

Я посвящаю эту книгу всем бескорыстным членам сообщества 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 на юго-запад от Дублина.

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

Пол 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 автором (или соавтором) 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 забавный акцент (впрочем, если вы тоже «с севера», тогда взгляды Пола и его акцент покажутся вам вполне нормальными).

Вы можете найти Пола в *Твиттере* (@barrypj). У него есть также своя домашняя страничка <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 базы 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
11¾	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
Метапознание: размышления 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!

Назначение 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
Какие варианты может 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
Устраиваем паузу выполнения	64
Генерация случайных чисел на Python	66
Создание серьезного бизнес-приложения	74
Отступы вас бесят?	76
Попросим интерпретатор помочь с функцией	77
Эксперименты с диапазонами	78
Код из главы 1	82



2

Списки

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.

Числа, строки... и объекты	84
Встречайте: четыре встроенные структуры данных	86
Словарь: неупорядоченная структура данных	88
Множество: структура данных, не позволяющая дублировать объекты	89
Создание литеральных списков	91
Если работаете с фрагментом кода большим, чем пара строк, используйте редактор	93
Заполнение списка во время выполнения	94
Проверка принадлежности с помощью in	95
Удаление объектов из списка	98
Добавление элементов в список	100
Вставка элементов в список	101
Как скопировать структуру данных	109
Списки расширяют нотацию с квадратными скобками	111
Со списками можно использовать диапазоны	112
Начало и конец диапазона в списках	114
Работаем со срезами в списке	116
Использование цикла «for» со списками в Python	122
Срезы в деталях	124
Когда не нужно использовать списки	127
Код из главы 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

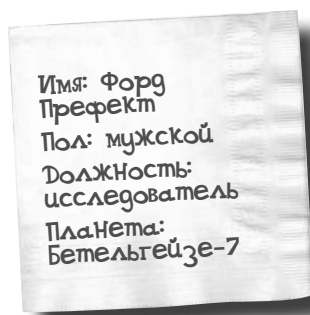
3

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 счетчика	141
Итерации по записям в словарях	143
Итерации по ключам и значениям	144
Итерации по словарям с использованием items	146
Насколько динамичны словари?	150
Предотвращение ошибок KeyError во время выполнения	152
Проверка вхождения с помощью in	153
Не забывайте инициализировать ключ перед использованием	154
Замена in на not in	155
Работа с методом setdefault	156
Создаем множества 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
Вызываем 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
Задание значений 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
Разбираемся 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 нашего веб-приложения?	236
Давайте установим 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 Веб	245
Построение HTML-формы	249
Шаблоны связаны с веб-страничками	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 редактирование/остановка/запуск/проверка	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.

Работа с данными из веб-приложения	280
Python позволяет открывать, обрабатывать и закрывать	281
Чтение данных из существующего файла	282
Лучше «with», чем открыть, обработать, закрыть	284
Просмотр журнала в веб-приложении	290
Исследуем исходный код страницы	292
Пришло время экранировать (ваши данные)	293
Просмотр всего журнала в веб-приложении	294
Журналирование отдельных атрибутов веб-запроса	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 шаблон	311
Создание читаемого вывода с помощью Jinja2	312
Текущее состояние кода нашего веб-приложения	314
Задаем 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

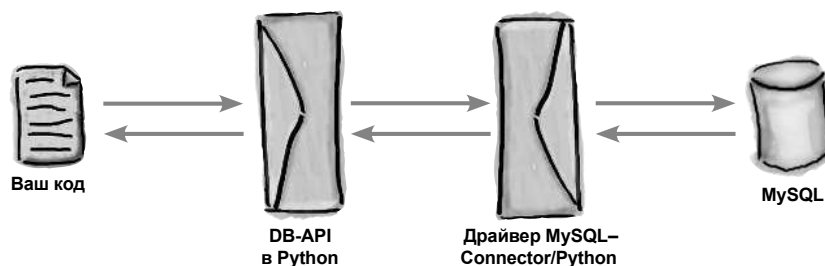
7

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.

Включаем поддержку баз данных в веб-приложении	318
Задача 1. Установка сервера MySQL	319
Введение в Python DB-API	320
Задача 2. Установка драйвера базы данных MySQL для Python	321
Установка MySQL-Connector/Python	322
Задача 3. Создание базы данных и таблиц для веб-приложения	323
Выбираем структуру для журналируемых данных	324
Убедимся, что таблица готова к использованию	325
Задача 4. Программирование операций с базой данных и таблицами	332
Хранение данных — только половина дела	336
Как организовать код для работы 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 такая уж неприятная	342
Код из главы 7	343



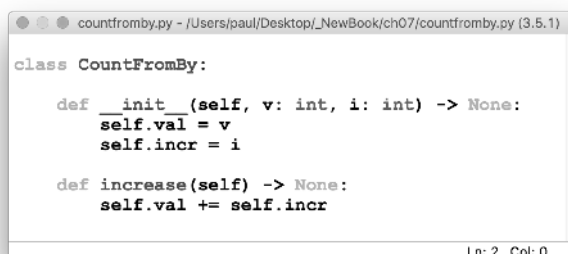
8

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.

Подключаемся к инструкции «with»	346
Объектно-ориентированный пример	347
Создание объектов из классов	348
Объекты обладают общим поведением, но не состоянием	349
Расширяем возможности CountFromBy	350
Вызов метода: подробности	352
Добавление метода в класс	354
Важность «self»	356
Область видимости	357
Добавляйте к именам атрибутов приставку «self»	358
Инициализация атрибута перед использованием	359
Инициализация атрибутов в «init» с двойными подчеркиваниями	360
Инициализация атрибутов в «__init__»	361
Представление CountFromBy	364
Определение представления CountFromBy	365
Определение целесообразных умолчаний для CountFromBy	366
Классы: что мы знаем	368
Код из главы 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