

Грег Уилсон

**РАЗРАБОТКА
ПРОГРАММНОГО
ОБЕСПЕЧЕНИЯ**
НА ПРИМЕРАХ
С ПОМОЩЬЮ PYTHON

УДК 004.42
ББК 32.973.26-018
У36

Software Design by Example
A Tool-Based Introduction with Python 1st edition

by Greg Wilson
© 2024 Greg Wilson
All Rights Reserved.

Authorised translation from the English language edition published
by CRC Press, a member of the Taylor & Francis Group LLC.

Уилсон, Грег.
У36 Разработка ПО на примерах с помощью Python / Грег Уилсон ;
[перевод с английского О. Д. Сайфудиновой, О. И. Перфильева]. —
Москва : Эксмо, 2025. — 400 с. — (Мировой компьютерный бестсел-
лер).

ISBN 978-5-04-204718-3

Книга Грега Уилсона посвящена практическому освоению принципов проектирования и создания программ. Автор, известный преподаватель и сооснователь Software Sargentry, объясняет, как на основе реальных примеров — от систем резервного копирования до движков браузеров — понять устройство современных инструментов и научиться писать чистый, повторно используемый код. Издание рассчитано на разработчиков, знакомых с основами Python, и содержит упражнения, примеры и советы, помогающие применять теорию на практике.

УДК 004.42
ББК 32.973.26-018

ISBN 978-5-04-204718-3

© Сайфудинова О.Д., Перфильев О.И., перевод на русский язык, 2025
© Оформление. ООО «Издательство «Эксмо», 2025

Разработка программного обеспечения на примерах с помощью Python

Лучший способ разобраться в любом проектировании — это поучиться на примерах. А самые качественные примеры проектирования ПО¹ можно найти в инструментах, с которыми работают программисты. Таким образом, в «**Разработке ПО на примерах с помощью Python**» создаются небольшие версии того, чем пользуются программисты, раскрываются их тайны и проливается свет на образ мышления опытных специалистов. Мы начнем с системы резервного копирования файлов и фреймворка для тестирования и дойдем до обработчика регулярных выражений, а также движка браузера и очень небольшого компилятора. На их примере изучим общие шаблоны проектирования, покажем, как упрощение кода для тестирования облегчает и его повторное использование, поможем читателям понять, как работают отладчики, профилировщики, пакетные менеджеры и системы контроля версий, и подскажем, как работать с ними эффективнее.

Данным материалом можно воспользоваться для самостоятельного изучения, на курсах бакалавриата по проектированию ПО, либо взять за основу недельного интенсива для практикующих программистов. В каждой главе дается набор упражнений разного объема и сложности — от нескольких десятков строк до полноценного дня работы. Читателям должны быть знакомы основы современного Python, но более продвинутые возможности языка объясняются и иллюстрируются.

Все письменные материалы данного проекта можно свободно воспроизводить в рамках лицензии Creative Commons (с указанием авторства), а само программное обеспечение доступно в рамках Hippocratic License. Все доходы от продажи этой книги пойдут на поддержку семейного приюта Red Door в Торонто.

Основные моменты:

- объясняется проектирование ПО; программистам показывается, как создавать инструменты для своей каждодневной работы;
- в каждой главе есть упражнения, чтобы читатели могли проверить и углубить свои знания;
- весь код из примеров можно скачать, повторно использовать и отредактировать в рамках открытой лицензии.

¹ Программное обеспечение — *Прим. пер.*

Доктор Грег Уилсон — программист, автор и преподаватель из Торонто. Сооснователь и первый исполнительный директор Software Carpentry — компании, обучившей базовым навыкам работы с ПО десятки тысяч исследователей по всему миру. Грег выступал в роли автора или редактора для более чем десятка книг (в том числе двух детских). Является членом фонда Python Software Foundation и лауреатом Ассоциации вычислительной техники SIGSOFT в номинации «Авторитетный педагог года».

Посвящение

Эта книга посвящается Майку и Джону:
я рад, что вы всегда находили время поболтать.

Оглавление

Разработка программного обеспечения на примерах с помощью Python	5
1 Введение	13
1.1. Целевая аудитория	13
1.2. Главные мысли	15
1.3. Форматирование	16
1.4. Использование	17
1.5. Отзывы	17
1.6. Благодарности	18
1.7. Упражнения	20
2 Объекты и классы	21
2.1. Объекты	21
2.2. Классы	25
2.3. Аргументы	26
2.4. Наследование	28
2.5. Итоги	31
2.6. Упражнения	32
3 Поиск повторяющихся файлов	34
3.1. Первые шаги	35
3.2. Хеширование файлов	37
3.3. Улучшаем хеширование	39
3.4. Итоги	41
3.5. Упражнения	42
4 Сопоставление шаблонов	44
4.1. Простые шаблоны	45
4.2. Новый подход	50
4.3. Итоги	53
4.4. Упражнения	53
5 Синтаксический анализ текста (парсинг)	55
5.1. Токенизация	56
5.2. Парсинг	59
5.3. Итоги	62
5.4. Упражнения	62
6 Выполнение тестов	63
6.1. Хранение и выполнение тестов	64
6.2. Нахождение функций	67
6.3. Итоги	69

6.4. Упражнения	70
7 Интерпретатор	73
7.1. Выражения	74
7.2. Переменные	77
7.3. Интроспекция	78
7.4. Итоги	81
7.5. Упражнения	81
8 Функции и замыкания	84
8.1. Определение и хранение	84
8.2. Вызов функций	85
8.3. Замыкания	88
8.4. Итоги	91
8.5. Упражнения	91
9 Протоколы	95
9.1. Моск-объекты	95
9.2. Протоколы	98
9.3. Декораторы	100
9.4. Итераторы	104
9.5. Итоги	106
9.6. Упражнения	107
10 Архиватор файлов	108
10.1. Сохранение файлов	109
10.2. Тестирование	110
10.3. Отслеживание резервных копий	113
10.4. Рефакторинг	115
10.5. Итоги	116
10.6 Упражнения	117
11 Валидатор HTML	119
11.1. HTML и DOM	119
11.2. Шаблон «Посетитель»	122
11.3. Проверка стилей	125
11.4. Итоги	127
11.5. Упражнения	127
12 Генератор шаблонов	129
12.1. Синтаксис	130
12.2. Управление переменными	132
12.3. Посещение узлов	132
12.4. Реализация обработчиков	135
12.5. Поток управления	138
12.6 Итоги	140
12.7. Упражнения	141

13	Анализатор кода («линтер»)	144
	13.1. Алгоритмы обработки данных	145
	13.2. Поиск дубликатов ключей	148
	13.3. Поиск неиспользуемых переменных	149
	13.4. Итоги	152
	13.5. Упражнения	152
14	Верстка страниц	154
	14.1. Определение размеров	154
	14.2. Размещение (позиционирование)	157
	14.3. Отрисовка (визуализация)	159
	14.4. Перенос элементов	162
	14.5. Итоги	165
	14.6. Упражнения	166
15	Профилирование производительности	168
	15.1. Варианты	168
	15.2. Построчное хранение	170
	15.3. Колоночное хранение	173
	15.4. Производительность	176
	15.5. Итоги	180
	15.6. Упражнения	181
16	Персистентность (сохранение) объектов	184
	16.1. Встроенные типы	185
	16.2. Конвертация в классы	188
	16.3. Ссылочная эквивалентность («алиасинг»)	191
	16.4. Итоги	196
	16.5. Упражнения	196
17	Двоичные данные	198
	17.1. Целые числа	198
	17.2. Битовые операции	200
	17.3. Текст	201
	17.4. И снова персистентность	203
	17.5. Итоги	209
	17.6. Упражнения	210
18	База данных	211
	18.1. С чего начать	211
	18.2. Сохранение записей	215
	18.3. База данных с хранением в файле	217
	18.4. Работа с блоками	218
	18.5. Персистентность блоков	221
	18.6. Очистка	223
	18.7. Итоги	224
	18.8. Упражнения	225

19 Менеджер сборки	226
19.1. Основные понятия	226
19.2. Первая реализация	227
19.3. Топологическая сортировка	229
19.4. Более удачная реализация	231
19.5. Итоги	234
19.6. Упражнения	235
20 Менеджер пакетов	237
20.1. Семантическое версионирование	238
20.2. Полный перебор	238
20.3. Генерация вариантов вручную	242
20.4. Инкрементальный поиск	244
20.5. Доказатель теорем	246
20.6. Итоги	249
20.7. Упражнения	250
21 Передача файлов	252
21.1. TCP/IP	253
21.2. Разбиение на блоки («чанки»)	256
21.3. Тестирование	259
21.4. Итоги	261
21.5. Упражнения	261
22 Обслуживание веб-страниц	263
22.1. Протокол	263
22.2. Привет, веб	266
22.3. Отправка файлов	268
22.4. Тестирование	270
22.5. Итоги	272
22.6. Упражнения	273
23 Просмотрщик файлов	275
23.1. Curses	275
23.2. Оконный интерфейс	278
23.3. Прокрутка	280
23.4. Рефакторинг	282
23.5. Ограничение перемещений	286
23.6. Область просмотра	288
23.7. Итоги	290
23.8. Упражнения	290
24 Отмена и повтор	292
24.1. Начало работы	292
24.2. Вставка и удаление	294
24.3. Перемещение назад	297
24.4. Итоги	301
24.5. Упражнения	302

25	Виртуальная машина	303
	25.1. Архитектура	303
	25.2. Исполнение	306
	25.3. Ассемблерный код	308
	25.4. Массивы	313
	25.5. Итоги	315
	25.6. Упражнения	316
26	Отладчик	318
	26.1. По шагу за раз	318
	26.2. Тестирование	321
	26.3. Расширяемость	324
	26.4. Точки останова	326
	26.5. Итоги	329
	26.6. Упражнения	330
27	Заключение	332
A	Библиография	334
Б	Дополнительные материалы	336
	Б.1. Использование атрибутов функций	336
	Б.2. Ленивые вычисления	339
	Б.3. Расширение	341
	Б.4. Отслеживание наследования	343
	Б.5. Исследование функций	346
	Б.6. Пользовательские классы	346
	Б.7. Вещественные числа (с плавающей точкой)	350
	Б.8. Порядок байт от старшего к младшему и от младшего к старшему	352
	Б.9. Генерация тестовых данных	353
В	Основные идеи (учебная программа)	355
Г	Лицензия	362
	Г.1. Текстовые материалы	362
	Г.2. Программное обеспечение	363
Д	Кодекс поведения	365
	Д.1. Наши стандарты	365
	Д.2. Наши обязанности	365
	Д.3. Область применения	366
	Д.4. Применение	366
	Д.5. Источник	366
Е	Вклад в проект	367
	Е.1. Редактирование контента	367
	Е.2. Принятие решений	368
	Е.3. Вопросы и ответы	369
	Глоссарий	371
	Предметный указатель	392

1

Введение

- Сложность системы растет быстрее, чем ее размер.
- Лучший способ изучать проектирование — разбирать примеры. Самые полезные примеры — это программы, которыми программисты пользуются каждый день.
- Предполагается, что читатели уже умеют писать небольшие программы и хотят перейти на более крупные проекты, либо ищут материалы для своих уроков по проектированию ПО.
- Все материалы доступны для чтения и повторного использования в рамках открытой лицензии, а все авторские отчисления от продаж данной книги будут направлены на благотворительность.

Упомянутые термины: **когнитивная нагрузка**

Лучший способ разобраться в любом проектировании — это поучиться на примерах [Schon1984; Petre2016], а самые понятные примеры — те, с которыми читатели уже знакомы. Именно поэтому в уроках создают небольшие версии инструментов, которыми программисты пользуются каждый день², и на их примере показывается, как думают более опытные проектировщики ПО. Попутно уроки знакомят с ключевыми идеями в информатике — такими, с которыми многие программисты-самоучки еще не сталкивались. Надеемся, данные уроки помогут вам разрабатывать более качественное ПО. И если вы понимаете, как работают программные средства, то с большей вероятностью начнете ими пользоваться, причем будете делать это гораздо эффективнее.

1.1. Целевая аудитория

Лучше всего целевую аудиторию данной книги описывает следующий образ обучающегося [Wilson2019]:

У Майи есть степень магистра в области геномики. Майя знает Python на том уровне, чтобы анализировать данные своих экспериментов, но с трудом пишет код, которым могли бы пользоваться другие. Эти уроки покажут ей, как проектировать, создавать и тестировать крупные приложения быстрее и легче.

² https://en.wikipedia.org/wiki/Programming_tool

1.2. Главные мысли

Наш подход к проектированию базируется на трех главных идеях. Первая: с ростом количества компонентов система быстро усложняется (Иллюстрация 1.2). Но объем информации, которую мы можем хранить в рабочей памяти в любой момент времени, постоянный и сравнительно небольшой [Hermans2021]. Если мы хотим писать более крупные программы, которые сможем понять, то нужно составлять их из частей с ограниченным взаимодействием. И основа того, что мы называем «проектированием» — чтобы понять, какими именно должны быть эти части и их взаимодействия.

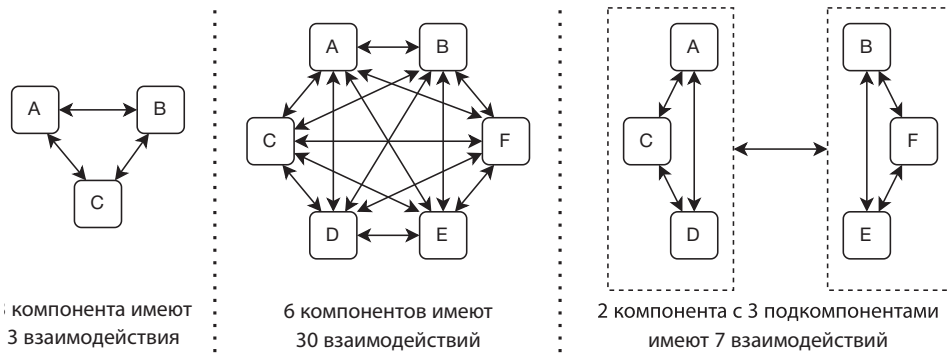


Иллюстрация 1.2. Как с увеличением размера растет сложность

Во-вторых, «понятность» зависит от нас самих. В низкоуровневом языке огромная важность придается **когнитивной нагрузке** по сборке микрошагов во что-то более осмысленное. С другой стороны, при использовании высокоуровневого языка мы получаем ту же нагрузку, переводя назначения функций в фактические операторы реальных данных.

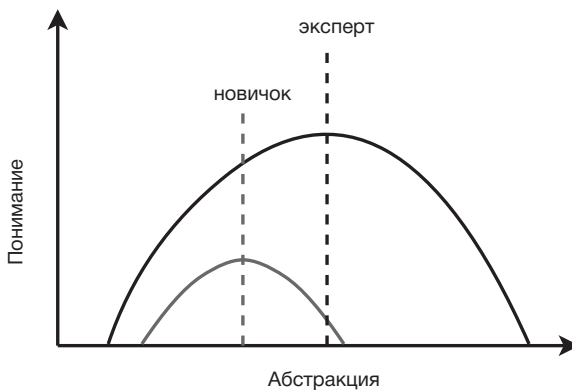


Иллюстрация 1.3. Кривые понимания новичков и экспертов

Более опытные программисты обладают большими способностями на обоих концах кривой, но это не единственное, что меняется. Кривая понимания новичков больше похожа на нижнюю (Иллюстрация 1.3), тогда как у экспертов она выглядит как та, что показана выше. Эксперты не просто больше понимают на всех уровнях абстракции; *предпочтительный* для них уровень также меняется, поэтому прочитать современную запись $\sqrt{x^2 + y^2}$ для них проще, чем ее средневековый аналог «сторона квадрата, площадь которого равна сумме площадей двух квадратов со сторонами из первой и второй части». Данная кривая означает, что для любой конкретной задачи код, понятный новичку, почти гарантированно будет отличаться от кода, который быстрее всего поймет эксперт.

Наша третья главная мысль состоит в том, что программы — это лишь еще один тип данных. Исходный код — это просто текст, который можно обрабатывать также, как любые текстовые файлы. Точно также программа в памяти — это просто структура данных, которую можно проверять и изменять, как и любую другую. Отношение к коду как к данным помогает нам решать сложные проблемы элегантным путем, но за счет повышения уровня абстракции наших программы. И еще раз: под «проектированием» («дизайном») ПО мы подразумеваем как раз поиск этого баланса.

1.3. Форматирование

Исходный код на языке Python отображается в книге следующим образом:

```
for ch in "example":
    print(ch)
```

Команды оболочки Unix отображается следующим образом:

```
for filename in *.dat
do
    cut -d, -f 10 $filename
done
```

Файлы данных и вывод программ отображаются вот так:

```
- name: read
  params:
  - sample_data.csv
```

```
alpha
beta
gamma
delta
```

Многоточие (...) показывает, где опущены строки, и иногда разрывает строки нелогичным способом, чтобы они поместились на странице. В таких случаях все строки, кроме последней, заканчиваются обратной косой чертой \. И, наконец, записи глоссария выделяются **жирным шрифтом**, а функции прописываются как *имя_функции*, а не *имя_функции()*. Последний вариант в технической литературе используется чаще, но из-за пустых круглых скобок, сложнее понять, идет ли речь о самой функции, либо о вызове функции без параметров.

1.4. Использование

Источник этой книги доступен в нашем репозитории Git⁵, и все материалы можно почитать на сайте⁶. Письменные материалы данной книги распространяются по лицензии Creative Commons Attribution — NonCommercial 4.0 International license⁷ (CC — BY — NC-4.0), а программное обеспечение подпадает под Hippocratic License⁸. Первая лицензия позволяет вам использовать и переделывать этот материал в некоммерческих целях в исходном или в адаптированном виде, со ссылкой на оригинальный источник. Если вы хотите продавать копии или зарабатывать на этом материале каким-либо иным способом, вы должны сначала связаться с нами⁹ и получить разрешение. Вторая лицензия позволяет вам использовать и переделывать приложение на этом сайте при условии, что вы не нарушаете международные соглашения, регулирующие права человека; подробнее см. Приложение Г.

Если вы хотите улучшить имеющийся материал, добавить что-то новое или задать вопрос, пожалуйста, откройте тикет (issue) в нашем GitHub-репозитории¹⁰ или отправьте нам электронное письмо¹¹. Все участники должны соблюдать наш этический кодекс (Приложение Д).

1.5. Отзывы

Вот что говорят люди о JavaScript-версии этой книги [Wilson2022a]:

- Джессика Керр¹²: «Это книга, которую я порекомендую каждому новому разработчику. Она хочет, чтобы у вас все получилось... это мостик от учебы до работы программиста».

⁵ <https://github.com/gvwilson/sdхpy>

⁶ <https://third-bit.com/sdхpy>

⁷ <https://creativecommons.org/licenses/by-nc/4.0>

⁸ <https://firstdonoharm.dev/>

⁹ <mailto:gvwilson@third-bit.com>

¹⁰ <https://github.com/gvwilson/sdхpy/>

¹¹ <mailto:gvwilson@third-bit.com>

¹² <https://jessitron.com/2023/02/20/book-review-software-design-by-example/>