1 Python: с чем его едят

Начнем с одной небольшой тайны и ее разгадки. Что, по-вашему, означают следующие две строки?

```
(Ряд 1): (RS) K18, ssk, k1, turn work.
(Ряд 2): (WS) Sl 1 pwise, p5, p2tog, p1, turn.
```

Выглядит как какая-то компьютерная программа. На самом деле это схема для вязания, а если точнее, фрагмент, который описывает, как связать пятку носка. Для меня эти строки имеют не больше смысла, чем кроссворд из газеты New York Times для моего кота, но моя жена понимает их совершенно точно. Если вы вяжете, то тоже их поймете.

Рассмотрим еще один пример. Вы сразу поймете его предназначение, хотя и не сразу сможете распознать результат:

```
½ столовой ложки масла или маргарина;
½ столовой ложки сливок;
2 ½ стакана муки;
1 чайная ложка соли;
1 чайная ложка сахара;
4 стакана картофельного пюре (охлажденного).
Перед тем как добавить муку, убедитесь, что все ингредиенты охлаждены.
Смешайте все ингредиенты.
Тщательно замесите.
Сделайте 20 шариков. Держите их охлажденными до следующего этапа.
Для каждого шарика разровняйте муку на тряпочке.
Раскатайте шарик при помощи рифленой скалки.
Жарьте на сковороде до подрумянивания.
Переверните и обжарьте другую сторону.
```

Даже если вы не готовите, вы сможете распознать *кулинарный рецепт*: список продуктов, за которым следуют указания по приготовлению. Но что получится в итоге? Это *лефсе*, норвежский деликатес, который напоминает тортилью. Полейте блюдо маслом, вареньем или чем-нибудь еще, сверните и наслаждайтесь.

Схема для вязания и рецепт имеют несколько похожих моментов:

- фиксированный словарь, состоящий из слов, аббревиатур и символов. Некоторые могут быть знакомы, другие же покрыты тайной;
- правила, описывающие, что и где можно говорить, — синтаксис;
- последовательность операций, которые должны быть выполнены по порядку;
- в некоторых случаях — повторение определенных операций *(цикл)*, например способ приготовления каждого кусочка лефсе;
- в некоторых случаях — ссылка на другую последовательность операций (говоря компьютерными терминами, *функция*). Например, когда вы прочтете приведенный выше рецепт, вам может понадобиться рецепт приготовления картофельного пюре;
- предполагаемое знание контекста. Рецепт подразумевает, что вы знаете, что такое вода и как ее кипятить. Схема для вязания подразумевает, что вы умеете вязать, не укалываясь слишком часто;
- ожидаемый результат. В наших примерах результатом будет предмет для ног и предмет для желудка. Главное — не перепутать.

Все эти идеи вы можете встретить и в компьютерных программах. Я воспользовался этими «непрограммами», чтобы показать, что программы не так страшны, как может показаться. Нужно всего лишь выучить верные слова и правила.

Теперь оставим этих дублеров и рассмотрим настоящую программу. Что она делает?

```
for countdown in 5, 4, 3, 2, 1, "hey!":
    print(countdown)
```

Если вы считаете, что это программа, написанная на языке программирования Python, которая выводит на экран следующее:

```
5
4
3
2
1
hey!
```

то вы знаете, что язык программирования Python выучить проще, чем понять рецепт или схему для вязания. К тому же вы можете тренироваться писать на языке программирования Python, сидя за удобным и безопасным столом, избегая опасностей вроде горячей воды и острых палочек.

Программа, написанная на языке программирования Python, содержит несколько специальных слов и символов — `for`, `in`, `print`, запятые, точки с запятой, скобки

и т. д., — которые являются важной частью синтаксиса языка. Хорошая новость заключается в том, что язык программирования Python имеет более приятный и менее объемный синтаксис, чем большинство других языков программирования. Он кажется более естественным — почти как рецепт.

Вот еще одна небольшая программа, написанная на языке программирования Python, которая выбирает новостные клише из *списка* и выводит их на экран:

```
cliches = [
    "At the end of the day",
    "Having said that",
    "The fact of the matter is",
    "Be that as it may",
    "The bottom line is",
    "If you will",
    ]
print(cliches[3])
```

Эта программа выведет четвертое клише:

```
Be that as it may
```

Списки — вроде `cliches` — представляют собой последовательность значений, доступ к которым осуществляется с использованием *смещения* от начала списка. Смещение для первого элемента списка равно 0, а для четвертого — 3.

Люди считают с единицы, поэтому может показаться странным считать с нуля. При программировании удобнее оперировать смещениями, чем позициями.

Списки широко распространены в языке программирования Python. О том, как ими пользоваться, будет рассказано в главе 3.

Далее приведена еще одна программа, которая также выводит цитату, но в этот раз цитата выбирается в зависимости от того, кто ее произнес, а не с помощью позиции в списке:

```
quotes = {

    "Moe": "A wise guy, huh?",
    "Larry": "Ow!",
    "Curly": "Nyuk nyuk!",
    }
stooge = "Curly"
print(stooge, "says:", quotes[stooge])
```

Если вы запустите эту небольшую программу, она выведет следующее:

```
Curly says: Nyuk nyuk!
```

`quotes` — это *словарь*, коллекция уникальных *ключей* (в этом примере ключом является имя участника Stooge) и связанных с ними *значений* (в этом примере — значимая цитата участника Stooge). Используя словарь, вы можете сохранять элементы и выполнять их поиск по именам, что часто удобнее, чем список. Более подробно о словарях можно прочитать в главе 3.

В примере с клише для создания списка используются квадратные скобки (`[` и `]`), а в примере со Stooge для создания словаря — фигурные скобки (`{` и `}`). Все это — примеры синтаксиса языка программирования Python, и в нескольких следующих главах вы увидите гораздо больше.

А теперь рассмотрим кое-что совершенно иное: в примере 1.1 показана программа, написанная на языке программирования Python, которая выполняет несколько более сложных задач. Не ждите, что сразу поймете, как работает программа, — для этого и предназначена данная книга. Мы рассматриваем пример для того, чтобы увидеть и прочувствовать обычную нетривиальную программу, написанную на языке программирования Python. Если вы знаете другие языки программирования, то можете сравнить их с Python прямо сейчас.

В примере ниже происходит подключение к сайту YouTube и получение информации о видеороликах, имеющих в данный момент самые высокие оценки. Если бы там получали обычную веб-страницу, заполненную текстом, отформатированным как HTML, было бы трудно получить всю необходимую информацию (я говорю об *извлечении данных* в разделе «Веб-сервисы и автоматизация» главы 9). Вместо этого пример получает данные, представленные в формате JSON, который предназначен для обработки компьютером. JSON, или JavaScript Object Notation — это читабельный для человека текстовый формат, который описывает типы и значения, а также выстраивает значения в определенном порядке. Он немного похож на языки программирования и уже стал популярным способом обмена данными между разными языками программирования и системами. Вы можете прочитать о JSON больше в подразделе «JSON» раздела «Структурированные текстовые файлы» главы 8.

Программы, написанные на языке Python, могут преобразовывать текст формата JSON в *структуры данных* — их вы увидите в следующих двух главах, — как если бы вы написали программу, чтобы создавать их самостоятельно. В полученном от YouTube ответе данных очень много, поэтому в рамках этого примера я выведу названия лишь первых шести видеороликов. И вновь перед вами полноценная программа, которую вы можете запустить самостоятельно.

```
import json
from urllib.request import urlopen
url = "https://gdata.youtube.com/feeds/api/standardfeeds/top_rated?alt=json"
response = urlopen(url)
contents = response.read()
text = contents.decode('utf8')
data = json.loads(text)
for video in data['feed']['entry'][0:6]:
    print(video['title']['$t'])
```

Когда я запускал эту программу в последний раз, я получил следующий результат:

```
Evolution of Dance – By Judson Laipply
Linkin Park – Numb
Potter Puppet Pals: The Mysterious Ticking Noise
"Chocolate Rain" Original Song by Tay Zonday
Charlie bit my finger – again !
The Mean Kitty Song
```

Эта небольшая программа, написанная на языке Python, выполняет много работы с помощью всего лишь девяти строк. Если вы не знаете всех этих терминов, не волнуйтесь — вы познакомитесь с ними в следующих главах.

- Строка 1: импортируем весь код из *стандартной библиотеки*, которая называется json.
- Строка 2: импортируем только функцию `urlopen` из стандартной библиотеки urllib.
- Строка 3: присваиваем URL сайта YouTube переменной `url`.
- Строка 4: соединяемся с веб-сервером, расположенным по этому адресу, и запрашиваем определенный *веб-сервис*.
- Строка 5: получаем ответ и присваиваем его переменной `contents`.
- Строка 6: *дешифруем* содержимое переменной `contents` в текстовую строку формата JSON и присваиваем ее переменной `text`.
- Строка 7: преобразуем переменную `text` в `data` — структуру данных языка Python, предназначенную для работы с видео.
- Строка 8: получаем информацию для одного видеоролика единовременно в переменную `video`.
- Строка 9: используем двухуровневый словарь (`data['feed']['entry']`) и функцию `slice([0:6])`.
- Строка 10: используем функцию `print`, чтобы вывести на экран только название видеоролика.

Информация о видеоролике представляет собой различные структуры данных, которые вы уже видели; все они демонстрируются в главе 3.

В предыдущем примере мы задействовали стандартные библиотечные модули (программы, включаемые в Python при установке), но в них нет ничего таинственного. Следующий фрагмент кода показывает переписанный пример, использующий внешний пакет ПО для Python, который называется `requests`:

```
import requests
url = "https://gdata.youtube.com/feeds/api/standardfeeds/top_rated?alt=json"
response = requests.get(url)
data = response.json()
for video in data['feed']['entry'][0:6]:
    print(video['title']['$t'])
```

Новая версия содержит всего шесть строк и, я полагаю, более читабельна для большинства людей. Я расскажу гораздо больше о `requests` и других авторских программах для Python в главе 5.

Python в реальном мире

Стоит ли тратить на изучение Python время и силы? Может быть, это игра в бирюльки? Язык программирования Python существует примерно с 1991 года (он появился раньше Java) и является одним из десяти самых популярных языков программирования. Людям платят деньги за то, что они пишут программы на Python, которыми мы пользуемся каждый день, — Google, YouTube, Dropbox, Netflix и Hulu. Я использовал Python для создания как поискового устройства для электронной почты, так и интернет-магазина. Python имеет репутацию высокопроизводительного языка программирования, что нравится динамично развивающимся организациям.

Вы можете найти множество приложений, написанных на Python, например:

- командную строку на мониторе или в окне терминала;
- пользовательские интерфейсы, включая сетевые;
- веб-приложения, как клиентские, так и серверные;
- бэкэнд-серверы, поддерживающие крупные популярные сайты;
- *облака* (серверы, управляемые сторонними организациями);
- приложения для мобильных устройств;
- приложения для встроенных устройств.

Программы, написанные на языке программирования Python, могут быть как одноразовыми *сценариями* — вы видели их ранее в этой главе, — так и сложными системами, содержащими миллионы строк. Мы рассмотрим применение языка программирования Python для создания сайтов, системного администрирования и манипулирования данными. Рассмотрим также использование Python в искусстве, науке и бизнесе.

Python против языка X

Насколько Python хорош по сравнению с другими языками программирования? Где и когда следует использовать тот или иной язык? В этом разделе я приведу примеры кода, написанные на других языках, чтобы вы могли понять, с чем соревнуется Python. Вы **не обязаны** понимать каждый из этих фрагментов, если не работали с этими языками. (Когда вы увидите последний фрагмент, написанный на Python, то почувствуете облегчение из-за того, что не работали с некоторыми

языками.) Если вам интересен только Python, вы ничего не пропустите, если не будете читать этот раздел.

Каждая программа должна вывести число и немного рассказать о языке, на котором она написана.

Если вы пользуетесь терминалом или терминальным окном, программа, которая читает то, что вы вводите, выполняет это и отображает результат, называется программой-*оболочкой*. Оболочка операционной системы Windows называется `cmd`, она выполняет *пакетные* файлы, имеющие расширение `.bat`. Для Linux и других операционных систем семейства Unix (включая Mac OS X) существует множество программ-оболочек, самая популярная из которых называется `bash` или `sh`. Оболочка обладает небольшими возможностями вроде выполнения простой логики и разворачивания символа-джокера наподобие * в полноценные имена файлов. Вы можете сохранять команды в файлы, которые называются *сценариями оболочки*, и выполнять их позже. Эти программы могли быть самыми первыми в вашей карьере программиста. Проблема заключается в том, что со сценариями оболочки трудно работать, если они содержат как минимум несколько сотен строк, а сами сценарии выполняются гораздо медленнее, чем программы, написанные на других языках. В следующем фрагменте кода демонстрируется небольшая программа-оболочка:

```
#!/bin/sh
language=0
echo "Language $language: I am the shell. So there."
```

Если вы сохраните этот файл под именем `meh.sh` и запустите его с помощью команды `sh meh.sh`, на экране увидите следующее:

```
Language 0: I am the shell. So there.
```

Старые добрые C и C++ являются довольно низкоуровневыми языками программирования, которыми пользуются в том случае, когда важна скорость. Их труднее выучить, и вам придется отслеживать множество деталей, что может привести к падениям программы и проблемам, которые трудно диагностировать. Так выглядит небольшая программа на языке C:

```
#include <stdio.h>
int main(int argc, char *argv[]) {
    int language = 1;
    printf("Language %d: I am C! Behold me and tremble!\n", language);
    return 0;
}
```

C++ происходит из одного семейства с C, но имеет несколько отличительных особенностей:

```
#include <iostream>
using namespace std;
```

```
int main() {
    int language = 2;
    cout << "Language " << language << \
       ": I am C++!  Pay no attention to that C behind the curtain!" << \
       endl;
    return(0);
}
```

Java и C# являются преемниками языков C и C++, избавленными от некоторых проблем предшественников. Однако они в некоторой степени избыточны и ограничительны. Следующий пример написан на Java:

```
public class Overlord {
    public static void main (String[] args) {
        int language = 3;
        System.out.format("Language %d: I am Java! Scarier than C!\n", language);
    }
}
```

Если вы никогда не писали программ ни на одном из этих языков, вам может быть интересно, **что все это такое**. Некоторые языки нагружены весомым синтаксическим багажом. Их иногда называют статическими языками, поскольку они требуют, чтобы вы указали компьютеру некоторые низкоуровневые детали. Позвольте мне объяснить.

Языки программирования имеют *переменные* — имена значений, которые вы хотите использовать в программе. Статические языки заставляют вас указывать *тип* каждой переменной, который определяет, сколько места переменная займет в памяти и что можно с ней сделать. Компьютер использует эту информацию, чтобы *скомпилировать* программу в очень низкоуровневый *машинный язык* (характерный для определенного аппаратного обеспечения, машины понимают его лучше, а люди — хуже). Дизайнеры языков программирования часто должны решать, кому их язык должен быть понятнее: людям или компьютерам. Объявление типов переменных помогает компьютеру найти некоторые ошибки и работать быстрее, но это требует предварительного продумывания и набора кода. Большая часть кода примеров, написанного на языках C, C++ и Java, требует объявления типов переменных. Например, в каждом из примеров объявление типа `int` было необходимо для того, чтобы переменная `language` считалась целым числом. (Другие типы включают в себя числа с плавающей точкой, вроде `3.14159`, и символьные или текстовые данные, которые хранятся по-разному.)

Почему же они называются *статическими* языками? Потому что переменные в этих языках не могут изменять свой тип, они статичны. Целое число — это целое число, раз и навсегда.

Динамические языки — полная противоположность статических (они также называются *скриптовыми языками*). Эти языки программирования не заставляют вас определять тип переменной перед тем, как ее использовать. Если вы напишете

что-то вроде `x = 5`, динамический язык определит, что 5 — это целое число, поэтому переменная `x` имеет тип `int`. Эти языки позволяют вам достичь большего, написав меньшее количество строк кода. Вместо того чтобы компилироваться, они *интерпретируются* программой, которая называется — сюрприз! — *интерпретатором*. Динамические языки обычно медленнее, чем статические, но их скорость повышается, поскольку интерпретаторы становятся более оптимизированными. Долгое время динамические языки использовались для коротких программ *(сценариев)*, которые часто предназначались для того, чтобы подготовить данные для обработки более длинными программами, написанными на статических языках. Такие программы назывались *связующим кодом*. Несмотря на то что динамические языки больше годятся для этой задачи, в наши дни они могут решать и самые трудные задачи по обработке данных.

Многоцелевым динамическим языком многие годы был Perl. Язык программирования Perl очень мощный и имеет множество библиотек. Однако его синтаксис может быть трудным для понимания, а сам язык теряет в популярности из-за появления языков программирования Python и Ruby. А вот извольте: острый код с привкусом Perl:

```
my $language = 4;
print "Language $language: I am Perl, the camel of languages.\n";
```

Язык программирования Ruby (http://www.ruby-lang.org/) появился немного позже. Он отчасти заимствует функционал у языка Perl, а свою популярность приобрел благодаря фреймворку для веб-разработки *Ruby on Rails*. Он используется примерно в тех же областях, что и Python, и, выбирая между этими языками, вам придется руководствоваться в большей степени вкусом и доступными библиотеками. Следующий фрагмент кода написан на Ruby:

```
language = 5
puts "Language #{language}: I am Ruby, ready and aglow."
```

Язык программирования PHP (http://www.php.net/), который вы можете увидеть в следующем примере, очень популярен в области веб-разработки, поскольку он позволяет довольно легко объединить HTML и код. Однако язык PHP имеет несколько подводных камней, и его довольно трудно применить за пределами веб-разработки:

```
<?PHP
$language = 6;
echo "Language $language: I am PHP. The web is <i>mine</i>, I say.\n";
?>
```

Следующий пример показывает ответ Python этим языкам программирования:

```
language = 7
print("Language %s: I am Python. What's for supper?" % language)
```