Содержание

Глава 17

Отправка сообщений электронной почты

В ЭТОЙ ГЛАВЕ...

- Последовательность событий при отправке электронной почты
- Создание приложения электронной почты
- Тестирование приложения электронной почты

В этой главе рассматривается процесс рассылки электронных писем с использованием Python. Вы сможете понять, что происходит в процессе дистанционного общения. Несмотря на то что глава посвящена электронной почте, в ней излагается универсальный материал, который можно применять для решения других задач. Например, при работе с внешними службами часто требуется создавать пакеты того же типа, что и для электронной почты. Таким образом, информация, изложенная в этой главе, поможет вам создавать самые разные коммуникационные приложения.

Электронная почта была фактически смоделирована на основе реальной почты. Первоначально термин “электронная почта” использовался для любого вида передачи электронных документов, и в некоторых приложениях отправитель и получатель должны были находиться в сети одновременно. Это привело к тому, что в Интернете появилась масса невразумительной информации о происхождении и развитии электронной почты. В данной главе электронная почта рассматривается в том виде, в каком она существует сегодня, — как механизм обмена документами различных типов.

Примеры этой главы рассчитаны на использование сервера SMTP (Simple Mail Transfer Protocol — простой протокол передачи почты). Если этот термин вам незнаком, обратитесь к приведенной ниже врезке. Исходный код примеров этой главы находится в файле `BPPD_17_Sending_an_Email.ipynb` (см. введение).

SMTP (ПРОСТОЙ ПРОТОКОЛ ПЕРЕДАЧИ ПОЧТЫ)

Имея дело с электронной почтой, вы будете постоянно встречать упоминание SMTP. Конечно, этот термин звучит довольно загадочно, но разбираться в технических деталях вовсе не обязательно, достаточно лишь знать, для чего применяется данный протокол. С другой стороны, полезно немного разобраться в принципах работы SMTP-сервера, чтобы не воспринимать его на уровне "черного ящика", который принимает письмо от отправителя и пересылает получателю. Рассмотрим отдельные компоненты, образующие аббревиатуру SMTP, начав с конца.

- **Protocol (Протокол).** Это стандартный набор коммуникационных правил. Работа с электронной почтой требует соблюдения общепринятых правил, в противном случае почтовая система станет ненадежной.
- **Mail Transfer (Передача почты).** Электронные документы пересылаются из одного места в другое почти так же, как письма между почтовыми отделениями. В случае электронной почты процесс передачи основан на коротких командах, которые ваше почтовое приложение выдает SMTP-серверу. Например, команда `MAIL FROM` сообщает серверу имя отправителя электронной почты, а команда `RCPT TO` указывает, куда ее отправлять.
- **Simple (Простой).** Означает, что доставки почтовых отправлений выполняется с наименьшим количеством возможных усилий. Это позволяет повысить надежность процесса в целом.

Если вы изучите правила передачи информации, то обнаружите, что они вовсе не такие простые. Например, RFC1123 — это стандарт, который определяет, как должны работать интернет-хосты (`http://www.faqs.org/rfcs/rfc1123.html`). Подобные правила лежат в основе многих интернет-технологий, что объясняет, почему большинство из них работает примерно одинаково (несмотря на внешние различия).

Другой стандарт, RFC2821, описывает, каким образом в SMTP реализуются правила, описанные в RFC1123 (`http://www.faqs.org/rfcs/rfc2821.html`). Иными словами, существует множество всевозможных правил, написанных на языке, который понятен лишь узкому кругу специалистов (и то не всегда). Если хотите получить более внятное объяснение принципов работы электронной почты, обратитесь по адресу `https://ru.bmstu.wiki/SMTP_(Simple_Mail_Transfer_Protocol)`.

Что происходит при отправке электронной почты

Электронная почта стала настолько надежной и обыденной, что большинство людей даже не представляют, какое это чудо. В сущности, то же самое можно сказать и о реальной почтовой службе. Если задуматься, то вероятность того, что одно конкретное письмо будет доставлено именно туда, куда нужно, кажется мизерной. Но на самом деле это не так. И электронная почта, и ее физический эквивалент имеют несколько общих особенностей, которые резко повышают их надежность. В следующих разделах объясняется, что происходит, когда вы пишете электронное письмо, щелкаете на кнопке Отправить и письмо доставляется получателю. Вы удивитесь тому, сколько там всего интересного.

Структура электронного письма

Лучше всего рассматривать электронную почту как ее физический аналог. Когда вы пишете письмо, вам нужны как минимум две бумажки: первая — само письмо, вторая — конверт. Если почтовая служба работает честно, то содержимое письма никогда не просматривается никем, кроме получателя. То же самое можно сказать и об электронной почте. Она содержит следующие компоненты.

- **Сообщение**. Содержимое электронного письма, которое фактически состоит из двух частей.
  - *Заголовок*. Включает тему, список получателей и другие атрибуты, такие как срочность сообщения.
  - *Тело*. Фактическое сообщение. Может быть представлено в формате простого текста или HTML и содержать вложенные документы.
- **Конверт**. Контейнер для сообщения. Содержит информацию об отправителе и получателе так же, как и конверт, пересылаемый обычной почтой. Только на него не клеится почтовая марка.

При работе с электронной почтой вы создаете сообщение с помощью почтового приложения. В процессе конфигурирования приложения вы также задаете информацию о своей учетной записи. После щелчка на кнопке Отправить выполняются следующие действия.

1. Приложение электронной почты упаковывает ваше сообщение, содержащее заголовок, в конверт, который включает в себя информацию как об отправителе, так и о получателе.

2. Приложение электронной почты использует информацию учетной записи, чтобы связаться с SMTP-сервером и отправить сообщение получателю.
3. SMTP-сервер считывает только ту информацию, которая найдена в конверте сообщения, и перенаправляет вашу электронную почту получателю.
4. Приложение электронной почты получателя подключается к локальному серверу, загружает электронную почту и отображает сообщение пользователю.

В действительности все происходит немного сложнее, чем здесь описано, но вам главное понять суть. Это примерно то же самое, что происходит с физическими письмами. В случае обычной почты приложение электронной почты заменяется отправителем письма на одном конце и получателем письма — на другом. SMTP-сервер заменяется почтовым отделением и работающими в нем сотрудниками (включая почтальонов). В обоих случаях кто-то пишет письмо, кто-то его доставляет, а кто-то — получает.

Компоненты конверта электронного сообщения

Есть разница в том, как сконфигурирован конверт для электронной почты и как он фактически обрабатывается. Когда вы просматриваете конверт электронного письма, он выглядит подобно реальному письму в том смысле, что он содержит адрес отправителя и адрес получателя, т.е. те же самые компоненты, что и физический конверт. В реальном письме указываются имя отправителя, его почтовый адрес, город, область и почтовый индекс, и то же самое — для получателя. Эти элементы определяют адрес, по которому почтальон должен доставить письмо или вернуть письмо, если оно не может быть доставлено.

SMTP-сервер, обрабатывая конверт электронной почты, должен проанализировать адреса, и здесь аналогия с физическим конвертом начинает немного нарушаться. Адрес электронной почты содержит совсем не ту информацию, которую мы привыкли видеть в физическом адресе. Ниже описано содержимое электронного адреса.

- **Хост.** Подобен городу и области, указанным на почтовом конверте. Адрес хоста — это адрес, используемый сетевой картой, которая физически подключена к Интернету и обрабатывает весь сетевой трафик для этого конкретного компьютера. Пользователь компьютера может взаимодействовать с различными интернет-ресурсами, но адрес хоста во всех случаях будет одинаков.
- **Порт.** Напоминает адрес квартиры, указанный на почтовом конверте. Он определяет, какая часть почтовой системы должна получить сообщение. Например, SMTP-сервер, обрабатывающий исходящую почту, обычно использует порт 25. В то же время сервер POP3, обрабатывающий входящие сообщения, использует порт 110. Браузер,

как правило, взаимодействует с веб-сайтами через порт 80. Однако для защищенных веб-сайтов (в адресе которых указан протокол `https`, а не `http`) используется порт 443. Список типичных портов можно просмотреть по следующему адресу:

`https://ru.wikipedia.org/wiki/Список_портов_TCP_и_UDP`

» **Имя локального хоста.** Это комбинация хоста и порта. Например, имя веб-сайта `http://www.myplace.com` разрешается в адрес `55.225.163.40:80` (где первые четыре числа — это адрес хоста, а номер после двоеточия — порт). Python заботится об этих деталях в фоновом режиме, поэтому вы о них не будете знать.

Теперь, когда вы получили представление о том, как формируется адрес, рассмотрим его подробнее.

Хост

Адрес хоста — это идентификатор подключения к серверу. Точно так же, как адрес на конверте не является фактическим местоположением, так и адрес хоста не является фактическим сервером. Он просто указывает на местоположение сервера.


ЗАПОМНИ!

Объект подключения, соответствующий комбинации адреса хоста и порта, называется *сокетом*. С помощью сокета можно получить всю информацию, полезную для понимания работы электронной почты. В следующей пошаговой инструкции показано, как узнать используемые имена и адреса хостов. Благодаря этому вы начнете понимать всю идею конверта электронной почты и адресов, которые он содержит.

1. **Откройте новый блокнот.**

   Можете также воспользоваться файлом исходного кода `BPPD_17_Sending_an_Email.ipynb`, содержащим код приложения (см. введение).

2. **Введите `import socket`.**

   Прежде чем начать работать с сокетами, следует импортировать библиотеку сокетов. Она содержит множество интересных функций, которые помогут нам увидеть, как работают интернет-адреса.

3. **Введите `print(socket.gethostbyname("localhost"))`.**

   На экране отобразится адрес хоста. В данном случае вы должны увидеть `127.0.0.1`, поскольку `localhost` — это стандартное имя локального хоста, которому всегда соответствует адрес `127.0.0.1`.

4. **Введите `print(socket.gethostbyaddr("127.0.0.1"))` и щелкните на кнопке Run Cell (Выполнить ячейку).**

Приготовьтесь к сюрпризу. Результатом работы функции будет кортеж (рис. 17.1). Вместо `localhost` в качестве имени хоста вы увидите имя своего компьютера. Имя `localhost` — это универсальное обозначение локального компьютера, но если вы указываете адрес, то получаете имя компьютера. Учтите, что имя вашего компьютера будет отличаться от имени, показанного на рис. 17.1.

Рис. 17.1. Адрес локального хоста соответствует вашему компьютеру

5. **Введите `print(socket.gethostbyname("www.johnmuellerbooks.com"))` и щелкните на кнопке Run Cell.**

   На экране появится результат, показанный на рис. 17.2. Это адрес моего сайта. Он остается одинаковым, где бы вы ни находились, точно так же, как и адреса почтовых отправлений. Почтовая служба использует адреса, которые являются уникальными во всем мире. То же самое касается и Интернета.

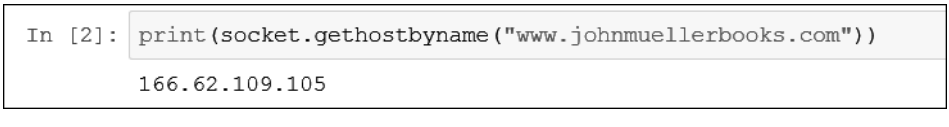

Рис. 17.2. Адреса, используемые для отправки электронных сообщений, уникальны в Интернете

Порт

Порт — это точка входа на сервер. Адрес хоста указывает местоположение, а порт определяет, как именно осуществляется доступ к серверу. Даже если вы не указываете порт при обращении к адресу хоста, подставляется стандартный порт. Доступ всегда предоставляется с использованием комбинации адреса хоста и порта. В следующей пошаговой инструкции показано, как работать с портами для получения доступа к серверу.

1. **Введите `import socket`.**

   Помните о том, что сокет предоставляет адрес хоста и информацию о порте. Именно с помощью сокета создается объект подключения, содержащий оба элемента.

2. **Введите `socket.getaddrinfo("localhost", 110)` и щелкните на кнопке Run Cell (Выполнить ячейку).**

   Первое значение — это имя хоста, о котором вы хотите получить информацию. Второе значение — порт на указанном хосте. В данном случае вы запрашиваете информацию о порте 110 локального хоста.

   Результат показан на рис. 17.3. Выходные данные состоят из двух кортежей: один — для протокола IP версии 6 (IPv6), второй — для протокола IP версии 4 (IPv4). Каждый из этих кортежей содержит пять записей, о четырех из которых вам действительно не нужно беспокоиться, потому что они вам, вероятно, никогда не понадобятся. А вот самая последняя запись, `('127.0.0.1', 110)`, содержит адрес и порт локального хоста.

Порт

```
In [3]: import socket
        socket.getaddrinfo("localhost", 110)

Out[3]: [(<AddressFamily.AF_INET6: 23>, 0, 0, '', ('::1', 110, 0, 0)),
         (<AddressFamily.AF_INET: 2>, 0, 0, '', ('127.0.0.1', 110))]
```

Рис. 17.3. Хост `localhost` поддерживает адреса IPv6 и IPv4

3. **Введите `socket.getaddrinfo("johnmuellerbooks.com", 80).`**

   Результат показан на рис. 17.4. В данном случае предоставляется только адрес IPv4, которому соответствует порт 80. Метод `socket.getaddrinfo()` позволяет определить, можно ли получить доступ к определенному интернет-ресурсу. Протокол IPv6 обеспечивает значительные преимущества по сравнению с IPv4, но большинство веб-сайтов в настоящее время реализуют только поддержку IPv4.

```
In [4]: socket.getaddrinfo("johnmuellerbooks.com", 80)

Out[4]: [(<AddressFamily.AF_INET: 2>, 0, 0, '', ('166.62.109.105', 80))]
```

Рис. 17.4. Большинство веб-сайтов поддерживает только адреса IPv4

4. **Введите `socket.getservbyport(25).`**

   Результат показан на рис. 17.5. Метод `socket.getservbyport()` позволяет определить, как используется заданный порт. Порт 25 всегда зарезервирован для поддержки SMTP. Поэтому, обратившись по адресу `127.0.0.1:25`, вы

запрашиваете доступ к SMTP-серверу на локальном хосте. Другими словами, порт обеспечивает доступ к конкретной службе.

```
In [5]: socket.getservbyport(25)

Out[5]: 'smtp'
```

Рис. 17.5. Стандартные порты служат для доступа к конкретным службам на каждом сервере

Некоторые пользователи думают, будто информация о порте предоставляется всегда. Однако это не совсем так. Python предоставит порт, заданный по умолчанию, если вы явно не укажете номер порта, но полагаться на такой порт не стоит, ведь вы не будете точно знать, с какой службой он связан. Кроме того, некоторые системы используют нестандартные номера портов исходя из соображений безопасности. Выработайте привычку всегда указывать номер порта и проверять, что работаете с корректным портом.

Имя локального хоста

Имя хоста — это текстовая форма адреса хоста. Люди плохо понимают смысл адреса 127.0.0.1 (адреса IPv6 еще менее понятны). А вот имя хоста понятно всем. Есть специальная служба, отвечающая за перевод понятных человеку имен хостов в адреса, но в этой книге мы не будем затрагивать данную тему.

В разделе “Хост” для определения имени хоста применялся метод `socket.gethostbyaddr()`, в качестве аргумента которого задавался сетевой адрес. Мы также выполняли и обратное преобразование с помощью метода `socket.gethostbyname()`. В следующей пошаговой инструкции показано, как работать с именем хоста.

1. **Введите `import socket`.**
2. **Введите `socket.gethostname()` и щелкните на кнопке Run Cell (Выполнить ячейку).**

   Отобразится имя локального компьютера (рис. 17.6). В вашем случае имя наверняка будет другим.

```
In [7]: import socket
        socket.gethostname()

Out[7]: 'Alex'
```

Рис. 17.6. Иногда нужно знать имя локального компьютера

3. **Введите `socket.gethostbyname(socket.gethostname())` и щелкните на кнопке Run Cell.**

   Вы увидите IP-адрес локальной системы, как показано на рис. 17.7. Опять же, ваши настройки наверняка отличаются от настроек автора книги, поэтому будет отличаться и результат. Данный прием можно использовать в приложениях для определения адреса отправителя, когда это необходимо. Поскольку здесь не используются жестко заданные значения, прием будет работать в любой системе.

```
In [7]: socket.gethostbyname(socket.gethostname())
Out[7]: '192.168.0.101'
```

Рис. 17.7. По возможности избегайте жестко заданных значений для локальной системы

Компоненты сообщения

Конверт электронной почты используется SMTP-сервером для маршрутизации сообщения. Он не включает собственно содержимое, находящееся в письме. Многие разработчики путают эти два элемента, потому что письмо также содержит информацию об отправителе и получателе. Эти сведения появляются в письме подобно адресной информации, включаемой в деловое письмо, и служат исключительно для удобства пользователя. Когда вы отправляете деловое письмо, почтальон не будет вскрывать конверт, чтобы увидеть информацию об адресе внутри. Для него имеют значение только сведения, указанные на конверте.

Из-за того что информация в электронном письме отделена от информации, указанной в конверте, злоумышленники могут подделать адреса электронной почты. В результате конверт будет содержать корректную информацию об отправителе и получателе, а письмо — нет. (Когда вы получаете электронное сообщение, вы видите само письмо, но не конверт — он удаляется приложением электронной почты.)

Сообщение электронной почты состоит из нескольких компонентов, как и конверт. Ниже приведен краткий обзор трех компонентов сообщения.

- **Отправитель.** Информация об отправителе сообщения. Включает только адрес электронной почты отправителя.
- **Получатель.** Информация о получателе сообщения. В действительности это список адресов получателей. Даже если вы хотите

отправить сообщение только одному человеку, нужно добавить один адрес в список.

- **Сообщение.** Информация, которую должен увидеть получатель. Может включать следующие элементы.
  - **От.** Информация об отправителе в текстовом виде.
  - **Кому.** Информация о получателе в текстовом виде.
  - **Копия.** Отображаемые получатели, которым тоже адресуется сообщение, даже если они не указаны как основные получатели.
  - **Тема.** Назначение сообщения.
  - **Документы.** Один или несколько документов, включая текстовое сообщение, которые прилагаются к письму.

Электронные письма могут быть довольно сложными и длинными и включать множество дополнительных элементов. Но чаще всего они содержат лишь вышеперечисленные простые компоненты, требуемые для отправки письма из почтового приложения. В следующих разделах процесс создания письма и его компонентов будет описан более подробно.

Структура сообщения

Можно, конечно, успешно отправить кому-либо пустой конверт, но вряд ли в этом есть смысл. Чтобы от письма была польза, оно должно содержать сообщение. В Python поддерживается несколько способов создания сообщений. Самый простой и надежный способ — воспользоваться средствами MIME (Multipurpose Internet Mail Extensions — многоцелевые расширения интернет-почты).

Как и многие другие протоколы электронной почты, MIME стандартизирован, поэтому работает одинаково вне зависимости от платформы. Существуют многочисленные форматы MIME, включенные в модуль `email.mime`, который описан по адресу `https://docs.python.org/3/library/email.mime.html`. Чаще всего вам понадобятся следующие форматы MIME.

- **`MIMEApplication`**. Позволяет отправлять и получать входные и выходные данные приложения.
- **`MIMEAudio`**. Содержит звуковой файл.
- **`MIMEImage`**. Содержит файл изображения.
- **`MIMEMultipart`**. Позволяет создавать сообщения, состоящие из нескольких частей, например текстовой и графической.
- **`MIMEText`**. Содержит текстовые данные в формате ASCII, HTML или другом стандартизированном формате.

Несмотря на то что с помощью Python можно создавать сообщения любого типа, проще всего начать с сообщения, содержащего обычный текст. Отсутствие форматирования позволит нам сосредоточиться на создании самого сообщения, а не его содержимого. В следующей пошаговой инструкции показано, как создается сообщение электронной почты, которое пока что никуда не отправляется.

1. **Введите следующий код в блокноте.**

```
from email.mime.text import MIMEText
msg = MIMEText("Hello there")
msg['Тема'] = "A Test Message"
msg['От'] = 'John Mueller <John@JohnMuellerBooks.com>'
msg['Кому'] = 'John Mueller <John@JohnMuellerBooks.com>'
```

ЗАПОМНИ!

Это простое текстовое сообщение, поэтому для него нужно импортировать класс `MIMEText`. Если бы создавался какой-то иной контент, нужно было бы импортировать другие классы или же весь модуль `email.mime`.

Конструктору `MIMEText()` нужно передать текст сообщения. Это тело сообщения, которое может быть довольно длинным. В данном случае сообщение относительно короткое и представляет собой простое приветствие.

Далее мы присваиваем значения стандартным атрибутам. В примере показаны три универсальных атрибута, которые определяется всегда: `Тема`, `От` и `Кому`. Два поля адреса, `От` и `Кому`, содержат как текстовое имя, так и адрес электронной почты. Обязательным является только адрес электронной почты.

2. **Введите `msg.as_string()` и щелкните на кнопке Run Cell (Выполнить ячейку).**

Результат показан на рис. 17.8. Вот как на самом деле выглядит сообщение. Если вы когда-либо просматривали исходное содержимое сообщений, создаваемых вашим почтовым приложением, то подобная структура будет вам знакома.

Параметр `Content-Type` определяет тип сообщения, которое в данном случае является текстовым. Параметр `charset` сообщает о том, какой набор символов используется в сообщении, чтобы получатель знал, как их обрабатывать. Параметр `MIME-Version` задает версию MIME, используемую для создания сообщения. Наконец, параметр `Context-Transfer-Encoding` определяет, каким образом сообщение преобразуется в битовый поток перед отправкой получателю.

```
In [4]: from email.mime.text import MIMEText
        msg = MIMEText("Hello there")
        msg['Тема'] = "A Test Message"
        msg['От']='John Mueller <John@JohnMuellerBooks.com>'
        msg['Кому'] = 'John Mueller <John@JohnMuellerBooks.com>'

        msg.as_string()

Out[4]: 'Content-Type: text/plain; charset="us-ascii"\nMIME-Version: 1.0\nContent-Transfer-Encoding: 7bit\nТема: A Test Messa
        ge\nОт: John Mueller <John@JohnMuellerBooks.com>\nКому: John Mueller <John@JohnMuellerBooks.com>\n\nHello there'
```

Рис. 17.8. Python добавляет служебную информацию, необходимую для создания сообщения

Процесс передачи сообщения

Выше уже говорилось о том, каким образом конверт используется для передачи сообщения из одного места в другое. При отправке сообщения необходимо определить способ передачи. Python создает сам конверт и выполняет передачу сообщения, но именно пользователь определяет особенности передачи. В следующей пошаговой инструкции демонстрируется самый простой подход к отправке сообщения в Python. У вас не получится отправить письмо, если вы не внесете коррективы в соответствии с конфигурацией вашего почтового сервера. Прочитайте приведенную ниже врезку “Знакомство с SMTP-сервером” для получения дополнительной информации.

1. **Введите следующий код в блокноте.**

   ```
   import smtplib
   s = smtplib.SMTP('localhost')
   ```

   Модуль `smtplib` содержит все, что нужно для создания и отправки конверта сообщения. Первым шагом в этом процессе является создание подключения к SMTP-серверу, который указывается в строковом виде в конструкторе. Если заданного вами SMTP-сервера не существует, приложение аварийно завершится, сообщив, что хост отказал в установке соединения.

2. **Введите `s.sendmail('`*`адрес_отправителя`*`', ['`*`адрес_получателя`*`'], msg.as_string())` и щелкните на кнопке Run Cell (Выполнить ячейку).**

ВНИМАНИЕ!

Чтобы этот код был работоспособным, нужно заменить *`адрес_отправителя`* и *`адрес_получателя`* реальными адресами. На этот раз не используйте описательные имена, поскольку серверу нужен только адрес. Если не указать реальный адрес, будет выдано сообщение об ошибке. Другие возможные причины появления сообщения об ошибке: почтовый сервер временно отключен от сети, произошел сетевой сбой или случилась какая-то другая нештатная ситуация. Если вы уверены, что ввели корректную информацию, попробуйте отправить сообщение еще раз, прежде чем начинать паниковать. Для получения дополнительной

информации обратитесь к приведенной ниже врезке "Знакомство с SMTP-сервером".

На этом шаге Python создает конверт, упаковывает сообщение электронной почты и отправляет его получателю. Обратите внимание на то, что информация об отправителе и получателе указывается отдельно от сообщения, которое SMTP-сервер не читает.

Подтипы сообщений

Выше рассматривались основные типы сообщений электронной почты. Если бы электронная почта полагалась только на эти типы, то передавать согласованные сообщения было будет затруднительно. Проблема в том, что одного типа недостаточно. Когда вы отправляете текстовое сообщение, вы должны знать, что это за текст, прежде чем сможете его корректно обработать. Текстовое сообщение может быть отформатировано как обычный текст или как HTML-страница. Вы не узнаете этого, просто прочитав метку типа, поэтому для сообщений требуется указывать подтип. Тип сообщения — текст, а подтип — HTML, если вы отправляете кому-либо HTML-страницу. Тип и подтип разделяются косой чертой, поэтому вы увидите `text/html`, если просмотрите исходное содержимое сообщения.

ЗАПОМНИ!

Теоретически количество подтипов не ограничено, если для платформы определены их обработчики. Но на практике все пользователи должны прийти к соглашению о подтипах, иначе обработчики не появятся (если только вы не имеете дело с пользовательским приложением, для которого обе стороны заранее договорились насчет собственного подтипа). Вы сможете найти список стандартных типов и подтипов на сайте `http://www.freeformatter.com/mime-types-list.html`. Приятной особенностью опубликованной там таблицы является то, что в ней приведены стандартные расширения файлов, связанные с каждым подтипом, и даны ссылки для получения дополнительной информации.

Создание электронного сообщения

До сих пор конверт и сообщение рассматривались по отдельности. Теперь пришло время соединить их и посмотреть, как работает электронная почта. В следующих разделах будет показано, как создать два сообщения. Первое из них — текстовое, а второе использует форматирование HTML. Оба сообщения должны поддерживаться большинством клиентов электронной почты.

Работа с текстовым сообщением

Текстовые сообщения — это наиболее эффективный и наименее ресурсоемкий способ передачи информации. С другой стороны, в них содержится меньше всего информации. Да, можно использовать смайлики для выражения эмоций, но отсутствие форматирования в определенных ситуациях становится проблемой. Ниже описано, как создать простое текстовое сообщение, используя Python.

1. **Введите следующий код в блокноте.**

```
from email.mime.text import MIMEText
import smtplib
msg = MIMEText("Hello There!")
msg['Тема'] = 'A Test Message'
msg['От'] = 'SenderAddress'
msg['Кому'] = 'RecipientAddress'
s = smtplib.SMTP('localhost')
s.sendmail('SenderAddress',
           ['RecipientAddress'],
           msg.as_string())
print("Сообщение отправлено!")
```

   Этот пример представляет собой квинтэссенцию всего, что было рассмотрено в главе. Мы впервые свели все вместе. Заметьте, что сначала мы создаем сообщение, а затем — конверт (как и в реальной жизни).

ВНИМАНИЕ!

   Программа выдаст ошибку, если не заменить `SenderAddress` и `RecipientAddress` реальными адресами. Эти записи используются только в качестве заполнителей. Как и в примере из предыдущего раздела, ошибки могут возникать по разным причинам, поэтому всегда пытайтесь отправить сообщение как минимум дважды, если видите ошибку впервые. Для получения дополнительной информации по этой теме обратитесь к врезке "Знакомство с SMTP-сервером".

2. **Щелкните на кнопке Run Cell (Выполнить ячейку).**

   Приложение уведомит о том, что сообщение было отправлено получателю.

Работа с HTML-сообщением

HTML-сообщение — это текстовое сообщение со специальным форматированием. В следующей пошаговой инструкции показано, как создать электронное письмо в формате HTML и отправить его.

ЗНАКОМСТВО С SMTP-СЕРВЕРОМ

Если вы попробовали выполнить пример этой главы, не внеся никаких изменений, то, скорее всего, у вас ничего не получилось. Маловероятно, чтобы имеющийся в вашей системе SMTP-сервер был подключен к сайту `localhost`. Причина, по которой в примерах используется адрес `localhost`, заключается в предоставлении заполнителя, который позже должен быть заменен реальным адресом в соответствии с конфигурацией конкретной системы.

Для того чтобы пример был работоспособным, нужны SMTP-сервер и реальная учетная запись электронной почты. Можете установить все программное обеспечение, необходимое для создания такой среды, в своей системе, и некоторые разработчики, которые интенсивно работают с почтовыми приложениями, именно так и поступают. В большинстве платформ имеется пакет электронной почты, доступный для установки, или же можно воспользоваться бесплатной альтернативой — открытым пакетом Sendmail, который можно скачать по следующей ссылке:

`https://downloads.tomsguide.com/SendMail,0301-2037.html`

Самый простой способ заставить пример заработать — использовать тот же самый SMTP-сервер, с которым взаимодействует ваше почтовое приложение. Когда вы конфигурировали приложение электронной почты, вы либо проинструктировали его обнаруживать SMTP-сервер, либо задали SMTP-сервер самостоятельно. Параметры конфигурации вашего почтового приложения должны содержать необходимую информацию. Где именно находится эта информация сильно зависит от конкретного приложения, так что сверьтесь с документацией.

Независимо от того, какой тип SMTP-сервера в конечном итоге будет использован, в большинстве случаев необходимо иметь учетную запись на этом сервере, чтобы работать с ним. Используйте в примерах адрес вашего SMTP-сервера, например `smtp.myisp.com`, и укажите ваш адрес электронной почты для отправителя и получателя. В противном случае пример работать не будет.

1. **Введите следующий код в блокноте.**

```
from email.mime.text import MIMEText
import smtplib
msg = MIMEText(
   "<h1>A Heading</h1><p>Hello There!</p>", "html")
msg['Тема'] = 'A Test HTML Message'
msg['От'] = 'SenderAddress'
msg['Кому'] = 'RecipientAddress'
s = smtplib.SMTP('localhost')
s.sendmail('SenderAddress',
```

```
                ['RecipientAddress'],
                msg.as_string())
print("Сообщение отправлено!")
```

Это фактически тот же пример, что и пример текстового сообщения в предыдущем разделе. Но обратите внимание на то, что сообщение теперь содержит HTML-теги. Мы создаем только тело HTML-сообщения, а не всю страницу. Сообщение содержит заголовок `<h1>` и абзац `<p>`.

ЗАПОМНИ!

Наиболее важной частью примера является текст, который следует за сообщением. Аргумент `"html"` изменяет подтип с `text/plain` на `text/html`, давая получателю понять, что сообщение нужно обрабатывать как HTML-содержимое. Если не внести это изменение, получатель не увидит HTML-контент.

2. **Щелкните на кнопке Run Cell (Выполнить ячейку).**

   Приложение уведомит о том, что сообщение было отправлено получателю.

Просмотр электронного сообщения

На данный момент в вашей папке Входящие должно находиться от одного до трех сообщений, сгенерированных приложением. Чтобы вы могли просмотреть сообщения, созданные в предыдущих разделах, ваше почтовое приложение должно получить их от сервера — так же, как и любую другую электронную почту. На рис. 17.9 показан пример HTML-версии сообщения.

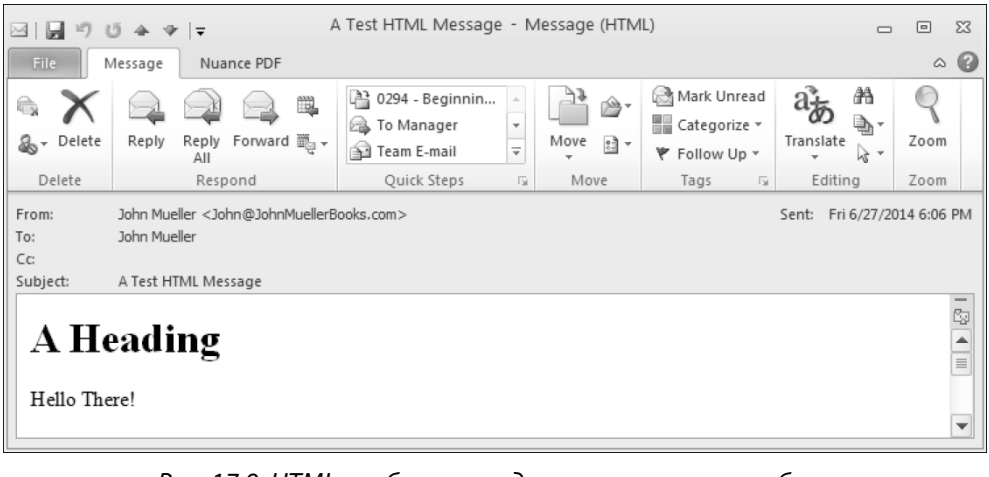

Рис. 17.9. HTML-сообщение содержит заголовок и абзац

Если почтовое приложение позволяет просмотреть источник сообщения, вы обнаружите, что сообщение содержит ту информацию, которую вы уже видели ранее. Ничего не поменялось, поскольку сообщение не подвергается никаким изменениям в процессе передачи получателю.


ЗАПОМНИ!

Создавать собственное приложение для отправки и получения электронной почты неудобно — гораздо лучше использовать готовое приложение. Целью примеров, приведенных в этой главе, было продемонстрировать гибкость Python. При создании собственного приложения вы контролируете каждый аспект сообщения. Python скрывает большую часть деталей, давая вам возможность сконцентрироваться на том, как создать и передать сообщение, используя корректные аргументы.