

Введение

Популярными становятся только уродливые языки. Python — исключение из этого правила.

Дональд Кнут

Итак, вы здесь. И это хорошо.

Возможно, вы уже знаете что-то о компьютерах и Python, а может, и нет. Если какой-то раздел покажется вам слишком нудным, просто пропустите его. Вы не обидите меня. Итак, начнем.

Тайны

Начнем с двух небольших тайн и их разгадок. Что, по-вашему, означают следующие две строки?

Ряд 1 (ЛС): 18 лиц, 2 вм лиц влево, 1 лиц, пов.

Ряд 2 (ИС): 1 п снять как изн, 5 изн, 2 вм изн, 1 изн, пов.

Выглядят они как некая компьютерная программа. На самом деле это схема для вязания, точнее, фрагмент, который описывает, как связать пятку носка. Похожие носки показаны на рис. 1.1.



Рис. 1.1. Вязанные носки

Для меня эти строки имеют не больше смысла, чем sudoku для одного из моих котов, но вот моя жена совершенно точно понимает написанное. Если вы вяжете, то тоже поймете.

Рассмотрим еще один таинственный текст, который можно увидеть записанным на листочке из блокнота. Вы сразу поймете его предназначение, даже если и не догадаетесь о том, каким будет конечный продукт:

- 1/2 столовой ложки масла или маргарина;
- 1/2 столовой ложки сливок;
- 2 1/2 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.

Сохраняйте их охлажденными до следующего этапа.

Для каждого шарика:

- присыпьте разделочную доску мукой;
- раскатайте шарик при помощи рифленной скалки;
- жарьте на сковороде до подрумянивания;
- переверните и обжарьте другую сторону.

Даже те, кто совсем не готовит, без труда распознают кулинарный *рецепт*¹ — список продуктов, за которым следуют указания по приготовлению. Но что получится в итоге? Это *лефсе*, норвежский деликатес, который напоминает тортилью (рис. 1.2). Полейте блюдо маслом, вареньем или чем-либо еще, сверните и наслаждайтесь.

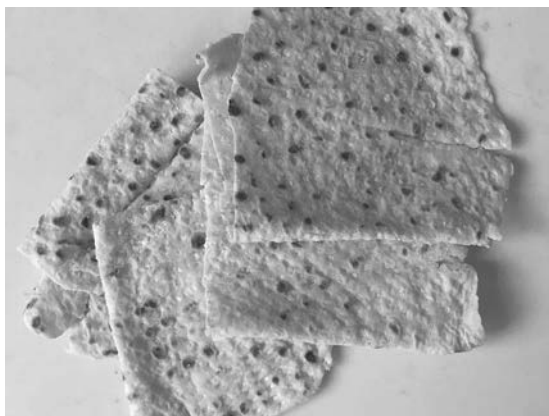


Рис. 1.2. Лефсе

¹ Такие рецепты публикуются в кулинарных книгах и уютных детективах.

Схема для вязания и рецепт имеют некоторое сходство:

- фиксированный *словарь*, состоящий из слов, аббревиатур и символов. Какие-то могут быть вам знакомы, какие-то — нет;
- правила, описывающие, что и где можно говорить, — *синтаксис*;
- *последовательность* операций, которые должны быть выполнены в определенном порядке;
- в некоторых случаях — повторение определенных операций (*цикл*), например способ приготовления каждого кусочка лефсе;
- в некоторых случаях — ссылка еще на одну последовательность операций (говоря компьютерными терминами, *функцию*). Например, когда вы прочтете приведенный 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 держать спицы в руках;
- кое-какие *данные*, которые нужно использовать, создать или изменить, — картофель и нитки;
- *инструменты*, с помощью которых идет работа с данными, — горшки, миксеры, духовки, вязальные спицы;
- ожидаемый *результат*. В наших примерах результатом будет предмет для надевания на ноги и предмет для желудка. Главное — не перепутать.

Как ни назови — идиомы, жаргон, — примеры их использования можно встретить везде. Жаргон помогает сэкономить время тем, кто его знает, а для других людей оставляет информацию совершенно непонятной. Попробуйте расшифровать колонку газеты, посвященную бриджу, если вы не играете в эту игру, или научную статью, если вы не ученый (или ученый, но в другой области).

Маленькие программы

Схему для вязания и рецепт я использовал для демонстрации того, что писать программы не так сложно, как может показаться, — нужно всего лишь выучить верные слова и правила.

Понять этот маленький язык гораздо легче, если в нем не очень много слов и правил и если вам не нужно изучать их все одновременно: за один раз наш мозг может воспринять лишь ограниченное количество знаний.

Далее перечислены некоторые факты о Python:

- это язык программирования;
- создан в 1991 году голландским ученым-информатиком Гвидо ван Россумом;
- с тех пор его развивали и направляли ван Россум и др.;

- он хорошо спроектирован;
- он легче читается, чем большинство других языков, и на нем легче писать;
- стал основой современной науки о данных и искусственного интеллекта (ИИ);
- владение этим языком — ценный навык на современном рынке труда в сфере ИТ;
- на нем удобно писать и небольшие сценарии, и большие системы, такие как Instagram.

А теперь рассмотрим небольшую, но самую настоящую программу (пример 1.1). Как вы думаете, что она делает?

Пример 1.1. `countdown.py`

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

Если вы угадали, что эта программа на Python выведет на экран:

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

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

Программа на Python, содержит некоторое количество специальных слов и символов: `for`, `in`, `print`, запятые, двоеточия, скобки и т. д. — все они являются важной частью синтаксиса (правил) языка. Хорошая новость заключается в том, что Python имеет более доступный и менее объемный синтаксис по сравнению с большинством других языков программирования: текст кажется почти понятным, как и рецепт.

Пример 1.2 — еще одна небольшая программа на Python: она позволяет выбрать одно из заклинаний Гарри Поттера, хранящееся в списке, и вывести его на экран.

Пример 1.2. `spells.py`

```
spells = [  
    "Riddikulus!",  
    "Wingardium Leviosa!",  
    "Avada Kedavra!",  
    "Expecto Patronum!",  
    "Nox!",  
    "Lumos!",  
]  
print(spells[3])
```

Отдельные заклинания являются в Python *строками* (последовательностями текстовых символов, заключенных в кавычки). Они разделены запятыми и помещены в *список* — это можно определить по квадратным скобкам ([и]). Слово `spells` — это *переменная*, являющаяся именем списка, — с ее помощью мы можем работать со списком. В нашем случае на экран будет выведено четвертое заклинание:

```
Expecto Patronum!
```

Почему мы сказали 3, если нам нужно было четвертое заклинание? Списки Python, такие как `spells`, представляют собой последовательность значений, доступ к которым осуществляется с использованием *смещения* от начала списка. Смещение первого элемента списка равно 0, а четвертого — 3.



Люди обычно считают с единицы, поэтому может показаться странным считать с нуля. Однако в программировании удобнее оперировать смещениями, а не позициями. Да, это пример того, чем компьютерная программа иногда отличается от обычного языка.

Список — очень распространенная структура данных в Python. О том, как им пользоваться, будет рассказано в главе 8.

Программа из примера 1.3 выводит на экран слова одного из участников комедийного трио The Three Stooges («Три балбеса»), однако на выбор фразы влияет не позиция в списке, а то, кто ее сказал.

Пример 1.3. quotes.py

```
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:

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

`quotes` — переменная, играющая роль имени словаря Python — коллекции уникальных ключей (в примере ключом является имя участника трио) и связанных с ними значений (в нашем примере — значимое высказывание участника «Балбесов»). Словарь позволяет сохранять элементы и выполнять их поиск по именам: зачастую это удобнее, чем работать со списком.

В примере с заклинаниями для создания списка использовались квадратные скобки ([и]), а в примере с цитатами для создания словаря — фигурные скобки ({ и }). Также мы задействовали двоеточие (:), чтобы связать каждый ключ словаря с соответствующим значением. Более подробно о словарях можно прочитать в главе 9.

Надеюсь, я не перегрузил вас синтаксисом. В следующих нескольких разделах вы познакомитесь и с другими простыми правилами.

Настройка окружения

Скорее всего, вы используете одну из трех основных платформ:

- Microsoft Windows;
- Apple macOS;
- Linux или другую Unix-подобную систему.

Программирование — это процесс, требующий ввода большого объема текста. Для выполнения большинства операций современные ОС предлагают графический пользовательский интерфейс (graphical user interface, GUI), но для создания программ вам на компьютере понадобятся два инструмента:

- *окно терминала* для ввода команд;
- *текстовый редактор* для создания и изменения файлов.

В главе 13 мы рассмотрим более сложные инструменты, в том числе *интегрированные среды разработки* (или просто среды разработки) (integrated development environments, IDE), которые предоставляют эти и другие функции в одном приложении, но в части I мы используем только окно терминала и текстовый редактор. Все они доступны на трех основных платформах, перечисленных ранее.

По мере необходимости я буду вставлять в текст книги любые замечания, касающиеся конкретной платформы.

Установка Python

К сожалению, многие операционные системы не включают Python в свой состав по умолчанию, поэтому вам может потребоваться установить его. Даже если в вашей системе установлен Python, он может быть устаревшей версии. Рекомендую скачать последнюю версию Python с официального сайта python.org. Нажмите кнопку **Downloads** (Загрузки), затем кнопку, соответствующую вашей платформе, и следуйте инструкциям.

Обновление Python

В индустрии программного обеспечения программам принято присваивать номер версии в формате *x.y.z*, где *x* обозначает *старший номер* версии, *y* — *младший номер* (номер подверсии в пределах версии со старшим номером *x*), а *z* — *номер исправления* в пределах подверсии *y*.

На момент написания книги последней была версия Python со *старшим.младшим* номерами 3.13. Версии со старшим номером 3 потребовалось около десяти лет, чтобы заменить предыдущую версию со старшим номером 2. Смена младших номеров версий происходит примерно ежегодно¹.

В окне терминала ваша *командная оболочка* (интерпретатор команд) выводит *приглашение*, сообщающее, что сразу вслед за ним можно ввести команду. Вводимый текст выделен **полужирным**. В этой книге я использую в качестве приглашения простой символ `$`. Введите команду `python -V`. Вы должны увидеть примерно такой результат:

```
$ python -V
Python 3.13
```



Если у вас установлена версия Python ниже 3.13, то обращайтесь внимание на примечания в этой книге, сообщающие, в каких случаях более ранние версии могут действовать иначе.

Запуск программ на Python

Запускать программы на Python можно двумя способами:

- интерактивно — вводить строку за строкой и следить за происходящим;
- запустить файл с программой на Python.

Попробуем оба варианта.

Интерактивный интерпретатор Python

Иногда интерпретатор называют *оболочкой Python* (не путайте с командной оболочкой операционной системы, которая выводит приглашение `$`). Вот что получится, если ввести простую команду `python` (в следующем выводе я заменил информацию о версии, потому что фактическая информация намного длиннее и может различаться в зависимости от машины и версии):

```
$ python
Python ... version info ...
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

¹ Ожидается, что Python 4 никогда не появится, — слишком уж сложным оказался переход с Python 2 на Python 3.

Каждая строка, начинающаяся с `>>>`, — это приглашение интерпретатора Python ввести строку кода на Python.

Вы должны ввести код на Python после этого приглашения, и интерпретатор выполнит его, строку за строкой. Такой способ позволяет быстро протестировать фрагменты кода, и я буду использовать его на протяжении всей книги для иллюстрации коротких примеров. Вы можете вводить выделенный полужирным текст в интерпретатор Python на своем компьютере по мере чтения.

Давайте попробуем:

```
$ python
Python ... version info ...
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello? World?")
Hello? World?
>>>
```

Функция `print()` встроена в Python и выведет все, что вы поместите в круглые скобки. В данном случае мы передали ей текстовую строку. Такие строки в Python обозначаются кавычками (`"`).

Файлы с кодом на Python

Код на Python принято сохранять в файлах с расширением `.py`. Чтобы запустить такой файл, введите команду `python`, а затем имя файла.

Сохраните ту единственную строку, которую вы ввели в предыдущем разделе, в файл `hello.py` (пример 1.4).

Пример 1.4. `hello.py`

```
print("Hello? World?")
```



Для создания файлов с программами на Python всегда используйте простой текстовый редактор или IDE. Такие программы, как Microsoft Word, могут добавлять служебные символы управления форматированием, заменять кавычки и производить другие эффекты. Вам же нужен обычный текст.

Затем в окне терминала сообщите интерпретатору Python, что он должен запустить файл:

```
$ python hello.py
Hello? World?
```

Обычно интерактивный способ используется только для тестирования небольших фрагментов кода, а для чего-то большего — файлы Python.

Встроенные возможности Python

Помимо функции `print()`, интерпретатор Python имеет встроенную поддержку структур данных и кода, которые вы увидите в следующих главах части I.

Стандартная библиотека Python

При установке интерпретатора Python вместе с ним вы получаете *стандартную библиотеку* Python: набор файлов Python, официально поддерживаемых в каждой версии. Они решают многие задачи, которые вы увидите в части III. Доступ к ним из программы осуществляется с помощью оператора `import`, который подробно описан в главе 12 и представлен в примере 1.5.

Сторонние пакеты Python

Вы можете установить код Python, написанный кем угодно, и получить к нему доступ так же, как к стандартной библиотеке. В части III вы увидите множество примеров таких пакетов.

Более сложный пример

Когда-то я работал над созданием Internet Archive (<https://archive.org>), бесплатного сервиса, сохраняющего содержимое Интернета. Его самая известная функция — Wayback Machine¹, которая годами архивирует веб-страницы и является бесценным ресурсом для поиска старых копий сайтов, даже давно исчезнувших. Пример 1.5 обращается к Wayback Machine и иллюстрирует многие задачи, которые можно выполнять с помощью Python без дополнительных настроек. Все подробности вы узнаете позже, по мере чтения этой книги. (Номера строк не являются частью кода, а были добавлены, чтобы можно было сослаться на них в дальнейших пояснениях.)

Пример 1.5. archive.py

```
1 import webbrowser
2 import json
3 from urllib.request import urlopen
4
5 print("Let's find an old website.")
6 site = input("Type a website URL: ")
7 era = input("Type a year, month, and day, like 20150613: ") + "0000"
8 url = f"http://archive.org/wayback/available?url={site}&timestamp={era}"
9 response = urlopen(url)
```

¹ Отсылка к эпизодам с Шерманом и мистером Пибоди — персонажами старых мультфильмов *The Rocky and Bullwinkle*.

```
10 contents = response.read()
11 text = contents.decode("utf-8")
12 data = json.loads(text)
13 try:
14     old_site = data["archived_snapshots"]["closest"]["url"]
15     print("Found this copy: ", old_site)
16     print("It should appear in your browser now.")
17     webbrowser.open(old_site)
18 except:
19     print("Sorry, no luck finding", site)
```

Такая небольшая программа, написанная на языке Python, делает многое с помощью всего нескольких строк. (В рабочей версии будет больше комментариев и проверок на ошибки.) Вы пока знаете не все термины, однако сможете познакомиться с ними в следующих главах. Вот что происходит в этих строках.

1. Импортируется (делается доступным для этой программы) весь код из модуля стандартной библиотеки, который называется `webbrowser`.
2. Импортируется весь код из модуля стандартной библиотеки, который называется `json`.
3. Импортируется только функция `urlopen` из модуля `urllib.request` стандартной библиотеки.
4. Пустая строка (чтобы код было проще читать).
5. Выводится текст приветствия.
6. Выводится предложение ввести URL, затем пользовательский ввод считывается и сохраняется в переменной с именем `site`.
7. Выводится еще одно предложение ввести год, месяц и день, которые затем сохраняются в переменной с именем `era`.
8. Создается строковая переменная с именем `url`, чтобы сайт Wayback Machine смог найти копию требуемого сайта по дате. Здесь используется так называемая *f-строка*, которая встраивает содержимое указанных переменных.
9. Устанавливается соединение с сервером, находящимся по сконструированному адресу, и запрашивается определенный *веб-сервис*.
10. Полученный ответ присваивается переменной `contents`.
11. *Содержимое* переменной `contents` преобразуется в текстовую строку в формате JSON и сохраняется в переменной `text`.
12. JSON-строка, сохраненная в переменной `text`, преобразуется в `data` — структуру данных языка Python.
13. Выполняется проверка на ошибки: следующие четыре строки помещены в блок `try`, и если при их выполнении возникнет ошибка, то будет выполнена последняя строка программы (она следует сразу за ключевым словом `except`).

14. После получения содержимого, соответствующего указанному сайту и дате, из трехуровневого *словаря* Python извлекается нужное значение. Обратите внимание на отступы в этой и трех последующих строках — так мы сообщаем интерпретатору Python, что эти строки находятся в блоке `try`.
15. Полученный URL выводится на экран.
16. Сообщается, что произойдет, когда будет выполнена следующая строка.
17. Полученное содержимое отображается в браузере.
18. Если во время выполнения предыдущих четырех строк что-то пойдет не так, то интерпретатор Python перейдет сюда.
19. Если программа дала сбой, то выводится сообщение и имя искомого сайта. Эта строка тоже имеет отступ, потому что должна выполняться, только если выполняется строка `except`.

Вам не обязательно понимать все это прямо сейчас! Этот пример я привел, только чтобы показать, как много можно сделать на Python, используя относительно небольшое количество строк, которые читаются легче (и безопаснее!), чем заклинание из древнего манускрипта.



В большинстве языков программирования для обозначения начала и конца многострочных блоков кода используются символьные разделители. В некоторых языках это слова `start` и `end`, а в языках с фигурными скобками (C, C++, Java, JavaScript и многих других) используются скобки `{}` и `}`. В Python для той же цели служат отступы. Это делает код менее громоздким и более простым для понимания.

Когда я сам запустил эту программу в окне терминала, то ввел URL сайта и дату и получил следующий результат:

```
$ python archive.py
Let's find an old website.
Type a website URL: xkcd.com
Type a year, month, and day, like 20150613: 20240728
Found this copy: http://web.archive.org/web/20240727204346/https://xkcd.com/
It should appear in your browser now.
```

На рис. 1.3 показано то, что появилось в моем браузере.

По разным причинам на сохраненных архивных страницах иногда отсутствует что-то, что было на исходном сайте, например изображения. В верхней части страницы из архива находится дополнительная панель, позволяющая получить доступ к любому архиву этого сайта. Архив Internet Archive — отличный инструмент, помогающий узнать, что люди говорили и делали в конкретные моменты времени.

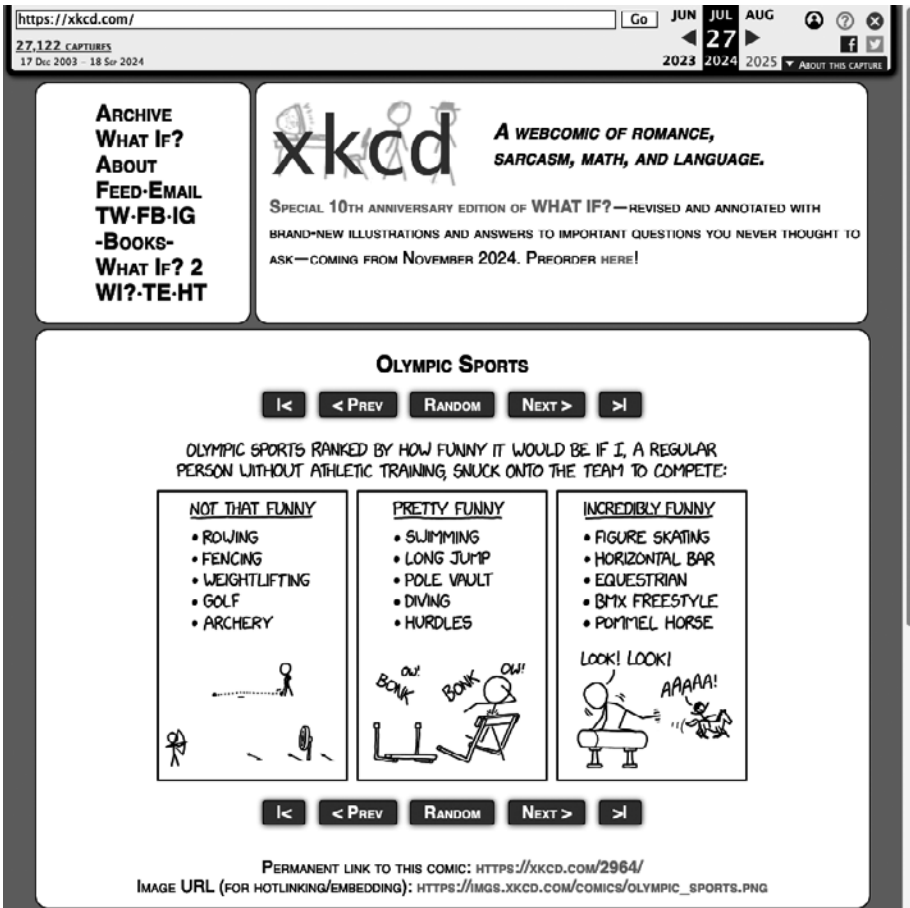


Рис. 1.3. Результат обращения к Wayback Machine

Итоги и взгляд вперед

В этой главе представлен минимальный набор инструментов, необходимых для написания программ на Python, а также примеры кода, демонстрирующие некоторые базовые типы данных Python, такие как списки и строки. Глава включает пример довольно крупной программы и ее вывод.

В следующей главе мы подробнее рассмотрим типы данных, а также переменные — основные концепции любого языка программирования.