Содержание

Об авторе

Эрик Вестра уже больше 25 лет является профессиональным разработчиком программного обеспечения, который в течение последнего десятилетия работает почти исключительно с Python. Первоначальный интерес Эрика к проектированию графического интерфейса пользователя привел к разработке одной из самых продвинутых систем срочной курьерской доставки, используемых службами и компаниями курьерской доставки по всему миру. В последние годы Эрик участвует в разработке и реализации систем по подбору поставщиков для потребителей товаров и услуг по целому ряду регионов с широкой географией и с использованием разных систем обмена мгновенными сообщениями и платежных систем. Эта работа включает в себя создание геокодеров в реальном масштабе времени и режимов просмотра постоянно меняющихся данных географических карт. Эрик проживает в Новой Зеландии и работает на компании, расположенные по всему миру.

Кроме того, он является автором выпущенных в издательстве Packt книг «*Геоанализ на Python*» и «*Создание картографических приложений*» на основе QGIS, а также предстоящей публикации «*Модульное программирование на языке Python*».

Я хотел бы поблагодарить Рут за ее великолепие, а моих дочерей – за их терпение. Без вас ничего из этого не было бы возможным.

О рецензенте

Лу Може получил образование в области информатики очень давно в Университете штата Мичиган, где учился использовать программное обеспечение для разработки циклотрона. Затем он в течение 34 лет проработал в компании IBM и потом продолжил карьеру, работая на несколько консалтинговых фирм, включая долгосрочное сотрудничество с железнодорожной отраслью индустрии. В настоящее время он консультирует компанию Keyhole Software, расположенную в Ливуд, шт. Канзас. Прошлой весной для этой компании он создал программный инструмент MockOlap для каркасного прототипирования на основе операций перетаскивания. Лу программирует на C++, Java и более новых языках и в настоящее время интересуется микросервисами, Docker, Node.js, NoSQL, геопространственными системами, функциональным программированием, мобильными и одностраничными веб-приложениями – любым новым языком или платформой. Лу иногда ведет блог, посвященный программным технологиям. Он является соавтором трех книг по информатике, написал два учебных руководства по XML для IBM DeveloperWorks и учебное руководство по LDAP для журнала WebSphere. Кроме того, Лу является соавтором нескольких сертификационных тестов по J2EE для IBM, а также рецензентом ряда других издателей.

Предисловие

С ростом популярности картографических веб-сайтов и пространственно ориентированных устройств и приложений методы разработки геопространственного программного обеспечения образовали быстро развивающуюся область информатики – геоинформатику. Будучи разработчиком на Python, вы не можете позволить себе отстать от новейших тенденций. В сегодняшнем информированном о местоположении мире каждый разработчик на Python может извлечь выгоду из понимания концепций и методики разработки геопространственных приложений.

Работа с геопространственными данными бывает непростой, потому что она связана с математическими моделями земной поверхности. Но поскольку Python – это мощный язык программирования с большим количеством высокоуровневых программных инструментов, он идеально подходит для разработки геопространственных приложений. Эта книга познакомит вас с инструментами языка Python, которые требуются для разработки геопространственных приложений. Она проведет вас по ключевым геопространственным понятиям, таким как географическое положение, расстояние, единицы измерения, картографические проекции, геодезические датумы и форматы геопространственных данных. Затем мы займемся изучением ряда программных библиотек Python и воспользуемся ими и общедоступными геопространственными данными для решения самых разнообразных задач. Книга предоставляет углубленный анализ методов хранения пространственных данных в базе данных и приемов использования пространственных баз данных в качестве инструментов для решения широкого диапазона геопространственных задач.

В ней подробно рассматриваются методы генерирования карт при помощи инструмента визуализации цифровых карт – программной библиотеки Mapnik. Кроме того, книга поможет вам создать высокотехнологичное геопространственное веб-приложение с функционалом редактирования карты на основе географического модуля GeoDjango для веб-платформы Django, программной библиотеки Mapnik и геопространственного расширения PostGIS для СУБД PostgreSQL. К концу книги вы научитесь интегрировать пространственный функционал в свои собственные приложения и создавать полнофункциональные картографические приложения с нуля.

Эта книга представляет собой практическое руководство, которое научит вас приемам получения доступа к геоданным, управления ими и их визуализации, используя широкий диапазон инструментов Python для разработки геоинформационных систем (ГИС).

О чем эта книга рассказывает

Глава 1 «*Разработка геопрограмм на Python*» предлагает обзор языка программирования Python и концепций, лежащих в основе процесса разработки геопро-

странственного приложения. Кроме того, будут затронуты основные прецеденты использования разработок геопространственных приложений, последние достижения и тенденции ближайшего будущего в данной области.

Глава 2 «*Геоинформационные системы*» посвящена ознакомлению с базовыми понятиями, включая географическое положение, расстояние, единицы измерения, картографические проекции, географические фигуры, геодезические датумы и форматы геоданных, и затем обсуждению процесса работы с геоданными в ручном режиме.

Глава 3 «*Библиотеки Python для геопрограммирования*» разбирает основные библиотеки Python, предназначенные для разработки геоприложений, в том числе функционал библиотек, способы их установки, важные понятия, необходимые для понимания работы библиотек, и способы их применения.

Глава 4 «*Источники геоданных*» посвящена исследованию главных общедоступных источников геоданных, характеристике имеющейся в распоряжении информации, используемым форматам данных и приемам импортирования данных после их скачивания.

Глава 5 «*Решение задач с геоданными на Python*» посвящена применению ранее представленных библиотек для выполнения различных задач с использованием геоданных, включая смену картографических проекций, импорт и экспорт данных, конвертирование и стандартизацию единиц геометрий и расстояний, выполнение геопространственных расчетов.

Глава 6 «*Пространственные базы данных*» вводит понятия, лежащие в основе пространственных баз данных, и затем подробно рассматривает расширение PostGIS для СУБД PostgreSQL с поддержкой пространственных данных, способы его установки и использования из программы на Python.

Глава 7 «*Генерирование карт при помощи Python и библиотеки Mapnik*» предлагает подробный анализ инструмента картографирования Mapnik и приемов его применения для генерирования разнообразных карт.

Глава 8 «*Работа с пространственными данными*» познакомит с процессом проектирования и реализации полнофункционального геоприложения под названием DISTAL, используя общедоступные геоданные, хранящиеся в пространственной базе данных.

Глава 9 «*Совершенствование приложения DISTAL*» посвящена развитию функционала приложения из предыдущей главы для решения ряда задач, связанных с удобством использования и производительностью.

Глава 10 «*Инструменты для разработки геопространственных веб-приложений*» исследует понятия платформ веб-приложений, веб-служб, библиотек пользовательского интерфейса на JavaScript и скользящих карт. Данная глава познакомит с рядом стандартных веб-протоколов, используемых в геопространственных приложениях, и закончится обзором инструментария, при помощи которого в трех заключительных главах книги будет создано полнофункциональное картографическое приложение.

Глава 11 «*Собираем все вместе – полнофункциональная картографическая система*» знакомит с полноценным и высокотехнологичным веб-приложением

ShapeEditor, созданным с использованием геопространственного расширения PostGIS, библиотеки Mapnik и географического модуля GeoDjango. Работа начинается с проектирования законченного приложения и затем продолжается созданием моделей базы данных приложения ShapeEditor.

Глава 12 «*ShapeEditor – импорт и экспорт файлов фигур*» продолжает реализацию системы ShapeEditor, концентрируясь на отображении списка импортированных файлов фигур и подпрограммах импорта и экспорта файлов фигур через веб-браузер.

Глава 13 «*ShapeEditor – выбор и правка геообъектов*» завершает реализацию системы ShapeEditor добавлением подпрограмм, которые предлагают пользователю функционал для выбора и редактирования геообъектов внутри импортированного файла фигур. Он включает в себя создание специального сервера сборных цифровых карт и использование картографической библиотеки OpenLayers на JavaScript для визуализации геоданных на экране компьютера и взаимодействия с ними.

Что требуется для этой книги

Третье издание данной книги было расширено с целью поддержки Python 3, хотя, если хотите, вы можете продолжить пользоваться Python 2. Вам также понадобятся следующие инструменты и библиотеки, которые следует скачать и установить; подробные инструкции даны в соответствующих разделах книги:

- программный пакет **GDAL/OGR**;
- динамическая библиотека **GEOS**;
- библиотека Python **Shapely**;
- динамическая библиотека **Proj**;
- библиотека Python **pyproj**;
- СУБД **PostgreSQL** + программа администрирования **pgAdmin III**;
- геопространственное расширение **PostGIS** для СУБД PostgreSQL;
- адаптер СУБД PostgreSQL **psycopg2** для Python;
- динамическая библиотека **Mapnik**;
- веб-платформа **Django**;
- географический модуль **GeoDjango** для веб-платформы Django;
- картографическая библиотека **OpenLayers** на JavaScript.

Для кого эта книга

Эта книга предназначена для опытных разработчиков на языке Python, которые хотели бы освоить свободно распространяемый инструментарий и методику разработки геопространственных приложений с целью создания своих собственных геоприложений либо интеграции геопространственной технологии в свои существующие программы на Python.

Условные обозначения

В этой книге вы найдете ряд текстовых стилей, которые выделяют различные виды информации. Вот некоторые примеры этих стилей и объяснение их значения.

Фрагменты программного кода в тексте показаны следующим образом: «Набор данных `gdal.Dataset` представляет файл, который содержит данные в растровом формате».

Блок кода выглядит следующим образом:

```
import pyproj

lat1,long1 = (37.8101274,-122.4104622)
lat2,long2 = (37.80237485,-122.405832766082)

geod = pyproj.Geod(ellps="WGS84")
angle1,angle2,distance = geod.inv(long1, lat1, long2, lat2)

print("Расстояние равно {:0.2f} метров".format(distance))
```

Когда требуется привлечь ваше внимание к конкретной части блока кода, соответствующие строки или элементы выделяются полужирным шрифтом:

```
for value in values:
    if value != band.GetNoDataValue():
        try:
            histogram[value] += 1
        except KeyError:
            histogram[value] = 1
```

Любой ввод команды из командной строки или вывод результатов их вычисления оформляется следующим образом:

```
% python3 calcBoundingBoxes.py
Afghanistan (AFG) lat=29.4061..38.4721, long=60.5042..74.9157
Albania (ALB) lat=39.6447..42.6619, long=19.2825..21.0542
Algeria (DZA) lat=18.9764..37.0914, long=-8.6672..11.9865
```

Новые термины и **важные слова** показаны полужирным шрифтом. Слова, которые вы видите на экране, например в меню или диалоговых окнах, выглядят в тексте следующим образом: «Нажмите на гиперссылке **Download Domestic Names**, чтобы скачать национальные географические названия».


Предупреждения или важные примечания появляются в этом поле.


Подсказки и приемы появляются тут.


Дополнения к тексту оригинала книги.

Отзывы читателей

Отзывы наших читателей всегда приветствуются. Сообщите нам, что вы думаете об этой книге – что вам понравилось, а что нет. Обратная связь с читателями для нас очень важна, поскольку она помогает нам формировать названия книг, из которых вы действительно получите максимум полезного.

Отзыв по общим вопросам можно отправить по адресу feedback@packtpub.com, упомянув заголовок книги в теме вашего электронного сообщения.

Если речь о теме, в которой у вас есть экспертные знания, и вы интересуетесь написанием либо содействием в написании книги, то обратитесь к нашему перечню авторов по адресу www.packtpub.com/authors.

Служба поддержки

Теперь, когда вы являетесь довольным владельцем книги издательства Packt, мы предложим вам ряд возможностей с целью помочь вам получать максимум от своей покупки.

Скачивание исходного кода примеров

Вы можете скачать файлы с кодом примеров из вашего аккаунта по адресу http://www.packtpub.com/ для всех книг издательства Packt Publishing, которые вы приобрели. Если вы купили эту книгу в другом месте, то можно посетить http://www.packtpub.com/support и зарегистрироваться там, чтобы получить файлы прямо по электронной почте.

Загрузить файлы с примерами программного кода можно, выполнив следующие шаги:

1. Войдите на наш веб-сайт или зарегистрируйтесь там, используя ваш адрес электронной почты и пароль.
2. Наведите указатель мыши на вкладку **SUPPORT** вверху страницы.
3. Щелкните по разделу **Code Downloads & Errata**, посвященному примерам программного кода и опечаткам.
4. Введите название книги в **Поле поиска**.
5. Выберите книгу, для которой вы хотели бы скачать файлы с примерами.
6. Из раскрывающего меню выберите место, где вы купили эту книгу.
7. Щелкните по **Code Download**, чтобы скачать примеры.

Кроме того, файлы примеров можно скачать, нажав на кнопку **Code Files** на странице книги на веб-сайте издательства Packt Publishing. Доступ к этой странице можно получить, введя наименование книги в **Поле поиска**. Обращаем внимание, что для этого вы должны войти в ваш аккаунт в издательстве Packt.

Скачав файл, пожалуйста, убедитесь, что вы разархивировали или извлекли папку, воспользовавшись последней версией указанных ниже архиваторов:

- WinRAR / 7-Zip для Windows;
- Zipeg / iZip / UnRarX для Mac;
- 7-Zip / PeaZip для Linux.

Помимо этого, пакет примеров программного кода, прилагаемый к данной книге, размещен на GitHub по адресу https://github.com/PacktPublishing/Python-Geospatial-Development-Third-Edition. Мы также располагаем другими пакетами примеров программного кода, которые можно выбрать из нашего богатого каталога книг и видео, предлагаемого на странице https://github.com/PacktPublishing/. Можете убедиться сами!

Опечатки

Несмотря на то что мы приняли все меры, чтобы гарантировать корректность нашего информационного материала, ошибки действительно случаются. Если вы найдете ошибку в одной из наших книг, возможно, в тексте или коде, то мы будем благодарны, если вы сообщите об этом нам. Поступая так, вы можете уберечь других читателей от разочарования и помочь нам улучшать последующие версии этой книги. Если вы найдете какие-либо опечатки, пожалуйста, сообщите о них, посетив страницу http://www.packtpub.com/submit-errata, выбрав вашу книгу, нажав на ссылку формы **errata submission form** для предоставления информации об опечатке и введя ваши данные об опечатках. Как только ваши данные будут проверены, сообщение будет принято к рассмотрению, и данные об опечатке будут загружены на наш веб-сайт или добавлены к любому списку из существующих списков опечаток в разделе Errata под соответствующим заголовком.

Чтобы просмотреть ранее предоставленные опечатки, перейдите на страницу https://www.packtpub.com/books/content/support и в поле поиска введите название книги. Запрошенная информация появится под разделом Errata.

Нарушение авторских прав

Пиратство защищенного авторским правом материала в Интернете является хронической проблемой во всех средствах массовой информации. В издательстве Packt мы очень серьезно относимся к защите нашего авторского права и лицензий. Если вы сталкиваетесь с какими-либо недопустимыми копиями наших работ в какой-либо форме в Интернете, пожалуйста, просим вас незамедлительно предоставить нам адрес размещения или название веб-сайта, чтобы мы могли добиваться правовой защиты.

Пожалуйста, свяжитесь с нами по электронному адресу copyright@packtpub.com со ссылкой на предполагаемый пиратский материал.

Мы ценим вашу помощь в защите наших авторов и в наших усилиях предоставлять вам ценный информационный материал.

Вопросы

Если у вас есть вопрос по каким-либо аспектам этой книги, то вы можете связаться с нами по электронному адресу questions@packtpub.com, и мы приложим все усилия, чтобы решить ваш вопрос.

Комментарий переводчика

Весь материал книги приведен в соответствие с последними действующими версиями библиотек (время перевода книги – август-сентябрь 2016 г.), дополнен свежей информацией и протестирован в среде Windows 10 и Fedora 24. При тестировании программного кода за основу взят Python версии 3.5.2.

Книга содержит много аббревиатур и технических терминов из разных областей науки. Для удобства большинство аббревиатур кратко определено в сносках, а для некоторых терминов в силу отсутствия единой терминологии приведены соответствующие варианты наименований или пояснения. В конце книги основные термины и аббревиатуры собраны в *глоссарии*.

Поскольку в оригинале книги и, главное, во всех инструментальных средах программирования для отделения целой части числа от дробной используется не запятая, а точка, и чтобы не вносить путаницу при работе с книгой, в переводе оставлена форма записи чисел как есть, без изменений.

Книга может быть интересной широкому кругу специалистов, в том числе начинающим аналитикам данных, преподавателям, студентам, а также всем, кто интересуется геопрограммированием.

Далее приведены особенности установки и работы некоторого используемого программного обеспечения.

Установка и настройка дистрибутива Anaconda

Anaconda — это полностью свободный дистрибутив Python (предназначенный в том числе для коммерческого использования и повторного распространения). Он содержит более 400 самых популярных библиотек Python для вычислений в области естественных наук, математики, инженерии и анализа данных.

Установка дистрибутива в Windows выполняется стандартным образом после загрузки с сайта бинарного файла дистрибутива. В Linux сначала необходимо скачать установщик — скриптовый файл с расширением .sh для оболочки Bash (https://www.continuum.io/downloads#_unix). На примере Anaconda 4.2.0 (с Python 3.5.2) для 64-разрядных операционных систем команда установки выглядит так: `bash Anaconda3-4.2.0-Linux-x86_64.sh`.

Отметим, что инсталляция вступит в силу после того, как вы закроете и заново откроете окно терминала.

Для обновления дистрибутива Anaconda следует набрать в терминале следующую команду:

```
conda update conda
```

Удаление дистрибутива Anaconda:

```
rm -rf ~/anaconda
```

Более подробная информация по деинсталляции Anaconda содержится по указанной выше ссылке.

Чтобы удостовериться, что дистрибутив Anaconda установлен успешно, воспользуйтесь следующей командой:

```
conda --version
```

В результате будет выведен номер установленной версии.

В случае если вы в основном работаете в среде Python 3, но иногда возникает необходимость переключиться в среду Python 2 и использовать ее для работы с библиотеками, которые предназначены только для Python 2, то Anaconda предлагает функционал создания и активации новой среды, куда можно установить нужную версию языка. В нижеследующей таблице приведено несколько команд, которые можно выполнить прямо в Spyder во встроенном окне командной строки (**Инструменты ⇨ Открыть командную строку**):

`conda create -n py27 python=2.7.12 anaconda`	Установить другую версию Python (2.7.12) в новую среду с именем py27 (имя может быть любым)
`conda info --envs`	Проверить, что среда с именем py27 установлена (команда выводит список всех сред, при этом активная среда выделяется знаком *)
`source activate py27` (Linux, OS X), `activate py27` (Windows)	Переключиться в среду py27 с другой версией Python (команда activate добавляет в начало строки путь к среде py27)
`deactivate`	Деактивировать текущую среду и вернуться к среде по умолчанию
`python --version`	Проверить, что среда использует нужную версию Python
`conda remove -n py27 --all`	Удалить среду py27 (после выполнения команды выполнить conda clean --lock)
`conda install --name py27 scypy`	Установить библиотеку scypy в среду py27
`conda remove --name py27 scypy`	Удалить библиотеку scypy в среде py27
`conda clean --lock`	Очистить блокировку, если произошел сбой при установке среды (иногда сперва требуется удалить conda в Диспетчере задач)

Установка и настройка инструментальной среды Spyder

Spyder (в октябре 2016 г. появилась версия 3.0.0 с локализованным на русский язык графическим интерфейсом) — это инструментальная среда для научных вычислений для языка Python (Scientific PYthon Development EnviRonment) для Windows, Mac OS X и Linux. Это простая, легковесная и бесплатная интерактив-

ная среда разработки на Python, которая предлагает функционал, аналогичный среде разработки на MATLAB, включая готовые к использованию виджеты `PyQt5` и `PySide`: редактор исходного кода, редактор массивов данных `NumPy`, редактор словарей, консоли Python и IPython и многое другое.

Для пользователей Windows хорошая новость заключается в том, что Spyder уже включен в состав дистрибутива Anaconda Python, и исполнимый файл находится в папке `C:\Users\[ИМЯ_ПОЛЬЗОВАТЕЛЯ\Anaconda3\Scripts\spyder.exe`. Чтобы установить среду Spyder без установки Anaconda, необходимо скачать соответствующий файл whl (https://pypi.python.org/pypi/spyder) и установить, как объяснено в разделе *Установка библиотек Python из whl-файла* ниже. Чтобы установить среду Spyder в Ubuntu Linux, используя официальный менеджер пакетов, нужна всего одна команда:

```
sudo apt-get install spyder3
```

Чтобы установить с использованием менеджера пакетов pip:

```
sudo apt-get install python-qt5 python-sphinx
sudo pip3 install spyder
```

И чтобы обновить:

```
sudo pip3 install -U spyder
```

Установка среды Spyder в Fedora 24 приведена ниже в разделе «*Подготовка среды Python 3 в ОС Fedora 24*». Во всех вышеперечисленных случаях речь идет о версии Spyder для Python 3 (на момент инсталляции это был Python 3.5.2). Чтобы установить версию Spyder для Python 2, нужно просто поменять `spyder3` на `spyder`.

Настройка среды Spyder с Python 3 для работы с Python 2

При инсталляции дистрибутива Anacona3 по умолчанию базовым является интерпретатор Python 3 (в данном случае версии 3.5.2). В случае если нужно на время переключиться в режим работы в интерпретаторе Python версии 2 (в данном случае версии 2.7.12), существуют два варианта: дополнительно инсталлировать дистрибутив Anaconda для Python 2 либо выполнить небольшую настройку инструментальной среды Spyder, по умолчанию работающей на основе Python 3.

Для такой настройки нужно в основном меню выбрать **Инструменты** и затем **Параметры**. В открывшемся окне **Параметры** выбрать **Интерпретатор Python**. В разделе **Интерпретатор Python** с двумя переключателями установить переключатель **Использовать следующий интерпретатор Python** и затем нажать кнопку напротив текстового поля, чтобы выбрать путь к интерпретатору Python 2.7.12, который находится в папке C:\Users*[имя_пользователя]*\Anaconda3\envs\py27, где py27 — это имя среды, созданной вами для Python 2.7.12 (см. раздел «*Настройка дистрибутива Anaconda*» ранее). Если открыть новую консоль (**Консоли ⇨ Открыть консоль Python**), то в строке приветствия будет указана нужная версия Python.

Теперь в этой консоли можно набирать команды и вставлять целые программы, однако запускать программы из рабочего окна редактора Spyder не получится. Для

сценария, работающего с Python 2, требуется еще одна, и последняя, настройка. В основном меню нужно выбрать **Запуск**, затем **Настроить** и в разделе **Консоль** выбрать второй переключатель **Выполнить во внешнем системном терминале** (то есть в новой специально выделенной консоли). Теперь этот конкретный спенарий будет выполняться в консоли с Python 2.

Закончив работу с интерпретатором Python 2.7.12, следует не забыть вернуться к исходному интерпретатору Python 3.5.2. Для этого нужно в окне **Интерпретатор Python** просто установить переключатель **По умолчанию** (тот же, что и для Spyder).

Напомним, что месторасположение базового интерпретатора следующее: `C:\Users\[ИМЯ_ПОЛЬЗОВАТЕЛЯ]\Anaconda3\python.exe`. Разумеется, такой режим работы вносит некоторые неудобства, но, по крайней мере, он не такой громоздкий, как установка еще одной версии Anaconda 4.2.0, но уже с Python 2.7.12 в качестве базового интерпретатора.

Установка библиотек Python из whl-файла

Библиотеки для Python можно разрабатывать не только на чистом Python. Довольно часто библиотеки пишутся на C (динамические библиотеки), и для них пишется обертка Python, или же библиотека пишется на Python, но для оптимизации узких мест часть кода пишется на C. Такие библиотеки получаются очень быстрыми, однако библиотеки с вкраплениями кода на C программисту на Python тяжелее установить ввиду банального отсутствия соответствующих знаний либо необходимых компонентов и настроек в рабочей среде (в особенности в Windows). Для решения описанных проблем разработан специальный формат (файлы с расширением .whl) для распространения библиотек, который содержит заранее скомпилированную версию библиотеки со всеми ее зависимостями. Формат whl поддерживается всеми основными платформами (Mac OS X, Linux, Windows).

Установка производится с помощью менеджера пакетов pip. В отличие от обычной установки командой `pip3 install <имя_пакета>`, вместо имени пакета указывается путь к whl-файлу `pip3 install <путь/к/whl_файлу>`. Например:

```
pip3 install C:\temp\networkx-1.11-py2.py3-none-any.whl
```

Откройте окно командной строки и при помощи команды `cd` перейдите в каталог, где размещен ваш whl-файл. В Anaconda по умолчанию подобные файлы лежат в каталоге Scripts. Просто скопируйте туда ваш whl-файл. В этих случаях полный путь указывать не понадобится. Например:

```
pip3 install networkx-1.11-py2.py3-none-any.whl
```

При выборе пакета важно, чтобы разрядность устанавливаемой библиотеки и разрядность интерпретатора совпадали. Пользователи Windows могут брать whl-файлы с сайта http://www.lfd.uci.edu/~gohlke/pythonlibs/. Библиотеки там постоянно обновляются и в архиве содержатся все, какие только могут понадобиться.

Подготовка среды Python 3 в ОС Fedora 24

Действие	Команда
Обновление ОС Fedora 24 (http://losst.ru/nastrojka-fedora-24-posle-ustanovki)	`$ dnf update` `$ dnf upgrade`

Интерпретатор языка Python 3 и менеджер пакетов	`$ dnf install python3`
	`$ dnf -y install python3-tools`
	`$ pip3 install --upgrade pip`

Оболочка Python IDLE3	`$ dnf install idle3-tools`
Основные научные библиотеки	`$ dnf install python3-numpy python3-matplotlib python3-scipy python3-pandas python3-sympy python3-scikit-learn`
Записные книжки Jupyter	`$ pip3 install jupyter --upgrade`

Инструментальная среда Spyder (файл whl в каталоге пакетов Python https://pypi.python.org/pypi/spyder)	`$ dnf install python3-spyder`
	или
	`$ dnf install PyQt5`
	`(pip3 install PyQt5)`
	`$ sudo dnf install python3-devel`
	`$ pip3 install psutil`
	`$ sudo pip3 install /путь/к/файлу/spyder-3.0.0-py3-none-any.whl`

СУБД PostgreSQL, геопространственного расширения PostGIS, адаптера базы данных psycopg2 и администратора БД pgadmin3	`$ sudo dnf list postgresql\*`
	`$ sudo dnf install postgresql postgresql-contrib post-gresql-server postgis`
	`$ sudo dnf install python3-psycopg2`
	`$ sudo dnf install pgadmin3`

Установка геопространственных библиотек в ОС Fedora 24

Библиотека GDAL

Скачать файл RPM из каталога библиотек RPM https://www.rpmfind.net/linux/rpm2html/search.php?query=gdal-python3:

```
$ dnf install /путь/к/файлу/gdal-python3-2.1.1-1.fc26.x86_64.rpm
```

или

```
$ dnf install gdal gdal-devel
```

Библиотека pyproj

Скачать файл RPM из каталога библиотек RPM https://www.rpmfind.net/linux/rpm2html/search.php?query=python3-pyproj&submit=Search+...&system=&arch=:

```
$ dnf install /путь/к/файлу/python3-pyproj-1.9.5.1-2.fc24.x86_64.rpm
```

Библиотека Shapely

Скачать файл RPM из каталога библиотек RPM https://www.rpmfind.net/linux/rpm2html/search.php?query=python3-shapely&submit=Search+...&system=&arch=:

```
$ dnf install /путь/к/файлу/python3-shapely-1.5.16-1.fc24.x86_64.rpm
```

Библиотека Mapnik

Автор книги будет использовать библиотеку Mapnik в главах 3, 7, 8, 9 и 13. Стоит отметить, что версия 3 библиотеки Mapnik, являясь мощным средством картографирования, пока работает только в Linux и Mac OS X, и разработка бинарников для Windows стоит в ближайших планах разработчиков. В связи с этим примеры программ в этих главах тестировались в ОС Fedora 24. Кроме того, предыдущие издания этой книги вышли в 2010 и 2013 гг., и с тех пор парадигма написания кода с использованием библиотеки Mapnik поменялась с процедурного на декларативный на основе XML и Geojson. Тем не менее в книге авторские примеры программного кода оставлены без изменений, а альтернативные реализации на XML приведены в комментариях и в прилагаемых к книге примерах.

Скачать файл RPM из каталога библиотек RPM https://www.rpmfind.net/linux/rpm2html/search.php?query=mapnik&submit=Search+...&system=&arch=:

```
$ dnf install /путь/к/файлу/mapnik-3.0.10-3.fc24.x86_64.rpm
$ dnf install /путь/к/файлу/python3-mapnik-0.1-7.20160202git1cb6851.fc24.x86_64.rpm
```

Запись справочной информации по библиотеке mapnik в HTML-файл:

```
$ python3
>>> import pydoc
>>> import mapnik
>>> pydoc.writedoc("mapnik")
>>> exit()
```

Протокол настройки СУБД PostgreSQL в ОС Fedora 24

```
[root@localhost ~]$ sudo postgresql-setup --initdb --unit postgresql
[root@localhost ~]$ sudo systemctl start postgresql
[root@localhost ~]$ sudo -i -u postgres
-bash-4.3$ psql
psql (9.5.4)
Введите "help", чтобы получить справку.
postgres=# alter user postgres password 'geo';
ALTER ROLE
postgres=# \c
Вы подключены к базе данных "postgres" как пользователь "postgres".
postgres=# create table test (id int);
CREATE TABLE
postgres=# \d
          Список отношений
 Схема  | Имя  |   Тип   | Владелец
--------+------+---------+----------
 public | test | таблица | postgres
(1 строка)
postgres=# drop table test;
DROP TABLE
postgres=# \d
```

```
Отношения не найдены.
postgres=# \q
-bash-4.3$ psql -c "GRANT ALL PRIVILEGES ON DATABASE postgres TO postgres"
GRANT
-bash-4.3$ psql -d postgres -c "CREATE EXTENSION postgis;"
CREATE EXTENSION
-bash-4.3$ exit
logout
[root@localhost ~]$ sudo systemctl stop postgresql
[root@localhost /]# nano /var/lib/pgsql/data/pg_hba.conf
  Прим. Отредактировать 2 строки в pg_hba.conf и сохранить файл:
  local all all peer  поменять на local all all trust
  host  all all 127.0.0.1/32 ident поменять на host  all all 127.0.0.1/32 md5
[root@localhost ~]$ sudo systemctl start postgresql
```

Протокол установки и настройки Django в ОС Fedora 24

```
[user@localhost /]$ sudo pip3 install django
[user@localhost /]$ cd путь/к/djex             # djex - это пример приложения
[user@localhost djex]$ django-admin startproject example
[user@localhost djex]$ ls
example
[user@localhost djex]$ cd example
[user@localhost example]$ python3 manage.py startapp hello
    (обновить settings.py)
    (обновить models.py)
[user@localhost example]$ python3 manage.py makemigrations hello
Migrations for 'hello':
  hello/migrations/0001_initial.py:
    - Create model Counter
[user@localhost example]$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: auth, contenttypes, hello, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying hello.0001_initial... OK
  Applying sessions.0001_initial... OK
    (обновить examples/urls.py)
    (обновить hello/views.py)
    (создать hello/templates/say_hello.html)
```

```
[user@localhost example]$ python3 manage.py runserver
Performing system checks...
System check identified no issues (0 silenced).
October 07, 2016 - 07:41:34
Django version 1.10.2, using settings 'example.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

(набрать в браузере http://127.0.0.1:8000)

Глава 1

Разработка геопрограмм на Python

Эта глава предлагает обзор языка программирования Python и процесса разработки геопространственных приложений. Отметим, что данная глава не является учебным руководством по использованию языка Python; язык Python прост в изучении, однако связанные с этим подробности выходят за рамки данной книги.

В этой главе будут рассмотрены следующие темы:

- что из себя представляет язык Python и чем он отличается от других языков программирования;
- каким образом стандартная библиотека Python и его каталог пакетов делают этот язык еще более мощным;
- что обозначают термины **геопространственные данные** и **разработка геопространственных приложений**;
- обзор процесса получения доступа к геопространственным данным, управления ими и их отображения на экране компьютера, и каким образом он осуществляется;
- несколько основных сфер приложения разработок в области геопространственного программирования;
- некоторые современные тенденции развития процесса разработки геопространственного программного обеспечения.

Python

Python (http://python.org) – это современный, высокоуровневый язык программирования, подходящий для широкого спектра задач программирования. Он часто используется в качестве языка сценариев для автоматизации и упрощения задач на уровне операционной системы и одинаково подходит для создания больших и сложных программ. Python использовался при написании веб-систем, настольных приложений, игр, научных программ и даже утилит и других высокоуровневых элементов различных операционных систем.

Python поддерживает широкий спектр идиом программирования от прямого процедурного программирования до объектно-ориентированного и функционального программирования.

Python иногда критикуют за то, что он – интерпретируемый и бывает медленным, по сравнению с компилирующими языками, такими как C. Однако использование байткодовой компиляции и тот факт, что подавляющая часть тяжелой работы выполняется программными библиотеками, означают, что производительность Python часто бывает удивительно хороша – к тому же имеется множество приемов, которые позволяют улучшить производительность программ, если это необходимо.

Версии интерпретатора Python с открытым исходным кодом находятся в свободном доступе для всех основных операционных систем. Python идеально подходит для всех видов программирования, от оперативных разовых сценариев до создания огромных и сложных систем. Python может даже выполняться в интерактивном режиме (в командной строке), позволяя вводить одноразовые команды и короткие программы и сразу получать результаты. Это идеальный сценарий для выполнения оперативных вычислений или выяснения того, как работает конкретная библиотека.

Первое, на что разработчик на Python обращает внимание, по сравнению с другими языками, такими как Java или C++, – это выразительность языка: то, что на Java потребует 20–30 строк программного кода, на Python часто занимает менее 10 строк. Например, предположим, необходимо напечатать сортированный список слов, которые встречаются в конкретной части текста. В Python это достигается легко:

```
words = set(text.split())
for word in sorted(words):
    print(word)
```

Реализация этого вида задачи на других языках часто, к удивлению, представляет трудности.

В то время как непосредственно сам язык Python позволяет программировать быстро и легко, давая возможность фокусироваться на поставленной задаче, стандартная библиотека Python делает программирование еще более эффективным. Эта библиотека облегчает выполнение таких процедур, как конвертирование значений даты и времени, обработка строковых значений, загрузка данных с вебсайтов, выполнение сложных математических вычислений, работа с электронными письмами, кодирование и декодирование данных, разбор XML, шифрование данных, файловые операции, сжатие и распаковка файлов, работа с базами данных – и далее по списку. То, что можно сделать с использованием стандартной библиотеки Python, воистину впечатляет.

Наряду со встроенными в стандартную библиотеку Python модулями можно легко загрузить и установить пользовательские модули, которые могут быть написаны на Python или C. Каталог библиотек языка Python (http://pypi.python.org) предлагает тысячи дополнительных модулей, которые можно загрузить

и установить. И если этого недостаточно, то множество других систем предлагает привязки Python[1], которые обеспечивают к ним доступ непосредственно из ваших программ. Кстати, в этой книге мы будем интенсивно использовать питоновские привязки.

Python – это во многих отношениях идеальный язык программирования. Познакомившись с языком и пару раз использовав его в деле, вы осознаете, насколько он легок при написании программ для решения разнообразных задач. Не опасаясь похоронить себя в болоте определений типов и среди низкоуровневых методов обработки строк, можно просто сконцентрироваться на том, чего вы хотите достигнуть. И в итоге придете к тому, что начнете думать непосредственно в терминах питоновского программного кода. Программирование на Python прямолинейное, эффективное и, не побоюсь этого слова, увлекательное.

Python 3

Сегодня используются две главные разновидности Python: Python версии 2.x существует уже в течение многих лет и все еще широко используется, в то время как Python версии 3.x, которая не является обратно несовместимой с Python 2, становится все более популярным, учитывая, что эта версия Python взята за основу для дальнейшего развития.

Один из главных факторов, сдерживающих повсеместное принятие Python 3, – это отсутствие поддержки сторонних библиотек. Эта проблема было особенно острой относительно библиотек Python, использующихся для разработки геопространственных приложений, поскольку они часто зависят от отдельных разработчиков или имеют требования, которые не были совместимыми с Python 3 в течение достаточно продолжительного времени. Однако все основные библиотеки, которые используются в этой книге, теперь в равной степени можно выполнять, используя Python 3, и поэтому все примеры программного кода в этой книге преобразованы таким образом, чтобы использовать синтаксис этой версии Python.

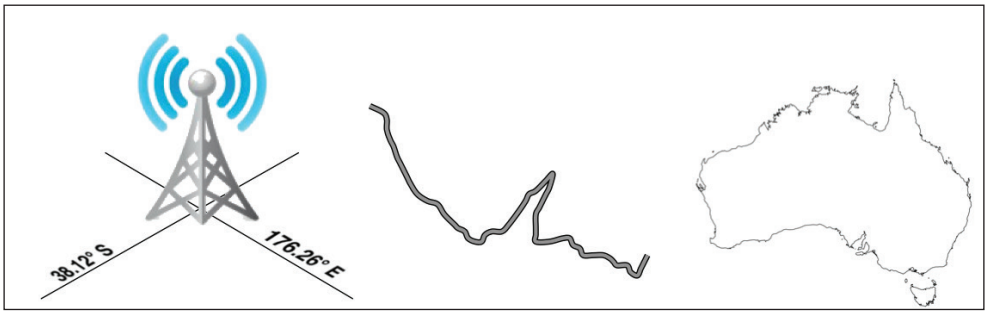
Если ваш компьютер работает под Linux или Mac OS X, то вы можете непосредственно использовать Python 3 со всеми этими библиотеками. Если же ваш компьютер работает под Windows, то совместимость с Python 3 выглядит проблематичнее. В этом случае имеются две возможности: можно попытаться скомпилировать библиотеки самостоятельно, чтобы получить возможность работать с Python 3, либо вернуться к использованию Python 2 и внести изменения в примеры программного кода, где это требуется. К счастью, различия в синтаксисе между Python 2 и Python 3 предельно простые и поэтому потребуют незначительных изменений, если вы действительно примете решение использовать Python 2.x, а не Python 3.x.

---

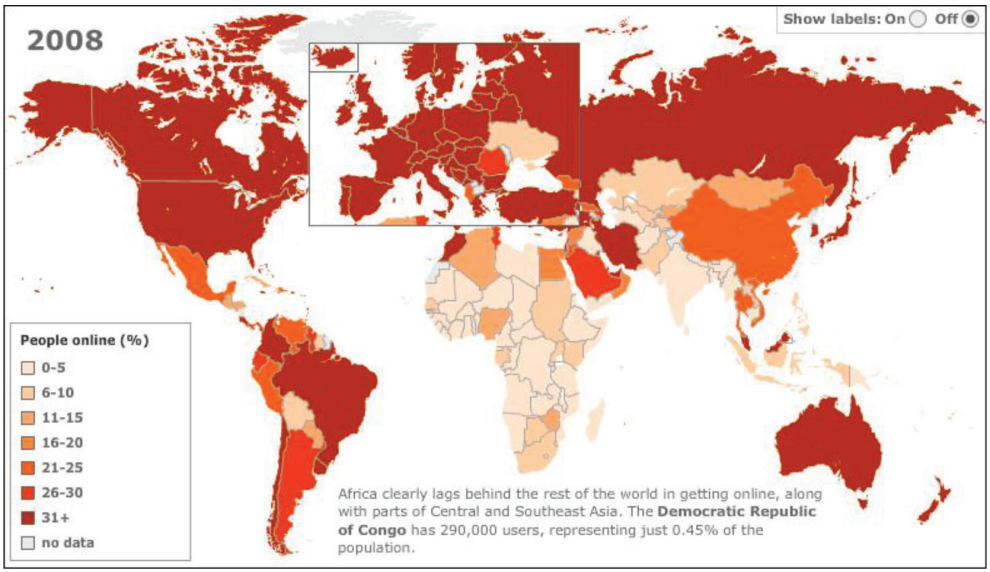
[1] Привязка (binding) – программный код, благодаря которому программа получает доступ к методам и свойствам программного объекта. В сущности, это оберточный или промежуточный слой между средой разработки на Python и прикладным программным интерфейсом (API), который предоставляется веб-сервисом, динамической библиотекой или СУБД. – *Прим. перев.*

Разработка геопространственных программ

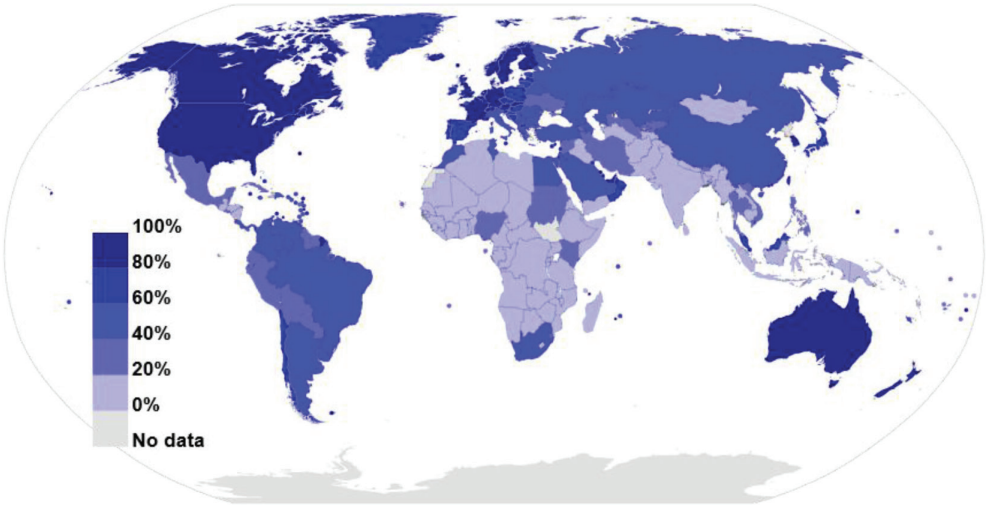
Термин «**геопространственный**» относится к отысканию информации, которая расположена на земной поверхности. Например, она может состоять из географического положения вышек сотовой связи, географической формы дороги или контура страны:



Геопространственные данные часто связывают некую порцию информации с конкретным географическим положением. Например, ниже приведена интерактивная карта с веб-сайта http://www.bbc.co.uk, которая показывает процент населения в каждой стране с доступом в Интернет в 2008 г.



Аналогичная карта из Википедии с данными за 2012 г. (https://en.wikipedia.org/wiki/Global_Internet_usage):



Геопространственные данные, или просто *геоданные*, как правило, связывают некоторое количество информации с конкретным географическим положением. Например, следующая ниже карта, взятая с http://opendata.zeit.de/nuclear-reactors-usa, показывает число людей, проживающих в пределах 50 миль от ядерного реактора в границах восточной части США:

Термин «**разработка геопространственных приложений**» обозначает процесс написания компьютерных программ, которые способны получать доступ к такому типу информации, управлять ею и ее визуализировать.

Внутренне геоданные представлены в виде серии **координат**, часто в виде значений широты и долготы. Кроме того, нередко также присутствуют дополнительные **атрибуты**, такие как температура, тип почвы, высота или название ориентира. Одиночный набор геоданных может описывать до нескольких тысяч (или даже миллионов) точек данных. Например, следующий ниже контур Новой Зеландии состоит почти из 12 000 отдельных точек данных:

По причине того, что в работе задействовано такое большое количество данных, геопространственную информацию принято хранить в базах данных. Значительная часть этой книги будет посвящена приемам хранения вашей геоинформации в базе данных и получения к ней доступа эффективным образом.

Геоданные поступают в самых различных формах. Разные поставщики **географических информационных систем** (ГИС), или, точнее, геопространственных информационных систем, за последние годы создали свои собственные форматы файлов, а разного рода организации установили свои собственные стандарты. И поэтому, чтобы прочитать файлы нужного формата, необходимо при импорте геоданных в вашу базу данных пользоваться библиотекой Python.

К сожалению, не все точки геоданных совместимы. Точно так же, как расстояние величиной 2.8 единицы может иметь совсем разные значения в зависимости от того, используете вы километры или мили, конкретное значение координаты может указывать на любое число точек на искривленной поверхности Земли, в зависимости от того, какая **картографическая проекция** используется.

Картографическая проекция – это метод представления земной поверхности в двух измерениях. Мы рассмотрим проекции более подробно в *главе 2 «Геоинформационные системы»*, но на данный момент просто следует иметь в виду, что с любым элементом геоданных связана проекция. Чтобы сравнить или совместить два набора геоданных, часто необходимо конвертировать данные из одной проекции в другую.

Значения широты и долготы иногда упоминаются как неспроецированные, или географические, координаты. Мы узнаем подробнее об этом в следующей главе.

В дополнение к прозаическим задачам импорта геоданных из различных внешних форматов файлов и перевода данных из одной проекции в другую геоданными можно управлять, чтобы решать разного рода содержательные задачи. Очевидные примеры включают в себя задачу вычисления расстояния между двумя точками, задачу вычисления длины дороги или нахождения всех точек данных, находящихся в пределах заданного радиуса от выбранной точки. Чтобы решать все эти и другие задачи, мы будем пользоваться библиотеками Python.

И наконец, сами по себе геоданные не очень интересны. Длинный список координат практически ни о чем не говорит; понять их смысл можно, только когда эти числа используются для создания изображений. Составление карт, размещение на них точек данных и предоставление пользователям возможности с ними взаимодействовать – все эти задачи являются важными аспектами процесса разработки геоприложений.

Мы рассмотрим их все в более поздних главах.

Сферы применения геоприложений

Вкратце рассмотрим некоторые наиболее распространенные задачи, решаемые геоприложениями, с которыми вы, по всей видимости, уже встречались.

Анализ геоданных

Предположим, у вас есть база данных, которая содержит набор геоданных по г. Сан-Франциско. Эта база данных может состоять из географических объектов, дорожной сети, географических положений видных строений и других антропогенных объектов, таких как мосты, аэропорты и т. д.

Такая база данных может быть ценным ресурсом для ответа на различные вопросы, такие как:

- Какая самая длинная дорога в г. Саусалито?

- Сколько мостов находится в г. Окленд?
- Какова общая площадь городского парка «Золотые ворота» (Golden Gate Park) в г. Сан-Франциско?
- Каково расстояние от Пирса 39 до Башни Койт[1]?

Многие из этих типов задач можно решить при помощи таких инструментов, как расширение PostGIS со встроенной поддержкой пространственных данных. Например, чтобы рассчитать общую площадь городского парка «Золотые ворота», можно воспользоваться следующим запросом SQL:

```
select ST_Area(geometry) from features
where name = "Golden Gate Park";
```

Чтобы рассчитать расстояние между двумя географическими положениями, их сначала необходимо геокодировать, получив их значения широты и долготы. Это можно выполнить разными методами; один из самых простых состоит в использовании бесплатной веб-службы геокодирования, как, например, эта:

```
http://nominatim.openstreetmap.org/search?format=json&q=Pier 39,San Francisco, CA
```

Для Пирса 39 она возвращает (среди прочего) значения широты `37.8101274` и долготы `-122.4104622`.

Приведенные значения широты и долготы исчисляются десятичными градусами. Если вы не знаете, что это такое, то не переживайте; мы займемся ими в главе 2 «Геоинформационные системы».

Аналогичным образом можно установить географичесое положение Башни Койт, используя для этого следующий ниже запрос:

```
http://nominatim.openstreetmap.org/search?format=json&q=Coit Tower, San Francisco, CA
```

Он возвращает значение широты `37.80237485` и долготы `-122.405832766082`.

Располагая координатами двух нужных географических точек, мы можем рассчитать между ними расстояние, используя для этих целей библиотеку Python `pyproj`:

Если вы захотите выполнить этот пример, то сначала необходимо инсталлировать библиотеку `pyproj`. Мы рассмотрим инсталляцию библиотек в *главе 3* «*Библиотеки Python для геопрограммирования*».

```
import pyproj

lat1,long1 = (37.8101274,-122.4104622)
lat2,long2 = (37.80237485,-122.405832766082)

geod = pyproj.Geod(ellps="WGS84")
```

---

[1] Башня Койт (Coit Tower) – башня-мемориал, расположенная в парке пионеров (Pioneer Park) на вершине Телеграфного холма (Telegraph Hill). Со смотровой площадки башни открывается уникальный вид на Сан-Франциско. Благодаря этому факту башня является излюбленным местом посещения туристами. – *По материалам Википедии.*

```
angle1,angle2,distance = geod.inv(long1, lat1, long2, lat2)

print("Расстояние составляет {:0.2f} метров".format(distance))
```

В результате будет выведено расстояние между двумя точками:

Расстояние составляет 952.17 метра.

На данном этапе вам не стоит обращать внимания на обозначение `WGS84`; его смысл будет рассмотрен в *главе 2* «*Геоинформационные системы*».

Разумеется, в обычных условиях такого рода анализ не выполняется на разовой основе, как тут, – гораздо чаще на Python создают программу, которая отвечает на подобные вопросы для *любого* нужного набора данных. Можно, к примеру, создать веб-приложение, которое выводит на экран меню доступных численных расчетов. Одна из опций в этом меню могла бы выполнять расчеты расстояния между двумя точками; при выборе этой опции веб-приложение предложит пользователю ввести географические положения этих двух точек, попытается их геокодировать, вызвав для этого соответствующую веб-службу (и выведет на экран компьютера сообщение об ошибке, если геокодировать географическое положение не получилось), затем рассчитает между двумя точками расстояние, используя библиотеку `pyproj`, и, наконец, выведет пользователю результаты.

С другой стороны, если у вас есть база данных, которая содержит соответствующие геоданные, то можно предоставить пользователю возможность выбрать географические положения этих двух точек из базы данных, вместо того чтобы заставлять пользователя в произвольном режиме набирать наименования географических положений или уличные адреса на клавиатуре.

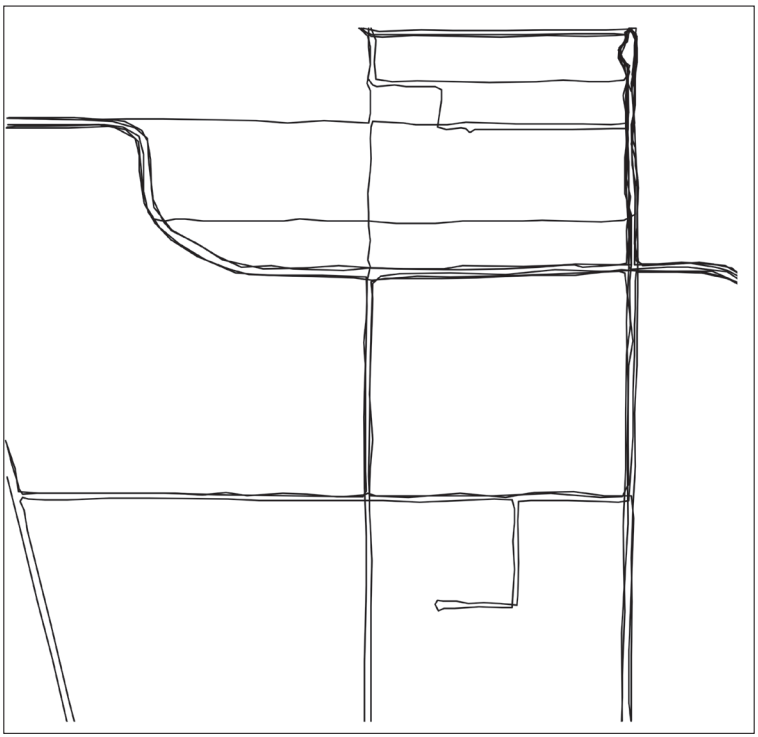
Каким бы ни было ваше решение о том, как структурировать ввод данных, выполнение подобного рода расчетов будет главной составной частью вашего геопространственного приложения.

Визуализация геоданных

Предположим, вам надо посмотреть, какие районы города обычно покрываются услугами такси в течение среднего рабочего дня. В автомобиль такси можно установить записывающее устройство GPS и оставить его там на несколько дней, чтобы оно записывало его географические положения. В результате получится серия временных меток, а также значения широт и долгот, как показано ниже:

```
03-21 9:15:23  -38.16614499  176.2336626
03-21 9:15:27  -38.16608632  176.2335635
03-21 9:15:34  -38.16604198  176.2334771
03-21 9:15:39  -38.16601507  176.2333958
...
```

Сами по себе эти голые цифры почти ничего не означают. Но когда вы визуализируете эти данные на экране компьютера, цифры начинают приобретать смысл:



Процедура скачивания примеров программного кода подробно описана в *предисловии* к этой книге.

Пакет примеров программного кода, прилагаемый к данной книге, размещен на странице GitHub https://github.com/PacktPublishing/Python-Geospatial-Development-Third-Edition. Мы также располагаем другими пакетами примеров, которые можно выбрать из нашего богатого каталога книг и видео, предлагаемого на странице https://github.com/PacktPublishing/. Можете убедиться сами!

Можно сразу увидеть, что такси имеет тенденцию снова и снова перемещаться вдоль одних и тех же улиц, и если вы наложите эти данные поверх карты города, то увидите, где именно такси находилось: