Грег Уилсон

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

НА ПРИМЕРАХ С ПОМОЩЬЮ PYTHON

БОМБОРА
ИЗДАТЕЛЬСТВО
Москва

УДК 004.42
ББК 32.973.26-018
У36

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

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 бестселлер).

ISBN 978-5-04-204718-3

Книга Грега Уилсона посвящена практическому освоению принципов проектирования и создания программ. Автор, известный преподаватель и сооснователь Software Carpentry, 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

Разработка программного обеспечения на примерах с помощью Python

Лучший способ разобраться в любом проектировании — это поучиться на примерах. А самые качественные примеры проектирования ПО[1] 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.

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. Соосновательи 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

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 создаются небольшие версии инструментов, которыми программисты пользуются каждый день[2], 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

Вы, как и Майя, должны уметь:

- писать программы на Python с помощью списков, циклов, условных конструкций, словарей и функций;
- разбираться в программах на Python с классами и исключениями;
- запускать базовые команды Unix-оболочки: ls и mkdir;
- немного читать и писать HTML;
- сохранять файлы и делиться ими на Git[3] (ничего страшного, если вы не знаете более сложных команд[4]).

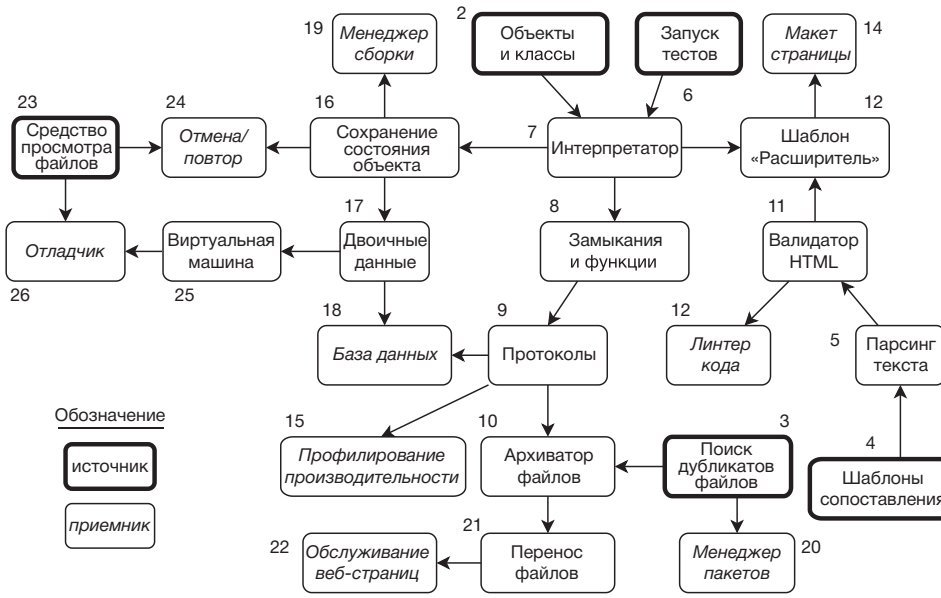

Иллюстрация 1.1. Темы и зависимости урока

Эти главы (Иллюстрация 1.1) также помогут и читателям, соответствующим другому образу:

> *Йим преподает в колледже два курса по разработке ПО. Он разочарован тем, что во многих учебниках говорится о нюансах, а не о самом проектировании, и приводятся примеры, непонятные ученикам. Эта книга даст учителям тот материал, которым можно пользоваться в классе в качестве отправной точки для курсовых проектов.*

[3] https://git-scm.com/

[4] https://git-man-page-generator.lokaltog.net/

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.

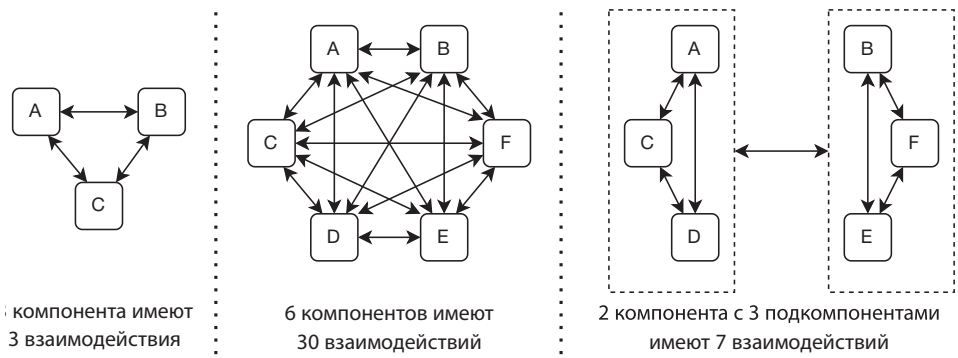

Иллюстрация 1.2. Как с увеличением размера растет сложность

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 Git[5], и все материалы можно почитать на сайте[6]. Письменные материалы данной книги распространяются по лицензии Creative Commons Attribution — NonCommercial 4.0 International license[7] (CC — BY — NC-4.0), а программное обеспечение подпадает под Hippocratic License[8]. 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 нами[9] 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-репозитории[10] или отправьте нам электронное письмо[11]. Все участники должны соблюдать наш этический кодекс (Приложение Д).

1.5. Отзывы

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

- Джессика Керр[12]: «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/sdxpy

[6] https://third-bit.com/sdxpy

[7] https://creativecommons.org/licenses/by-nc/4.0

[8] https://firstdonoharm.dev/

[9] mailto: gvwilson@third-bit.com

[10] https://github.com/gvwilson/sdxpy/

[11] mailto: gvwilson@third-bit.com

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