей программе, чтобы начать работу. Если вы не определитесь, какое мороженое хотите, мороженщик ничего вам не даст!

Или так: представьте, что вы играете с друзьями в футбол. Что нужно, чтобы начать игру? Футбольный мяч, две команды игроков и какое-то футбольное поле или хотя бы асфальтированная площадка. Все это и будет «вводом данных» для вашего футбольного матча.

ВЫВОД РЕЗУЛЬТАТОВ

Вывод – это результат работы программы. Например, если вы заказываете «среднюю порцию малинового мороженого», именно ее вы и ожидаете получить.

А что является выводом результатов при игре в футбол? Каждая команда стремится выиграть, но счастливчика определяет счет. Так что в этом случае выводом может быть количество голов, забитых командами.

Говоря о вводе данных и выводе результатов, мы смотрим на то, с чего программа начинается и чем она заканчивается. Пока нам нет нужды знать, как добраться из пункта А в пункт Б, нам нужно выяснить, что в нашем случае пункт А и пункт Б.

ВЫЧИСЛЕНИЕ

В каждой компьютерной программе есть какие-нибудь арифметические расчеты. Иногда их видно сразу: например, маленькая порция мороженого отличается от большой весом, или у вас имеется 200 грамм мороженого, которые вы хотите честно разделить на троих. Или, быть может, вы хотите купить четыре рожка, и надо посчитать, сколько на это понадобится денег. Сложение, вычитание, умножение и деление – для компьютера это пара пустяков!

Но иногда еще пойди найди эти вычисления! На самом деле все, с чем работает компьютер, будь это текст или изображение, состоит из цифр. К счастью, большинство преобразований (из цифр в текст или картинку) происходит автоматически, и нам не надо беспокоиться о том, как видео с симпатичными котиками превращается в единички и нули благодаря невидимому коду. Мы просто полагаемся на арифметические расчеты и верим, что они не подведут.

Можно написать программу-инструкцию по продаже мороженого, мойке оборудования и замене пустых подносов из-под мороженого на полные. Но на самом деле, где-то за кулисами, все это превращается в цифры и арифметику.

ВЫЧИСЛЕНИЕ В ЗАВИСИМОСТИ ОТ УСЛОВИЯ

Если вы закажете мороженое в рожке, будет странно, если мороженщик вручит вам и рожок с мороженым, и стаканчик. Если вы заказываете фисташковое мороженое, значит, в этот раз вы не попробуете ни клубничное, ни ванильное.

Программы предназначены для того, чтобы удовлетворять вкусы разных пользователей. И, скажем, если ваша программа

состоит из 1000 строк, то хорошо, если в обычный день будут использоваться 500 из них (или хотя бы 100).

Если программа может вести себя по-разному в зависимости от ситуации, это называется условным исполнением. Без него программа бы делала для всех пользователей одно и то же, и пришлось бы вам все время есть только фисташковое мороженое или играть в одну и ту же футбольную игру. Скучновато, как по мне.

ПОВТОРНОЕ ВЫПОЛНЕНИЕ

Каждую программу можно представить как последовательность действий или решений определенных задач. Например, если вы играете в футбол, то задачи могут быть такие: ударить по мячу, отобрать мяч у противника, забить гол — или не дать противнику забить гол в ваши ворота. Наверняка, когда вы играете в футбол, вы неоднократно совершаете все эти действия.

Точно так же и большинство программ выполняет одни и те же действия снова и снова. Поэтому, если мы пишем код для зачерпывания мороженого, то разумно использовать один и тот же блок кода и для орехового мороженого, и для черничного: совершенно незачем писать лишние строчки!

Основы

Может быть, вы уже слышали о JavaScript, Scratch, C++, Ruby или каких-то еще языках программирования, которых в мире существует не один десяток.

Если так, то у меня для вас есть хорошая новость: на самом деле неважно, какой из этих языков учить первым. Если вы выучили один, то со вторым и третьим справиться будет легче. У каждого языка программирования есть свои хитрости, но все языки опираются на одни и те же базовые принципы кодирования. Другими словами, все языки используют один и тот же набор инструментов.

Так что давайте присмотримся к этим самым инструментам, сравнить которые можно с отвертками, сверлами, молотками и гаечными ключами, и поучимся с ними работать. Как только

вы поймете, как именно применяются те или иные инструменты для решения разных задач, вы сможете создать все, что угодно: дворец, избушку или автофургон для путешествий.

При написании кода самыми важными принципами, или инструментами, являются следующие:

- переменные;
- типы и структуры данных;
- условные инструкции;
- петли;
- функции.

Каждая программа, начиная с тех, что управляют беспилотными автомобилями, и заканчивая игрой «Морской бой», использует эти пять инструментов. Если вы научитесь ими пользоваться, то и научитесь мыслить как программист. Чем больше кода вы создадите с помощью этих инструментов, тем лучше будете понимать, как работают программы, – и сможете воплощать свои идеи в реальность! Быть может, в один прекрасный день вы, столкнувшись со сложной задачей, например, «Как найти самый короткий маршрут от дома до школы?», сможете превратить решение этой проблемы в программное обеспечение GPS*.

Главное – не забывать о том, что программирование – дело непростое. Все разработчики делают ошибки. Если с первого раза ваш код не работает так, как нужно, что с того? Со временем вы научитесь исправлять ошибки и добиваться успеха, пусть и не сразу. Главное – не бросать дело на полдороге. Ведь когда мы делаем неверные шаги, мы учимся чему-то новому.

Поговорим подробнее о принципах кодирования

ПЕРЕМЕННЫЕ

Переменные – это инструменты, с помощью которых сохраняется вся информация, используемая программой. Переменные можно сравнить с многоразовыми пластиковыми контейнерами.

---

* GPS – спутниковая система навигации, позволяет вам определять свое местоположение в любом месте Земли. – *Прим. пер.*

Закончили одно дело? Моете контейнер — и в него можно положить новые данные! Какое мороженое вы выбрали? Сколько голов забили, играя в футбол? Где-то ведь эта информация должна храниться. Если поместить ее в переменные, то код получается аккуратный и хорошо организованный, вдобавок его легко править.

ТИПЫ И СТРУКТУРЫ ДАННЫХ

Структуры данных — это инструменты, которые позволяют создавать наборы переменных. Если у вас есть миллион различных элементов, вам не обойтись без системы, с помощью которой вы будете ими управлять.

Если переменные можно сравнить с многоразовыми контейнерами, то структуры данных — это что-то вроде целого холодильника или ланч-бокса. С ними можно быть уверенным, что никакой контейнер не завалится в самый низ вашего школьного рюкзака, и «еда» в нем не протухнет.

Кроме того, каждый фрагмент данных может иметь какой-либо тип. **Тип данных** определяет, сколько места требуется данным в памяти компьютера и как данные реагируют на те или иные **математические операторы**, например, «плюс» (+), «минус» (-), «умножить» (*) и «разделить» (/). Очевидно, что для хранения остатков супа и нежных пирожных требуются разные контейнеры. Точно также числа, текст и переменные «да или нет» хранятся в разных емкостях.

УСЛОВНЫЕ ИНСТРУКЦИИ

С помощью **условных инструкций** вы можете создавать в своем коде развилки. Например, если пользователь выбирает карамельный сироп, развилка выполнит одну часть кода, а если шоколадный — то другую, и каждый выбор влечет за собой определенные последствия. Помните, пользователям обычно нужен не весь функционал кода, а только его отдельные части. С помощью конструкций условия мы можем решать, какие команды будет выполнять наш код и в каких случаях.

ЦИКЛЫ

Циклы — важный инструмент повторения кода. Чтобы понять, что такое «цикл», представьте себе петлю на «американских горках»: когда проходишь по одному и тому же маршруту снова и снова.

Если в процессе создания программы вам приходится решать очень похожие задачи, то лучше использовать цикл, чтобы заново не писать один и тот же код. Например, процедура зачерпывания мороженого с манго вряд ли сильно отличается от зачерпывания мороженого со вкусом жевательной резинки. Вот и отлично, можно сделать цикл!

ФУНКЦИИ

Функции – это инструмент, с помощью которого мы можем определять для разных фрагментов кода их назначение. Например, за прилавком могут работать двое мороженщиков: один делает мороженое, а второй принимает деньги. В футболе бывают нападающие, защитники и вратари, ведь если не определить каждому его задачу, будет сплошной беспорядок!

Как и циклы, функции применяются для повторного использования кода. Еще они нужны, чтобы код был четким и понятным. Записывая код, задавайтесь не только вопросом: «Какие задачи повторяются снова и снова?» (это повод для создания цикла), но думайте и о том, как разбить сложную задачу, вроде игры в футбол, на задачи поменьше. Так вы сможете понять, какие функции вам нужны.

Говори как программист: слова, которые надо знать

Поначалу процесс создания кода может выглядеть устрашающе, потому что программисты используют кучу странных словечек.

Давайте побеседуем про самые распространенные термины программирования

АЛГОРИТМЫ

Алгоритм – это набор точных пошаговых инструкций, нацеленный на решение какой-либо задачи. Погодите! А чем же тогда алгоритм отличается от программы?

Дело в том, что компьютерные программы предназначены для решения больших и неконкретных «человеческих» задач («где взять идею для поделки» или «как добраться до зоопарка»), а алгоритмы нужны для решения задач поменьше, так сказать, «компьютерных». Что это может быть? Например, «как умножить друг на друга два очень больших числа» или «как найти самое маленькое число в списке, в котором находится миллион чисел».

В одной программе может использоваться много алгоритмов, каждый из которых помогает программе продвинуться на шаг ближе к достижению цели, для которой человек и создал эту программу.

КОМПИЛЯЦИЯ

Сами по себе компьютеры не знают никаких языков – ни русского, ни английского. И, на самом деле, не понимают они и языков программирования, таких как Python, JavaScript или C#. Сильно упрощая, можно сказать, что компьютеры состоят из электронных переключателей, у которых есть два положения: «включено» (1) и «выключено» (0). **Компилирование** – это преобразование кода, который могут читать люди, в последовательность нолей и единиц, которую способен понять компьютер.

ИСПОЛНЕНИЕ ЗАДАЧИ

Чтобы код начал выполняться, его нужно запустить: это то же самое, что сказать «давай, вперед!» и позволить компьютеру выполнить данные ему инструкции – строка за строкой. Представьте, что водитель запускает двигатель автомобиля, выезжает из гаража и объезжает вокруг квартала. Если двигатель запущен и машина едет, значит, автомобиль работает, а не стоит в гараже без дела. Точно также и запущенный код не бездельничает, а выполняет поставленную перед ним задачу.

Время исполнения задачи – это период между началом работы программы (запуск двигателя автомобиля) и окончанием этой работы (машина завершила поездку, объехав вокруг квартала).

ОТЛАДКА

Люди несовершенны, и иногда наш код делает не то, что мы хотим. А иногда он вообще делает то, чего мы не хотим! В таких