Programming Python

Second Edition

Mark Lutz

O'REILLY®

Программирование на Python

Второе издание

Марк Лутц

Символ

Санкт-Петербург
2002

Марк Лутц

Программирование на Python, 2 издание

Перевод С. Маккавеева

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*М. Деркачев*
Редактор	*А. Лосев*
Корректура	*С. Беляева*
Верстка	*Н. Гриценко*

Лутц М.
Программирование на Python. – Пер. с англ. – СПб: Символ-Плюс, 2002. – 1136 с., ил.
ISBN 5-93286-036-7

Python – это широко распространенный язык программирования, применяемый при решении многих важных задач, диапазон которых простирается от коммерческих сценариев установки Linux и программирования веб-приложений до анимации фильмов и создания спецэффектов. Он доступен на всех ведущих вычислительных платформах, в том числе на основных коммерческих версиях Unix, Linux, Windows и Mac OS. Кроме того, он является языком с открытым исходным кодом.

Второе издание самого известного бестселлера по Python, прорецензированное и одобренное Гвидо ван Россумом, создателем Python, представляет собой наиболее полный на сегодняшний день источник для серьезно программирующих на Python. Основное внимание здесь сосредоточено на практическом применении языка. Читатель обнаружит, что одна книга фактически содержит в себе четыре, которые глубоко освещают создание сценариев для Интернета, системное программирование, программирование GUI с использованием Tkinter и интеграцию с C. Кроме того, обсуждаются новые инструменты и приложения: Jython – версия Python, компилируемая в виде байт-кодов Java; расширения Active Scripting и COM; Zope – система веб-приложений с открытым исходным кодом; генераторы кода HTMLgen и SWIG; поддержка потоков; модули CGI и протоколов Интернета. В книге приводится большое количество примеров кода, которые вы сможете использовать при разработке на Python сложных приложений. Прилагается CD для платформ PC, Macintoch и Unix.

ISBN 5-93286-036-7
ISBN 0-596-00085-5 (англ)

Издательство «Символ-Плюс». 193148, Санкт-Петербург, ул. Пинегина, 4,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.
Подписано в печать 18.09.2002. Формат 70x100 1/16 . Печать офсетная.
Объем 71 печ. л. Тираж 2000 экз. Заказ N
Отпечатано с диапозитивов в Академической типографии «Наука» РАН
199034, Санкт-Петербург, 9 линия, 12.

Оглавление

Вступительное слово

Менее пяти лет назад я написал вступительное слово к первому изданию «Программирования на Python». За истекшее время книга изменилась примерно в той же степени, что и сам язык и сообщество Python! Мне больше не нужно защищать Python: статистика и события, приведенные Марком в предисловии, говорят сами за себя.

За последний год Python быстро развивался. Мы выпустили Python 2.0 – большой шаг вперед – с новыми возможностями, включенными в стандартную библиотеку, такими как поддержка Unicode и XML, и несколькими новыми синтаксическими конструкциями, в том числе улучшенным присвоением: теперь вместо `x = x+1` можно писать `x += 1`. Кое-кто считал, что это мелочь (представьте себе, однако, не `x`, а `dict[key]` или `list[index]`), но в целом это имело успех у тех пользователей, которые привыкли к использованию улучшенного присвоения в других языках.

Менее теплый прием получило расширение команды print, `print>>file` – сокращенная форма для вывода в файл, отличный от стандартного вывода. Лично я использую эту возможность Python 2.0 очень часто, но большинство высказавшихся по этому поводу считает ее отвратительной. Обсуждение данного простого расширения языка в телеконференции было одним из самых длинных в истории, исключая вечную тему «Python против Perl».

И это подсказывает следующую тему. (Нет, не противостояние Python и Perl. Предисловие – не место для стычек.) Я имею в виду скорость, с которой происходит развитие Python, – эту тему автор книги принимает близко к сердцу. Всякий раз, когда я ввожу в Python новую функцию, у Марка появляется еще одна седая прядь в волосах – еще одна глава устарела! Особое его беспокойство вызвало появление в Python 2.0 множества новых функций, добавленных как раз в то время, когда он работал над данным, вторым изданием книги. Что если в Python 2.1 появится столько же нововведений? Книга устареет в тот день, когда будет опубликована!

Марк, расслабься. Python продолжит свое развитие, но я обещаю, что те вещи, которые активно используются, сохранятся! Например, много беспокойства вызвал модуль для работы со строками. Сейчас, с появлением методов у объектов строк, этот модуль стал в значительной мере избыточным. Хотелось бы мне объявить его устаревшим и тем самым побудить программистов Python начать использовать вместо него методы строк. Но с учетом того, что большая часть существующего кода Python и даже многие стандартные модули используют модуль строк, такая перемена не может, очевидно, произойти очень быстро. Скорее всего, модуль строк мы сможем убрать не раньше выхода в свет Python 3000, и даже тогда, вероятно, этот модуль сохранится в библиотеке для поддержки обратной совместимости, чтобы иметь возможность использовать старый код.

Python 3000?! Да, это условное название интерпретатора Python следующего поколения. Название можно считать игрой слов с Windows 2000 или ассоциацией с «Mystery Science Theater 3000» – ТВ-шоу вполне в духе Python, ставшее культовым. Когда выйдет Python 3000? О-о-о-чень нескоро, хотя не обязательно придется ждать 3000-го года.

Первоначально Python 3000 должен был быть спроектирован и переписан заново. Это позволило бы мне осуществить несовместимые изменения, чтобы исправить те проблемы архитектуры языка, которые нельзя решить с сохранением обратной совместимости. В настоящее время, однако, планируется постепенное введение необходимых изменений в рамках продолжения текущей линии Python 2.x с ясным переходным периодом, в течение которого поддерживается обратная совместимость.

Возьмем, к примеру, деление целых чисел. Так же как и в C, в настоящее время в Python деление x/y для двух целочисленных аргументов имеет целочисленный результат. Иными словами, 1/2 дает 0! Для закоренелых программистов это естественно, но для новичков, составляющих все большую часть пользователей Python (количество которых растет экспоненциально), это служит беспрестанным источником недоразумений. С точки зрения вычислений более разумно, чтобы оператор / давал одно и то же значение независимо от типа его операндов: в конце концов, так поступают все числовые операторы. Но нельзя просто взять и изменить Python, чтобы 1/2 давало 0.5, потому что (как и в случае удаления модуля string) слишком большая часть существующего кода стала бы неработоспособной. Где же выход?

Решение – слишком сложное, чтобы описывать его здесь подробно, – охватит несколько последующих версий Python и состоит в постепенном усилении давления на программистов Python (сначала в документации, затем через предупреждения об использовании устаревших конструкций и, наконец, через сообщения об ошибках) с целью заставить их переработать свой код. Кстати, структура вывода предупредительных сообщений будет включена в версию Python 2.1. Прости, Марк!

Поэтому не ожидайте в скором времени объявления о выходе Python 3000. Вместо этого вы можете в один прекрасный день обнаружить, что *уже* используете Python 3000, только название его будет не таким, а скажем, чем-нибудь вроде Python 2.8.7. И большая часть того, что вы узнаете из этой книги, будет по-прежнему действовать! Тем не менее, в ссылках на Python 3000 не будет недостатка: просто помните, что это намеренно превозносимая химера в буквальном смысле слова. Вместо беспокойства по поводу Python 3000 лучше продолжайте использовать и изучать ту версию Python, которая у вас есть.

Хотелось бы сказать несколько слов о существующей в данное время модели разработки Python. До начала 2000 года разработчиков Python были сотни, но, в сущности, все нововведения проходили через мой почтовый ящик. Предлагая внести в Python изменение, вы должны были прислать мне по почте файл различий, который я применял к своей рабочей версии Python, и если изменение меня удовлетворяло, я вносил его в свое дерево исходного кода CVS. (CVS – это система управления версиями кода, которой посвящено несколько книг.) Сообщения о выявленных ошибках следовали тем же путем, за исключением того, что в итоге я выпускал заплатку. Очевидно, что с ростом числа предлагаемых изменений мой почтовый ящик стал узким местом процесса. Что было делать?

К счастью, Python – не единственный проект open source, в котором возникла эта проблема, и несколько умных людей в VA Linux предложили свое решение: SourceForge! Это динамичный веб-сайт с полным набором средств управления распределенными проектами: открытое хранилище CVS, почтовые списки рассылки (использующие Mailman, очень популярное приложение Python!), дискуссионные форумы, средства администрирования для ошибок и патчей и область для загрузки с сервера – все доступно для любого проекта open source, которому это может понадобиться.

В данное время у нас есть группа из 30 добровольцев с правами записи в SourceForge и почтовый список рассылки для разработчиков, в котором вдвое больше народа. Все привилегированные добровольцы поклялись на верность BDFL (Benevolent Dictator For Life – Великодушному Пожизненному Диктатору, то есть мне :-). Введение существенных новых функций регулируется с помощью облегченной системы предложений и обратной связи под названием Python Enhancement Proposals (PEPs). Наша система PEP оказалась столь успешной, что почти буквально была скопирована сообществом Tcl, когда оно совершило аналогичный переход с Cathedral на Bazaar.

Итак, с уверенностью в будущем Python я передаю слово Марку Лутцу. Отличная работа, Марк! Завершаю моей любимой цитатой из Monty Python: «Take it away, Eric, the orchestra leader»!

Гвидо ван Россум

Рестон, Вирджиния, январь 2001

Предисловие ко второму изданию

«А теперь нечто совершенно новое... опять»

Предыдущее издание этой книги стало одним из первых, представивших язык программирования Python. Второе издание является фактически новой книгой, посвященной более сложным темам Python, и должно послужить продолжением изложения основ языка в «Learning Python», дополненного справочными материалами из «Python Pocket Reference».

Это означает, что настоящее издание посвящено скорее возможностям *использования* Python, чем самому языку. Попутно исследуются идеи, лежащие в основе разработки программного обеспечения на Python; в действительности они обретают смысл только в контексте более крупных примеров, которые включены в данное издание. Но в целом, предполагается наличие у читателя хотя бы беглого знакомства с основами языка Python, которые помогут перейти к оставшейся части рассказа о нем.

В данном предисловии будут объяснены некоторые причины такой существенной переработки текста, более подробно описана структура издания и дан краткий обзор возможностей использования программ Python, имеющихся на прилагаемом компакт-диске. Однако вначале нам предстоит осуществить исторический экскурс.

Вехи распространения Python

Последние пять лет стали примечательными в мире Python. С тех пор как я написал первое издание данной книги в период между 1995 и 1996 годами, Python из новичка в семействе языков сценариев превратился в солидный и широко распространенный инструмент, используемый компаниями, разбросанными по всему свету. Хотя не всегда легко оценить популярность свободно распространяемого программного обеспечения с открытым кодом (*http://opensource.org*), такого как Python, имеющаяся статистика показывает экспоненциальный рост его популярности за последние пять лет. Вот некоторые из самых свежих признаков взрывного роста интереса к нему:

Книги

В 2001 году, когда пишется эта книга, в продаже имеется свыше десятка книг, посвященных Python, и еще почти столько же готовится к изданию (в 1995 книг не было). Некоторые из этих книг посвящены определенным областям (например, Windows), а некоторые изданы на немецком, французском и японском языках.

Пользователи

В 1999 году один из ведущих обозревателей индустрии программного обеспечения предположил исходя из различных статистических данных, что в мире насчитывается 300 000 пользователей Python. Другие оценки являются еще более оптимистичными. В начале 2000 года, например, уже действовал веб-сайт Python, с которого к концу года было произведено 500 000 новых загрузок интерпретатора (помимо того, что есть и другие носители дистрибутивов Python). Вероятно, на момент написания данной книги последняя цифра ближе к подлинной численности контингента пользователей.

Периодическая печать

Сейчас Python регулярно служит темой публикаций в периодических изданиях по программированию. В действительности после 1995 года создатель Python Гвидо ван Россум (Guido van Rossum) появился на обложке известных специальных журналов, например *Linux Journal* и *Dr. Dobb's Journal*; в публикации последнего создание Python стало основанием для награды за выдающиеся успехи в программировании.[1]

Приложения

Настоящие компании применяют Python для создания настоящих продуктов. Он был использован в построении спецэффектов для последнего фильма «Звездные войны» (Industrial Light & Magic), при выводе карт и каталогов в Интернете (Yahoo), в инсталляционной программе операционной системы Linux (Red Hat), при проверке микросхем и плат (Intel), в управлении дискуссионными форумами Интернета (Egroups), в сценариях сетевых игр (Origin), в интерфейсе к серверу CORBA (TCSI), в реализации инструментария веб-сайтов (Digital Creations' Zope), в сценариях для беспроводных устройств (Agilent) и во многих других местах.[2]

Телеконференции

Объем переписки в главной телеконференции по Python, *comp.lang.python*, также значительно вырос. Например, согласно eGroups (см. *http://www.egroups.com/group/python-list*), в январе 1994 года в этот список было подано 76 сообщений против 2678 в январе 2000 года – 35-кратное увеличение. В последующее время активность еще более возросла (4226 сообщений только за июнь 2000 – примерно 140 в сутки), и наблюдается постоянный рост с момента начала работы списка. К тому времени, когда вы будете читать это, цифры, относящиеся к числу пользователей и приведенные в данном предисловии, вероятно, возрастут. Но даже при настоящей интенсивности обмена форумы по Python достаточно загружены, чтобы полностью занять время того, кто готов полностью посвятить себя им.

Конференции

Сейчас ежегодно проводятся две конференции по Python, одну из которых организует O'Reilly & Associates. Число участников конференций, посвященных Python, примерно удваивалось каждый год. Кроме того, в Европе теперь проводится ежегодный «День Python».

Групповая терапия

Региональные группы пользователей Python стали возникать во многих местах в США и за границей, в том числе в Орегоне, Сан-Франциско, Вашингтоне, Италии, Корее и Англии. Такие группы работают над усовершенствованиями языка, проводят общественные мероприятия по Python и делают многое другое.

---

[1] Когда я писал эту книгу, *Linux Journal* напечатал также специальное приложение, посвященное Python, к майскому изданию 2000 года, на обложке которого, конечно, красовался голый мужчина, сидящий на улице перед компьютером вместо пианино. Если вам не понятно, почему это должно быть смешно, нужно посмотреть повторные показы телесериала «Monty Python», от которого получил свое название Python (считайте это первым предложенным упражнением). Я подробнее расскажу о последствиях, к которым привело название Python, в первой главе.

[2] Подробнее см. на *http://www.python.org*. Некоторые компании скрывают использование ими Python по соображениям конкуренции, хотя для многих это в итоге выявляется, когда происходит отказ на какой-либо из веб-страниц и в броузере появляется сообщение Python об ошибке. Обычно среди компаний, «засветившихся» подобным образом, упоминается Hewlett Packard.

Области применения

Развитие Python объединило как разработчиков для Microsoft Windows, с включением поддержки COM и Active Scripting, так и разработчиков Java благодаря новой, основанной на Java реализации языка – JPython (переименованного в «Jython»). Как будет показано в данной книге, появившаяся в языке поддержка COM позволяет сценариям Python являться как серверами компонентов, так и клиентами этих серверов; Active Scripting позволяет встраивать код Python в страницы HTML и выполнять его у клиента или на сервере, а JPython компилирует сценарии Python в код виртуальной машины Java, благодаря чему они могут выполняться в системах, поддерживающих Java, и беспрепятственно интегрировать библиотеки классов Java для использования в коде Python. В качестве инструментального средства, упрощающего создание сайтов, внимание вебмастеров и программистов CGI также привлек основанный на Python сервер веб-приложений Zope, описываемый в данной книге.

Поддержка

Что касается практической стороны, то коммерческая поддержка, консультации, готовые дистрибутивы и профессиональное обучение в настоящее время предоставляются многими фирмами. Например, интерпретатор Python можно получить на CD и в пакетах, продаваемых различными компаниями (в том числе Walnut Creek, «Dr. Dobb's Journal» и ActiveState); кроме того, Python обычно бесплатно поставляется в собранном виде во многих дистрибутивах операционной системы Linux.

Работа

Сейчас можно зарабатывать деньги в качестве Python-программиста (даже не прибегая к необходимости писать большие книги, содержащие интересные идеи). Когда писалась эта книга, на доске объявлений о работе на *http://www.python.org/Jobs.html* было перечислено около 60 компаний, которым требуются программисты Python в США или за рубежом. Поиск «Python» на популярных сайтах по трудоустройству дает еще более внушительные результаты: 285 связанных с Python рабочих мест на Monster.com и 369 на dice.com. Конечно, не обязательно менять работу, но приятно знать, что теперь можно зарабатывать на жизнь благодаря знанию языка, использование которого к тому же доставляет удовольствие.

Инструменты разработки

Python лег в основу многих проектов разработки инструментальных средств. Самыми заметными из них на момент написания книги являются проект Software Carpentry, в котором разрабатываются новые базовые программные инструментальные средства для Python; ActiveState, которая близка к выпуску продуктов разработки на Python для Windows и Linux[1]; и PythonWare, собирающаяся выпустить для Python интегрированную среду разработки и средство создания графического интерфейса пользователя.

Компиляторы

Когда писалось это предисловие, ActiveState объявила о создании нового компилятора Python для структуры Microsoft.NET и языковой среды C# – действительного компилятора Python и независимой реализации языка Python, в которой генерируются файлы DLL и EXE, можно разрабатывать код Python в Visual Studio и обеспечивается гладкая интеграция .NET со сценариями Python. Это будет третья реализация Python, наряду со стандартным Python, основанным на C, и системой Jpython, основанной на Java.

---

[1] ActiveState выпустила IDE Komodo, поддерживающий Python, для платформ Windows и Linux. – *Примеч. науч. ред.*

Так что же такое Python?

Если вам нужно исчерпывающее определение темы данной книги, воспользуйтесь таким:

Python является языком программирования общего назначения с открытым исходным кодом (open source), оптимизированным для качества, производительности, переносимости и интеграции. Им пользуются сотни тысяч разработчиков по всему миру в таких областях, как создание интернет-сценариев, системное программирование, проектирование пользовательских интерфейсов, настройка программных продуктов под пользователя и др.

Помимо всего остального Python поддерживает объектно-ориентированное программирование (ООП); обладает синтаксисом, который очень прост, легко читается и сопровождается; интегрируется с компонентами, написанными на языке C; обладает большим собранием уже запрограммированных интерфейсов и утилит. Несмотря на свое общее назначение, Python часто называют языком сценариев (*scripting language*), поскольку в нем просто использовать другие программные компоненты и управлять ими. Возможно, самым большим достоинством Python является просто то, что с его помощью разработка программного обеспечения становится более быстрой и приятной. Как это происходит, станет ясно из дальнейшего изложения.

Обучение

Python начал также привлекать внимание образовательных учреждений, многие из которых рассматривают его как «Pascal 2000-х годов» – язык, идеальный для обучения программированию благодаря его простоте и структуре. Отчасти это направление порождено проектом Гвидо ван Россума «Computer Programming for Everybody» (CP4E, компьютерное программирование для всех), который имеет целью сделать Python предпочтительным языком для начинающих программистов во всем мире. Будущее проекта CP4E пока остается неясным, но сформировалась группа особого интереса (SIG) к Python, которая должна заняться вопросами, связанными с образованием. Независимо от исхода той или иной инициативы, Python может сделать программирование более доступным для тех многочисленных людей, которым скоро наскучит щелкать по заранее запрограммированным ссылкам по мере их превращения из пользователей компьютеров в создателей сценариев.

Иными словами, 1995 год давно прошел. Большая часть приведенного перечня была невыполнима, когда было задумано первое издание этой книги. Естественно, этот список окажется устаревшим даже раньше, чем эта книга попадет на полки, но тем не менее, он показывает те вехи, которыми отмечено развитие Python за последние пять лет и которые ожидают нас в ближайшие годы. В качестве языка, оптимально подходящего к сегодняшним требованиям при создании программного обеспечения, Python, несомненно, еще ждет вершины своего успеха.

Для чего потребовалось это издание?

Одним из следствий растущей популярности Python стал наплыв новых пользователей, стилей программирования и приложений, из-за чего некоторые части первого издания данной книги потребовали обновления. Сам Python изменился незначительно, но важные расширения упростили различные стороны разработки на Python и заслуживают того, чтобы о них было рассказано.

Возможно, главной причиной этого издания является то, что изменилась «аудитория» Python. За последние пять лет Python превратился из развивающегося языка, которым интересуются преимущественно первопроходцы, в широко используемый программистами инструмент для решения повседневных задач. Данное издание переориентировано на эту новую аудиторию Python. Вы увидите, что теперь это более посвященная техническим деталям книга, в меньшей степени нацеленная на знакомство с языком и его популяризацию, чем на рассказ о том, как применять его к задачам программирования реального масштаба.

Объем изменений таков, что данное издание отчасти представляет совершенно новую книгу. Хочу выразить признательность тем читателям, которым понравилась первая книга; надеюсь, что тот же дух они обнаружат во втором издании. И хотя книга в значительной мере переработана, я попытался сохранить возможно большую часть материала и стиля первоначальной книги (в особенности шутки :-).

После выхода пять лет назад первого издания мне предоставилась возможность преподавать Python в США и за рубежом, и некоторые новые примеры отражают опыт, полученный в результате этого учебного процесса. Новые примеры областей применения отражают часто возникающие вопросы и интересы – как мои собственные, так и моих учеников. Преподавание Python практическим работникам, многие из которых теперь *вынуждены* использовать Python в своей работе, обусловило новый уровень связи с практикой, что вы заметите в выборе примеров и тем данного издания.

Другие примеры появились на свет как результат удовольствия, получаемого мной при программировании на Python. Да, удовольствия: я твердо уверен, что одним из самых больших неосязаемых достоинств Python является его способность вызывать радость программирования у новичков и снова вызывать эту радость у тех, кто годами трудился с использованием более требовательных инструментов. В данном издании будет продемонстрировано, что Python чрезвычайно облегчает применение более сложных, но полезных инструментов, таких как потоки, сокеты, GUI, веб-сайты и ООП – областей, которые могут показаться и утомительными и пугающими в традиционных компилируемых языках, таких как C и C++.

Честно говоря, даже после восьми лет пребывания в качестве добросовестного «питонисты» (*Pythonista*)[1] мне все еще приятно заниматься программированием, когда я делаю это на Python. Python является чрезвычайно продуктивным языком, а рассмотрение его приложений доставляет в первую очередь эстетическое удовольствие. Надеюсь, что данное издание, так же как и предыдущее, продемонстрирует, как использовать преимущества Python в продуктивности, и отчасти передаст удовлетворение и восхищение, доставляемые таким средством быстрой разработки приложений, как Python.

Основные изменения, внесенные во второе издание

Конечно, лучше всего составить впечатление о какой-либо книге, прочтя ее. Но для тех, кто читал первое издание, в последующих нескольких разделах специально более подробно описывается, что нового внесено в данное издание.

---

[1] После выхода первого издания приверженцы Python стремились найти для себя название. Чаще всего встречается *Pythonista*, но горстка ветеранов упорно называют себя *Pythoneer* или еще как-нибудь. Pythonista имеет псевдовоинственный оттенок, что не обязательно отражает истинное лицо политики языка сценариев.

Издание обновлено в соответствии с Python 2.0

Данное издание обновлено в соответствии с Python 2.0, а материал по графическим интерфейсам пользователя (GUI) обновлен для Tk версий 8.0 и более поздних. Формально это обновление началось с Python 1.5.2, но все примеры перед публикацией были проверены под версией 2.0.

Для тех, кто любит мелочи: выпуск 2.0 был первым выпуском Python после перехода Гвидо в BeOpen, а 1.6 был последним выпуском у предыдущего работодателя Гвидо, CNRI. Перед самым завершением мной окончательного наброска книги и после выхода версии 2.0 Гвидо и основная команда разработчиков Python перешли из BeOpen в Digital Creations, родину Zope – инструментального набора создания веб-приложений, но это перемещение не зависит от выпусков Python (дополнительные подробности смотрите в главе 1 «Введение в Python»).

В версии 2.0 появились новые расширения языка, но 2.0 и 1.6 по содержанию сходны, и обновление только добавляет несколько функций. Примечательно, что большинство примеров из первого издания по-прежнему действует пять лет спустя с последними выпусками Python; для тех, которые не работали, потребовались незначительные исправления (например, формат вызова GUI и интерфейсы C API).

С другой стороны, хотя основы языка не сильно изменились с момента первого издания, в него был добавлен ряд новых конструкций, и мы их всех применим тут. В число этих новых характеристик Python входят пакеты модулей, классы исключений, псевдо-приватные аттрибуты классов, строки в Unicode, новый модуль регулярных выражений, новые функции Tkinter, такие как построитель таблиц, стандартные диалоги и меню верхнего уровня и т. д. В новом приложении приводится сводка всех важнейших изменений в Python между первым и вторым изданиями этой книги.

Помимо изменений в языке в данной книге представлены новые инструменты и приложения Python, появившиеся за последние годы. В их число входят интерфейс программирования IDLE, компилятор JPython (известный также как «Jython»), расширения Active Scripting и COM, структура создания веб-приложений Zope, серверные страницы Python (Python Server Pages – PSP), режим ограниченного исполнения, генераторы кода HTMLgen и SWIG, поддержка потоков, модули CGI и протоколов Интернета и многое другое (эти пять лет были активными). Такие приложения составляют самую суть второго издания.

Книга переориентирована на более подготовленную аудиторию

В данном издании программирование на Python представлено *усложненными примерами*. Чтобы стать профессионалом в Python, нужно решить две различные задачи: овладеть основами самого языка, а затем научиться применять его в приложениях. Решение второй (и более крупной) задачи в данном издании осуществляется путем представления библиотек Python, инструментальных средств и технологий программирования. Поскольку это совершенно иная задача, следует сказать здесь несколько слов о том, почему она была поставлена.

Поскольку во время появления первого издания этой книги других изданий по Python на горизонте заметно не было, оно было обращено сразу к очень широкой аудитории: и к новичкам и к гуру одновременно. С тех пор появилась другая книга O'Reilly, «Learning Python» («Изучаем Python»), которая была ориентирована на новичков, и был опубликован «Python Pocket Reference» («Карманный справочник по Python») для читателей, нуждающихся в кратком справочнике по Python. В результате ввод-

ный материал по основам языка и первоначальные справочные приложения были изъяты из этой книги.

«Изучаем Python» знакомит с основами языка – синтаксисом, типами данных и т. д. – с помощью намеренно упрощенных примеров. Многие считают эту книгу идеальной для изучения самого языка, но Python может стать еще интереснее, когда вы овладеете базовым синтаксисом и сможете писать простые примеры в интерактивном интерфейсе. Научившись разрезать список, вы весьма скоро захотите делать реальные вещи, например писать сценарии для сравнения каталогов файлов, отвечать на запросы пользователя через Интернет, выводить графические изображения в окнах, читать электронную почту и т. д. Повседневная работа состоит в основном в применении языка, а не в самом языке.

Данная книга «Программирование на Python» сконцентрирована на «всем остальном» в разработках на Python. В ней рассказывается о библиотеках и инструментарии, находящихся за рамками основного языка, которые приобретают первостепенное значение при написании реальных приложений. Она также рассматривает вопросы проектирования более крупных приложений, например повторное использование кода и ООП, которые можно проиллюстрировать только в контексте программ более реального масштаба. Иными словами, книга «Программирование на Python», особенно в этом новом издании, призвана подхватить изложение там, где оно закончилось в «Изучаем Python».

Поэтому, если данная книга покажется вам слишком сложной, я советую предварительно прочесть «Изучаем Python» и после овладения основами вернуться сюда за продолжением рассказа. Если только у вас нет значительного опыта в программировании, это издание лучше всего использовать в качестве второй своей книги по Python.

В книге раскрыты новые темы

Большинство изменений в данном издании было сделано для того, чтобы осветить новые темы. В нем появились новые главы и разделы, рассказывающие о сценариях в Интернете, сценариях CGI, интерфейсах операционных систем, генераторе интегрирующего кода SWIG, более сложных разделах Tkinter, генераторе веб-страниц HTMLgen, JPython, потоках, режиме ограниченного исполнения и многом другом. Весь объем можно уяснить, обратившись к оглавлению, а здесь перечислены некоторые из новых тем и структурных изменений, которые появились в данном издании:

Темы

Уделено значительно большее внимание Интернету, системному программированию, графическим интерфейсам Tkinter и интеграции с C. Можно утверждать, что на них теперь в основном сосредоточен данный текст. Например, появилось шесть новых глав по сценариям в Интернете, в которых рассказано об инструментах на стороне клиента, сценариях на стороне сервера, веб-сайтах и более сложных темах и системах, связанных с Интернетом. Четыре новые главы посвящены системным вопросам: потокам, обработке каталогов, запуску программ и т. д. Материал по GUI также вырос с одной главы до значительно более полного представления в четырех главах, охватывая теперь все графические элементы (в том числе текстовые и холст), а также новую поддержку таблиц, меню и диалогов.

Интеграция с C

Главы, посвященные расширениям и встраиванию C, дополнены новым материалом, охватывающим такие темы, как SWIG (способ, которым в настоящее время следует соединять Python с библиотеками C/C++), и представляющим новые примеры смешанного режима, такие как диспетчер функций обратного вызова (рас-

ширение и встраивание). Интеграция с C лежит в сердцевине многих систем Python, но примеры из этой области неизбежно сложны и включают в себя большие программы на C, интересные только пользователям C. Из уважения к читателям, которым не требуется интегрировать Python с C, этот материал теперь помещен отдельно в конце книги. Некоторые листинги кода на C также удалены, чтобы сократить объем книги: вместо них я предпочитал по возможности отсылать читателей к исходным файлам на C на прилагаемом CD.

Хотя более поздние главы основываются на материале более ранних глав, темы в данном издании освещаются достаточно независимо и объединяются в части книги. Благодаря этому не будет большой натяжкой считать, что в данном издании объединены в одно целое четыре или пять книг. Структура основных разделов книги подчеркивает ее концентрацию на темах приложения:

Предисловие (в котором вы находитесь)

- Глава 1 «Знакомство с Python»

Часть I «Системные интерфейсы»

- Глава 2 «Системные инструменты»
- Глава 3 «Системные средства параллельного выполнения»
- Глава 4 «Более крупные системные примеры, часть 1»
- Глава 5 «Более крупные системные примеры, часть 2»

Часть II «Программирование GUI»

- Глава 6 «Графические интерфейсы пользователя»
- Глава 7 «Обзор Tkinter, часть 1»
- Глава 8 «Обзор Tkinter, часть 2»
- Глава 9 «Более крупные примеры GUI»

Часть III «Создание сценариев для Интернета»

- Глава 10 «Сетевые сценарии»
- Глава 11 «Сценарии на стороне клиента»
- Глава 12 «Сценарии, выполняемые на сервере»
- Глава 13 «Более крупные примеры сайтов, часть 1»
- Глава 14 «Более крупные примеры сайтов, часть 2»
- Глава 15 «Более сложные темы Интернета»

Часть IV «Разные темы»

- Глава 16 «Базы данных и постоянное хранение»
- Глава 17 «Структуры данных»
- Глава 18 «Текст и язык»

Часть V «Интеграция»

- Глава 19 «Расширяем Python»
- Глава 20 «Встраиваем Python»

Часть VI «Финал»

- Глава 21 «Заключение: Python и цикл разработки»
- Приложение A «Последние изменения в Python»
- Приложение B «Прагматика»
- Приложение C «Python и C++»

Два замечания. Не дайте заглавиям ввести вас в заблуждение: хотя большинство разделов относится к темам приложений, все же попутно рассматриваются характеристики и общие идеи проектирования языка Python в контексте практических задач. И второе: читатели, использующие Python в качестве самостоятельного инструмента, могут благополучно пропустить главы, связанные с интеграцией, хотя все же рекомендуется бегло взглянуть на них. Программирование на C не дает такого удовольствия и легкости, как программирование на Python. Однако поскольку интеграция играет центральную роль в использовании Python в качестве языка сценариев, некоторое понимание ее может оказаться полезным независимо от того, занимаетесь ли вы интеграцией, написанием сценариев или тем и другим вместе.

Читатели первого издания обратят внимание на то, что материал по большей части является новым, причем даже главы, сохранившие свое название, сильно обновились. Заметно отсутствие в данном издании первоначального краткого обзора, мини-справочника и учебного приложения, а также полное отсутствие прежней части II, что отражает новую направленность книги и предполагаемую аудиторию.

Книга более ориентирована на примеры

Эта книга в значительной мере состоит из примеров. В настоящем издании старые примеры расширены, чтобы придать им большую реалистичность (например, PyForm и PyCalc); кроме того, всюду добавлены новые примеры. В число основных примеров входят:

PyEdit

Объект и программа редактора текстовых файлов на Python/Tk.

PyView

Слайд-шоу для фотоизображений и файлов заметок.

PyDraw

Графический редактор для рисования и перемещения объектов изображений.

PyTree

Программа для изображения древовидных структур данных.

PyClock

Графический элемент для изображения аналоговых и цифровых часов на Python/Tk.

PyToe

Графическая программа игры в крестики-нолики с искусственным интеллектом.

PyForm

Броузер для таблиц постоянных объектов.

PyCalc

Графический элемент калькулятора на Python/Tk.

PyMail

Почтовый клиент с поддержкой POP и SMTP на Python/Tk.

PyFtp

Простой GUI для пересылки файлов на Python/Tk.

PyErrata

Размещаемая в Сети система сообщений об ошибках.

PyMailCgi

Размещаемый в Сети интерфейс электронной почты.

Есть также новые примеры интеграции с C в смешанном режиме (например, регистрация функции обратного вызова (callback) и работа с объектом класса), примеры SWIG (с «теневыми» классами для C++ и без них), дополнительные примеры Интернета (сценарии отправки и получения по FTP, примеры NNTP и HTTP, средства работы с электронной почтой и новые примеры модулей `socket` и `select`), много новых примеров потоков в Python и новое освещение JPython, HTMLgen, Zope, Active Scripting, COM и интерфейсов Python к базам данных. Многие новые примеры являются довольно продвинутыми, впрочем, как и весь текст.

Кроме того, прежний API для встраивания в C кода на Python (называемый теперь *ppembed*) расширен с целью поддержки предварительной компиляции строк в байт-коды, а первоначальный пример калькулятора (имеющий теперь название *PyCalc*) усилен и стал поддерживать ввод с клавиатуры, буфер команд, цвета и другие функции.

На практике примеры в новой книге, распространяемые на прилагаемом к данному изданию CD-ROM, *сами по себе* являются довольно сложной системой программ на Python, входящие в которую примеры структурно изменены в ряде важных направлений:

Дерево примеров

Весь дистрибутив примеров организован в виде одного большого *пакета модулей* Python, что облегчает импорт из других каталогов и устраняет конфликт имен с другим кодом Python, установленным на компьютере. Использование путей каталогов в командах import (вместо сложного PYTHONPATH) облегчает также установление происхождения модулей. Более того, теперь нужно добавить в путь поиска PYTHONPATH только один каталог для всего дерева примеров в книге: каталог, содержащий корневой каталог примеров *PP2E*. Для использования примеров, приведенных в книге, в собственных приложениях нужно просто импортировать их через корень пакета *PP2E* (например, `from PP2E.Launcher import which`).

Имена файлов примеров

Имена модулей стали значительно менее таинственными. Я давно перестал обращать внимание на совместимость с форматом DOS 8.3 и использую более описательные имена. Были также исправлены некоторые старые имена файлов, полностью записанные в верхнем регистре – последнее наследие MS-DOS.

Названия примеров

В метках листингов примеров теперь указывается полный путь к файлу исходного кода примера, что помогает найти его в дистрибутиве примеров. Например, файл с исходным кодом примера, имя которого указано как *Example N-M: PP2E\Internet\Ftp\sousa.py*, указывает на файл *sousa.py* в подкаталоге *PP2E\Internet\Ftp* каталога дистрибутива примеров.[1]

Командные строки примеров

Аналогично, когда показана командная строка, введенная после приглашения, например `C:\...\PP2E\System\Streams>`, она в действительности должна быть введе-

[1] «Каталог дистрибутива примеров» является каталогом, содержащим каталог верхнего уровня *PP2E* дерева примеров книги. На CD это самый верхний каталог *Examples*; если вы скопировали примеры на диск, это то место, куда вы скопировали (или распаковали) корневой каталог *PP2E*. Большинство этих примеров можно непосредственно запускать с CD, но они должны быть скопированы на жесткий диск для внесения в них изменений и чтобы позволить Python сохранять файлы компилированного байт-кода *.pyc*, допускающие более быстрый начальный запуск.

на так, чтобы указывать на подкаталог *PP2E\System\Streams* в вашем дереве примеров. Пользователям Unix и Linux: пожалуйста, имейте в виду / при встрече с \ в путях к файлам (официальное извинение по этому поводу выражено в следующем разделе).

Средства запуска примеров

Поскольку приятно иметь возможность сразу щелкнуть куда нужно мышкой, есть новые самоконфигурирующиеся программы запуска демонстрационных примеров (описываемые ниже в этом предисловии в разделе «Вкратце»), позволяющие быстро взглянуть на сценарии Python в действии с минимальной предварительной настройкой. Обычно можно запускать их прямо с CD, не устанавливая каких-либо переменных окружения.

Книга более нейтральна в отношении используемых платформ

За исключением все тех же примеров с интеграцией с C, большинство программ в этом издании было разработано на моем портативном компьютере под Windows 98 с прицелом на переносимость под Linux и другие платформы. В действительности некоторые примеры порождены моим желанием создать на Python переносимые эквиваленты некоторых инструментов, отсутствующих в Windows (например, программ для разбивки файлов на части). Когда программы показываются в действии, это обычно делается на Windows; на платформе Red Hat Linux 6.x они демонстрируются только при использовании специфических для Unix интерфейсов.

Это вовсе не политическое заявление: Linux мне тоже нравится. Это в основном следствие того, что я писал эту книгу с помощью MS Word; когда время ограничено, удобнее запускать сценарии на той же платформе, что и издательские средства, чтобы не перегружаться слишком часто в Linux. К счастью, поскольку Python стал теперь в большой степени переносим между Windows и Linux, используемая операционная система меньше заботит разработчиков Python, чем было ранее. Python, его библиотеки и система GUI Tkinter в настоящее время прекрасно работают на обеих платформах.

Однако поскольку я не занимаюсь политикой, то постарался сделать примеры возможно более независимыми от платформы и попутно указать на специфические для каждой платформы проблемы. Вообще говоря, большинство сценариев должно работать на обычных платформах Python без внесения изменений. Например, все примеры GUI были проверены под Windows (98, 95) и Linux (KDE, Gnome), а большинство примеров командной строки и потоков были разработаны в Windows, но работают и в Linux. Поскольку системные интерфейсы Python обычно строятся в расчете на переносимость, сделать это легче, чем может показаться.

В то же время, эта книга при необходимости погружается в специфические для платформ темы. Заново изложены многие специальные темы Windows: Active Scripting, COM, варианты запуска программ и т. д. Читатели, работающие под Linux и Unix, найдут и материал, относящийся к их платформе: ветвление, конвейеры и тому подобное. Заново изложены также способы редактирования и запуска программ Python на большинстве главных платформ.

Единственное место, где читатели смогут усмотреть уклон в сторону определенной платформы, – это примеры интеграции Python с C. Для простоты детали компиляции программ на C, о которых говорится в данной книге, все еще имеют некоторый уклон в сторону Unix/Linux. Во всяком случае этому можно найти разумное объяснение: Linux не только поставляется с бесплатными компиляторами C, но вокруг этого языка выросла вся его среда разработки. Код расширений на C, приведенных в этой книге,

Но все равно это не справочное руководство

Обратите, пожалуйста, внимание на то, что это издание, как и первое, все же скорее *учебник*, чем справочное руководство (несмотря на название, сходное с популярным справочником по Perl). Эта книга должна учить, а не документировать. С помощью ее оглавления и предметного указателя можно найти конкретные детали, и новая структура облегчает это. Но все же данное издание задумано как книга, которую используют вместе со справочниками по Python, а не вместо них. Поскольку руководства по Python бесплатны, написаны хорошо, доступны в Сети и часто редактируются, было бы безумием тратить место на копирование их содержания. Исчерпывающий список всех инструментов, имеющихся в системе Python, можно найти в других книгах (например, изданной O'Reilly «Python Pocket Reference») или стандартных руководствах на веб-сайте Python либо на компакт-диске, прилагаемом к данной книге.

будет работать и под Windows, но может потребоваться использование различных процедур сборки программ в зависимости от используемого в Windows компилятора. Издательство O'Reilly опубликовало замечательную книгу «Python Programming on Win32», в которой освещены специфические для Windows темы Python вроде этой, которая должна помочь справиться с некоторыми наблюдаемыми здесь расхождениями. Если вы занимаетесь программированием под Windows, то найдете все относящиеся к Windows подробности, которые здесь пропущены.

Использование примеров и демонстрационных программ

Хочу здесь кратко описать, как использовать имеющиеся в книге примеры. В целом, однако, посмотрите, пожалуйста, следующие текстовые файлы в каталоге дистрибутива примеров, в которых вы найдете дополнительные сведения:

- *README-root.txt* – замечания о структуре пакета
- *PP2E\README-PP2E.txt* – общие замечания об использовании
- *PP2E\Config\setup-pp.bat* – конфигурация для Windows
- *PP2E\Config\setup-pp.csh* – конфигурация для Unix и Linux

Из этих файлов наиболее информативен *README-PP2E.txt*, а в каталоге *PP2E\Config* содержатся все файлы примеров конфигурации. Здесь дается обзор, но в перечисленных файлах находится полное описание.

Вкратце

Если вы сразу хотите посмотреть некоторые примеры Python, поступите так:

1. Установите Python с компакт-диска, прилагаемого к книге, если он еще не установлен на вашем компьютере. В Windows щелкните на имени самоустанавливающейся программы на CD и сделайте установку по умолчанию (отвечайте «yes» или «next» на все приглашения). Для других систем посмотрите файл README (заархивированный дистрибутив исходного кода на CD можно использовать для локальной сборки Python).
2. Запустите один из следующих *самоконфигурирующихся сценариев*, располагающихся в каталоге CD верхнего уровня Examples\PP2E. Щелкните по соответству-

ющим пиктограммам в проводнике или запустите из системной подсказки (например, окна консоли DOS или Linux Xterm) посредством командной строки формата `python имя-сценария` (может потребоваться указать полный путь к `python`, если он не включен в систему):

- *Launch_PyDemos.pyw* – главная инструментальная панель для запуска демонстрационных программ Python/Tk
- *Launch_PyGadgets_bar.pyw* – инструментальная панель для запуска утилит Python/Tk
- *Launch_PyGadgets.py* – запускает стандартные утилиты Python/Tk
- *LaunchBrowser.py* – открывает указатель веб-примеров в веб-броузере

Сценарии `Launch_*` запускают программы Python переносимым образом[1] и требуют только наличия установленного Python: для их запуска не требуется предварительной настройки переменных окружения или изменений в прилагаемых файлах настройки *PP2E\Config*. `LaunchBrowser` сможет работать, если найдет на вашей машине веб-броузер, даже если у вас нет соединения с Интернетом (хотя некоторые интернет-примеры работают неполным образом в отсутствие работающего соединения).

Если отказаться от установки Python, все же можно запустить несколько веб-демонстраций Python, отправив броузер на *http://starship.python.net/~lutz/PyInternetDemos.html*. Так как эти примеры предназначены для выполнения сценариев на сервере, лучше всего они работают при запуске на этом сайте, а не с CD, прилагаемого к книге.

Подробности

Для лучшей организации новых примеров я поместил в каталог верхнего уровня дистрибутива примеров *PP2E* программу запуска демонстраций, *PyDemos.pyw*. На рис. П.1 показана PyDemos в действии под Windows после нажатия нескольких кнопок. Панель запуска появляется в левой части экрана; с ее помощью можно щелчком мыши запустить большинство графических примеров, приведенных в книге. Если на вашей машине удастся обнаружить броузер, то с помощью панели запуска демонстраций можно также запускать основные примеры для Интернета (смотрите описание программы запуска далее).

Помимо запуска демонстрационных программ исходный код PyDemos дает указатели на основные примеры в дистрибутиве; детали ищите в исходном коде программы. В каталоге верхнего уровня примеров вы также обнаружите сценарии для автоматической компиляции под Linux примеров интеграции Python и C, которые служат указателями для основных примеров на C.

Я также включил программу верхнего уровня под названием *PyGadgets.py* и родственную ей *PyGadgets_bar.pyw*, чтобы запускать для реального использования, а не для демонстрации некоторые наиболее полезные примеры GUI из книги (в большинстве своем часто используемые мной программы; переконфигурируйте их, если захотите). На рис. П.2 показано, как выглядит в Windows `PyGadgets_bar`, а также некоторые утилиты, которые могут запускаться его кнопками. Все эти программы представлены в данной книге и включены в дистрибутив примеров. Для большинства из

---

[1] Все демо- и запускающие сценарии написаны переносимым образом, но на момент написания книги известно, что они работают только под Windows 95/98 и Linux; для других платформ могут потребоваться незначительные изменения. Приношу извинения, если вы используете платформу, которую я не смог проверить: Tk работает под Windows, X11 и на Macs; сам Python работает всюду – от карманных PDA до мэйнфреймов; и мой вклад в написание этой книги был не столь велик, как вы могли подумать.

Рис. П.1. Запускающая программа PyDemos со всплывающими окнами и демонстрационными программами (фотография Гвидо перепечатана с разрешения Dr. Dobb's Journal)

них требуется Python с поддержкой Tkinter, но это и есть стандартная конфигурация для запуска материалов с CD этой книги под Windows.

Для непосредственного запуска файлов, перечисленных в предыдущем параграфе, требуется настроить путь поиска модулей Python (подсказку ищите в файлах `PP2E/Config/setup*`). Но если вы хотите запустить собрание демонстрационных программ Python из данной книги и не хотите утруждать себя предварительной настройкой окружения, просто выполните сценарии утилит в каталоге *PP2E*: *Launch_PyDemos.pyw*, *Launch_PyGadgets.py* и *Launch_ PyGadgets_bar.pyw*.

Эти сценарии для запуска программ, написанные на Python, предполагают, что Python уже установлен, и автоматически найдут интерпретатор языка и дистрибутив примеров из книги, а также настроят пути к модулям Python и системные пути поиска так, как это требуется для запуска демонстрационных программ. Вероятно, вы сможете запустить эти сценарии, просто щелкая по их именам в проводнике, а также сможете запустить их непосредственно с прилагаемого CD-ROM. Дополнительные сведения можно найти в комментариях в начале *Launcher.py* (или прочесть об этих сценариях в главе 4 «Более крупные системные примеры, часть 1»).

Многие из примеров для Интернета, использующих броузеры, можно найти в сети на *http://starship.python.net/~lutz/PyInternetDemos.html*, где можно проверить их работу. Поскольку эти примеры выполняются в броузере, их можно посмотреть, даже если на машине не установлен Python (или поддержка Tk для Python).

Программа PyDemos также пытается запустить броузер с веб-страницами основных примеров путем выполнения сценария *LaunchBrowser.py* в корневом каталоге примеров. Этот сценарий обычно успешно находит на вашей машине броузер, которым можно воспользоваться; если поиск окажется неудачным, смотрите дополнительные

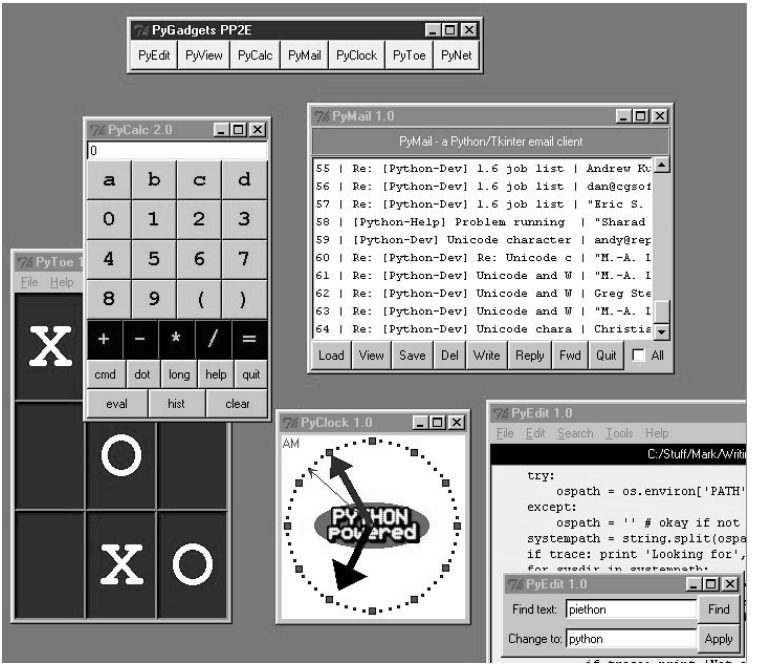

Рис. П.2. Панель запуска утилит PyGadgets вместе с программами

сведения в сценарии. Если `LaunchBrowser` сможет найти на вашей машине броузер, некоторые кнопки автоматически выведут веб-страницы независимо от наличия действующего соединения с Интернетом (если соединения нет, в броузере будут показаны локальные файлы). На рис. П.3 показано, как выглядит страница PyInternetDemos в Internet Explorer под Windows.

Особенно интересно, что ссылка *getfile.html* на этой странице позволяет просмотреть исходный код любого другого файла на сайте книги – код HTML, CGI-сценарии Python и т. д.; подробности смотрите в главе 12. Подведем итоги; вот что вы найдете в каталоге верхнего уровня *PP2E* дистрибутива примеров книги:

PyDemos.pyw

Панель кнопок для запуска основных примеров GUI и Интернета.

PyGadgets.py

Запуск программ в недемонстрационном режиме для обычного использования.

PyGadgets_bar.pyw

Панель кнопок для запуска PyGadgets по требованию.

*Launch_*.py**

Запускает программы PyDemos и PyGadgets с помощью *Launcher.py*, автоматически конфигурирующего пути поиска (запускайте их, чтобы бегло взглянуть).

Launcher.py

Используется для запуска программ без настройки окружения: находит Python, устанавливает PYTHONPATH, запускает программы Python.

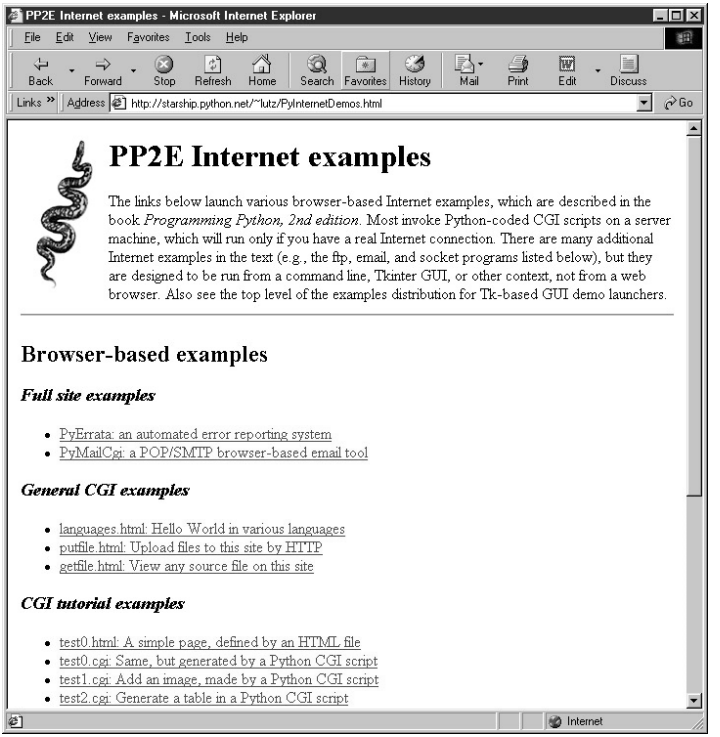

Рис. П.3. Веб-страница PyInternetDemos

LaunchBrowser.py

Открывает веб-страницы примеров в автоматически найденном броузере, загружая их из Сети или из локальных файлов; при непосредственном запуске открывает страницу указателя PyInternetDemos.

Существуют также подкаталоги для примеров из каждой основной группы тем, рассматриваемых в книге.

Кроме того, каталог верхнего уровня *PP2E\PyTools* содержит написанные на Python утилиты командной строки для преобразования символов перевода строки во всех текстовых файлах примеров в формат DOS или Unix (полезно использовать, если они странно выглядят в вашем текстовом редакторе), превращения всех файлов примеров в доступные для записи (полезны, если вы перетаскиваете файлы с прилагаемого CD), удаления старых файлов байт-кода *.pyc* в дереве каталогов и другие. Дополнительные сведения, касающиеся всех вопросов, связанных с примерами, ищите в файле *README-PP2E.txt*.

Где все это находится

Дистрибутив примеров книги находится на сопровождающем ее компакт-диске. Детали использования описаны в файле README на верхнем уровне каталогов CD. Для беглого ознакомления можно просмотреть корневой каталог примеров на компакт-диске в своем любимом файловом проводнике.

Помимо примеров из книги CD содержит также различные относящиеся к Python пакеты, в том числе полную *программу самоустановки* под Windows с поддержкой Python и Tk (для установки дважды щелкните по ней и ответьте «yes» на все приглашения), полный *дистрибутив исходного кода* Python (распакуйте и откомпилируйте на своей машине) и набор *стандартной документации* Python в формате HTML (щелкните для просмотра в своем веб-броузере).

Дополнительные пакеты открытого исходного кода, например последние версии (на момент публикации) генератора кода SWIG и Jpython, тоже помещены на CD, но вы всегда можете найти новейшие версии Python и других пакетов на веб-сайте Python *http://www.python.org*.

Использованные в книге типографские обозначения

В данной книге принято следующее использование шрифтов:

Курсив

Используется в именах файлов и каталогов, URL, командах, для выделения новых терминов при первом знакомстве и в некоторых комментариях в фрагментах кода.

Моноширинный

Используется в листингах кода и для обозначения модулей, методов, опций, классов, функций, утверждений, программ, объектов и тегов HTML.

Моноширинный полужирный

Используется в приводимом коде для показа данных, вводимых пользователем.

Моноширинный курсив

Используется для обозначения заменяемого пользователем текста.

Изображение совы обозначает замечание, относящееся к близлежащему тексту.

Изображение индюшки обозначает предупреждение, относящееся к близлежащему тексту.

Где можно найти обновления

Как и прежде, обновления, исправления и дополнения для данной книги поддерживаются на веб-сайте автора *http://www.rmi.net/~lutz*. Найдите на этой странице ссылку на второе издание, чтобы получить всю дополнительную информацию, касающуюся данной версии книги. Как и для первого издания, я буду вести на этом сайте журнал регистрации изменений, производимых в Python, который следует рассматривать как дополнение к данной книге.

Начиная с этого издания я открываю в Сети систему сообщения пользователями об ошибках, найденных в книге, на сайте:

http://starship.python.net/~lutz/PyErrata/pyerrata.html

Там вы найдете формы для сообщений о проблемах, возникших с книгой, и комментариев, а также сможете просматривать базу данных сообщений, отсортированную по различным ключам. По умолчанию сообщения хранятся в базе данных общего доступа, но при желании можно направлять их закрытым письмом. Система PyErrata тоже написана на Python и составила пример, вошедший в данную книгу; смотрите главу 14. Рис. П.4 показывает внешний вид корневой страницы PyErrata.

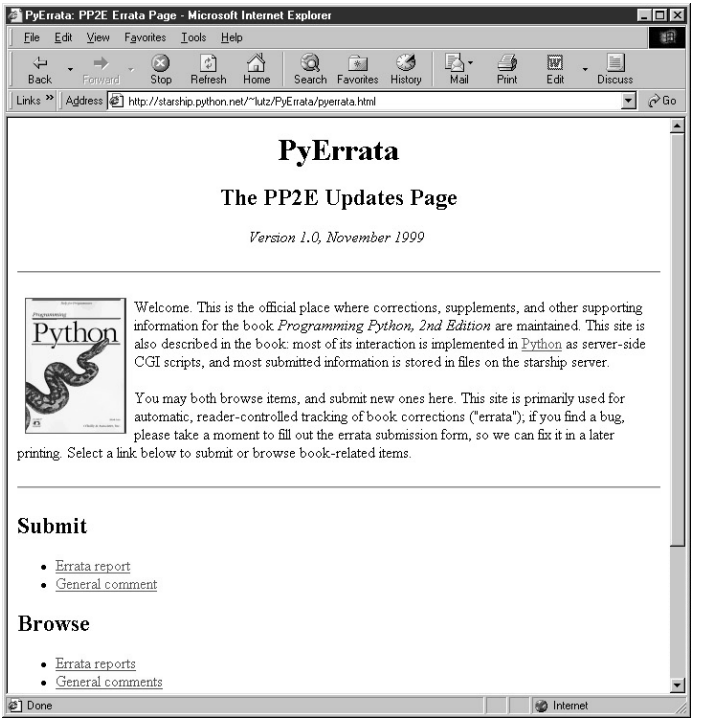

Рис. П.4. Сайт PyErrata вносимых в книгу исправлений

Если какие-либо из этих адресов окажутся недействующими, то получить доступ к этим страницам можно также на веб-сайте издательства O'Reilly *http://www.oreilly.com*.[1] Я по-прежнему буду рад получать прямые письма читателей, но надеюсь, что PyErrata усовершенствует процесс передачи сообщений об ошибках.

Связь с O'Reilly

Можно также обратиться с комментариями и вопросами относительно данной книги к издателю:

O'Reilly & Associates, Inc.
101 Morris Street
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)

[1] На веб-сайте O'Reilly тоже есть система приема сообщений об ошибках, и оба эти списка в совокупности следует рассматривать как официальное заключение о найденных ошибках и внесенных изменениях.

(707) 829-0515 (international/local)
(707) 829-0104 (fax)

В O'Reilly есть веб-страница, посвященная данной книге, на которой приведены ошибки, примеры и любая дополнительная информация. Эта страница находится на

http://www.oreilly.com/catalog/python2/

Для комментариев или технических вопросов по данной книге пишите по адресу:

bookquestions@oreilly.com

Дополнительную информацию о книгах, конференциях, программном обеспечении, центрах ресурсов и сети O'Reilly Network можно найти на веб-сайте O'Reilly на

http://www.oreilly.com

Благодарности

Помимо людей, упомянутых в первом издании, я хотел бы дополнительно выразить признательность тем, кто так или иначе помогал мне во время данного проекта второго издания:

- Редактору первого издания Фрэнку Уиллисону (Frank Willison) за то, что он просмотрел эту редакцию и продвигал Python как в O'Reilly, так и за его пределами. Следующему редактору данной книги Лоре Левин (Laura Lewin), перенявшей у него эстафету.
- Создателю Python Гвидо ван Россуму (Guido van Rossum), благодаря которому работа над книгой снова стала удовольствием.
- Участникам рецензирования раннего проекта данного издания: Эрику Рэймонду (Eric Raymond), Марку Хэммонду (Mark Hammond), Дэвиду Эшеру (David Ascher), Тиму Петерсу (Tim Peters) и Дэйву Бизли (Dave Beazley).
- Тиму О'Рейлли (Tim O'Reilly) и служащим O'Reilly & Associates как за выход этой книги, так и за поддержку программного обеспечения с открытым кодом в целом.
- Сообществу Python вообще за прилежание, напряженную работу и юмор – в прежние годы и сейчас. Мы далеко продвинулись, но вспомним строчку из 1970-х: «Вы еще ничего не видите».
- Студентам многочисленных курсов Python, которым я преподавал, а также многочисленным читателям, не пожалевшим времени, чтобы послать мне комментарии по поводу первого издания: ваше мнение помогло определить характер данной новой редакции.

Наконец, несколько личных выражений благодарности. Моим детям, Майклу, Саманте и Роксане, за целеустремленность. Если они представляют свое поколение, то будущее нашего вида находится в хороших руках. Вам придется извинить меня за гордыню, но с такими детьми, как мои, невозможно чувствовать иначе.

А самая большая благодарность Лайзе, матери этих изумительных детей. Я в самом большом долгу перед ней за все – от терпеливого отношения к моим уходам от реальности, когда я пишу книги вроде этой, до того, что она спасла меня от тюрьмы в годы нашей молодости. Что бы ни таило нам будущее, я благодарен судьбе, соединившей нас два десятилетия назад.

Марк Лутц
Ноябрь 2000
Где-то в Колорадо

«Когда Билли пойдет вниз, с ним это произойдет быстро»

За последние пять лет наблюдался также подъем движения за программное обеспечение с открытым кодом (open source) – это программы, бесплатно распространяемые вместе с полным исходным кодом, обычно являющиеся результатом работы многих разработчиков, сотрудничающих друг с другом в свободной манере. В эту категорию попадают Python, операционная система Linux и многие другие инструменты, например Perl и веб-сервер Apache. Частично благодаря вызову, брошенному им преобладанию на рынке мегакомпаний, движение программного обеспечения с открытым кодом быстро и глубоко проникло в общество.

Позвольте рассказать о событии, недавно подчеркнувшем степень влияния на меня этого движения. Чтобы вам стал понятен этот рассказ, следует сообщить, что я писал эту книгу в небольшом городке в Колорадо, о котором нельзя сказать, что он находился на переднем крае технологических нововведений. Сказать образнее, это одно из тех мест, которые называют «ковбойский городок».

Я зашел в небольшую местную книжную лавку в поисках последнего номера *Linux Journal*. После недолгих поисков я нашел требуемое и направился к кассе. За прилавком стояли двое служащих, внешний вид которых более соответствовал занятию родео, чем нахождению за прилавком этого заведения. Старший из них был седым, с усами и с задубелой кожей человека, привыкшего к жизни на ранчо. Оба носили обязательные бейсбольные кепки. Вылитые ковбои.

Когда я положил журнал, старший из служащих на некоторое время поднял глаза и сказал, типично по-ковбойски растягивая слова: «Угу, Линукс? Вот что я скажу: когда Билли пойдет вниз, с ним это произойдет быстро!» Разумеется, речь шла о широко комментировавшемся соревновании между Linux и Microsoft Windows Билла Гейтса, усилившемся благодаря движению open source.

В другое время эти двое могли бы рассуждать о коровах и револьверах, сидя за чашкой крепкого кофе. Однако каким-то странным образом они стали страстными защитниками Linux – операционной системы с открытым исходным кодом. Подобрав отвалившуюся челюсть, я вступил с ними в оживленный разговор о Linux, Microsoft, Python и всяких открытых вещах. Можно сказать, мы славно поболтали.

Я не хочу отдавать предпочтение одной операционной системе перед другой: у каждой из них есть свои преимущества, а Python одинаково хорошо работает на той и другой платформе (в действительности вошедшие в книгу примеры разрабатывались на обеих системах). Но меня поразило, что мысль, которую разработчики программного обеспечения часто считают само собой разумеющейся, имела такое глубокое влияние на средних американцев. Это вселяет в меня большие надежды: если технология действительно должна повысить качество жизни в следующем тысячелетии, нам нужно побольше таких ковбоев.

1

Знакомство с Python

«А теперь нечто совершенно новое»

Эта книга посвящена использованию Python, объектно-ориентированного языка программирования весьма высокого уровня с открытым исходным кодом[1], предназначенного для оптимизации скорости разработки. Несмотря на то что Python в полной мере является языком общего назначения, его часто называют объектно-ориентированным *языком сценариев* – отчасти просто из-за легкости использования, отчасти из-за частого применения при «оркестровке» или «склеивании» в приложении других программных компонентов.

Если вы новичок в Python, то, возможно, все же слышали где-нибудь об этом языке, но точно не знаете, в чем его особенности. В помощь начинающим эта глава неформально знакомит с характеристиками Python и той ролью, которую он играет. В этом знакомстве будет больше пользы, если познакомиться с реальными программами Python, но прежде чем заняться деревьями, взглянем на лес в целом.

В предисловии говорилось, что для Python характерны такие понятия, как качество, продуктивность, переносимость и интеграция. Поскольку в эти четыре слова сведено большинство причин использования Python, хотелось бы определить их более подробно:

Качество

С помощью Python легко писать программы, которые можно повторно использовать и сопровождать. Он специально спроектирован так, чтобы повысить стандарт качества разработок в области языков сценариев. Понятный синтаксис Python и ясная архитектура почти вынуждают программистов писать код, отличающийся хорошей читаемостью, что важно для программного обеспечения, которое могут изменять другие программисты. Язык Python действительно создает впечатление планомерности, а не накопления характеристик. Кроме того, Python обладает хорошими возможностями для поддержки современных методологий повторного использования программного обеспечения. В действительности создание высококачественных компонентов Python, которые могут применяться во многих контекстах, происходит почти автоматически.

Продуктивность

Python оптимизирован по скорости разработки программного обеспечения. На нем можно быстро писать программы, потому что детали, которые в языках более

---

[1] Системы с открытым исходным кодом иногда называются *freeware*, поскольку их исходный код распространяется бесплатно и контролируется сообществом. Однако пусть это понятие не вводит вас в заблуждение, так как при наличии на сегодняшний день в этом сообществе примерно полумиллиона пользователей Python имеет очень хорошую поддержку.

низкого уровня вы должны кодировать явно, берет на себя интерпретатор. В сценариях Python не найти таких вещей, как объявление типов, управление памятью или процедуры компиляции. Но быстрая первоначальная разработка – лишь одна составляющая продуктивности. В реальном мире программисты пишут код как для компьютера, который будет его выполнять, так и для других программистов, которые будут его читать и сопровождать. Поскольку синтаксис Python напоминает *исполняемый псевдокод*, создаваемые программы легко понять через длительное время после того, как они написаны. Кроме того, Python поддерживает (но не навязывает) продвинутые парадигмы, например объектно-ориентированное программирование, которое еще больше повышает производительность и сокращает сроки разработки.

Переносимость

Большинство программ Python без изменений работает почти на всех используемых на сегодняшний день компьютерных системах. Фактически программы Python сегодня работают везде – от мэйнфреймов IBM и суперкомпьютеров Cray до ноутбуков и карманных PDA. Хотя для некоторых платформ существуют непереносимые расширения, базовый язык Python и его библиотеки нейтральны в отношении платформы. Например, сценарии Python, разработанные на Linux, в большинстве своем могут сразу выполняться под Windows, и наоборот – просто скопируйте сценарий. Более того, программа с графическим интерфейсом пользователя (GUI), написанная со стандартной библиотекой Python Tkinter, будет выполняться в системе X Windows, Microsoft Windows и Macintosh, выглядя при этом в каждом случае как родная, без всякой модификации исходного кода.

Интеграция

Python спроектирован с возможностью интеграции с другими инструментами. Написанные на Python программы можно легко смешивать с другими компонентами системы и управлять ими. Например, сценарии Python могут сейчас обращаться к имеющимся библиотекам C и C++, работать с классами Java, интегрироваться с компонентами COM и CORBA и многое другое. Кроме того, программы, написанные на других языках, могут с такой же легкостью запускать сценарии Python, вызывая функции API C и Java, обращаясь к COM-серверам, написанным на Python, и т. д. Python – это не запертый ящик.

В эпоху неуклонно сжимающихся графиков разработки, более быстрых машин и гетерогенных приложений такие достоинства оказываются сильными союзниками как в малых, так и в больших проектах. Естественно, есть и другие особенности Python, привлекательные для разработчиков, например легкость изучения для разработчиков и пользователей, библиотеки готовых инструментов, минимизирующие объем предварительных разработок, и полная бесплатность, сокращающая расходы на разработку продукта и его развертывание.

Но самым привлекательным и определяющим качеством Python является, вероятно, его нацеленность на продуктивность. Когда я пишу эти строки, главная проблема, существующая в мире разработки программного обеспечения, состоит не просто в том, чтобы быстро писать программы, а в том, чтобы найти разработчиков, у которых вообще есть время для разработки программ. Время, затрачиваемое разработчиками, приобрело первостепенное значение, гораздо более важное, чем скорость выполнения. Существующих проектов просто больше, чем программистов, которыми можно их укомплектовать.

В качестве языка, оптимизированного для продуктивности разработчиков, Python дает правильный ответ на вопросы, возникающие в мире разработки программного обеспечения. Программируя на Python, разработчики не только могут быстро реали-

зовывать системы, но получаемые системы легче сопровождать, они переносимы и просто интегрируются с другими программными компонентами.

История жизни Python

Python был изобретен примерно в 1990 году Гвидо ван Россумом во время его работы в CWI[1] в Амстердаме. Рептилии здесь не в счет: язык назван в честь комедийного сериала Би-Би-Си «Monty Python's Flying Circus», любителем которого является Гвидо (смотрите следующую глупую врезку). Гвидо занимался распределенной операционной системой Amoeba и языком ABC. В действительности первоначальной мотивацией для Python было создание развитого языка сценариев для системы Amoeba.

Но конструкция Python оказалась настолько общей, что его можно было применять в разнообразных сферах. Сейчас он используется сотнями тысяч специалистов по всему свету во все новых и новых областях. Сегодня компании используют Python в коммерческих продуктах для таких задач, как тестирование микросхем и плат, разработка GUI, поиск в Сети, анимация в фильмах, пользовательская настройка библиотек клас-

Что в имени тебе моем?

Python получил свое название благодаря британскому комедийному телесериалу 1970-х «Monty Python's Flying Circus». Фольклор Python утверждает, что Гвидо ван Россум, создатель Python, смотрел повторные показы этого сериала примерно в то же время, когда ему понадобилось дать имя новому языку, который он разрабатывал. И, как говорят в шоу-бизнесе, «случившееся навсегда вошло в историю».

Такая наследственность служит причиной частого появления в примерах и тексте ссылок на комедийную игру. Например, имя «Spam» особо относится к пользователям Python, а конфронтации иногда называют «Испанской инквизицией». Как правило, когда пользователь Python начинает произносить фразы, не имеющие отношения к реальности, они оказываются заимствованными из сериала или фильмов «Monty Python». Некоторые из этих фраз могут встретиться даже в данной книге. Необязательно, конечно, бежать и брать в прокате «The Meaning of Life» или «The Holy Grail», чтобы делать что-то полезное на Python, но и хуже от этого не будет.

Хотя имя «Python» быстро прижилось, оно послужило причиной некоторых интересных побочных эффектов. Например, когда в 1994 году возникла телеконференция по Python *comp.lang.python*, первые несколько недель она была почти полностью оккупирована людьми, желавшими обсуждать темы, касающиеся телевизионной постановки. Позднее в специальном приложении к журналу *Linux Journal* красовались фотографии Гвидо, облаченного в обязательную «красивую красную форму».

Иногда в конференции Python все еще появляются случайные письма от поклонников сериала. Например, в одном письме невинно предлагалось обмениваться сценариями Monty Python с другими поклонниками. Если бы автор понимал назначение этого форума, то хотя бы указал, выполняются его сценарии под DOS или Unix.

---

[1] Национальном научно-исследовательском институте математики и информатики Нидерландов. – *Примеч. пер.*

сов C++ и многих других.[1] На самом деле, поскольку Python в полной мере является языком общего назначения, область его применения ограничивается только областью применения компьютеров в целом.

Со времени своего появления в свободном доступе в 1991 году Python продолжает привлекать верных сторонников и в 1994 году породил в Интернет специальную телеконференцию, *comp.lang.python*. А когда в 1995 году писалось первое издание этой книги, дебютировала домашняя страничка Python в WWW на *http://www.python.org*. Она остается официальным местом для всего, связанного с Python.

Для руководства развитием Python со временем образовались организации, нацеленные на поддержку разработчиков Python. В их число входят Python Software Activity (PSA), содействующая конференциям и веб-сайтам по Python, и Python Consortium, образованный организациями, заинтересованными в содействии развитию Python. Когда я пишу эти строки, будущее PSA неясно, но в первые годы она способствовала поддержке Python.

Сегодня Гвидо и несколько других главных разработчиков Python работают в компании под названием Digital Creations и занимаются разработками на Python. Digital Creations, располагающаяся в Вирджинии, является родиной основанного на Python инструментального набора создания веб-приложений Zope (смотрите *http://www.zope.org*). Однако владеет и управляет языком Python независимая организация, и он остается общественно управляемой, подлинной системой open source.

Есть и другие компании, также использующие Python в своих проектах. Например, ActiveState и PythonWare разрабатывают инструменты Python, O'Reilly (издатель данной книги) и компания под названием Foretech организуют ежегодные конференции по Python, а O'Reilly управляет дополнительным веб-сайтом Python (посмотрите Python DevCenter в сети O'Reilly Network на *http://www.oreillynet.com/python*). O'Reilly Python Conference проводится в составе ежегодной конференции по программному обеспечению с открытым кодом (Open Source Software Convention). Хотя в мире профессиональных организаций и компаний изменения происходят чаще, чем в публикуемых книгах, мало сомнений, что язык Python продолжит удовлетворять потребности сообщества его пользователей.

Обязательный список характеристик

Одним из способов описания языка является перечисление его характеристик. Конечно, он станет более осмысленным, когда вы посмотрите на Python в работе; сейчас же мне приходится говорить абстрактно. И то, что делает Python самим собой, это взаимодействие данных характеристик между собой. Но, рассмотрев некоторые атрибуты Python, будет легче дать ему определение. В табл. 1.1 перечислены некоторые наиболее упоминаемые причины привлекательности Python.

Таблица 1.1. Характеристики языка Python

Характеристики	Преимущества
Отсутствие этапов компиляции или компоновки	Короткий цикл разработки
Отсутствие объявлений типов	Более простые, короткие и гибкие программы

[1] Примеры других компаний, использующих Python, есть в предисловии. Более полный список коммерческих приложений можно также найти на *http://www.python.org*.

Таблица 1.1 (продолжение)

Характеристики	Преимущества
Автоматическое управление памятью	Благодаря уборке мусора не нужен код, ведущий учет памяти
Типы данных и операции высокого уровня	Быстрая разработка с помощью встроенных типов объектов
Объектно-ориентированное программирование	Повторное использование кода, интеграция с C++, Java и COM
Встраивание и расширение в C	Оптимизация, подгонка, системный «клей»
Классы, модули, исключительные ситуации	Поддержка модульного программирования
Простой понятный синтаксис и архитектура	Легкость чтения, сопровождения и изучения
Динамическая загрузка модулей C	Упрощенность расширений, меньшие двоичные файлы
Динамическая перезагрузка модулей Python	Можно модифицировать программы, не останавливая их
Универсальная «первоклассная» (first-class) модель объектов	Меньше ограничений и особых правил
Построение программ на этапе исполнения	Кодирование конечным пользователем при возникновении непредвиденных потребностей
Интерактивная динамическая природа	Наращиваемые разработка и тестирование
Доступ к данным интерпретатора	Метапрограммирование, интроспективные объекты
Широкая переносимость интерпретатора	Программирование для разных платформ без отдельных переносов
Компиляция в переносимый байт-код	Скорость выполнения, защита исходного кода
Стандартная переносимая структура GUI	Сценарии Tkinter выполняются на X, Windows и Mac
Поддержка стандартных протоколов Интернета	Легкость доступа к e-mail, FTP, HTTP, CGI и т. д.
Стандартные переносимые системные вызовы	Системные сценарии, безразличные к платформе
Встроенные библиотеки и библиотеки сторонних разработчиков	Большая коллекция готовых программных компонентов
Подлинная открытость исходного кода	Можно бесплатно встраивать и распространять

По правде говоря, в действительности Python является конгломератом функций, заимствованных из других языков. В нем есть элементы, взятые из C, C++, Modula-3, ABC, Icon и других языков. Например, модули пришли в Python из Modula, а опера-

ция среза (slicing) из Icon (по крайней мере, насколько можно вспомнить). А предшествующий опыт Гвидо послужил тому, что Python заимствовал многие идеи ABC, но в него были добавлены и собственные практические возможности, например поддержка расширений на C.

Где хорош Python?

Поскольку области применения Python очень разнообразны, почти невозможно дать надежный ответ на этот вопрос. В общих чертах, любое приложение, которое может выиграть от использования языка, оптимизированного для скорости разработки, может послужить хорошей областью применения Python. С учетом все более сжатых сроков разработки программного обеспечения, это очень широкая категория.

Более точный ответ сформулировать нелегко. Некоторые, например, используют Python как язык встраиваемых расширений, в то время как другие применяют его исключительно в качестве самостоятельного инструмента программирования. В какой-то степени вся данная книга должна ответить на этот самый вопрос: в ней исследуются некоторые наиболее часто выполняемые Python роли. Но сейчас приведем сводку некоторых наиболее распространенных на сегодняшний день способов применения Python:

Системные утилиты

Переносимые инструменты командной строки, тестирование систем.

Сценарии для Интернета

Веб-сайты CGI, апплеты Java, XML, ASP, инструменты электронной почты.

Графические интерфейсы пользователя

С такими API, как Tk, MFC, Gnome, KDE.

Интеграция компонентов

Клиенты библиотек C/C++, настройка продуктов под пользователя.

Доступ к базам данных

Постоянное хранение объектов, интерфейсы баз данных SQL.

Программирование распределенных приложений

С такими API клиент/сервер, как CORBA, COM.

Быстрое создание прототипов и разработка

Одноразовые или поставляемые прототипы.

Языковые модули

Использование Python вместо специализированных синтаксических анализаторов.

И другие

Обработка изображений, численное программирование, ИИ и т. д.

С другой стороны, Python вообще не привязан к какой-либо конкретной области приложений. Например, поддержка интеграции в Python делает его полезным почти в любой системе, которая может выиграть от программируемого оконечного интерфейса. Абстрактно говоря, Python предоставляет сервисы, соединяющие области приложений. Python – это:

- Динамический язык программирования для ситуаций, в которых этап компиляции/сборки невозможен (настройка на месте) или неудобен (создание прототипа, быстрая разработка, системные утилиты)

«Автобусы признаны опасными»

Упоминавшаяся выше организация PSA первоначально была образована как ответ на когда-то давно возникшее в телеконференции Python обсуждение полусерьезного вопроса: «Что будет, если Гвидо попадет под автобус?»

В настоящее время Гвидо ван Россум по-прежнему является верховным арбитром для предложений о внесении в Python изменений, но масса пользователей Python помогает поддерживать язык, работать над расширениями, исправлять ошибки и т. д. Фактически разработка Python сейчас является совершенно открытым процессом: изучать новейшие файлы с исходным кодом или посылать патчи может каждый посетитель веб-сайта (детали смотрите на *http://www.python.org*).

Являясь пакетом с открытым исходным кодом, Python действительно разрабатывается очень большим составом программистов, которые согласованно работают, находясь в разных частях света. С учетом популярности Python нападение со стороны автобуса уже не кажется таким опасным, как раньше, но Гвидо, возможно, считает иначе.

- Мощный, но простой язык, предназначенный для ускоренной разработки и ситуаций, в которых сложность более крупных языков может стать обременительной (прототипирование, кодирование конечным пользователем)
- Обобщенный языковый инструмент для таких ситуаций, в которых иначе потребовалось бы изобретать и реализовывать очередной «язычок» (программируемые системные интерфейсы, инструменты конфигурирования)

При наличии таких общих свойств Python может применяться в любой нужной области путем расширения его библиотеками, относящимися к области применения, встраивания в приложение или самостоятельного использования. Например, роль Python как языка для системных средств обусловлена как его встроенными интерфейсами к сервисам операционной системы, так и самим языком. В действительности, поскольку Python создавался с учетом возможностей интеграции, он естественным образом породил растущую библиотеку расширений и инструментальных средств, доступных разработчикам на Python в виде готовых компонентов. В табл. 1.2 перечислены лишь некоторые из них, о большинстве подробнее рассказывается в данной книге или на веб-сайте Python.

Таблица 1.2. Некоторые популярные инструментальные средства и расширения Python

Область применения	Расширения
Системное программирование	Сокеты, потоки, сигналы, конвейеры, вызовы RPC, интерфейс к POSIX
Графические интерфейсы пользователя	Tk, PMW, MFC, X11, wxPython, KDE, Gnome
Интерфейсы баз данных	Oracle, Sybase, PostGres, mSQL, постоянное хранение, dbm
Инструменты Microsoft Windows	MFC, COM, ActiveX, ASP, ODBC, .NET
Инструменты Интернета	JPython, средства CGI, анализаторы HTML/XML, средства e-mail, Zope

Таблица 1.2 (продолжение)

Область применения	Расширения
Распределенные объекты	DCOM, CORBA, ILU, Fnorb
Другие популярные инструменты	SWIG, PIL, регулярные выражения, NumPy, криптография

Для чего Python не годится?

Опять-таки, если говорить честно, некоторые задачи находятся за пределами возможностей Python. Как и все динамические языки, Python (в настоящей его реализации) не настолько быстр и эффективен, как статические компилируемые языки типа C. Во многих областях эта разница не имеет значения: в программах, занятых в основном взаимодействием с пользователем или передачей данных по сетям, производительности Python обычно более чем достаточно для удовлетворения потребностей всего приложения. Но в некоторых областях производительность сохраняет первостепенное значение.

Поскольку в настоящее время Python является интерпретируемым языком,[1] одного только Python обычно оказывается недостаточно для создания компонентов, критическим свойством которых является быстродействие. Операции, интенсивно выполняющие вычисления, можно реализовывать как компилированные *расширения* (*extensions*) Python и кодировать на языке низкого уровня типа C. Python нельзя использовать в качестве единственного языка, на котором реализуются такие компоненты, но он хорошо работает в качестве клиентского интерфейса сценариев для них.

Например, поддержка численного программирования и обработки изображений введена в Python путем объединения оптимизированных расширений с интерфейсом языка Python. В такой системе после того, как разработаны оптимизированные расширения, программирование происходит в основном на более высоком уровне сценариев Python. В итоге получается инструмент численного программирования, который одновременно эффективен и прост в использовании.

Кроме того, в таких областях применения Python по-прежнему можно использовать для создания прототипов. Система может быть сначала реализована на Python, а затем перед поставкой полностью или частично переведена на такой язык, как C. У C и Python есть свои сильные стороны и задачи. Гибридный подход, при котором C используется для модулей с интенсивными вычислениями, а Python для создания прототипов и клиентских интерфейсов, может усилить выгоды от использования обоих языков.

В некотором смысле, Python находит компромисс между эффективностью и гибкостью тем, что вообще не ищет его. Есть язык, оптимизированный для простоты ис-

---

[1] Python «интерпретируется» таким же образом, как Java: исходный код Python автоматически компилируется (транслируется) в промежуточный формат, называемый «байт-код», который выполняется затем виртуальной машиной Python (то есть системой исполнения Python). Благодаря этому сценарии Python лучше переносятся и выполняются быстрее, чем чистый интерпретатор, который выполняет исходный или древовидный код. Но в результате Python также становится более медленным, чем подлинные компиляторы, транслирующие исходный код в двоичный машинный код для конкретного ЦП. Имейте, однако, в виду, что эти детали касаются стандартной реализации Python; JPython (или. «Jython») компилирует сценарии Python в байт-код Java, а новая реализация для C#/.NET компилирует сценарии Python в двоичные *.exe*-файлы. Оптимизирующий компилятор Python может разрушить все высказанные в данной главе опасения относительно производительности (будем на это надеяться).

пользования, и есть средства интеграции его с другими языками. Сочетая компоненты, написанные на Python и компилируемых языках типа C и C++, разработчики могут выбирать нужную пропорцию простоты использования и эффективности для каждого конкретного случая. Хотя навряд ли Python когда-либо станет таким же быстрым, как C, в большинстве сегодняшних программных проектов скорость разработки, обеспечиваемая Python, не менее важна, чем скорость выполнения, обеспечиваемая C.

О честности в рекламе

В конце книги мы вернемся к некоторым важнейшим идеям, представленным в этой главе, после того как изучим Python в действии. Однако я хочу заранее подчеркнуть, что занимаюсь информатикой, а не маркетингом. Я намерен соблюдать в данной книге предельную откровенность как в отношении достоинств Python, так и его недостатков. Несмотря на то что Python является одним из самых простых в использовании языков программирования из всех когда-либо созданных, некоторые ловушки действительно существуют, и о них будет говориться в книге.

Сразу и начнем. Самая большая ловушка, о которой, возможно, следует знать, такова: *с помощью Python чрезвычайно легко быстро создать плохую конструкцию*. Это действительно проблема. Поскольку писать программы на Python так просто и быстро в сравнении с обычными языками, очень легко увлечься актом программирования, уделяя недостаточно внимания самой задаче, которую на самом деле нужно решить.

В действительности Python может оказаться явно соблазнительным – настолько, что потребуется сознательно противостоять искушению быстро набросать на Python программу, которая работоспособна и, как можно утверждать, «крута», но с ней вы окажетесь столь же далеко от сопровождаемой реализации первоначального замысла, как вы были в начале работы. Естественные задержки, присущие разработке на компилируемом языке, – исправление ошибок, выявленных компилятором, компоновка библиотек и тому подобное, – отсутствуют при работе с Python и не помогут вам притормозить и подумать.

Не всегда все так плохо. В большинстве случаев первые наброски на скорую руку послужат переходу к лучшим решениям, которые вы впоследствии сохраните. Но имейте в виду: даже такой язык быстрой разработки приложений, как Python, *не может заменить необходимость думать*. Прежде чем начать вводить код, лучше сесть и подумать. По крайней мере, на сегодняшний день ни одному языку программирования не удалось сделать ненужным интеллект.

I

Системные интерфейсы

В этой первой технической части книги представлены инструменты Python для системного программирования – интерфейсы к сервисам базовой операционной системы, а также к контексту выполнения программы. Часть I состоит из следующих глав:

- Глава 2 «Системные инструменты». Эта глава всесторонне рассматривает часто используемые инструменты системных интерфейсов и показывает, как работать с потоками, файлами, каталогами, аргументами командной строки, переменными оболочки и многим другим. Глава ведет изложение неторопливо и частично предназначена служить справочником.
- Глава 3 «Системные средства параллельного выполнения». Служит введением в поддержку библиотекой Python параллельного выполнения программ. Здесь вы найдете описание потоков выполнения (нитей), ветвления процессов, каналов, сигналов и тому подобного.
- Глава 4 «Более крупные системные примеры, часть 1» и глава 5 «Более крупные системные примеры, часть 2». В этих главах собраны типичные примеры системного программирования, основанные на материале первых двух глав. Среди прочего, представленные здесь сценарии Python демонстрируют, как разрезать и соединять файлы, сравнивать и копировать каталоги, генерировать веб-страницы по шаблонам и запускать программы и веб-броузеры переносимым образом. Вторая из глав сосредоточена на более сложных примерах файлов и каталогов; в первой представлены разнообразные конкретные примеры использования системных инструментов.

Хотя в этой части книги особое значение придается задачам системного программирования, показанные в ней средства являются универсальными и часто используются в последующих главах.