

УДК 004.43
ББК 32.973.26-018.2
Х45

Dane Hillard
PUBLISHING PYTHON PACKAGES

© 2024 by Eksmo Publishing House. Authorized translation of the English edition © 2023 by Manning Publications. This translation is published and sold by permission of Manning Publications, the owner of all rights to publish and sell the same.

Хиллард, Дэй.
Х45 Публикация пакетов Python. Тестирование, распространение и автоматизация проектов / Дэйв Хиллард ; [перевод с английского В. Краснянской]. — Москва : Эксмо, 2024. — 288 с. — (Мировой компьютерный бестселлер).

ISBN 978-5-04-189146-6

Книга «Публикация пакетов Python» описывает практический процесс масштабируемого совместного использования кода Python с высокой эффективностью и помогает получить опыт работы с новейшими инструментами упаковки. Пособие дает возможность изучить все тонкости тестирования и непрерывной интеграции пакетов, а также предлагает профессиональные советы по созданию поддерживаемого проекта с открытым исходным кодом, включая вопросы лицензирования, документации и создания сообщества участников.

УДК 004.43
ББК 32.973.26-018.2

ISBN 978-5-04-189146-6

© Виктория Краснянская, перевод на русский язык, 2024
© Оформление. ООО «Издательство «Эксмо», 2024

Краткое содержание

Предисловие	11
Предисловие автора	13
Благодарности	15
О книге	17
Об авторе	22
Об иллюстрации на обложке	23
ЧАСТЬ 1. ОСНОВЫ	25
Глава 1. Зачем и почему нужны пакеты Python	27
Глава 2. Подготовка к разработке пакета	42
Глава 3. Анатомия минимального пакета Python	49
ЧАСТЬ 2. СОЗДАНИЕ ЭФФЕКТИВНОГО ПАКЕТА	71
Глава 4. Обработка зависимостей пакета, точек входа и расширений	73
Глава 5. Создание и поддержка пакета программ для тестирования	92
Глава 6. Автоматизация инструментов проверки качества кода	122
ЧАСТЬ 3. ВЫХОД НА ПУБЛИКУ	145
Глава 7. Автоматизация работы с помощью непрерывной интеграции	147
Глава 8. Создание и поддержка документации	174
Глава 9. Поддержка актуальности пакета	207
ЧАСТЬ 4. ДОЛГИЙ РЕЙС	231
Глава 10. Изменение масштабов и укрепление ваших методов	233
Глава 11. Создание сообщества	252
Приложение А. Установка asdf и python-launcher	268
Приложение В. Установка pipx, build, tox, pre-commit и cookiecutter	275
Предметный указатель	279

Содержание

Предисловие	11
Предисловие автора	13
Благодарности	15
О книге	17
Кому стоит прочитать книгу?	17
Как организована книга: схема	18
О коде	20
Обсуждения на форуме liveBook	21
Об авторе	22
Об иллюстрации на обложке	23

ЧАСТЬ 1. ОСНОВЫ

Глава 1. Зачем и почему нужны пакеты Python	27
1.1. Так что же такое пакет?	28
1.1.1. Стандартизация пакетов для автоматизации	29
1.1.2. Содержание распространяемого пакета	31
1.1.3. Трудности совместного пользования программным обеспечением	32
1.2. Как помогает сборка пакетов	33
1.2.1. Усиление сцепления и инкапсуляции с помощью пакетов	33
1.2.2. Четкое определение собственности кода	35
1.2.3. Отделение реализации от использования	36
1.2.4. Заполнение позиций путем создания маленьких пакетов	39
Краткие итоги	40
Глава 2. Подготовка к разработке пакета	42
2.1. Управление виртуальными окружениями Python	43
2.1.1. Создание виртуальных окружений с помощью venv	45
Краткие итоги	48
Глава 3. Анатомия минимального пакета Python	49
3.1. Рабочий процесс сборки Python	50
3.1.1. Части системы сборки Python	50
3.2. Создание метаданных пакета	55
3.2.1. Необходимые основные метаданные	56
3.2.2. Необязательные базовые метаданные	58
3.2.3. Указание лицензии	61

3.3. Контроль исходного кода и обнаружение файлов	63
3.4. Включение в пакет файлов не на Python	66
Краткие итоги	69

ЧАСТЬ 2. СОЗДАНИЕ ЭФФЕКТИВНОГО ПАКЕТА

Глава 4. Обработка зависимостей пакета, точек входа и расширений	73
4.1. Пакет для вычисления отклонения транспортного средства от прямолинейного движения	74
4.2. Создание расширения C для Python	76
4.2.1. Создание исходного расширения на C	77
4.2.2. Интеграция Cython в сборку пакета Python	78
4.2.3. Установка и профилирование расширения C	80
4.2.4. Создание целевых пакетов поставки двоичного кода wheel	82
4.2.5. Определение требуемых версий Python	83
4.3. Поставка инструментов командной строки из пакета Python	84
4.3.1. Создание команд с помощью точек входа <code>setuptools</code>	84
4.4. Управление зависимостями в пакетах Python	87
Ответы на упражнения	90
Краткие итоги	91
Глава 5. Создание и поддержка пакета программ для тестирования	92
5.1. Интеграция тестовой среды	93
5.1.1. Фреймворк для тестирования <code>pytest</code>	93
5.1.2. Добавление управления покрытия тестами	96
Тестовое покрытие ветвей	99
Охват отсутствующего тестового покрытия	101
Упрощение отчета о тестовом покрытии	102
5.1.3. Как увеличить покрытие кода тестами	103
Покрытие неблагоприятных сценариев	104
5.2. Решение проблемы утомительных и скучных тестов	107
5.2.1. Решение проблемы повторяющихся тестов, основанных на данных	107
5.2.2. Решение проблемы частой установки пакета	109
Начало работы с <code>tox</code>	109
Модель сред <code>tox</code>	111
5.2.3. Настройка тестовых сред	114
5.2.4. Советы по более быстрому и безопасному тестированию	116
Запускайте тестовые среды параллельно	116
Тестовое покрытие с фиксацией текущего состояния	117
Убедитесь, что маркеры <code>pytest</code> верны	118
Убедитесь, что ожидаемые отказы не происходят неожиданно	120

Ответы на упражнения	120
Краткие итоги	121
Глава 6. Автоматизация инструментов проверки качества кода	122
6.1. Настоящая сила сред tox	123
6.1.1. Создание сред tox не по умолчанию	124
6.1.2. Управление зависимостями в средах tox	126
6.2. Анализ безопасности типов	130
6.2.1. Создание среды tox для проверки типов	131
6.2.2. Настройка mypy	133
6.3. Создание среды tox для форматирования кода	135
6.3.1. Настройка black	138
6.4. Создание среды tox для проверки качества кода	139
6.4.1. Настройка flake8	141
Ответы на упражнения	142
Краткие итоги	143

ЧАСТЬ 3. ВЫХОД НА ПУБЛИКУ

Глава 7. Автоматизация работы с помощью непрерывной интеграции	147
7.1. Процесс непрерывной интеграции	148
7.2. Непрерывная интеграция с GitHub Actions	149
7.2.1. Высокоуровневый рабочий поток GitHub Actions	151
7.2.2. Терминология GitHub Actions	151
7.2.3. Начинаем конфигурацию рабочего потока GitHub Actions	154
7.3. Преобразование ручных операций в GitHub Actions	157
7.3.1. Многократный запуск джоба с помощью матрицы сборки (build matrix)	160
7.3.2. Создание дистрибутивов пакета Python для различных платформ	162
7.4. Публикация пакета	165
Краткие итоги	173
Глава 8. Создание и поддержка документации	174
8.1. Немного об общем подходе к документации	175
8.2. Создание документации с помощью Sphinx	177
8.2.1. Автоматизация обновления документации в процессе разработки	181
8.2.2. Автоматизация извлеченной из кода документации	182
8.3. Публикация документации на Read the Docs	191
8.3.1. Конфигурация Read the Docs	196
Запуск sphinx-apidoc на Read the Docs	198
Автоматическая сборка Read the Docs для пул-реквестов GitHub	199

8.4. Лучшие методы работы с документацией	201
8.4.1. Что документировать	201
8.4.2. Отдавайте предпочтение ссылкам, а не повторению	202
8.4.3. Используйте четкий и внятный язык	203
8.4.4. Избегайте предположений и приводите контекст	204
8.4.5. Поддерживайте визуальный интерес и последовательную структуру	204
8.4.6. Усиление документации	205
Краткие итоги	206
Глава 9. Поддержка актуальности пакета	207
9.1. Выбор стратегии контроля версий	208
9.1.1. Прямые и непрямые зависимости	208
Прямые и непрямые зависимости на практике	210
9.1.2. Идентификаторы зависимостей Python и ад зависимостей	212
9.1.3. Семантическое и календарное управление версиями	214
9.2. Берем от GitHub все	216
9.2.1. Граф зависимостей GitHub	217
9.2.2. Ослабление уязвимости зависимостей с помощью Dependabot	218
Включение обновления безопасности Dependabot	219
Включение сканирования кода GitHub	220
Автоматическое обновление зависимостей с помощью Dependabot	222
9.3. Пороговое значение тестового покрытия	223
9.4. Обновление синтаксиса Python с помощью ruyupgrade	226
9.5. Уменьшение повторной работы при использовании хуков перед коммитом	226
Ответы на упражнения	229
Краткие итоги	229

ЧАСТЬ 4. ДОЛГИЙ РЕЙС

Глава 10. Изменение масштабов и укрепление ваших методов	233
10.1. Создание шаблона проекта для будущих проектов	234
10.1.1. Создание конфигурации cookiecutter	235
Запрос данных у пользователя	237
Использование предыдущих значений в качестве основы	238
10.1.2. Извлечение шаблона cookiecutter из существующего проекта	239
10.2. Использование пакетов пространства имен	243
10.2.1. Преобразование существующего пакета в пакет пространства имен	246

10.3. Распространение пакетов в вашей организации	247
10.3.1. Серверы частных репозиториев пакетов	247
Конфигурация twine и pip для использования в частном репозитории	248
Краткие итоги	251
Глава 11. Создание сообщества	252
11.1. В файл README необходимо включить предлагаемые преимущества проекта	253
11.2. Предоставьте разную сопроводительную документацию для разных типов пользователей	255
11.3. Учредите кодекс правил поведения, поддерживайте его и контролируйте соблюдение правил	257
11.4. Обозначьте концепцию развития проекта, отмечайте его текущий статус и изменения	259
11.4.1. Использование проектов GitHub для управления в системе канбан	259
11.4.2. Использование меток GitHub для отслеживания статуса отдельных задач	260
11.4.3. Отслеживание высокоуровневых изменений в логе	262
11.5. Последовательный сбор информации с помощью шаблона тикета	264
11.6. Идите вперед	267
Краткие итоги	267
Приложение А. Установка asdf и python-launcher	268
А.1. Установка asdf	269
А.2. Установка python-launcher	272
Ответ на упражнение А.1	274
Приложение В. Установка pipx, build, tox, pre-commit и cookiecutter	275
В.1. Установка pipx	275
В.2. Установка build	276
В.3. Установка tox	277
В.4. Установка pre-commit	277
В.5. Установка cookiecutter	278
Предметный указатель	279

Предисловие

Каждый начинает дорогу к Python по-своему. Но в итоге все равно ищет путь вверх по течению, чтобы увидеть новые места. Когда река сужается и над водой начинают нависать ветви деревьев, приходит пора задуматься, не показать ли свои приложения Python другим людям. Тогда и обнаруживается множество раздробленных остовов проектов у входа в таинственную пещеру в истоке реки. Вы все же готовы броситься в эту пропасть, но вдруг слышите заданный бесстрастным голосом вопрос: «А какой ваш любимый инструмент оформления программ в пакеты?»

У Python для каждого найдется что-то свое, обычно нужное можно разыскать в Python Package Index (<https://pypi.org>). Однако PyPI приносит и новые проблемы, потому что вы начинаете волноваться об установке, зависимостях, средах и других важных вопросах, связанных с использованием вашего программного обеспечения в реальном мире. Здесь создаются и поддерживаются новые пакеты с открытым исходным кодом. Это и стало мотивом создания такой нужной книги.

Проблема большинства трудов по Python в том, что на самом деле никто не хочет уделять особое внимание оформлению программ в пакеты (я сужу об этом, опираясь на собственный опыт). О, авторы книг более чем рады поговорить о модулях и пакетах с точки зрения архитектуры программного обеспечения. Некоторые книги даже предлагают простые заготовки основного пакета, которые можно взять на вооружение. Однако сам процесс создания загружаемого пакета эксплуатационного уровня обычно оставляется

читателю «в качестве упражнения». В отличие от них данная книга занимает уникальную нишу, поскольку напрямую решает эту конкретную задачу.

Даже несмотря на то, что большинство программистов, использующих Python, не склонны выпускать общедоступные пакеты, им часто приходится предоставлять свой код на Python другим, а при решении таких задач зачастую возникают некоторые трудности. Они являются естественным последствием сложности экосистемы. В пакетах Python часто содержится не только сам Python. Например, они могут включать соединение таких языков программирования, как Python, C, C++, Fortran и Rust. Пакеты могут запускаться в самых разных операционных системах, таких как Linux, Mac и — в последнее время — WebAssembly. Добавляют проблем и совместимость многочисленных версий Python, зависимости между пакетами и постоянно меняющаяся ситуация с инструментами управления пакетами. Кажется, проблемы растут как снежный ком.

Прежде чем приступить к написанию этого предисловия, я задал себе один очень простой вопрос: получится ли использовать извлеченные из этой книги знания, чтобы улучшить какие-то из моих собственных проектов? Я работаю с Python уже двадцать пять лет. За это время я выпустил несколько небольших пакетов и продолжаю их поддерживать. Тем не менее это всегда являлось для меня дополнительной сферой деятельности. Честно говоря, в целом я стремился выполнять поменьше работы, связанной с оформлением программ в пакеты. Поэтому я начал читать данную книгу с определенной долей скептицизма и даже со старомодной косностью и не мог дать никакого вразумительного совета по поводу «лучших методов» композиции программ, потому что просто ничего о них не знал.

С радостью сообщаю, что эта книга многому меня научила. Во-первых, она применяет очень современный подход к существующему инструментарию оформления пакетов. Во-вторых, предоставляет контекст, связанный с задачами, с которыми сталкиваются разработчики пакетов, и только потом представляет практическое решение этих задач. Также я познакомился с несколькими новыми трюками, связанными с инструментами, уже использованными мною ранее (например, `pytest`, `coverage` и т. д.). Наконец, оценил советы по созданию и поддержке сообщества для своего проекта.

Все это свидетельствует, что перед вами нелегкое чтение. Даже при современном отношении все еще не существует универсального подхода к оформлению пакетов. Вы можете пробовать различные варианты и адаптировать написанное здесь под себя. Думаю, главное — сохранять неподвзятое отношение и воспринимать книгу как практическое руководство, открывающее определенные возможности. Поступив именно так, вы, полагаю, найдете в этом тексте полезный источник вдохновения.

Дэвид Бизли (<https://www.dabeaz.com>), автор книги Python Distilled

Предисловие автора

Осенью 2014 года я начал работать в компании ИТНАКА. Команда без устали трудилась над задачей освободить от проприетарного контента систему управления, цикл выпуска которой занимал месяцы, что существенно замедляло скорость ее изменения. Наши усилия окупились сторицей, дав жизнь новой платформе доставки приложений, способной поддерживать подвижность, к которой мы стремились.

Группа фронтенда использовала фреймворк Django для ориентированной на пользователя разработки на JSTOR (<https://www.jstor.org>). Этот выбор сыграл определенную роль в моих дальнейших действиях. Также команда вкладывалась в поддержку проекта с устанавливаемыми пакетами Python, что принесло немалую пользу, поскольку наши продуктовые предложения являлись фрагментированными, но при этом сохраняли общую функциональность. Предметная область бизнеса, связанная с нашим контентом, и модели доступа были сложными; пакеты устанавливали между ними полезные границы и давали возможность использования в других условиях. Однако несмотря на то, что эта концепция получила значительное развитие, мы по-прежнему страдали от нескольких недоработок.

Мы не имели частного хранилища пакетов, так что их требовалось устанавливать из нашей системы контроля версий. Также не существовало

семантического управления версиями, а хеши коммитов в Git давали не слишком много информации или возможностей регулирования изменений между версиями. Мы не сохраняли журнал изменений проекта: оставленные во время коммитов сообщения являлись для нас средством контроля изменений.

В последующие годы наша работа с Django и Python превратилась в тысячи строк кода, обслуживающих большую часть трафика на JSTOR. Наши пакеты росли в размерах, их становилось все больше, и несоответствия в них вынуждали создавать обходные пути. Мы сливали пакеты, чтобы дополнить код нашего приложения, отказываясь от возможности с относительной простотой рассматривать изменения, немедленно отражающиеся в процессе разработки. Код размещался там, где было проще всего это сделать.

Этот подход изменился, когда группа разработки базовой инфраструктуры вложила средства в профессиональную поддержку частных пакетов. Увидев новые возможности и проанализировав имеющуюся у нас организационную структуру, я попытался понять, как улучшить систему непрерывной интеграции и стандартизации, чтобы поддерживать гибкость и адаптивность и в то же время повышать качество и сохранять способность совершать изменения.

Проект `apiron` (<https://github.com/ithaka/apiron>) стал нашей первой попыткой использовать новые методы создания пакетов программ и превратился в первый активно разрабатываемый проект компании ИТНАКА с открытым исходным кодом, предназначенный для использования третьими лицами. По мере того как преимущества процесса создания пакетов становились все более явными, мы начали его широкое применение. Сегодня группа фронтенда ИТНАКА поддерживает более двух десятков пакетов Python, соответствующих такому же количеству приложений.

Цель ИТНАКА — облегчить доступ к знаниям, и я надеюсь, что эта книга внесет свой вклад, открыв вам глаза на некоторые вещи и предоставив то, что вы надеялись и мечтали отыскать в других источниках. Хотя это практическое руководство, я также рассчитываю, что вы приобретете теоретические и философские основы, которые помогут вам и вашим командам значительно продвинуться с автоматизацией и воспроизводимыми процессами. Возможно, в каких-то местах вы со мной не согласитесь, и это прекрасно: так появляется шанс внести что-то новое в коллективное знание, которое мы как творцы пополняем снова и снова. Я надеюсь, что книга вам понравится, вы извлечете из нее пользу, будете с ней не соглашаться и ее критиковать.

В любой момент вы можете обратиться ко мне с вопросами, рассказать об историях успеха и поспорить по адресу `pubwrapack@danehillard.com`.

Благодарности

Каких бы трудов мне ни стоило добраться до последней строки этой книги, она никогда не появилась бы на свет, если бы не громадная работа, проделанная людьми, составляющими Python Software Foundation и Python Packaging Authority. Ваши усилия, направленные на продвижение вперед оформления программ в пакеты Python, очень ценны. Здесь я собирал знания и методы, но во многом опирался на плечи гигантов.

Я мог позволить себе роскошь писать свою первую книгу в элитном магазине домашнего интерьера с уютной кофейней в задней части. С другой стороны, на свет она появилась дома — в основном за столом моей спутницы жизни Стефани. Спасибо тебе за спокойствие, доброту и твой особенный юмор. Если мы смогли справиться с вызванной пандемией повышенной раздражительностью от жизни в четырех стенах и с моими бурными жалобами и прокрастинацией, возможно, однажды мы вместе завоюем мир.

Спасибо команде ITNAKA за постоянное стремление к знаниям, усовершенствованиям и инновациям. Ваша потребность делать все хорошо побуждает меня двигаться вперед.

Эта книга могла вообще не появиться на свет. Спасибо моему редактору-консультанту по аудитории Тони Арритоле и рецензенту издательства Майку Стефенсу за то, что они дали мне второе дыхание. Ваша поддержка и обратная связь убеждали, что моя попытка может принести стоящий результат.

Алу Кринкеру, моему техническому редактору, я благодарен за постоянные вопросы о целях. Это, разумеется, улучшило наглядность работы и повысило ее воздействие.

Спасибо Марджан Бейс и всей команде Manning за то, что они дали жизнь этой книге и передали ее в руки тех, кто в ней нуждается.

Я очень благодарен храбрым людям, которые в самом начале вложили средства в создание этой книги и во время ее написания оставались со мной на связи. Вы наставляли меня на путь истинный и не давали спотыкаться о препятствия.

Спасибо всем рецензентам, работавшим над этой книгой: Алексей Агарков, Кейдж Слагель, Клиффорд Тербер, Даниэль Холт, Дэвид Кабреро Суто, Дэвид Кронкайт, Делена Малан, Эдгар Хасслер, Эммануэль Пиччинелли, Эрик Чанг, Ганеш Свамнатан, Хавард Уолл, Ховард Банди, Джим Амрхейн, Джонни Хопкинс, Хосе Апаблаза, Джошуа А. Макадамс, Катя Паткин, Кевин Этьенн, Кимберли Л. Уинстон-Джексон, Ларри Кей, Лаксман Сингх Томар, Марк-Энтони Тейлор, Магиас Аффуртит, Маттиас Буш, Майк Бэран, Мики Тебека, Ричард Дж. Тобиас, Ричард Мейнсен, Роберт Вандеруолл, Сатил Атали, Сириам Мачарла, Васудеван Сурендран, Видхья Винай, Врж Мохан и Зо-хоб Айнапур. Ваши предложения улучшили эту книгу.

И наконец, я благодарю тех, кто так или иначе оказал на эту книгу положительное влияние — напрямую, непреднамеренно или каким-то иным образом. Не надеюсь составить полный список этих людей: какие-то имена так в нем и не появились исключительно из-за ограниченных возможностей моей памяти. Тем не менее спасибо И Дурбину, Дастину Инграму, Бретту Кэннону, Полу Ганслу, Филипе Лаинсу, Бернату Габору, Лукашу Ланге, Себастьяну Эсташу, Томасу Клюйверу, Дональду Стафту, Симону Уиллисону, Уиллу Макгагену, Дон Уэйджес, Реувену Лернеру, Дэвиду Бизли, Бретту Слаткину, Тцу Пину Чунгу, Генри Шрейнеру, Прадиуну Гедаму, Полу Муру, Тусшару Садхвани, Сэнди Метцу, Джейсону Кумбсу, Джеффу Триpletу, Карлтону Гибсону, Крису Колосивски и Питеру Ангу.

О книге

«Публикация пакетов Python» рассказывает о нескольких направлениях, связанных конкретно с оформлением пакетов в Python, а также о нескольких концепциях, приложимых практически к любому языку программирования. Вместе они позволяют командам и отдельным разработчикам более эффективно осуществлять поставку программного обеспечения. Команды разработки, рабочие группы и инженеры SRE найдут здесь новые полезные в работе методы и инструменты. Если вы хотите автоматизировать, стандартизировать и организовать как можно больше своих проектов Python, возможно, эта книга для вас.

Кому стоит прочитать книгу?

«Публикация пакетов Python» предназначена для любого читателя, уже знакомого с языком программирования Python и имеющего желание поделиться своим кодом с друзьями, коллегами или незнакомыми людьми во всем мире. Приемы, описанные в этой книге, отобраны так, чтобы их мог использовать один человек, но приложимы и к работе команды практически любого размера. Сотрудничество — ключ к эффективной разработке программного обеспечения, так что эти методы способствуют избавлению от утомительного

однообразия и появлению возможности сосредоточиться на таком необходимом общении с помощью кода и текста.

По мере того как в научном сообществе растет роль программного обеспечения, повышается ценность пакетов программ. Успешные проекты с открытым исходным кодом использовались в последних знаковых достижениях науки, таких как посадка на Марс или фотографии черных дыр. Ищете ли вы крупный проект или просто хотите убедиться, что PI в вашей лаборатории может проверять код, который вы используете для получения результатов, воспроизводимые процессы — именно то, что может вам помочь.

Если ранее вы не работали с такими инструментами оценки качества программного обеспечения, как модульное или юнит-тестирование и линтеры, эта книга поможет подробнее ознакомиться с автоматизацией работы и улучшить проверку качества кода в высокоавтоматизированных модулях. Вам выпал шанс подумать о выявлении проблем до того, как они разрастутся, вместо того чтобы постоянно заниматься ликвидацией разгорающихся пожаров.

Как организована книга: схема

«Публикация пакетов Python» состоит из 11 глав, разделенных на четыре части. В части 1 рассказывается о самостоятельной ценности организованного в пакеты программного обеспечения любого рода. Часть 2 проведет вас через создание работающего пакета со всеми дополнительными компонентами, которые могут для этого понадобиться. Часть 3 погрузит в автоматизацию и сопровождение, необходимые для проектов, требующих высокой степени сотрудничества. Часть 4 покажет, как повторить процесс и привести к масштабу ваших пользователей и сотрудников.

Часть 1 «Основы» готовит почву для оформления в пакеты программ на Python и дает правильный настрой на самостоятельное создание пакета. В ней рассматриваются следующие вопросы.

- В главе 1 рассказывается, как появилось пакетирование и почему оно сохраняет свою ценность для совместного использования ПО и по сей день. Эта глава расширит понимание того, что можно отнести к пакетам, и поможет осознать, что такое оформление программ нацелено на многочисленную аудиторию.
- Глава 2 начинает рассказ об инструментах, которые можно использовать для оформления продукта в пакет.
- Глава 3 показывает подоплеку того, что означает оформление продукта Python в пакет, в том числе связанные с этим процессом файлы и метаданные, а также то, как они работают в нем.