1 Плавное погружение в Python

Дай человеку рыбу — и он будет сыт один день. Научи человека рыбачить — и он будет сыт всю жизнь.

Китайская пословица

Программирование — или, как его еще называют, кодинг — это процесс, в ходе которого вы даете компьютеру инструкции на понятном для него языке.

Компьютеры — это мощные устройства, которые не умеют самостоятельно мыслить. Им нужно объяснить все: как выполнять задачи, как проверять условия, чтобы выбрать нужный путь, как обрабатывать данные, поступающие, например, с жесткого диска или из сети, и как реагировать на непредвиденные ситуации — скажем, если что-то отсутствует или вышло из строя.

Писать код можно по-разному, используя различные языки. Это трудно? И да и нет. Это похоже на письмо: научиться может каждый. Но если вы хотите стать поэтом — одного умения писать недостаточно. Придется развить целый набор других навыков, и это займет больше времени и сил.

В конечном счете все зависит от того, как далеко вы готовы пойти. Кодинг — это не просто составление работающих инструкций. Это нечто гораздо большее!

Хороший код — это короткий, быстрый, элегантный, понятный и удобный для чтения код. Его легко менять, расширять, масштабировать, тестировать и рефакторить. Чтобы научиться писать такой код, потребуется время. Но хорошая новость в том, что, читая эту книгу, вы уже делаете первый шаг. И мы не сомневаемся, что у вас получится. На самом деле это под силу каждому: ведь все мы программируем — просто не всегда осознаем это.

Допустим, вы хотите заварить растворимый кофе. Вам нужно взять кружку, банку с кофе, чайную ложку, воду и чайник. Даже если вы не задумываетесь об этом, в процессе вы анализируете множество факторов. Вы проверяете, есть ли вода в чайнике, включен ли он в розетку, чистая ли кружка и достаточно ли кофе в банке. Затем вы кипятите воду и, возможно, параллельно насыпаете кофе в кружку. Когда вода закипит, вы наливаете ее в кружку и размешиваете.

При чем здесь программирование?

Ну, мы собрали необходимые ресурсы (чайник, кофе, воду, ложку и кружку) и проверили условия, связанные с ними (чайник подключен, кружка чистая, кофе достаточно). Затем запустили два действия: начали кипятить воду и засыпали кофе в кружку. А когда оба действия завершились, мы закончили процедуру — налили воду в кружку и размешали.

Видите параллель? Мы только что описали высокоуровневую «функциональность» программы для приготовления кофе. Это было несложно, поскольку наш мозг и так делает это постоянно: оценивает условия, принимает решения, выполняет действия, повторяет какие-то из них и в какой-то момент завершает выполнение.

Теперь вам нужно лишь научиться раскладывать такие действия, которые вы обычно выполняете автоматически, на шаги, понятные компьютеру. А еще — выучить язык, на котором можно давать компьютеру такие инструкции.

Именно в этом и поможет наша книга. Мы покажем один из возможных подходов к успешному программированию и используем для этого множество простых, но конкретных примеров (мы такие особенно любим).

Основы программирования

Когда мы обучаем программированию, стараемся использовать примеры из реальной жизни. Мы уверены, что такие параллели помогают лучше запоминать новые понятия. Однако сейчас пришло время подойти к вопросу более строго и взглянуть на программирование с технической точки зрения.

Когда мы пишем код, мы даем компьютеру инструкции: что и как он должен делать. Где все это происходит? В разных частях системы: в оперативной памяти, на жестких дисках, в сетевых кабелях, процессоре и т. д. Это целый мир, который, как правило, представляет собой модель части реального мира.

Если вы пишете программу, с помощью которой люди могут покупать одежду онлайн, то вам нужно будет отразить в ней реальные сущности: людей, одежду, бренды, размеры — все это должно существовать внутри программы.

Для этого вы будете создавать и использовать объекты. Человек — это объект. Автомобиль — тоже объект, как и брюки. К счастью, Python отлично работает с объектами.

Любой объект в программе обладает двумя основными характеристиками: *свойствами* и *методами*. Например, если мы представляем человека как объект, то в программе это может быть клиент или сотрудник. Он может обладать такими свойствами, как имя, номер социального страхования, возраст, наличие водительских прав, адрес электронной почты и т. д. В коде мы сохраняем всю информацию, необходимую для использования объекта в соответствии с его назначением. Например, если вы разрабатываете сайт по продаже одежды, то вам, скорее всего,

нужно будет хранить данные о росте и весе клиента, а также другие параметры, которые позволят подбирать подходящие для него вещи.

Свойства — это характеристики объекта. Мы постоянно пользуемся ими в жизни.

— Подай мне ручку.

— Какую?

— Черную.

Здесь мы использовали свойство цвета (черная), чтобы указать на нужную ручку, — вероятно, потому, что рядом лежали и другие, отличающиеся по цвету.

Методы — это действия, которые объект может выполнять. Как человек, я могу *говорить*, *ходить*, *спать*, *просыпаться*, *есть*, *мечтать*, *писать*, *читать* и т. д. Все мои действия можно представить как методы объекта, который меня описывает.

Теперь, зная, что такое объекты, что у них есть методы (действия) и свойства (характеристики), вы готовы к программированию. Ведь, по сути, программирование — это управление объектами, которые существуют в модели мира, реализованной в вашем программном обеспечении. Вы можете создавать объекты, использовать их (в том числе повторно) и удалять по своему усмотрению.

В разделе *Data Model* официальной документации Python (https://docs.python.org/3/reference/datamodel.html) сказано:

> «Объекты — это абстракция, с помощью которой Python работает с данными. Все данные в Python представлены объектами или связями между ними».

К объектам в Python мы еще вернемся в главе 6. Пока же вам нужно знать только одно: каждый объект в этом языке имеет *идентификатор* (ID), *тип* и *значение*.

Объектно-ориентированное программирование (ООП) — лишь один из множества подходов к разработке. Python позволяет писать код в функциональном, императивном и объектно-ориентированном стиле. Однако, как мы уже говорили, в Python все является объектом — поэтому мы так или иначе постоянно работаем с объектами, вне зависимости от выбранного стиля программирования.

После создания объекта его идентификатор никогда не меняется. Это его уникальный номер, который Python использует «за кадром», чтобы находить объект при обращении к нему. Не изменяется и тип объекта. Он определяет, какие операции поддерживает объект и какие значения ему можно присваивать. С самыми важными типами данных в Python мы познакомимся в главе 2. А вот значение некоторых объектов может меняться. Такие объекты называют *изменяемыми* (mutable). Если же значение изменить нельзя — объект является *неизменяемым* (immutable).

Чтобы использовать объект, надо дать ему имя! Вы сможете обращаться к нему по имени и использовать его в программе. В более общем смысле объекты — такие как числа, строки (текст) и коллекции — связываются с именами. В языках

программирования такие имена обычно называют *переменными*. Переменную можно представить себе как коробку, в которую вы помещаете данные.

Объекты — это представление данных. Данные хранятся в базах или передаются по сети. Это то, что вы видите, открывая веб-страницу или работая с документом. Компьютерные программы обрабатывают эти данные: выполняют с ними действия, управляют их потоком, проверяют условия, реагируют на события и т. п.

Чтобы делать все это, нужен язык. Именно для этого и существует Python. В книге мы будем рассматривать, как с его помощью давать компьютеру понятные инструкции.

Добро пожаловать в мир Python

Python — замечательное творение Гвидо ван Россума, нидерландского ученого и математика, который решил подарить миру проект, над которым он экспериментировал во время рождественских каникул 1989 года. Публике Python был представлен примерно в 1991 году и с тех пор превратился в один из самых популярных языков программирования.

Мы, авторы книги, начали программировать еще в юности. Фабрицио начал в семь лет — на компьютере Commodore VIC-20, который позже был заменен на его более мощного «старшего брата» — Commodore 64. В нем использовался язык *BASIC*. Генрих познакомился с программированием в школе, когда начал учить Pascal. Вместе мы успели поработать с Pascal, Assembly, C, C++, Java, JavaScript, Visual Basic, PHP, ASP, ASP.NET, C# и множеством других языков, названия которых уже и не вспомнить. Но только когда мы впервые столкнулись с Python, у нас появилось то самое чувство, которое бывает, когда находишь в магазине нужный диван — *все внутри говорит: бери этот, он идеален!*

Мы привыкли к Python примерно за день. Синтаксис оказался непривычным — не таким, как в языках, с которыми мы работали раньше. Но, преодолев это легкое ощущение неловкости, мы сразу же влюбились в Python. По-настоящему. А почему — расскажем далее.

Коротко о Python

Прежде чем погрузиться в технические детали, разберемся, почему вообще стоит использовать Python. Ниже описаны его ключевые достоинства.

Переносимость

Python работает везде. Перенос программы из Linux в Windows или macOS обычно сводится к корректировке путей и настроек. Язык изначально спроектирован как переносимый, и он сам заботится о различиях между *операционными*

системами (ОС), пряча их за удобными интерфейсами. Вам не нужно писать специальный код для каждой платформы.

Логичность

Python отличается высокой логичностью и внутренней согласованностью. Сразу видно, что этот язык создал талантливый специалист. Очень часто название метода можно просто угадать, даже если вы его не знаете.

Вы можете пока не осознавать, насколько важны логичность и согласованность (особенно если у вас еще не так много опыта), но они дают огромное преимущество. Благодаря им уменьшается необходимость поисков в документации и снижается когнитивная нагрузка при программировании.

Скорость разработки

Как пишет Марк Лутц в пятом издании книги *Learning Python*[1], код Python обычно в пять раз короче, чем эквивалентный код на Java или C++. А значит, работа выполняется быстрее. А быстрее — это хорошо. Это значит, что вы можете быстрее реагировать на изменения рынка. Меньше строк означает снижение объемов кода, который нужно не только писать, но и читать (а профессиональные программисты читают код гораздо чаще, чем пишут), а также сопровождать, отлаживать и рефакторить.

Кроме того, Python не требует долгих этапов компиляции и обработки зависимостей. Вам не нужно ждать, чтобы увидеть результат своей работы, — код запускается сразу.

Обширная библиотека

Python обладает невероятно богатой стандартной библиотекой — недаром о нем говорят, что он поставляется *«в комплекте с батарейками»*. Но и это еще не все: международное сообщество Python поддерживает огромное количество сторонних библиотек, ориентированных на самые разные задачи. Эти библиотеки свободно доступны через репозиторий *Python Package Index (PyPI)*. Когда вы пишете код и понимаете, что вам нужна некая функциональность, почти всегда найдется хотя бы одна библиотека, в которой она уже реализована.

Качество программ

В Python большое внимание уделено удобству чтения, логичности и качеству кода. Благодаря единообразию синтаксиса программы на Python легко читать, а это особенно важно сейчас, когда разработкой занимаются коллективы,

[1] *Лутц М.* Изучаем Python. — М., 2024.

а не один человек. Кроме того, Python — язык с мультипарадигменной основой. Вы можете использовать его как скриптовый язык, а можете программировать в объектно-ориентированном, императивном или функциональном стиле. Python — по-настоящему универсальный язык.

Интеграция с другим ПО

У Python есть еще одно важное преимущество: его можно интегрировать в другие языки программирования. Это означает, что, даже если основная разработка в компании ведется на другом языке, Python может стать связующим звеном между разными частями сложной системы, которым нужно «общаться» друг с другом. Это уже более сложный уровень, но в реальной практике такая гибкость играет важную роль.

Использование в Data Science

Python — один из самых популярных (если не *самый* популярный) языков в таких областях, как анализ данных, машинное обучение и искусственный интеллект. Поэтому знать Python сегодня практически обязательно всем, кто хочет строить карьеру в этих направлениях.

Удовольствие от работы

И последнее, но не менее важное. Работать с Python — действительно в радость. Мы можем писать код восемь часов подряд и уходить из офиса с ощущением удовлетворения и легкости. Нас не изматывают трудности, с которыми сталкиваются другие разработчики, использующие языки с менее удобными структурами и инструментами. Нет сомнений, что Python делает программирование приятным занятием. А удовольствие, как известно, — один из лучших стимулов к обучению и продуктивной работе.

Вот основные причины, по которым мы рекомендуем Python. Конечно, у него есть и множество других технических особенностей и расширенных возможностей, о которых мы могли бы рассказать. Но все они выходят за рамки вводной главы. Вы будете с ними знакомиться по мере изучения Python, читая эту книгу глава за главой.

А теперь рассмотрим ограничения Python.

Недостатки Python

Если не учитывать вопрос личных предпочтений, то главный недостаток Python — скорость выполнения. Как правило, этот язык работает медленнее, чем его компилируемые собратья. Стандартная реализация Python при запуске приложения сначала компилирует исходный код в байт-код (файлы с расширением `.pyc`),

а затем передает его на выполнение интерпретатору. Преимущество такого подхода — переносимость, но достигается она ценой более медленной работы, поскольку Python не компилируется в машинный код напрямую, как, например, C или C++.

Тем не менее в большинстве случаев низкая скорость Python не становится проблемой, и именно поэтому язык используется так широко. В реальной жизни стоимость оборудования уже давно некритична, и во многих случаях повысить производительность можно за счет распараллеливания задач. Кроме того, многие программы тратят бóльшую часть времени на завершение операций ввода-вывода, поэтому чистая скорость выполнения нередко оказывается второстепенным фактором по отношению к общей производительности.

Надо отметить, что разработчики ядра Python в течение последних нескольких лет приложили немало усилий, стремясь оптимизировать работу со стандартными структурами данных, и во многих случаях эти улучшения дали ощутимый результат.

Если же скорость действительно критична, то можно рассмотреть альтернативные реализации Python, такие как *PyPy*, которая в среднем дает четырехкратное ускорение за счет более сложных приемов компиляции (см. https://pypy.org). Кроме того, для участков кода, где важна максимальная производительность, можно использовать такие языки, как C или C++, и затем интегрировать эти компоненты с кодом Python. В библиотеках *pandas* и *NumPy*, предназначенных для обработки и анализа данных, активно используется такой подход.

Существует несколько различных реализаций Python. Мы будем использовать основную — CPython. Перечень альтернатив доступен на сайте https://www.python.org/download/alternatives/.

Если вся эта информация кажется недостаточно убедительной, то напомним: Python лежит в основе серверной логики таких гигантов, как Spotify и Instagram, где производительность имеет огромное значение. Так что этот язык отлично справляется со своей задачей — даже там, где требования весьма высоки.

Как Python используют сегодня

Python используется в самых разных областях: в системном программировании, веб-разработке и работе с API, в графических приложениях, играх и робототехнике, при быстром прототипировании, системной интеграции, анализе данных, в базах данных, работе с потоками в реальном времени и многом другом. Кроме того, несколько ведущих университетов выбрали Python в качестве основного языка в курсах по информатике.

Далее приведен список крупных компаний и организаций и дано пояснение, в каких целях они используют Python в своей технологической инфраструктуре, при разработке продуктов, анализе данных или автоматизации процессов.

- Технологические компании:
  - Google — для серверной логики, анализа данных и систем искусственного интеллекта;
  - Facebook — для управления инфраструктурой и автоматизации операций;
  - Instagram — в значительной степени построен на Python, активно использует фреймворк Django;
  - Spotify — для анализа данных и серверной разработки;
  - Netflix — для анализа данных, автоматизации и обеспечения безопасности.
- Финансовый сектор:
  - JP Morgan Chase — для финансового моделирования, анализа данных и алгоритмической торговли;
  - Goldman Sachs — в различных финансовых приложениях;
  - Bloomberg — для анализа финансовых данных и интерфейса Bloomberg Terminal.
- Технологии и ПО:
  - IBM — для решений в области ИИ, машинного обучения и кибербезопасности;
  - Intel — для тестирования оборудования и в процессах разработки;
  - Dropbox — клиентская часть классического приложения в значительной степени написана на Python.
- Космос и научные исследования:
  - NASA — для анализа данных и интеграции систем;
  - CERN — для обработки и анализа данных в физических экспериментах.
- Розничная и электронная торговля:
  - Amazon — для анализа данных, рекомендаций товаров и автоматизации операций;
  - eBay — для серверной логики и анализа данных.
- Индустрия развлечений и медиа:
  - Pixar — для разработки инструментов анимации и написания сценариев для производственного процесса;
  - Industrial Light & Magic (ILM) — в визуальных эффектах и обработке изображений.

- Образование и онлайн-платформы:
  - Coursera — для веб-разработки и серверных решений;
  - Khan Academy — для доставки учебного контента и серверной логики.
- Государственный сектор и некоммерческие организации:
  - Федеральное правительство США — различные департаменты используют Python для анализа данных, кибербезопасности и автоматизации;
  - Фонд Raspberry Pi — использует Python как основной язык программирования для образовательных проектов.

Настройка окружения

На наших компьютерах (MacBook Pro) установлена последняя версия Python:

```
>>> import sys
>>> print(sys.version)
3.12.2 (main, Feb 14 2024, 14:16:36) [Clang 15.0.0 (clang-1500.1.0.2.5)]
```

Как видите, это версия 3.12.2, выпущенная 2 октября 2023 года[1]. А перед вами — небольшой фрагмент кода на Python, введенный в консоли. Скоро мы поговорим об этом подробнее.

Все примеры из книги запускаются с использованием Python версии 3.12. Если вы хотите выполнять примеры и работать с исходным кодом из книги, то, пожалуйста, убедитесь, что используете ту же версию.

Установка Python

Процесс установки языка зависит от операционной системы, которую вы используете. Он уже установлен по умолчанию в большинстве дистрибутивов Linux. В современных версиях macOS он тоже, скорее всего, уже есть. А вот в Windows его, как правило, нужно устанавливать вручную.

Даже если Python уже установлен в системе, важно убедиться, что это именно версия 3.12.

Чтобы проверить установленную версию Python, введите в командной строке одну из следующих команд: python --version или python3 --version. (Об этом мы еще поговорим.)

[1] На самом деле в этот день была выпущена версия 3.12.0, а 3.12.2 — 6 февраля 2024 года. Время в выводе команды показывает не дату релиза, а дату компиляции интерпретатора. — *Примеч. науч. ред.*

Первое место, куда стоит заглянуть, — официальный сайт Python https://www.python.org. Там вы найдете официальную документацию и множество полезных ресурсов.

Полезные ссылки по установке

На сайте Python собраны инструкции по установке языка для различных операционных систем. Выберите ту, которая соответствует вашей системе.

Windows и macOS:

- https://docs.python.org/3/using/windows.html;
- https://docs.python.org/3/using/mac.html.

Linux:

- https://docs.python.org/3/using/unix.html;
- https://ubuntuhandbook.org/index.php/2023/05/install-python-3-12-ubuntu/.

Установка Python в Windows

В качестве примера рассмотрим установку Python в Windows. Перейдите на сайт https://www.python.org/downloads/ и скачайте установочный файл, подходящий для архитектуры вашего процессора.

После этого дважды щелкните на скачанном файле, чтобы запустить установку (рис. 1.1).

Рис. 1.1. Запуск установки в Windows