

Я посвящаю эту книгу всем бескорыстным членам сообществ Python, которые помогают этому языку идти в ногу со временем.

И всем тем, кто сделал изучение Python и связанных с ним технологий настолько сложным, что нужна *подобная* книга, чтобы справиться с ними.

## Пол Бэрри: «Изучаем программирование на Python», 2-е издание

На прогулке Пол остановился, чтобы обсудить правильное произношение слова «tuple» со своей терпеливой женой.



Обычная реакция Дейдры ☺

**Пол Бэрри** живет и работает в Карлоу (Ирландия), маленьком городке с населением около 35 тысяч человек в 80 км к юго-западу от Дублина.

Пол имеет степень бакалавра наук в области информационных систем и степень магистра в области вычислений. Он также закончил аспирантуру и получил свидетельство преподавателя и обучения.

Работает в Технологическом институте Карлоу с 1995 и читает лекции с 1997 года. Прежде чем начать преподавательскую деятельность, Пол десять лет посвятил ИТ-индустрии, работал в Ирландии и Канаде, большая часть его работы была связана с медицинскими учреждениями. Пол женат на Дейдре, у них трое детей (двое сейчас учатся в колледже).

Язык программирования Python (и связанные с ним технологии) составляют основу послевузовских курсов Пола с 2007 учебного года.

Пол является вторым (или со вторым) автором еще четырех книг: двух о Python и двух о Perl. В прошлом он подготовил довольно много статей для *Linux Journal Magazine*, в котором является пишущим редактором.

Пол вырос в Белфасте, Северная Ирландия, и это во многом объясняет некоторые его взгляды и забвения (впрочем, если вы тоже «с севера», тогда взгляды Пола и его забвения вполне нормальными).

Вы можете найти Пола в *Твиттере* (@barrypj). У него есть также своя домашняя страница <http://paulbarry.itcarlow.ie>.

## Оглавление (Краткое)

1	Основы. Начнем поскорее	37
2	Списки. Работа с упорядоченными данными	83
3	Структурированные данные. Работа со структурированными данными	131
4	Повторное использование. Функции и модули	181
5	Построение веб-приложения. Возвращение в реальный мир	231
6	Хранение и обработка данных. Где хранить данные	279
7	Использование баз данных. Используем DB-API в Python	317
8	Немного о классах. Абстракция поведения и состояния	345
9	Протокол управления контекстом. Подключение к инструкции with	371
10	Декораторы функций. Обертывание функций	399
11	Обработка исключений. Что делать, когда что-то идет не так	449
11 <sup>3</sup> / <sub>4</sub>	Немного о многопоточности. Обработка ожидания	497
12	Продвинутые итерации. Безумные циклы	513
A	Установка. Установка Python	557
B	Pythonanywhere. Развертывание веб-приложения	565
C	Топ-10 тем, которые мы не рассмотрели. Всегда есть чему поучиться	575
D	Топ-10 проектов, которые мы не рассмотрели. Еще больше инструментов, библиотек и модулей	587
E	Присоединяйтесь. Сообщество Python	599

## Содержание (Конкретное)

### Введение

**Ваш мозг и Python.** Вы пытаетесь чему-то научиться, а мозг делает вам одолжение и *сопротивляется* изо всех сил. Он думает: «Лучше оставить место для запоминания действительно важных вещей. Вдруг нам встретится голодный тигр или захочется покататься голышом на сноуборде. Я должен помнить об опасности». Как же нам *обмануть* ваш мозг, чтобы он считал программирование на Python важным для выживания?

Для кого эта книга ?	26
Мы знаем, о чем вы подумали	27
Мы знаем, о чем подумал ваш мозг	27
Метод познания: рэзмышления о мышлении	29
Вот что мы сделали	30
Прочти меня	32
Команда технических редакторов	34
Признательности и благодарности	35

## ОСНОВЫ

## 1

**Начнем поскорее****Начнем программировать на Python как можно скорее.**

В этой главе мы ознакомимся с основами программирования на Python и сделаем это в характерном для нас стиле: с места в карьер. Через несколько страниц вы запустите свою первую программу. К концу главы вы сможете не только запускать типичные программы, но также понимать их код (и это еще не все!). Попутно вы познакомитесь с некоторыми особенностями языка **Python**. Итак, не будем больше тратить время. Переверните страницу — и вперед!

Значение окон IDLE	40
Выполнение кода, одна инструкция за раз	44
Функции + модули = стандартная библиотека	45
Встроенные структуры данных	49
Вызов метода возвращает результат	50
Принятие решения о запуске блока кода	51
Ключевые слова могут иметь «if»?	53
Блоки кода могут содержать встроенные блоки	54
Возвращение в командную оболочку Python	58
Экспериментируем в оболочке	59
Перебор последовательности объектов	60
Повторяем определенное количество раз	61
Применим решение задачи № 1 к нашему коду	62
Упростим процесс выполнения	64
Генерация случайных чисел на Python	66
Создание серьезного бизнес-приложения	74
Отступы в 4-х местах?	76
Попросим интерпретатор помочь с функцией	77
Эксперименты с диапазонами	78
Код из главы 1	82



## Списки

## 2

## Работа с упорядоченными данными

Все программы обрабатывают данные, и программы на Python — не исключение.

На самом деле *данные повсюду*. Ведь в основном программирование — это работа с данными: *получение* данных, *обработка* данных, *интерпретация* данных. Чтобы работать с данными более эффективно, нужен какой-то контейнер, куда их можно *сложить*. Python предоставляет удобные структуры данных *широкого применения*: **списки**, **словари**, **кортежи** и **множества**. В этой главе мы бегло рассмотрим все четыре, а затем углубимся в изучение **списков** (остальные три структуры подробнее рассмотрены в следующей главе). Мы уже затрагивали эту тему ранее, поскольку все, с чем нам приходится сталкиваться при программировании на Python, так или иначе относится к работе с данными.

Числ , строки... и объекты	84
Встреч айте: четыре встроенные структуры д нных	86
Слов рь: неупорядоченн я структур д нных	88
Множество: структур д нных, не позволяющ я дублиров ть объекты	89
Созд ние литер льных списков	91
Если р бот ете с фр гментом код большим, чем п р строки, используйте ред ктор	93
З полнение список во время выполнения	94
Проверк прин джности с помощью in	95
Уд ление объектов из список	98
Доб вление элементов в список	100
Вст вк элементов в список	101
К к скопиров ть структуру д нных	109
Списки р ширают нот цию с кв др тными скобк ми	111
Со список ми можно использов ть ди п зоны	112
Н ч ло и конец ди п зон в список х	114
Р бот ем со срез ми в списке	116
Использов ние цикл «for» со список ми в Python	122
Срезы в дет лях	124
Когд не нужно использов ть списки	127
Код из гл вы 2	128

0	1	2	3	4	5	6	7	8	9	10	11
D	o	n	'	t		p	a	n	i	c	!
-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

## Структурированные данные

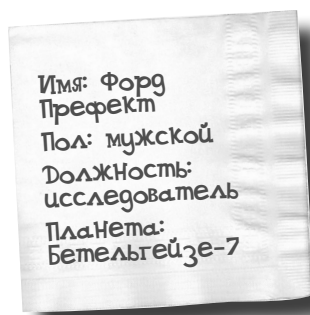
## 3

## Работа со структурированными данными

## Списки в Python очень удобны, но они не панацея.

Когда имеются *действительно* структурированные данные (для хранения которых список оказывается не лучшим выбором), спасение приходит от встроенных **словарей** Python. Словари «из коробки» позволяют хранить и обрабатывать данные, которые можно представить в виде *пар ключ/значение*. В этой главе мы изучим словари, а также **множества** и **кортежи**. Как и **списки** (которые мы изучили в главе 2), словари, множества и кортежи предоставляют встроенные инструменты для работы с данными, что делает Python еще более удобным языком.

Словари хранят пары ключ/значение	132
Как определяются словари в коде	134
Порядок добавления НЕ поддерживается	135
Выбор значений с помощью квадратных скобок	136
Работа со словарями во время выполнения программы	137
Изменение счетчик	141
Итерации по значениям в словарях	143
Итерации по ключам и значениям	144
Итерации по словарям с использованием items	146
Несколько динимичных словарей?	150
Предотвращение ошибок KeyError во время выполнения	152
Проверка вхождения с помощью in	153
Не забывайте инициализировать ключ перед использованием	154
Значения in и not in	155
Работа с методом setdefault	156
Создание множеств эффективно	160
Использование методов множеств	161
Сделаем пример с кортежами	168
Комбинирование встроенных структур данных	171
Доступ к данным, хранящимся в сложных структурах	177
Код из главы 3	179



## 4

## Повторное использование

## Функции и модули

## Повторное использование кода — ключ к построению стабильных систем.

В случае Python все повторное использование начинается и заканчивается **функциями**. Возьмите несколько строк кода, дайте им имя — и у вас готова функция (которую можно использовать повторно). Возьмите коллекцию функций и сохраните их в файле — у вас готовый **модуль** (который можно использовать повторно). Это правда, когда говорят, что *делиться приятно*, и в конце главы вы уже сможете создавать код для **многократного** и **совместного** использования, благодаря пониманию того, как работают функции и модули Python.

Повторное использование кода с помощью функций	182
Представляем функции	183
Вызов функции	186
Функции могут принимать аргументы	190
Возврат одного значения	194
Возврат более одного значения	195
Вспомним встроенные структуры данных	197
Создание универсальной и полезной функции	201
Создание другой функции	202
Значения по умолчанию для аргументов	206
Позиционные и именованные аргументы	207
Повторим, что мы узнали о функциях	208
Запуск Python из командной строки	211
Создание необходимых файлов установки	215
Создание файла дистрибутива	216
Установка пакетов при помощи «pip»	218
Демонстрация семантики вызова по значению	221
Демонстрация семантики вызова по ссылке	222
Установка инструментов разработчика для тестирования	226
Соответствует ли наш код рекомендациям в PEP 8?	227
Работаем с сообщениями об ошибках	228
Код из главы 4	230



модуль

## 5

## Построение веб-приложения

**Возвращение в реальный мир****Вы уже знаете Python достаточно, чтобы быть опасными.**

Четыре главы книги освоены, и сейчас вы в состоянии продуктивно использовать Python во многих областях применения (хотя многое еще предстоит узнать). Вместо того чтобы исследовать эти области, в этой и последующих главах мы изучим разработку веб-приложений — в этом Python особенно силен. Попутно вы еще больше узнаете о Python. Однако вначале позвольте кратко подытожить то, что вы уже знаете.

Python: что вы уже знаете	232
Чего мы хотим от нового веб-приложения?	236
Дайте установку новым Flask	238
Как работает Flask?	239
Первый запуск веб-приложения Flask	240
Создание объекта веб-приложения Flask	242
Декорирование функции URL	243
Запуск функций веб-приложения	244
Размещение функциональности в Web	245
Построение HTML-формы	249
Шablоны связаны с веб-страницами	252
Отображение шаблонов из Flask	253
Отображение HTML-формы веб-приложения	254
Подготовка к запуску кода с шаблонами	255
Коды состояния HTTP	258
Обработка отправленных данных	259
Оптимизация цикла рендеринга/отновки/запуска/проверки	260
Доступ к данным HTML-формы с помощью Flask	262
Использование данных запроса в веб-приложении	263
Выводим результат в виде HTML	265
Подготовка веб-приложения к развертыванию в облаке	274
Код из главы 5	277



## Хранение и обработка данных

# 6

### Где хранить данные

#### Рано или поздно появляется необходимость обеспечить надежное хранение данных.

И когда придет время **сохранить данные**, Python вам поможет. В этой главе вы узнаете о хранении и извлечении данных из *текстовых файлов*, которые — как механизм хранения — могут показаться слишком простыми, но тем не менее используются во многих проблемных областях. Кроме сохранения и извлечения данных из файлов, вы научитесь некоторым премудростям при работе с данными. «Серьезный материал» (хранение информации в базе данных) мы припасли для следующей главы, однако с файлами тоже придется потрудиться.

Рбот с данными из веб-приложения	280
Python позволяет открывать, обрабатывать и извлекать	281
Чтение данных из существующего файла	282
Лучше «with», чем открыть, обработать, закрыть	284
Просмотр журналов в веб-приложении	290
Исследуем исходный код строки	292
Пришло время экранировать (в шрифты)	293
Просмотр всего журнала в веб-приложении	294
Журналирование отдельных трибутов веб-зпрос	297
Журналирование данных в одну строку с разделителями	298
Вывод данных в читаемом формате	301
Генерируем читаемый вывод с помощью HTML	310
Встраиваем логику отображения в шрифт	311
Создание читаемого вывода с помощью Jinja2	312
Текущее состояние кода нашего веб-приложения	314
Задать вопросы о данных	315
Код из главы 6	316

Form Data	Remote_addr	User_agent	Results
ImmutableMultiDict([('phrase', 'hitch-hiker'), ('letters', 'aeiou')])	127.0.0.1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36	{'e', 'i'}

## Использование базы данных

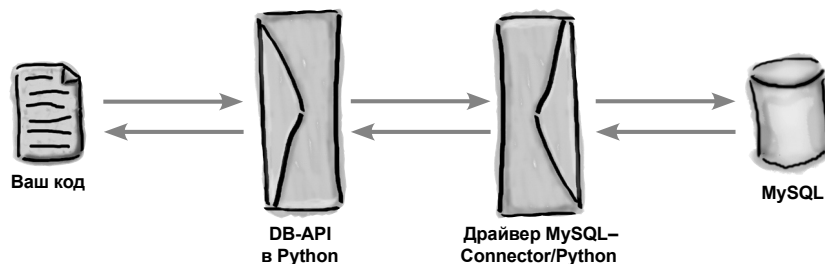
## 7

## Используем DB-API в Python

**Хранить информацию в реляционной базе данных очень удобно.**

В этой главе вы узнаете, как организовать взаимодействие с популярной базой данных (БД) **MySQL**, используя универсальный прикладной программный интерфейс, который называется **DB-API**. Интерфейс DB-API (входящий в состав стандартной библиотеки Python) позволяет писать код, не зависящий от конкретной базы данных... если база данных понимает SQL. Хотя мы будем использовать MySQL, ничто не мешает вам использовать код DB-API с вашей любимой реляционной базой данных, какой бы она ни была. Давайте посмотрим, как пользоваться реляционной базой данных в Python. В этой главе не так много нового в плане изучения Python, но использование Python для общения с БД — **это важно**, поэтому стоит поучиться.

Включить поддержку баз данных в веб-приложении	318
Задание 1. Установить сервер MySQL	319
Введение в Python DB-API	320
Задание 2. Установить драйвер баз данных MySQL для Python	321
Установка MySQL-Connector/Python	322
Задание 3. Создание баз данных и таблиц для веб-приложения	323
Выбор структуры для журналируемых данных	324
Убедимся, что таблица готова к использованию	325
Задание 4. Программирование операций с базой данных и таблицами	332
Хранение данных — только половина дела	336
Как организовать код для роботов с базой данных?	337
Подумайте, что вы собираетесь использовать повторно	338
А что с тем импортом?	339
Вы видели этот шаблон раньше	341
Неприятность не так уж неприятна	342
Код из главы 7	343



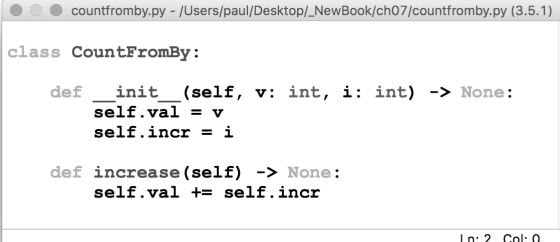
# 8 Немного о классах

## Абстракция поведения и состояния

### Классы позволяют связать поведение кода и состояние вместе.

В этой главе мы отложим веб-приложение в сторону и будем учиться создавать **классы**. Это умение понадобится вам при создании диспетчера контекста. Классы — настолько полезная штука, что вам в любом случае стоит ознакомиться с ними поближе, поэтому мы посвятили им отдельную главу. Мы не будем рассматривать классы во всех подробностях, а коснемся лишь тех аспектов, которые пригодятся для создания диспетчера контекста, которого ожидает наше веб-приложение. А теперь вперед, посмотрим, что к чему.

Подключ емся к инструкции «with»	346
Объектно-ориентиров нный пример	347
Созд ние объектов из кл ссов	348
Объекты обл д ют общим поведением, но не состоянием	349
Р ширяем возможности CountFromBy	350
Вызов метод : подробности	352
Доб вление метод в кл сс	354
В жность «self»	356
Обл сть видимости	357
Доб вляйте к имен м трибутов прист вку «self»	358
Иници лиз ция трибутов перед использов нием	359
Иници лиз ция трибутов в «init» с двойными подчеркив ниями	360
Иници лиз ция трибутов в «__init__»	361
Предст вление CountFromBy	364
Определение предст вления CountFromBy	365
Определение целесообр зных умолч ний для CountFromBy	366
Кл ссы: что мы зн ем	368
Код из гл вы 8	369



```

class CountFromBy:

    def __init__(self, v: int, i: int) -> None:
        self.val = v
        self.incr = i

    def increase(self) -> None:
        self.val += self.incr

```

Ln: 2 Col: 0

# Протокол управления Контекстом

## 9

### Подключение к инструкции with

#### Пора применить все изученное на практике.

В главе 7 мы обсудили использование **реляционных баз данных** в Python, а в главе 8 ознакомились с использованием **классов**. В главе 9 мы объединим обе методики и создадим **диспетчер контекста**, который позволит расширить инструкцию `with` для работы с системами реляционных баз данных. В этой главе вы подключитесь к инструкции `with`, создав новый класс, соответствующий требованиям **протокола управления контекстом** в языке Python.

Выбор лучшего способа повторного использования кода для работы с БД	372
Управление контекстом с помощью методов	374
Вы уже видели, как действует диспетчер контекста	375
Создание нового класса диспетчер контекста	376
Инициализация класса параметрами соединения с базами данных	377
Выполнение строки в « <code>__enter__</code> »	379
Выполнение завершающего кода с использованием « <code>__exit__</code> »	381
Повторное обсуждение кода веб-приложения	384
Вспомним функцию « <code>log_request</code> »	386
Изменение функции « <code>log_request</code> »	387
Вспомним функцию « <code>view_the_log</code> »	388
Изменился не только этот код	389
Изменение функции « <code>view_the_log</code> »	390
Ответы на вопросы о данных	395
Код из главы 9	396

```
File Edit Window Help Checking our log DB
$ mysql -u vsearch -p vsearchlogDB
Enter password:
Welcome to MySQL monitor...

mysql> select * from log;
+----+-----+-----+-----+-----+-----+-----+
| id | ts           | phrase                | letters | ip         | browser_string | results          |
+----+-----+-----+-----+-----+-----+-----+
| 1  | 2016-03-09 13:40:46 | life, the uni ... ything | aeiou   | 127.0.0.1 | firefox        | {'u', 'e', 'i', 'a'} |
| 2  | 2016-03-09 13:42:07 | hitch-hiker           | aeiou   | 127.0.0.1 | safari         | {'i', 'e'}        |
| 3  | 2016-03-09 13:42:15 | galaxy                | xyz     | 127.0.0.1 | chrome         | {'y', 'x'}        |
| 4  | 2016-03-09 13:43:07 | hitch-hiker           | xyz     | 127.0.0.1 | firefox        | set()              |
+----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.0 sec)

mysql> quit
Bye
```

# 10 Декораторы функций

## Обертывание функций

### Протокол управления контекстом из главы 9 — не единственная возможность улучшить код.

Python также позволяет использовать **декораторы** функций — технологию, с помощью которой можно добавлять код к существующим функциям, *не* изменяя имеющийся код. Если вы увидите в этом сходство с черной магией, не переживайте: ничего подобного. Однако многие программисты на Python считают создание декораторов функций одной из наиболее сложных сторон языка, поэтому пользуются этой возможностью реже, чем должны бы. В этой главе мы покажем, что создавать и использовать декораторы совсем не сложно, несмотря на всю их «продвинутость»

Веб-сервер (не в ш компьютер) з пуск ет в ш код	402
Поддержк се нсов в Flask позволяет хр нить состояние	404
Поиск по слов рю позволяет получить состояние	405
Орг низ ция вход в систему с помощью се нсов	410
Выход из системы и проверк состояния	413
Перед ем функцию в функцию	422
Вызыв ем перед нную функцию	423
Приним ем список ргументов	426
Обр ботк список ргументов	427
Приним ем слов рь ргументов	428
Обр ботк слов ря ргументов	429
Приним ем любое количество ргументов любого тип	430
Созд ние декор тор функции	433
Последний ш г: р бот с ргумент ми	437
Использов ние декор тор	440
Н з д к орг ничению доступ к /viewlog	444
Код из гл вы 10	446



## 11

## Обработка исключений

**Что делать, когда что-то идет не так**

**Каким бы хорошим ни был ваш код, иногда все равно что-то идет не так.**

Вы успешно выполнили все примеры в книге и, скорее всего, убедились, что имеющийся код работает. Значит ли это, что его можно считать надежным? По всей видимости, нет. Писать код и надеяться, что ничего плохого не случится, в лучшем случае наивно. В худшем случае — опасно, поскольку что-то непредвиденное порой все же происходит (и будет происходить). При написании кода лучше быть осторожным, а не доверчивыми. Осторожным, чтобы ваш код делал именно то, что вам нужно, и правильно реагировал, если что-то пойдет не так. В этой главе вы увидите, что может пойти не так, и узнаете, что делать, когда (и, чаще всего, прежде чем) такое произойдет.

Б зы д нных не всегда доступны	454
Веб- т ки — н стоящ я боль	455
Ввод-вывод быв ет медленным (иногда)	456
Вызовы функций могут ок нчив ться неуд чей	457
Всегда используйте try для код с высоким риском ошибок	459
Одн инструкция try, но несколько эксерт	462
Обр ботчик любых исключений	464
Узн ем об исключениях из «sys»	466
Универс льный обр ботчик, повторение	467
Н з д к н шему веб-приложению	469
Тих я обр ботк исключений	470
Обр ботк других ошибок в б з е д нных	476
Избег йте тесных связей в коде	478
Модуль DBcm, еще р з	479
Созд ние собственного исключения	480
Что еще может пойти не т к с «DBcm»?	484
Обр ботк SQLError отлич ется	487
Возбужд ем SQLError	489
Подведение итогов: доб вление н дежности	491
К к быть с ожид нием? Есть в ри нты...	492
Код из гл вы 11	493

```

...
Exception
+-- StopIteration
+-- StopAsyncIteration
+-- ArithmeticError
|   +-- FloatingPointError
|   +-- OverflowError
|   +-- ZeroDivisionError
+-- AssertionError
+-- AttributeError
+-- BufferError
+-- EOFError
...

```

11<sup>3/4</sup>

## Немного о многопоточности

**Обработка ожидания**

**Иногда для выполнения кода может потребоваться много времени.**

В одном случае это становится проблемой, в другом — нет. Если код работает «за кулисами» и ему нужно 30 секунд, чтобы сделать какие-то свои дела, ожидание не превращается в проблему. Однако если приложению требуется 30 секунд, чтобы ответить пользователю, это заметят все. Решение проблемы зависит от того, какие действия пытаетесь выполнить вы (и кто именно ожидает). В этой короткой главе мы обсудим некоторые варианты, а затем рассмотрим одно из решений данной проблемы: *как быть, если что-то требует слишком много времени?*

Ожидание: что делать?	498
Как выполняются запросы к базе данных?	499
Инструкции INSERT и SELECT быстрыми отличаются	500
Делать несколько дел сразу	501
Не грустим: используем потоки	502
В первую очередь: не паниковать	506
Не грустим: Flask нам поможет	507
Нужно ли в наше веб-приложение сейчас?	510
Код из главы 11 <sup>3/4</sup>	511

