

Краткое содержание

О создателях книги	14
Предисловие	15
От издательства	20

Часть I. Перед началом работы

Глава 1. Текущее состояние Python	22
Глава 2. Современные среды разработки на Python.....	39

Часть II. Ремесло Python

Глава 3. Современные элементы синтаксиса — ниже уровня класса.....	66
Глава 4. Современные элементы синтаксиса — выше уровня класса	123
Глава 5. Элементы метапрограммирования	152
Глава 6. Как выбирать имена	173
Глава 7. Создаем пакеты	195
Глава 8. Развертывание кода	231
Глава 9. Расширения Python на других языках	268

Часть III. Качество, а не количество

Глава 10. Управление кодом	308
Глава 11. Документирование проекта	339
Глава 12. Разработка на основе тестирования	366

Часть IV. Жажда скорости

Глава 13. Оптимизация — принципы и методы профилирования	404
Глава 14. Эффективные методы оптимизации.....	434
Глава 15. Многозадачность	461

Часть V. Техническая архитектура

Глава 16. Событийно-ориентированное и сигнальное программирование	504
Глава 17. Полезные паттерны проектирования	523
Приложение. reStructuredText Primer	552

Оглавление

О создателях книги	14
Об авторах.....	14
О научном редакторе.....	14
Предисловие	15
Для кого эта книга.....	15
Что мы рассмотрим	16
Как получить максимум от этой книги.....	17
Скачивание файлов с примерами кода.....	18
Скачивание цветных изображений.....	18
Условные обозначения	18
От издательства	20

Часть I. Перед началом работы

Глава 1. Текущее состояние Python	22
Технические требования.....	23
Где мы находимся и куда движемся.....	23
Почему и как изменился язык Python	23
Как не отставать от изменений в документации PEP	24
Внедрение Python 3 на момент написания этой книги.....	25
Основные различия между Python 3 и Python 2.....	26
Почему это должно нас волновать	26
Основные синтаксические различия и распространенные ошибки.....	27
Популярные инструменты и методы поддержания кросс-версионной совместимости	29
Не только CPython	33
Почему это должно нас волновать	33
Stackless Python	33
Jython	34
IronPython.....	35
PyPy.....	36
MicroPython	36
Полезные ресурсы	37
Резюме	38
Глава 2. Современные среды разработки на Python.....	39
Технические требования.....	40
Установка дополнительных пакетов Python с использованием pip.....	40
Изоляция сред выполнения.....	42
venv — виртуальное окружение Python	43
Изоляция среды на уровне системы	46
Виртуальные среды разработки, использующие Vagrant.....	47
Виртуальные среды, использующие Docker	49

Популярные инструменты повышения производительности	59
Пользовательские оболочки Python — ipython, bpython, ptpython и т. д.	60
Включение оболочек в собственные скрипты и программы	62
Интерактивные отладчики	63
Резюме	64

Часть II. Ремесло Python

Глава 3. Современные элементы синтаксиса — ниже уровня класса	66
Технические требования	67
Встроенные типы языка Python	67
Строки и байты	67
Контейнеры	73
Дополнительные типы данных и контейнеры	85
Специализированные контейнеры данных из модуля collections	85
Символическое перечисление с модулем enum	86
Расширенный синтаксис	88
Итераторы	88
Генераторы и операторы yield	91
Декораторы	94
Менеджеры контекста и оператор with	105
Функционально-стилевые особенности Python	109
Что такое функциональное программирование	110
Лямбда-функции	111
map(), filter() и reduce()	112
Частичные объекты и функция partial()	115
Выражения генераторов	116
Аннотации функций и переменных	117
Общий синтаксис	117
Возможные способы применения	118
Статическая проверка типа с помощью туру	118
Иные элементы синтаксиса, о которых вы, возможно, не знаете	119
Оператор for.. else...	119
Именованные аргументы	120
Резюме	122
Глава 4. Современные элементы синтаксиса — выше уровня класса	123
Технические требования	124
Протоколы в языке Python — методы и атрибуты с двойным подчеркиванием	124
Сокращение шаблонного кода с помощью классов данных	126
Создание подклассов встроенных типов	128
ПРМ и доступ к методам из суперклассов	131
Классы старого стиля и суперклассы в Python 2	133
Понимание ПРМ в Python	134
Ловушки суперкласса	138
Практические рекомендации	141
Паттерны доступа к расширенным атрибутам	141
Дескрипторы	142
Свойства	147
Слоты	150
Резюме	151

Глава 5. Элементы метапрограммирования	152
Технические требования	152
Что такое метапрограммирование	153
Декораторы как средство метапрограммирования	153
Декораторы класса	154
Использование <code>__new__()</code> для переопределения процесса создания экземпляра	156
Метаклассы	158
Генерация кода	165
Резюме	172
Глава 6. Как выбирать имена	173
Технические требования	174
PEP 8 и практические рекомендации по именованию	174
Почему и когда надо соблюдать PEP 8	174
За пределами PEP 8 — правила стиля внутри команды	175
Стили именования	175
Переменные	176
Руководство по именованию	184
Использование префиксов <code>is/has</code> в булевых элементах	184
Использование множественного числа в именах коллекций	185
Использование явных имен для словарей	185
Избегайте встроенных и избыточных имен	185
Избегайте уже существующих имен	186
Практические рекомендации по работе с аргументами	187
Сборка аргументов по итеративному принципу	187
Доверие к аргументам и тестам	188
Осторожность при работе с магическими аргументами <code>*args</code> и <code>**kwargs</code>	188
Имена классов	190
Имена модулей и пакетов	191
Полезные инструменты	191
PyLint	192
<code>pycodestyle</code> и <code>flake8</code>	193
Резюме	194
Глава 7. Создаем пакеты	195
Технические требования	195
Создание пакета	196
Странности в нынешних инструментах создания пакетов в Python	196
Конфигурация проекта	198
Пользовательская команда <code>setup</code>	207
Работа с пакетами в процессе разработки	208
Пакеты пространства имен	209
Почему это полезно	210
Загрузка пакета	214
PyPI — каталог пакетов Python	214
Пакеты с исходным кодом и пакеты сборок	216
Исполняемые файлы	220
Когда бывают полезны исполняемые файлы	221
Популярные инструменты	221
Безопасность кода Python в исполняемых пакетах	228
Резюме	230

Глава 8. Развертывание кода	231
Технические требования.....	232
Двенадцатифакторное приложение	232
Различные подходы к автоматизации развертывания	234
Использование Fabric для автоматизации развертывания.....	235
Ваш собственный каталог пакетов или зеркало каталогов	239
Зеркала PyPI	240
Объединение дополнительных ресурсов с пакетом Python	241
Общие соглашения и практики	249
Иерархия файловой системы	249
Изоляция	250
Использование инструментов мониторинга процессов	250
Запуск кода приложения в пространстве пользователя.....	252
Использование обратного HTTP-прокси.....	253
Корректная перезагрузка процессов	254
Контрольно-проверочный код и мониторинг	256
Ошибки журнала — Sentry/Raven	256
Метрики систем мониторинга и приложений	260
Работа с журнальными приложениями.....	262
Резюме	267
Глава 9. Расширения Python на других языках	268
Технические требования.....	269
Различия между языками C и C++	269
Необходимость в использовании расширений.....	272
Повышение производительности критических фрагментов кода	272
Интеграция существующего кода, написанного на разных языках.....	273
Интеграция сторонних динамических библиотек.....	274
Создание пользовательских типов данных	274
Написание расширений.....	275
Расширения на чистом языке C	276
Написание расширений на Cython	291
Проблемы с использованием расширений	295
Дополнительная сложность.....	296
Отладка	297
Взаимодействие с динамическими библиотеками без расширений	297
Модуль ctypes	298
CFFI	304
Резюме	306

Часть III. Качество, а не количество

Глава 10. Управление кодом	308
Технические требования.....	308
Работа с системой управления версиями	308
Централизованные системы	309
Распределенные системы.....	312
Распределенные стратегии	313
Централизованность или распределенность.....	314
По возможности используйте Git.....	315
Рабочий процесс GitFlow и GitHub Flow	316

Настройка процесса непрерывной разработки	320
Непрерывная интеграция	321
Непрерывная доставка	325
Непрерывное развертывание	326
Популярные инструменты для непрерывной интеграции	326
Выбор правильного инструмента и распространенные ошибки	335
Резюме	338
Глава 11. Документирование проекта	339
Технические требования	339
Семь правил технической документации	340
Пишите в два этапа	340
Ориентируйтесь на читателя	341
Упрощайте стиль	342
Ограничивайте объем информации	342
Используйте реалистичные примеры кода	343
Пишите по минимуму, но достаточно	344
Используйте шаблоны	344
Документация как код	345
Использование строк документации в Python	345
Популярные языки разметки и стилей для документации	347
Популярные генераторы документации для библиотек Python	348
Sphinx	349
MkDocs	352
Сборка документации и непрерывная интеграция	352
Документирование веб-API	353
Документация как прототип API с API Blueprint	354
Самодокументирующиеся API со Swagger/OpenAPI	355
Создание хорошо организованной системы документации	356
Создание портфеля документации	356
Ваш собственный портфель документации	362
Создание шаблона документации	363
Шаблон для автора	364
Шаблон для читателя	364
Резюме	365
Глава 12. Разработка на основе тестирования	366
Технические требования	366
Я не тестирую	367
Три простых шага разработки на основе тестирования	367
О каких тестах речь	372
Стандартные инструменты тестирования в Python	375
Я тестирую	380
Ловушки модуля unittest	380
Альтернативы модулю unittest	381
Охват тестирования	388
Подделки и болванки	390
Совместимость среды тестирования и зависимостей	396
Разработка на основе документации	400
Резюме	402

Часть IV. Жажда скорости

Глава 13. Оптимизация — принципы и методы профилирования	404
Технические требования.....	404
Три правила оптимизации.....	405
Сначала — функционал	405
Работа с точки зрения пользователя.....	406
Поддержание читабельности и удобства сопровождения	407
Стратегии оптимизации	408
Пробуем свалить вину на другого	408
Масштабирование оборудования	409
Написание теста скорости.....	410
Поиск узких мест	410
Профилирование использования ЦП	411
Профилирование использования памяти.....	419
Профилирование использования сети	430
Резюме	433
Глава 14. Эффективные методы оптимизации	434
Технические требования.....	435
Определение сложности	436
Цикломатическая сложность	437
Нотация «O большое».....	438
Уменьшение сложности через выбор подходящей структуры данных.....	440
Поиск в списке.....	440
Использование модуля collections	442
Тип deque	442
Тип defaultdict.....	444
Тип namedtuple.....	444
Использование архитектурных компромиссов	446
Использование эвристических алгоритмов или приближенных вычислений	446
Применение очереди задач и отложенная обработка.....	447
Использование вероятностной структуры данных	450
Кэширование	451
Детерминированное кэширование	452
Недетерминированное кэширование	455
Сервисы кэширования.....	456
Резюме	460
Глава 15. Многозадачность	461
Технические требования.....	461
Зачем нужна многозадачность	462
Многопоточность	463
Что такое многопоточность	464
Как Python работает с потоками.....	465
Когда использовать многопоточность	466
Многопроцессорная обработка	481
Встроенный модуль multiprocessing	483
Асинхронное программирование.....	489
Кооперативная многозадачность и асинхронный ввод/вывод.....	490
Ключевые слова <code>async</code> и <code>await</code>	491

Модуль <code>asuncio</code> в старых версиях Python	495
Практический пример асинхронного программирования	495
Интеграция синхронного кода с помощью фьючерсов <code>asunc</code>	498
Резюме	501

Часть V. Техническая архитектура

Глава 16. Событийно-ориентированное и сигнальное программирование	504
Технические требования.....	505
Что такое событийно-ориентированное программирование	505
Событийно-ориентированный != асинхронный.....	506
Событийно-ориентированное программирование в GUI.....	507
Событийно-ориентированная связь	509
Различные стили событийно-ориентированного программирования.....	511
Стиль на основе обратных вызовов.....	511
Стиль на основе субъекта	513
Тематический стиль	515
Событийно-ориентированные архитектуры	518
Очереди событий и сообщений	519
Резюме	521
Глава 17. Полезные паттерны проектирования	523
Технические требования.....	524
Порождающие паттерны	524
Синглтон.....	524
Структурные паттерны.....	527
Адаптер	528
Заместитель.....	542
Фасад.....	543
Поведенческие паттерны	544
Наблюдатель	544
Посетитель	546
Шаблонный метод.....	548
Резюме	550
Приложение. <code>reStructuredText</code> Primer	552
<code>reStructuredText</code>	552
Структура раздела	554
Списки	555
Форматирование внутри строк	556
Блок литералов.....	557
Ссылки.....	558

*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 SACI 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 SACI он работал в университете в ЕСПИ 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 Rughex. 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 по-прежнему 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. Конкуренция не спит. 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 не серьезно. 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 направлении 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.