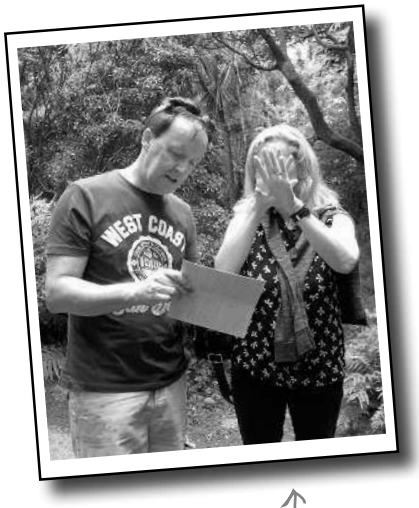
Я посвящаю эту книгу всем бескорыстным членам сообщества Python, которые помогают этому языку идти в ногу со временем.

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)

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?

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!

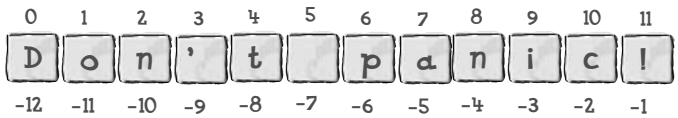
Списки

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.

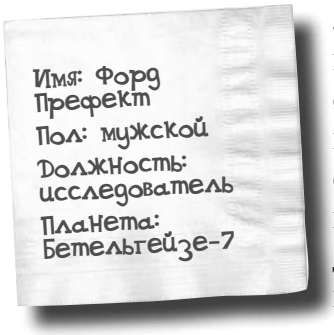


Структурированные данные

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.



Повторное использование

4 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 готовая 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.

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.

6 Хранение и обработка данны

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.

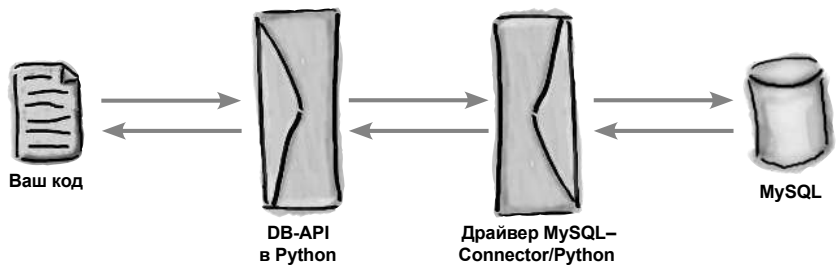
Form Data	Remote_addr	User_agent	Results
ImmutableMultiDict([('phrase', 'hitch-hiker'), ('letters', 'aeiou')])	127.0.0.1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36	{'e', 'i'}

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 помешает 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.



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.

countfromby.py - /Users/paul/Desktop/_NewBook/ch07/countfromby.py (3.5.1)

```
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