Краткое содержание

Оглавление

Часть I. Перед началом работы

Часть II. Ремесло Python

Часть III. Качество, а не количество

Часть IV. Жажда скорости

Часть V. Техническая архитектура

Благодарю любимую жену Оливию за ее любовь,
вдохновение и бесконечное терпение,
моих верных друзей Петра, Дэниела и Павла
за их поддержку,
мою мать, открывшую предо мной удивительный
мир программирования.

Михал Яворски

О создателях книги

Об авторах

Михал Яворски — программист на Python с десятилетним опытом. Занимал разные должности в различных компаниях: от обычного фулстек-разработчика, затем архитектора программного обеспечения и, наконец, до вице-президента по разработке в динамично развивающейся стартап-компании. В настоящее время Михал — старший бэкенд-инженер в Showpad. Имеет большой опыт в разработке высокопроизводительных распределенных сервисов. Кроме того, является активным участником многих проектов Python с открытым исходным кодом.

Тарек Зиаде — Python-разработчик. Живет в сельской местности недалеко от города Дижон во Франции. Работает в Mozilla, в команде, отвечающей за сервисы. Тарек основал французскую группу пользователей Python (называется Afpy) и написал несколько книг о Python на французском и английском языках. В свободное от хакинга и тусовок время занимается любимыми хобби: бегом или игрой на трубе.

Вы можете посетить его личный блог (Fetchez le Python) и подписаться на него в Twitter (tarek_ziade).

О научном редакторе

Коди Джексон — кандидат наук, основатель компании Socius Consulting, работающей в сфере IT и консалтинга по управлению бизнесом в Сан-Антонио, а также соучредитель Top Men Technologies. В настоящее время работает в CACI International ведущим инженером по моделированию ICS/SCADA. В IT-индустрии с 1994 года, еще со времен службы в ВМФ в качестве ядерного химика и радиотехника. До CACI он работал в университете в ECPI в должности ассистента профессора по компьютерным информационным системам. Выучился программированию на Python самостоятельно, написал книги *Learning to Program Using Python* и *Secret Recipes of the Python Ninja*.

Предисловие

Python — динамический язык программирования, применимый в широком спектре задач благодаря своей простой, но мощной сути. Писать на Python легко, но сделать код удобочитаемым, универсальным и простым в сопровождении — сложно. В третьем издании данной книги вы ознакомитесь с практическими рекомендациями, полезными инструментами и стандартами, используемыми профессиональными разработчиками на Python, так что сумеете преодолеть данную проблему.

Мы начнем эту книгу с новых возможностей, добавленных в Python 3.7. Изучим синтаксис Python и рассмотрим, как применять самые современные концепции и механизмы объектно-ориентированного программирования. Помимо этого, исследуем различные подходы к реализации метапрограммирования. Данная книга расскажет о присваивании имен при написании пакетов, создании исполняемых файлов, а также о применении мощных инструментов, таких как `buildout` и `virtualenv`, для развертывания кода на удаленных серверах. Вы узнаете, как создавать полезные расширения Python на языках C, C++, Cython и Pyrex. Кроме того, чтобы писать чистый код, вам будет полезно изучить инструменты управления кодом, написания ясной документации и разработки через тестирование.

Изучив эту книгу, вы станете экспертом в написании эффективного и удобного в сопровождении кода на Python.

Для кого эта книга

Книга написана для разработчиков на Python, желающих продвинуться в освоении этого языка. Под разработчиками мы имеем в виду в основном программистов, которые зарабатывают на жизнь программированием на Python. Дело в том, что книга сосредоточена на средствах и методах, наиболее важных для создания производительного, надежного и удобного в сопровождении программного обеспечения на Python.

Это не значит, что в книге нет ничего интересного для любителей. Она отлично подойдет для тех, кто хочет выйти на новый уровень в изучении Python. Базовых

навыков языка будет достаточно, чтобы понять изложенный материал, хотя менее опытным программистам придется приложить некоторые усилия. Книга также будет хорошим введением в Python 3.7 для тех, кто слегка отстал от жизни и пользуется версией Python 2.7 или еще более ранней.

Наибольшую пользу данная книга принесет веб- и бэкенд-разработчикам, поскольку в ней представлены две темы, особенно важные именно для этих специалистов: надежное развертывание кода и параллелизм.

Что мы рассмотрим

В главе 1 описано текущее состояние Python и его сообщества. Мы увидим, как меняется язык, из-за чего это происходит и почему данные факты очень важны для тех, кто хочет называть себя профессионалом в Python. Мы также рассмотрим наиболее известные и канонические способы работы с кодом Python, а именно популярные инструменты обеспечения производительности и правила, которые сегодня фактически являются стандартами.

В главе 2 представлены современные способы создания повторяемых и последовательных сред разработки для программистов на Python. Мы сосредоточимся на двух популярных инструментах для изоляции среды: средах типа `virtualenv` и контейнерах Docker.

В главе 3 даны практические рекомендации по написанию кода на Python (идиомы языка), а также краткое описание отдельных элементов синтаксиса Python, которые могут оказаться новыми для программистов, более привыкших к старым версиям Python. Кроме того, мы изложим пару полезных идей о внутренних реализациях типа CPython и их вычислительной сложности в качестве обоснования для рассмотренных идиом.

В главе 4 рассмотрены более сложные концепции и механизмы объектно-ориентированного программирования, доступные в Python.

В главе 5 представлен обзор общих подходов к метапрограммированию для программистов на Python.

В главе 6 приведено руководство по наиболее общепринятому стилю написания кода на Python (PEP 8), а также указано, когда и почему разработчики должны соблюдать его. Вдобавок мы рассмотрим некоторые общие рекомендации по назначению имен.

В главе 7 описаны особенности создания пакетов на Python и даны рекомендации по созданию пакетов, распространяемых в виде открытого исходного кода в **каталоге пакетов Python** (Python Package Index, PyPI). Мы также рассмотрим тему, которую часто игнорируют, — исполняемые файлы.

В главе 8 представлены некоторые облегченные инструменты для развертывания кода Python на удаленных серверах. Развертывание — это одна из областей,

где Python предстает во всей красе в реализации бэкенда для веб-сервисов и приложений.

В главе 9 объясняется, почему иногда удобно добавлять в код расширения на C и C++, и показывается, что при наличии подходящих инструментов сделать это будет проще, чем кажется.

В главе 10 рассказывается, как правильно управлять кодовой базой и почему следует использовать систему управления версиями. Мы опробуем на деле возможности такой системы (а именно, Git) в осуществлении непрерывных процессов, таких как непрерывная интеграция и непрерывная доставка.

В главе 11 приводятся общие правила написания технической документации, применимые к программному обеспечению, написанному на любом языке, а также различные инструменты, которые будут особенно полезны при создании документации для вашего кода на Python.

В главе 12 представлены плюсы подхода «разработка через тестирование» и подробно рассказывается о том, как использовать популярные инструменты Python для тестирования.

В главе 13 обсуждаются базовые правила оптимизации, которые должен знать каждый разработчик. Мы также научимся выявлять узкие места в производительности приложений и использовать общие инструменты профилирования.

В главе 14 показывается, как применить эти знания так, чтобы ваше приложение действительно работало быстрее или эффективнее с точки зрения используемых ресурсов.

В главе 15 объясняется, как реализовать параллелизм на Python с помощью различных подходов и готовых библиотек.

В главе 16 рассказывается, что такое событийно-ориентированное и сигнальное программирование и как оно связано с асинхронным и различными моделями параллелизма. Мы представим разные подходы к событийному программированию, доступные программистам на Python, а также полезные библиотеки, позволяющие применять эти шаблоны.

В главе 17 описаны несколько полезных паттернов проектирования и примеры их реализации на Python.

Приложение содержит краткое руководство по использованию языка разметки reStructuredText.

Как получить максимум от этой книги

Данная книга написана для программистов, работающих в любой операционной системе, где установлен Python 3.

Издание не подходит для начинающих, поэтому мы предполагаем, что вы установили Python в своей среде или вы знаете, как это сделать. Однако в книге учитывается

тот факт, что не все могут знать о последних функциях Python или официально рекомендованных инструментах. Именно поэтому в первой главе приведен обзор наиболее часто используемых утилит (например, виртуальных сред и pip), которые в настоящее время профессиональные разработчики на Python считают стандартными инструментами.

Скачивание файлов с примерами кода

Вы можете скачать файлы примеров кода для этой книги на сайте github.com/PacktPublishing/Expert-Python-Programming-Third-Edition.

Чтобы скачать файлы кода, выполните следующие действия.

1. Перейдите по указанной ссылке на сайт github.com.
2. Нажмите кнопку Clone or Download.
3. Щелкните кнопкой мыши на ссылке Download ZIP.
4. Скачайте архив с файлами примеров.

После скачивания файла распакуйте его с помощью последней версии одной из следующих программ:

- WinRAR/7-Zip для Windows;
- Zipeg/iZip/UnRarX для Mac;
- 7-Zip/PeaZip для Linux.

Скачивание цветных изображений

Мы также выложили оригинальный файл PDF, в котором приведены цветные изображения снимков экрана/схем, используемых в этой книге. Вы можете скачать его по ссылке www.packtpub.com/sites/default/files/downloads/9781789808896_ColorImages.pdf.

Условные обозначения

В этой книге используется ряд текстовых и символьных обозначений.

`Код в тексте`: такой формат обозначает кодовые слова в тексте, имена таблиц базы данных, имена папок и файлов, расширения файлов, пути к файлам, URL-адреса, пользовательский ввод и инструменты Twitter. Например: «Любая попытка запустить код, в котором есть такие проблемы, заставит интерпретатор завершить работу, выбросив исключение `SyntaxError`».

Блок кода выглядит следующим образом:

```
print("hello world")
print "goodbye python2"
```

Любой ввод или вывод из командной строки записывается так:

```
$ Python3 script.py
```

Новые термины и важные слова выделены *курсивом*.

Так помечаются предупреждения и важные примечания.

А так — советы и секреты.

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.

Часть I
Перед началом работы

Эта часть призвана помочь пользователю подготовиться к современным реалиям разработки на Python. Мы рассмотрим, как язык изменился за последние несколько лет и какими инструментами разработки пользуются современные программисты на Python.

1 Текущее состояние Python

Python удивителен.

На протяжении долгого времени одним из самых важных достоинств Python была его совместимость. Независимо от того, какую операционную систему используете вы или ваши клиенты, если у нее есть интерпретатор Python, то ваше написанное на Python ПО будет в ней работать. И что важнее всего — работать так, как нужно. Однако сейчас этим никого не удивишь. Современные языки, например Ruby и Java, предоставляют аналогичные возможности для взаимодействия. Но в наше время совместимость не самое важное качество языка программирования. С появлением облачных вычислений, веб-приложений и надежного программного обеспечения для создания виртуальных окружений вопросы совместимости и независимости от операционной системы отошли на второй план. Однако попрежнему важны инструменты, позволяющие программистам эффективно писать надежное и удобное в сопровождении ПО. К счастью, Python относится к тем языкам, благодаря которым программисты могут работать наиболее эффективно, и для развития компаний это лучший выбор.

Python так долго не теряет актуальности благодаря тому, что постоянно развивается. Эта книга ориентирована на последнюю версию Python 3.7, и все примеры кода написаны именно в ней, если не сказано иное. Поскольку Python имеет очень длинную историю и еще есть программисты, пишущие на Python 2, данная книга начинается с обзора текущего *статус-кво* Python 3. В этой главе вы узнаете, как и почему Python изменился и как писать программное обеспечение, совместимое и со старыми, и с последними версиями Python.

В этой главе:

- где мы находимся и куда движемся;
- почему и как изменился язык Python;
- как не отставать от изменений в документации PEP;
- принятие Python 3 на момент написания этой книги;
- основные различия между Python 3 и Python 2;
- не только CPython;
- полезные ресурсы.

Технические требования

Для этой главы скачать последнюю версию Python можно по ссылке www.python.org/downloads/.

Альтернативные реализации интерпретатора Python можно найти на следующих сайтах:

- Stackless Python: github.com/stackless-dev/stackless;
- PyPy: pypy.org;
- Jython: www.jython.org;
- IronPython: ironpython.net;
- MicroPython: micropython.org.

Файлы с примерами кода для этой главы можно найти по ссылке github.com/PacktPublishing/Expert-Python-Programming-Third-Edition/tree/master/chapter1.

Где мы находимся и куда движемся

История Python началась где-то в конце 1980-х годов, но релиз версии 1.0 состоялся в 1994 году. То есть это не молодой язык. Мы бы могли пройтись по всей хронологии версий Python, однако на самом деле нас интересует только одна дата: 3 декабря 2008 года — выход Python 3.0.

На момент написания этой книги прошло почти десять лет с появления первого релиза Python 3. Кроме того, прошло семь лет после выпуска PEP 404 — официального документа, в котором был *отменен выпуск* Python 2.8 и официально закрыта вся серия 2.x. С тех пор прошло много времени, однако сообщество Python все еще делится на два лагеря: несмотря на то что язык развивается очень быстро, есть большая группа пользователей, которые не хотят идти с ним в ногу.

Почему и как изменился язык Python

Ответ прост — Python изменяется, поскольку в этом есть необходимость. Конкуренты не спят. Каждые несколько месяцев из ниоткуда появляется новый язык, претендующий на решение всех проблем своих предшественников. Разработчики быстро утрачивают интерес к большинству подобных проектов, и их популярность часто вызвана исключительно хайпом.

За этим кроется более серьезная проблема. Люди берутся за разработку новых языков, поскольку считают, что существующие не решают их проблем. Было бы глупо отрицать необходимость в новых решениях. Кроме того, все более широкое использование Python показывает: язык можно и нужно улучшать.

Множество улучшений в Python обусловлены потребностями конкретных сфер, в которых он применяется. Наиболее значимая из них — веб-разработка. Так, постоянно растущий спрос на скорость и производительность в этой области привел к тому, что работа с параллелизмом в Python значительно упростилась.

Некоторые изменения вызваны попросту солидным возрастом и зрелостью проекта Python. На протяжении многих лет он обрастал множеством неорганизованных и избыточных модулей стандартных библиотек и даже плохими проектными решениями. То есть выпуск Python 3 был призван подчистить и освежить язык. К сожалению, время показало: данный план имел и неприятные последствия. В течение долгого времени Python 3 использовался многими разработчиками несерьезно. Будем надеяться, это изменится.

Как не отставать от изменений в документации PEP

Сообщество Python придумало устоявшийся способ реагирования на изменения. Хотя рискованные идеи языка Python в основном обсуждаются в рассылках (python-ideas@python.org), по-настоящему серьезные изменения сопровождаются выходом документа под названием *Python Enhancement Proposal (PEP)*.

Это формализованный документ, в котором подробно описывается предложение об изменении Python. Он также является отправной точкой для обсуждения в сообществе. Вся цель, формат и рабочий процесс вокруг данных документов также стандартизированы в документе PEP 1 (www.python.org/dev/peps/pep-0001).

PEP-документация очень важна для Python и, в зависимости от темы, выполняет разные функции:

- *информирования* — приводит информацию, необходимую разработчикам ядра Python, и графики выпуска версий Python;
- *стандартизации* — содержит указания по стилю кода, документации или другие руководящие принципы;
- *проектирования* — описывает предлагаемые функции.

Список всех предлагаемых PEP приведен в *постоянно обновляемом* документе PEP 0 (www.python.org/dev/peps/). Найти их легко, а ссылку на них нетрудно сформировать самостоятельно, поэтому в книге мы будем называть их лишь по номерам.

Документ PEP 0 — важный источник информации для тех, кому интересно, в каком направления движется язык Python, но некогда отслеживать каждое обсуждение в рассылках Python. В PEP 0 показано, какие документы уже были приняты, но еще не реализованы, а какие находятся на рассмотрении.