1 Знакомство с Python

В этой главе:

- ✓ Почему стоит использовать Python?
- ✓ Сильные стороны Python
- ✓ Что в Python стоит улучшить

Эта книга призвана помочь читателям как можно скорее обрести прочное общее представление о языке Python, избегая при этом углубления в продвинутые темы, но охватывая все необходимое для написания и чтения кода. В частности, она предназначена для тех, кто изучал другие языки программирования или уже немного знаком с Python, однако стремится расширить свои знания и практические навыки, поскольку популярность Python продолжает расти, особенно в таких областях, как data science, машинное обучение и научные исследования. Хотя никакого предварительного знакомства с Python не требуется, для извлечения максимальной пользы из этой книги желательно обладать некоторыми познаниями и опытом в сфере программирования.

После введения в Python и рекомендаций по началу работы в его среде приводится краткий обзор синтаксиса Python, а затем следуют главы, в которых рассматриваются встроенные типы данных, создание функций, классов и пакетов, а также некоторые расширенные возможности и практический пример работы с данными.

С развитием инструментов искусственного интеллекта (ИИ) на базе больших языковых моделей (LLM) появилась возможность автоматически создавать все бóльшие объемы программного кода при условии (и это немаловажно), что разработчик обладает достаточными знаниями в области программирования, чтобы грамотно управлять этим процессом. И хотя издание *не является* учебником по ИИ и его применению для генерации кода, задачи по программированию, поставленные в конце каждой главы, начиная с главы 5, содержат примеры ответов ИИ на те же вопросы, а также краткое обсуждение того, что искусственный интеллект сделал правильно (и что неправильно). Это поможет вам понять, как при помощи инструментов ИИ создавать программный код, который действительно работает.

1.1. Почему стоит использовать Python?

В современном мире существуют сотни языков программирования, от проверенных временем — таких, как C и C++, — до относительно новых — Rust, Go и C#, а также мощных корпоративных решений вроде Java и более веб-ориентированных, как JavaScript и Typescript. Такое изобилие затрудняет выбор языка программирования. Хотя ни один язык не может считаться идеальным вариантом для всех возможных ситуаций, я думаю, что Python хорошо подходит для решения многих задач программирования; кроме того, это отличный выбор для тех, кто только учится программировать. Миллионы программистов по всему миру используют Python, и их число растет год от года.

Python продолжает привлекать новых пользователей по целому ряду причин. Это полноценный кросс-платформенный язык, который одинаково хорошо работает на платформах Windows, Linux/UNIX и iOS, а также многих других, от суперкомпьютеров до мобильных устройств. Он может применяться для создания небольших приложений и быстрых прототипов, но также прекрасно масштабируется для разработки крупных программ. В состав Python входит мощный и удобный инструментарий для создания графических пользовательских интерфейсов (GUI), библиотеки для веб-программирования и многое другое. Python также стал важнейшим инструментом для выполнения научных вычислений, для науки о данных, машинного обучения и работы с искусственным интеллектом. *И все это* бесплатно.

1.2. С чем хорошо справляется Python

Python — современный язык программирования, созданный Гвидо ван Россумом (Guido van Rossum) в 1990-е годы (и получивший свое название в честь знаменитой комедийной труппы «Монти Пайтон» («Monty Python»), а точнее, в честь их популярного телешоу «Летающий цирк Монти Пайтона» («Monty Python's Flying Circus»)). Хотя Python нельзя назвать идеальным средством для создания любых приложений, благодаря своим сильным сторонам он хорошо подходит для решения множества различных задач.

1.2.1. Python прост в использовании

У программистов, знакомых с традиционными языками программирования, не возникнет особых трудностей с изучением Python. В нем содержатся все знакомые конструкции: циклы, условные операторы, массивы и т. п., однако многие из них проще в использовании. И вот почему.

- *Типы связываются с объектами, а не с переменными.* Переменная может указывать на объекты любого типа, а список может содержать объекты различных типов. Это также означает, что преобразования типов обычно не требуется, а ваш код не ограничен жесткими рамками заранее объявленных типов. И хотя для переменных типы не требуются, Python позволяет использовать аннотации типов, дающие разработчикам возможность отслеживать согласованность своего кода в отношении типов объектов, используемых для параметров, возвращаемых значений и т. п. Далее в книге мы уделим внимание аннотациям типов.
- *Python обычно работает на более высоком уровне абстракции.* Отчасти это обусловлено особенностями языка, а отчасти объясняется обширной стандартной библиотекой, включенной в дистрибутив Python. Программу для загрузки веб-страницы можно написать всего в две-три строки!
- *Правила синтаксиса очень просты.* Конечно, чтобы стать настоящим гуру, потребуется немало времени и усилий, однако даже новички способны освоить синтаксис Python в достаточной мере для того, чтобы свободно писать работающий код.

Python великолепно подходит для быстрой разработки приложений. На Python приложение нередко создается в пять раз быстрее, чем на C или Java, и при этом занимает впятеро меньше строк, чем аналогичная программа на C. Безусловно, все зависит от конкретного приложения; для числовых алгоритмов, выполняющих в основном целочисленные операции в циклах `for`, прирост производительности будет куда менее заметным. Однако в среднем такой прирост может оказаться весьма существенным.

1.2.2. Выразительность Python

Язык Python чрезвычайно выразителен. Под *выразительностью* в данном контексте понимается то, что одна строка кода Python может сделать намного больше, чем одна строка кода в большинстве других языков. Преимущества более выразительного языка очевидны: чем меньше строк кода вам придется написать, тем скорее вы сможете сделать проект. Чем меньше строк кода содержит программа, тем меньше проблем будет с ее поддержкой и отладкой.

Чтобы оценить, как выразительность Python упрощает код, возьмем задачу перестановки значений двух переменных, `var1` и `var2`. В таком языке, как Java, для этого потребуются три строки кода и дополнительная переменная:

```
int temp = var1;
var1 = var2;
var2 = temp;
```

Переменная `temp` необходима для сохранения значения `var1` перед тем, как ей будет присвоено значение `var2`, и последующего присвоения сохраненного значения `var2`. Процесс не особенно сложен, но, чтобы прочитать эти три строки и понять, что произошла перестановка, даже опытному программисту придется немного поразмыслить.

В отличие от этого, Python позволяет выполнить ту же перестановку в одной строке, причем так, что читатель программного кода сразу понимает, что значения меняются местами:

```
var2, var1 = var1, var2
```

Разумеется, это очень простой пример, однако язык Python просто изобилует такими преимуществами.

1.2.3. Удобочитаемость кода Python

Следующее преимущество кода Python заключается в том, что он легко читается. Казалось бы, язык программирования предназначен исключительно для компьютерной обработки, однако людям тоже приходится читать программный код: тому, кто занимается отладкой вашей программы (возможно, это будете вы сами), тому, кто осуществляет ее поддержку (опять-таки им можете оказаться вы), тому, кто, возможно, захочет модифицировать ее код в будущем. Во всех этих ситуациях чем легче прочесть и понять код, тем лучше.

Таким образом, чем понятнее код программы, тем его проще отлаживать, поддерживать и модифицировать. Главное преимущество Python в этом отношении — использование отступов. В отличие от большинства языков, Python *требует*, чтобы блоки кода снабжались отступами. Кому-то это требование может показаться странным, но оно гарантирует, что ваш код всегда будет отформатирован в удобочитаемом виде.

Ниже приведены две короткие программы, одна из которых написана на Perl, а другая — на Python. Обе получают два списка чисел одинакового размера и возвращают попарную сумму этих списков. По-моему, код Python читается лучше, чем код Perl; он визуально чище и содержит меньше непонятных символов:

```
# Версия Perl.
sub pairwise_sum {
    my($arg1, $arg2) = @_;
    my @result;
    for(0 .. $#$arg1) {
        push(@result, $arg1->[$_] + $arg2->[$_]);
    }
    return(\@result);
}
```

```
# Версия Python.
def pairwise_sum(list1, list2):
    result = []
    for i in range(len(list1)):
        result.append(list1[i] + list2[i])
    return result
```

Оба фрагмента кода выполняют одно и то же действие, однако код Python выигрывает в отношении удобочитаемости. (Справедливости ради, существуют и другие способы сделать это на Perl, некоторые из них гораздо компактнее, но читаются они, на мой взгляд, еще хуже.)

1.2.4. Полнота Python: «батарейки в комплекте»

Еще одно преимущество Python — его философия «батарейки в комплекте» в отношении модулей. Суть ее заключается в том, что при установке Python вы получаете весь рабочий инструментарий целиком, без необходимости подключать дополнительные модули. Вот почему стандартная библиотека Python поставляется с модулями для работы с электронной почтой, веб-страницами, базами данных, вызовами операционной системы, разработкой графического пользовательского интерфейса и т. п.

К примеру, на языке Python вам потребуются лишь две строчки кода, чтобы написать веб-сервер для обеспечения совместного доступа к файлам в каталоге:

```
import http.server
http.server.test(HandlerClass=http.server.SimpleHTTPRequestHandler)
```

Нет необходимости устанавливать модули для обработки сетевых подключений и HTTP, вся функциональность уже доступна в установочной версии Python.

1.2.5. Богатая экосистема сторонних библиотек

Несмотря на то что Python «укомплектован под завязку», неизбежны ситуации, когда все же приходится выходить за рамки полностью оснащенной стандартной библиотеки — это может быть узкоспециальная задача, новый формат данных, более сложные приложения и т. п. В этом отношении Python вышел на передовые позиции за последнее десятилетие. Во множестве областей, от веб-приложений и API до обработки и визуализации данных, машинного обучения и науки о данных, Python обладает одной из богатейших экосистем пакетов, модулей и фреймворков среди всех существующих языков программирования. И потому вероятность оказаться в ситуации, когда не найдется ни одного пакета Python, отвечающего вашим потребностям, ничтожно мала.

1.2.6. Кросс-платформенность

Python также является превосходным кросс-платформенным языком. Он работает на многих платформах: Windows, Mac, Linux, UNIX и др. Так как язык

является интерпретируемым, один и тот же код может выполняться на любой платформе с интерпретатором Python, а он сейчас реализован практически на всех современных платформах. Существуют даже версии Python, работающие на базе Java (Jython) и .NET (IronPython), а также на микроконтроллерах (MicroPython и CircuitPython), что дополнительно расширяет круг возможных платформ для работы с Python.

1.2.7. Свободное распространение

Наконец, за Python не нужно платить. Python изначально разрабатывался и продолжает разрабатываться по модели открытого исходного кода (open source) и находится в свободном доступе. Вы можете скачать и установить практически любую версию Python, использовать ее для разработки программного обеспечения как для коммерческих, так и для личных приложений, и вам не придется заплатить за это ни копейки.

Хотя отношение к бесплатному программному обеспечению постепенно меняется, некоторые по-прежнему относятся к нему настороженно из-за недостаточного уровня поддержки и веса в лице платных клиентов. Тем не менее многие авторитетные компании применяют Python в ключевых сферах своего бизнеса. Google, Bloomberg, NVIDIA, Capital One — вот лишь несколько примеров. Эти и многие другие компании справедливо считают Python весьма надежным, стабильным и хорошо поддерживаемым продуктом с активным и компетентным сообществом пользователей. На многочисленных интернет-форумах, посвященных Python, даже на самый трудный вопрос можно получить ответ быстрее, чем по большинству горячих линий технической поддержки, причем ответ будет правильным и бесплатным.

Python и ПО с открытым исходным кодом

Не только сам Python бесплатен, но и его исходный код находится в открытом доступе, и при желании вы вправе свободно модифицировать, улучшать и дополнять его. Вы можете заняться этим самостоятельно или нанять того, кто сделает это за вас. В случае с коммерческим программным обеспечением такая возможность редко предоставляется по разумной цене.

Если вы впервые сталкиваетесь с миром ПО с открытым исходным кодом, то вам следует понять, что вы можете не только свободно использовать и модифицировать Python, но и вносить свой вклад в его развитие и совершенствование (и это даже приветствуется). В зависимости от ваших обстоятельств, интересов и квалификации это может быть финансовый вклад (в частности, пожертвование в фонд Python Software Foundation), участие в одной из специальных групп по интересам (SIG — special interest group), тестирование и предоставление отзывов о релизах ядра Python или одного из вспомогательных модулей, а также предоставление сообществу части того, что разрабатываете вы или ваша компания. Конечно, уровень вклада зависит только от вас, но, если вы можете сделать что-то для других, это определенно того сто́ит. Здесь общими усилиями создается нечто весьма ценное, и у вас есть возможность внести свой посильный вклад.

Итак, у Python есть масса преимуществ: выразительность, удобочитаемость, богатый набор встроенных библиотек, кросс-платформенность, а также открытый исходный код. В чем же подвох?

1.3. Что в Python стоит улучшить

Хотя Python обладает многочисленными плюсами и постоянно совершенствуется, он неидеален, так как ни один язык не может удовлетворить абсолютно все потребности. Чтобы определить, станет ли Python подходящим языком для решения ваших задач, нужно обозначить и те области, в которых Python пока отстает.

1.3.1. Не самый быстрый язык

Один из возможных недостатков Python — скорость выполнения кода. Он не является полностью компилируемым языком. Вместо этого код сначала компилируется во внутренний байт-код, который затем выполняется интерпретатором Python. Есть ряд задач, в частности парсинг строк при помощи регулярных выражений, для которых Python имеет эффективную реализацию и работает так же быстро, а то и быстрее, чем любая программа на C, которую вы сможете написать. Тем не менее в большинстве случаев программы, написанные на Python, работают медленнее по сравнению с программами на C. Впрочем, на это следует взглянуть здраво: современные компьютеры обладают такой вычислительной мощностью, что для большинства приложений скорость разработки важнее скорости выполнения, а программы на Python, как правило, пишутся куда быстрее, чем на других языках. Кроме того, Python легко расширяется модулями, написанными на C или C++; они могут использоваться для выполнения ресурсоемких частей программы.

Основные разработчики Python также упорно трудятся над созданием его новых версий, которые являются более эффективными, загружаются и работают быстрее, а также лучше используют преимущества многоядерных процессоров. Эти усилия уже привели к значительному повышению быстродействия, и работа продолжится в будущем, так что, если вы собираетесь создать приложение, для которого критична производительность, вам стоит хорошенько подумать, подойдет ли Python для этой задачи (однако не стоит сразу же списывать его со счетов).

1.3.2. Python не применяет принудительное определение типов переменных во время компиляции

В отличие от некоторых других языков, переменные в Python не служат контейнерами для своих значений, скорее, они похожи на ссылки для указания на объекты различных типов: целых чисел, строк, экземпляров класса и т. д. Это

означает, что, хотя сами объекты обладают типами, ссылающиеся на них переменные не привязаны к этим конкретным типам. Можно (хотя это и не всегда желательно) использовать переменную x для ссылки на объект строкового типа в одной строке кода и на объект целочисленного типа в другой строке кода (примечание: вывод программы выделен **жирным** шрифтом):

```
x = "2"
x

'2'  ◄—— Вывод переменной x — строковое значение "2"

x = int(x)
x

2  ◄—— Теперь вывод переменной x — целочисленное значение 2
```

Поскольку в Python типы связаны с объектами, а не с переменными, интерпретатор не поможет вам выявить несоответствие типов объектов. Если вы включили в программу переменную `count` для хранения указания на целочисленный тип объекта, Python не станет возражать, если вы присвоите этому объекту строковый тип `"2"`. Программисты традиционной школы считают это недостатком, поскольку тем самым вы лишаетесь дополнительной проверки программного кода.

Приняв во внимание эти опасения, Python дополнил синтаксис и средства, позволяющие программистам указывать требуемый тип объекта, на который ссылается переменная, а также параметры функции, возвращаемые значения и т. п. С помощью этих *аннотаций типов*, как их называют, различные инструменты могут выявлять любые несоответствия в типах объектов еще до начала выполнения программы. В небольших программах ошибки несоответствия типов, как правило, несложно обнаружить и исправить даже без аннотаций типов, и в любом случае средства тестирования Python позволяют избегать таких ошибок. Многие программисты Python считают, что гибкость динамической типизации с лихвой перевешивает любые преимущества, которые может дать обязательная типизация самих переменных.

1.3.3. Слабая поддержка мобильных устройств

За последнее десятилетие появилось великое множество всевозможных мобильных устройств: смартфоны, планшеты, фаблеты, хромбуки и др. Новые мобильные устройства повсюду, и они работают под управлением различных операционных систем. Python не принадлежит к числу сильных игроков в этой области, однако различные проекты работают над данной проблемой, разрабатывая наборы инструментов и фреймворки, позволяющие писать приложения для платформ iOS и Android. Ситуация улучшается, но на момент написания книги писать и распространять коммерческие мобильные приложения при помощи Python все еще довольно хлопотно.

1.3.4. Слабая поддержка многоядерных систем

В наши дни многоядерные процессоры встречаются повсюду, и во многих случаях они обеспечивают значительный прирост производительности. Однако стандартная реализация Python не рассчитана на использование нескольких ядер из-за механизма *глобальной блокировки интерпретатора* (Global Interpreter Lock, GIL). Как уже упоминалось в связи с быстродействием, команда разработчиков Python в настоящее время работает над тем, чтобы Python более слаженно и эффективно справлялся с многоядерными процессорами, и развитие многоядерной поддержки Python будет продолжаться в течение последующих нескольких лет.

Итоги

- Python — современный высокоуровневый язык программирования с динамической типизацией и простым, логически ясным, последовательным синтаксисом и семантикой.
- Python — кросс-платформенный язык с высокой модульностью, хорошо подходящий как для быстрой разработки, так и для крупномасштабного программирования.
- Он достаточно быстрый, и может легко расширяться модулями C или C++ для повышения скорости.
- Python имеет встроенные расширенные возможности, такие как персистентное хранение объектов, продвинутые хеш-таблицы, расширяемый синтаксис классов и универсальные функции сравнения.
- Python включает в себя широкий набор модулей, предназначенных для обработки чисел, изображений, создания пользовательских интерфейсов и веб-скриптинга.
- Он поддерживается динамично развивающимся сообществом.