Марте, с любовью

Оглавление

Предисловие от издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте *www.dmkpress.com*, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу *dmkpress@gmail.com*; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу *http://dmkpress.com/authors/publish_book/* или напишите в издательство по адресу *dmkpress@gmail.com*.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в основном тексте или программном коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу *dmkpress@gmail.com*, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу *dmkpress@gmail.com*.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Об авторе

Лусиану Рамальо был веб-разработчиком до выхода компании Netscape на IPO в 1995 году, а в 1998 году перешел с Perl на Java, а затем на Python. В 2015 году пришел в компанию Thoughtworks, где работает главным консультантом в отделении в Сан-Паулу. Он выступал с основными докладами, презентациями и пособиями на различных мероприятиях, связанных с Python, в обеих Америках, Европе и Азии. Выступал также на конференциях по Go и Elixir по вопросам проектирования языков. Рамальо – член фонда Python Software Foundation и сооснователь клуба Garoa Hacker Clube, первого места для общения хакеров в Бразилии.

Колофон

На обложке изображена намакская песчаная ящерица (*Pedioplanis namaquensis*), встречающаяся в засушливых саваннах и полупустынях Намибии.

Внешние признаки: туловище черное с четырьмя белыми полосками на спине, лапы коричневые с белыми пятнышками, брюшко белое, длинный розовато-коричневый хвост. Одна из самых быстрых ящериц, активна в течение дня, питается мелкими насекомыми. Обитает на бедных растительностью песчано-каменистых равнинах. Самка откладывает от трех до пяти яиц в ноябре. Остаток зимы ящерицы спят в норах, которые роют в корнях кустов.

В настоящее время охранный статус намакской песчаной ящерицы – «пониженная уязвимость». Многие животные, изображенные на обложках книг O'Reilly, находятся под угрозой вымирания; все они важны для мира.

Предисловие

> План такой: если кто-то пользуется средством, которое вы не понимаете, просто пристрелите его. Это проще, чем учить что-то новое, и очень скоро в мире останутся только кодировщики, которые используют только всем понятное крохотное подмножество Python 0.9.6 <смешок>.
>
> – Тим Питерс, легендарный разработчик ядра и автор сборника поучений «The Zen of Python»[1]

«Python – простой для изучения и мощный язык программирования». Это первые слова в официальном «Пособии по Python» (*https://docs.python.org/3/tutorial/*). И это правда, но не вся правда: поскольку язык так просто выучить и начать применять на деле, многие практикующие программисты используют лишь малую часть его обширных возможностей.

Опытный программист может написать полезный код на Python уже через несколько часов изучения. Но вот проходят недели, месяцы – и многие разработчики так и продолжают писать на Python код, в котором отчетливо видно влияние языков, которые они учили раньше. И даже если Python – ваш первый язык, все равно авторы академических и вводных учебников зачастую излагают его, тщательно избегая особенностей, характерных только для этого языка.

Будучи преподавателем, который знакомит с Python программистов, знающих другие языки, я нередко сталкиваюсь еще с одной проблемой, которую пытаюсь решить в этой книге: нас интересует только то, о чем мы уже знаем. Любой программист, знакомый с каким-то другим языком, догадывается, что Python поддерживает регулярные выражения, и начинает смотреть, что про них написано в документации. Но если вы никогда раньше не слыхали о распаковке кортежей или о дескрипторах, то, скорее всего, и искать сведения о них не станете, а в результате не будете использовать эти средства лишь потому, что они специфичны для Python.

Эта книга не является полным справочным руководством по Python. Упор в ней сделан на языковые средства, которые либо уникальны для Python, либо отсутствуют во многих других популярных языках. Кроме того, в книге рассматривается в основном ядро языка и немногие библиотеки. Я редко упоминаю о пакетах, не включенных в стандартную библиотеку, хотя нынче количество пакетов для Python уже перевалило за 60 000 и многие из них исключительно полезны.

На кого рассчитана эта книга

Эта книга написана для практикующих программистов на Python, которые хотят усовершенствоваться в Python 3. Я тестировал примеры на Python 3.10,

[1] Сообщение в группе Usenet comp.lang.python от 23 декабря 2002: «Acrimony in c.l.p.» (*https://mail.python.org/pipermail/python-list/2002-December/147293.html*).

а большую их часть также на Python 3.9 и 3.8. Если какой-то пример требует версии 3.10, то это явно оговаривается.

Если вы не уверены в том, достаточно ли хорошо знаете Python, чтобы читать эту книгу, загляните в оглавление официального «Пособия по Python» (*https://docs.python.org/3/tutorial/*). Темы, рассмотренные в пособии, в этой книге не затрагиваются, за исключением некоторых новых средств.

На кого эта книга не рассчитана

Если вы только начинаете изучать Python, эта книга покажется вам сложноватой. Более того, если вы откроете ее на слишком раннем этапе путешествия в мир Python, то может сложиться впечатление, будто в каждом Python-скрипте следует использовать специальные методы и приемы метапрограммирования. Преждевременное абстрагирование ничем не лучше преждевременной оптимизации.

Пять книг в одной

Я рекомендую всем прочитать главу 1 «Модель данных в языке Python». Читатели этой книги в большинстве своем после ознакомления с главой 1, скорее всего, смогут легко перейти к любой части, но я зачастую предполагаю, что главы каждой части читаются по порядку. Части I–V можно рассматривать как отдельные книги внутри книги.

Я старался сначала рассказывать о том, что уже есть, а лишь затем о том, как создавать что-то свое. Например, в главе 2 части II рассматриваются готовые типы последовательностей, в том числе не слишком хорошо известные, например `collections.deque`. О создании пользовательских последовательностей речь пойдет только в части III, где мы также узнаем об использовании абстрактных базовых классов (abstract base classes – ABC) из модуля `collections.abc`. Создание собственного ABC обсуждается еще позже, поскольку я считаю, что сначала нужно освоиться с использованием ABC, а уж потом писать свои.

У такого подхода несколько достоинств. Прежде всего, зная, что есть в вашем распоряжении, вы не станете заново изобретать велосипед. Мы пользуемся готовыми классами коллекций чаще, чем реализуем собственные, и можем уделить больше внимания нетривиальным способам работы с имеющимися средствами, отложив на потом разговор о разработке новых. И мы скорее унаследуем существующему абстрактному базовому классу, чем будем создавать новый с нуля. Наконец, я полагаю, что понять абстракцию проще после того, как видел ее в действии.

Недостаток же такой стратегии в том, что главы изобилуют ссылками на более поздние материалы. Надеюсь, что теперь, когда вы узнали, почему я избрал такой путь, вам будет проще смириться с этим.

Как организована эта книга

Ниже описаны основные темы, рассматриваемые в каждой части книги.

Часть I «Структуры данных»

В главе I, посвященной модели данных в Python, объясняется ключевая роль специальных методов (например, `__repr__`) для обеспечения единообразного поведения объектов любого типа. Специальные методы более подробно обсуждаются на протяжении всей книги. В остальных главах этой части рассматривается использование типов коллекций: последовательностей, отображений и множеств, а также различие между типами `str` и `bytes` – то, что радостно приветствовали пользователи Python 3 и чего отчаянно не хватает пользователям Python 2, еще не модернизировавшим свой код. Также рассматриваются высокоуровневые построители классов, имеющиеся в стандартной библиотеке: фабрики именованных кортежей и декоратор `@dataclass`. Сопоставление с образцом – новая возможность, появившаяся в Python 3.10, – рассматривается в разделах глав 2, 3 и 5, где обсуждаются паттерны последовательностей, отображений и классов. Последняя глава части I посвящена жизненному циклу объектов: ссылкам, изменяемости и сборке мусора.

Часть II «Функции как объекты»

Здесь речь пойдет о функциях как полноправных объектах языка: что под этим понимается, как это отражается на некоторых популярных паттернах проектирования и как реализовать декораторы функций с помощью замыканий. Рассматриваются также следующие вопросы: общая идея вызываемых объектов, атрибуты функций, интроспекция, аннотации параметров и появившееся в Python 3 объявление `nonlocal`. Глава 8 содержит введение в новую важную тему – аннотации типов в сигнатурах функций.

Часть III «Классы и протоколы»

Теперь наше внимание перемещается на создание классов «вручную» – в отличие от использования построителей классов, рассмотренных в главе 5. Как и в любом объектно-ориентированном (ОО) языке, в Python имеется свой набор средств; какие-то из них, возможно, присутствовали в языке, с которого вы и я начинали изучение программирования на основе классов, а какие-то – нет. В главах из этой части объясняется, как создать свою коллекцию, абстрактный базовый класс (ABC) и протокол, как работать со множественным наследованием и как реализовать перегрузку операторов (если это имеет смысл). В главе 15 мы продолжим обсуждать аннотации типов.

Часть IV «Поток управления»

Эта часть посвящена языковым конструкциям и библиотекам, выходящим за рамки последовательного потока управления с его условными выражениями, циклами и подпрограммами. Сначала мы рассматриваем генераторы, затем – контекстные менеджеры и сопрограммы, в том числе трудную для понимания, но исключительно полезную новую конструкцию `yield from`. В главу 18 включен важный пример использования сопоставления с образцом в простом, но функциональном интерпретаторе языка. Глава 19 «Модели конкурентности в Python» новая, она посвящена обзору различных

видов конкурентной и параллельной обработки в Python, их ограничений и вопросу о том, как архитектура программы позволяет использовать Python в приложениях масштаба веба. Я переписал главу об *асинхронном программировании*, стремясь уделить больше внимания базовым средствам языка – `await`, `async dev`, `async for` и `async with` – и показать, как они используются совместно с библиотекой *asyncio* и другими каркасами.

Часть V «Метапрограммирование»

Эта часть начинается с обзора способов построения классов с динамически создаваемыми атрибутами для обработки слабоструктурированных данных, например в формате JSON. Затем мы рассматриваем знакомый механизм свойств, после чего переходим к низкоуровневым деталям доступа к атрибутам объекта с помощью дескрипторов. Объясняется связь между функциями, методами и дескрипторами. На примере приведенной здесь пошаговой реализации библиотеки контроля полей мы вскрываем тонкие нюансы, которые делают необходимым применение рассмотренных в этой главе продвинутых инструментов: декораторов классов и метаклассов.

Практикум

Часто для исследования языка и библиотек мы будем пользоваться интерактивной оболочкой Python. Я считаю важным всячески подчеркивать удобство этого средства для обучения. Особенно это относится к читателям, привыкшим к статическим компилируемым языкам, в которых нет цикла чтения-вычисления-печати (read-eval-print#loop – REPL).

Один из стандартных пакетов тестирования для Python, `doctest` (*https://docs.python.org/3/library/doctest.html*), работает следующим образом: имитирует сеансы оболочки и проверяет, что результат вычисления выражения совпадает с заданным. Я использовал `doctest` для проверки большей части приведенного в книге кода, включая листинги сеансов оболочки. Для чтения книги ни применять, ни даже знать о пакете `doctest` не обязательно: основная характеристика doctest-скриптов (или просто тестов) состоит в том, что они выглядят как копии интерактивных сеансов оболочки Python, поэтому вы можете сами выполнить весь демонстрационный код.

Иногда я буду объяснять, чего мы хотим добиться, демонстрируя тест раньше кода, который заставляет его выполниться успешно. Если сначала отчетливо представить себе, что необходимо сделать, а только потом задумываться о том, как это сделать, то структура кода заметно улучшится. Написание тестов раньше кода – основа методологии разработки через тестирование (TDD); мне кажется, что и для преподавания это полезно. Если вы незнакомы с `doctest`, загляните в документацию (*https://docs.python.org/3/library/doctest.html*) и в репозиторий исходного кода к этой книге (*https://github.com/fluentpython/example-code-2e*).

Я также написал автономные тесты для нескольких крупных примеров, воспользовавшись модулем `pytest`, который, как мне кажется, проще и имеет больше возможностей, чем модуль `unittest` из стандартной библиотеки. Вы увидите, что для проверки правильности большей части кода в книге до-