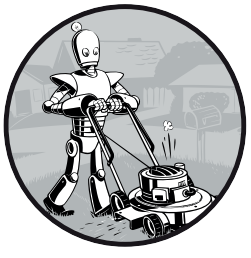
ВВЕДЕНИЕ



“За каких-то пару часов ты сделал то, на что у нас троих ушло бы два дня”. В начале 2000–х годов мой сосед по общежитию в колледже работал в магазине электроники. Время от времени они получали электронные таблицы с прайсами своих конкурентов, включающие тысячи наименований. Распечатка одной таблицы представляла собой толстую стопку бумаги. Обработкой данных занимались три сотрудника магазина. Они сравнивали цены, указанные в таблице, с ценами в своем магазине и отмечали тот товар, который конкуренты продавали по более низкой цене. На эту работу у них уходило примерно два дня.

“Слушайте, если вы дадите мне исходный файл таблицы, то я напишу программу, которая выполнит всю работу за вас”, — сказал мой товарищ, увидев, как они копошатся среди груды разбросанных на полу и сложенных в стопки листов.

Через пару часов у него была готова небольшая программа, которая считывала данные о ценах конкурентов из файла, находила для каждого продукта аналог в базе данных магазина и отмечала все товары, цены на которые у конкурентов были ниже. Здесь уместно сказать, что мой товарищ был всего лишь начинающим программистом и бо́льшую часть времени потратил на поиск нужной информации в книге по программированию. Сама программа выполнялась всего несколько секунд, что позволило моему товарищу и его коллегам насладиться в тот день удлиненным обеденным перерывом.

Этот пример наглядно демонстрирует возможности программирования. Компьютер подобен армейскому ножу, который можно использовать в самых разных ситуациях. Многие люди часами сидят за клавиатурой, выполняя одни и те же действия, и даже не догадываются, что компьютер, если снабдить его соответствующими инструкциями, способен сделать все то же самое за считанные секунды.

Для кого предназначена эта книга

В наши дни трудно найти сферу человеческой деятельности, в которой не использовалось бы программное обеспечение. Все мы общаемся в социальных сетях, наши смартфоны — это по сути компьютеры, подключенные к Интернету, а в офисах у всех установлены компьютеры. Как следствие, это привело к стремительному росту спроса на программистов. Бесчисленные книги по программированию, вебинары, семинары для разработчиков — все они обещают превратить амбициозных новичков в компьютерных инженеров, заработная плата которых выражается шестизначными числами.

Это книга не для них. Она предназначена для всех остальных.

Прочтение книги не сделает из вас профессионального разработчика, точно так же, как нескольких уроков игры на гитаре навряд ли будет достаточно для того, чтобы стать рок-звездой. Но если вы офисный работник, администратор, преподаватель или просто используете компьютер для развлечений, то, изучив основы программирования в том объеме, который предлагается в данной книге, вы сможете автоматизировать следующие простые задачи:

- перемещение и переименование тысяч файлов и их сортировка по папкам;
- заполнение веб-форм без ввода данных вручную;

- загрузка файлов или копирование текста с веб-сайта при его обновлении;
- отправка компьютером заранее подготовленных уведомлений;
- обновление и форматирование электронных таблиц Excel;
- проверка электронной почты и рассылка заранее подготовленных писем.

Все это простые задачи, но отнимают у нас массу времени. Кроме того, зачастую они настолько тривиальны или специфичны, что готовых программ для их решения нет. Вооружившись даже минимальными знаниями в области программирования, вы сможете заставить свой компьютер выполнять эти задачи вместо вас.

Исходные предположения

Эта книга — не справочник, а руководство для начинающих. Используемый в ней стиль программирования иногда идет вразрез с общепринятыми практиками (например, в некоторых программах используются глобальные переменные), но это компромиссное решение, позволяющее сделать код более легким для изучения. Книга предназначена для тех, кому будет достаточно научиться писать простой одноразовый код, поэтому стилю оформления программ и приданию им элегантного вида не уделяется особого внимания. В книге не рассматриваются продвинутые концепции программирования, такие как ООП, списковые включения или генераторы, чтобы не усложнять материал. Опытные программисты наверняка найдут в книге те места, где код можно сделать более эффективным, но нас в первую очередь интересует создание работоспособных программ с минимальными усилиями.

Что такое программирование

В сериалах и фильмах часто показывают потоки загадочных нулей и единиц, бегущих по экрану, но реальные компьютерные системы вовсе не такие таинственные, как в “Матрице”. *Программирование* — это всего-навсего процесс передачи инструкций компьютеру. Инструкции могут быть связаны с обработкой чисел, редактированием текста, поиском информации в файлах или передачей данных другим компьютерам по сети.

Строительными блоками любых программ служат элементарные инструкции. Вот как выглядят некоторые из них, если перевести их на понятный нам язык:

- “Сделай это, затем сделай то”;
- “Если данное условие соблюдается, выполни такое-то действие; в противном случае выполни другое действие”;

- “Выполни это действие столько-то раз”;
- “Продолжай выполнять эти действия до тех пор, пока данное условие соблюдается”.

Эти строительные блоки можно комбинировать для получения более сложных программ. В качестве примера ниже приведены инструкции (*исходный код*) простой программы, написанной на Python. Программа последовательно выполняет каждую строку кода от первой до последней (при этом некоторые инструкции выполняются, только *если* определенное условие выполняется, *иначе* выполняется другая инструкция).

```
❶ passwordFile = open('SecretPasswordFile.txt')
❷ secretPassword = passwordFile.read()
❸ print('Введите пароль.')
  typedPassword = input()
❹ if typedPassword == secretPassword:
❺     print('Доступ разрешен.')
❻     if typedPassword == '12345':
❼         print('Рекомендуем установить более сложный пароль!')
  else:
❽     print('В доступе отказано.')
```

Даже если вы ничего не смыслите в программировании, вы все равно сможете сделать разумные предположения относительно того, что делает этот код, просто читая его. Сначала программа открывает файл *SecretPasswordFile.txt* ❶, из которого считывается пароль ❷, после чего пользователю предлагается ввести свой вариант пароля (с помощью клавиатуры) ❸. Далее оба пароля сравниваются между собой ❹, и если они совпадают, то на экран выводится текст `'Доступ разрешен'` ❺. Затем программа проверяет, не равен ли введенный пароль строке `'12345'` ❻. Если это так, то программа выдает рекомендацию сменить пароль ❼. В случае несовпадения паролей программа выводит на экран сообщение `'В доступе отказано'` ❽.

Что такое Python

Python — это не только язык программирования (со своим синтаксисом, определяющим правила написания корректного кода), но и *интерпретатор*, т.е. программа, предназначенная для чтения исходного кода (написанного на языке Python) и выполнения содержащихся в нем инструкций. Различные версии интерпретатора Python, предназначенные для платформ Linux, macOS и Windows, доступны для бесплатной загрузки на сайте `https://python.org`.

Своим названием Python обязан вовсе не питону, а британской комедийной группе “Монти Пайтон” (Monty Python), работавшей в жанре

абсурдного юмора. Программистов на Python шутливо называют *питонистами*.

Программисту не обязательно в совершенстве знать математику

Многие, кто приступают к изучению программирования, боятся, что им придется интенсивно учить математику. Но в действительности большинству программистов не нужно быть математиками — достаточно знать арифметику. В этом смысле хорошему программисту понадобится не намного больший объем математических знаний по сравнению с тем, который требуется для решения головоломок судоку.

Суть судоку заключается в заполнении цифрами от 1 до 9 каждого из внутренних квадратов размером 3×3, расположенных на игровом поле размером 9×9, причем ни одна строка, ни один столбец и ни один внутренний квадрат игрового поля не должны содержать повторяющихся цифр. Для решения головоломки необходимо использовать дедуктивный метод, исходя из заданной начальной конфигурации цифр. Например, поскольку в головоломке, показанной на рис. 1, цифра 5 находится и в первой сверху, и второй сверху строке, она не может в них повторяться. А раз так, то в правом верхнем квадрате она может быть только в третьей сверху строке. При этом цифра 5 уже стоит в крайнем справа столбце, значит, она может находиться только слева от цифры 6. Последовательное применение подобной логики к строкам, столбцам и внутренним квадратам позволит находить подсказки для заполнения пустых клеток.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Рис. 1. Головоломка судоку (слева) и ее решение (справа). Несмотря на то что это числовая головоломка, никаких математических знаний для ее решения не требуется

Из того факта, что в судоку используются числа, вовсе не следует, что для решения головоломки необходимо быть хорошим математиком. То же самое справедливо и в отношении программирования. Как и в судоку, при

написании программ приходится разбивать задачу более мелкие этапы. Так же и при *отладке* (процесс обнаружения и исправления ошибок в программе) вы кропотливо анализируете действия программы, пытаясь выявить причину ошибки. И чем больше вы программируете, тем лучше у вас это будет получаться.

Учиться программированию никогда не поздно

Второе наиболее распространенное заблуждение относительно изучения программирования заключается в том, что люди думают, будто им уже слишком поздно заниматься этим. Причем так заявляют даже те, кому (вдумайтесь) аж 23 года! Не стоит так рано ставить на себе крест: многие осваивают искусство программирования в гораздо более зрелом возрасте.

Вам вовсе не нужно начинать программировать в школе, чтобы стать программистом. Тем не менее образ юного хакера весьма устойчив. Даже я вношу свою лепту в распространение этого мифа, когда рассказываю, что начал программировать еще в младших классах.

Как бы там ни было, но сегодня учиться программированию намного проще, чем в 1990-е годы. В наши дни доступно множество книг, онлайн-курсов и специализированных сайтов, посвященных программированию. Кроме того, сами языки программирования стали более удобными для изучения. **Все, что я узнал о программировании в школе, сегодня можно выучить за несколько месяцев.** Как видите, мой стартовый багаж знаний оказался не таким уж и большим.

Важно настраивать себя на постоянное обучение, т.е. понимать, что навыки программирования формируются благодаря практике. Люди не рождаются программистами, поэтому отсутствие опыта программирования вовсе не означает, что вы никогда не достигнете уровня эксперта.

Программирование — творческий вид деятельности

Программирование — творческое занятие, как рисование или вязание. Вы начинаете с чистого листа и поначалу ограничены его рамками, но затем перед вами открываются безграничные возможности.

Разница между программированием и другими творческими видами деятельности заключается в том, что все необходимое уже есть в компьютере и вам не нужно ничего докупать. Даже старенького компьютера десятилетней давности вполне достаточно для написания программ. А когда программа готова, ее можно легко скопировать произвольное количество раз. Вязаный свитер будет в конкретный момент времени носить только один человек, тогда как полезной программой можно легко поделиться со всем миром.

Структура книги

В части I рассматриваются основы программирования на Python, тогда как часть II посвящена различным задачам, которые можно автоматизировать. Каждая глава части II включает проекты, которые вам предстоит изучить. Ниже приведено краткое описание глав.

Часть I. Основы программирования на языке Python

- **Глава 1. Основные понятия языка Python.** Здесь рассматриваются выражения — базовые строительные блоки программы, а также описывается, как использовать интерактивную оболочку Python для экспериментов с кодом.
- **Глава 2. Порядок выполнения программы.** Объясняется, как заставить программу выполнять нужные инструкции в зависимости от тех или иных условий.
- **Глава 3. Функции.** Вы узнаете, как создавать собственные функции, разбивая код на логические блоки, с которыми проще работать.
- **Глава 4. Списки.** Вводится понятие *списка* и объясняется, как работать со структурами данных.
- **Глава 5. Словари.** Вводится понятие *словаря* и демонстрируются более мощные структуры данных.
- **Глава 6. Строки.** Описываются способы работы с текстовыми данными (в Python они называются *строками*).

Часть II. Автоматизация задач

- **Глава 7. Регулярные выражения.** Обсуждаются приемы обработки строк и способы поиска образцов текста, соответствующих заданному шаблону, с помощью регулярных выражений.
- **Глава 8. Проверка ввода.** Объясняется, каким образом программа может проверять информацию, которую пользователь предоставляет ей. Тем самым гарантируется, что пользовательские данные поступают в формате, который не вызовет проблем.
- **Глава 9. Чтение и запись файлов.** Будет рассказано, как организовать в программе чтение данных из текстовых файлов и сохранение информации в файлах на диске.
- **Глава 10. Управление файлами.** Рассматриваются автоматизированные способы копирования, перемещения, переименования и удаления файлов, благодаря которым эти операции будут выполняться гораздо быстрее, чем это можно сделать вручную. Также описываются принципы работы со сжатыми файлами.

- **Глава 11. Отладка.** Рассматриваются средства Python, предназначенные для обнаружения ошибок в программах.
- **Глава 12. Веб-скрейпинг.** Будет показано, как писать программы, способные автоматически загружать веб-страницы и извлекать из них данные. Этот процесс называется *веб-скрейпинг*.
- **Глава 13. Работа с таблицами Excel.** Описываются способы автоматизированной работы с электронными таблицами Excel, не требующие открытия самого приложения. Это очень полезно в тех случаях, когда количество обрабатываемых документов исчисляется сотнями или даже тысячами.
- **Глава 14. Работа с приложением Google Таблицы.** Вы узнаете, как загружать и обновлять электронные таблицы веб-приложения Google Таблицы.
- **Глава 15. Работа с документами PDF и Word.** Будут описаны программные методы чтения документов PDF и Word.
- **Глава 16. Работа с CSV-файлами и данными в формате JSON.** Продолжение темы программной обработки документов, только на этот раз в формате CSV и JSON.
- **Глава 17. Работа с датой и временем, планирование заданий и запуск программ.** Объясняется, как обрабатывать в программе значения даты и времени и как запрограммировать компьютер на выполнение задач по расписанию. Также будет показано, как запускать из сценариев Python программы, написанные на других языках.
- **Глава 18. Отправка электронной почты и текстовых сообщений.** Обсуждается написание программ, осуществляющих автоматическую рассылку электронной почты и текстовых сообщений.
- **Глава 19. Работа с изображениями.** Вы узнаете, как обрабатывать изображения, сохраненные в различных форматах, таких как JPEG или PNG.
- **Глава 20. Управление клавиатурой и мышью.** Речь пойдет об управлении клавиатурой и мышью путем программной эмуляции нажатий клавиш и щелчков.
- **Приложение А. Установка сторонних модулей.** Будет показано, каким образом можно расширить возможности Python за счет дополнительных модулей.
- **Приложение Б. Запуск программ.** Вы узнаете, как выполнять программы Python в среде Windows, macOS и Linux, не используя редактор кода.

- **Приложение В. Ответы на контрольные вопросы.** Здесь даны ответы на контрольные вопросы, приведенные в конце каждой главы.

Загрузка и установка Python

Дистрибутивы Python для Windows, macOS и Ubuntu доступны для бесплатной загрузки по адресу `https://python.org/downloads/`. Если вы загрузите текущую версию для своей системы, то все примеры программ, приведенные в книге, должны работать.

Предупреждение

Убедитесь в том, что загружаете версию Python 3 (например, 3.8.0). Все примеры программ в книге написаны с использованием Python 3, и если вы попытаетесь запустить их в версии Python 2, то они могут выполняться неправильно или не выполняться вовсе.

На странице загрузки для каждой операционной системы предлагаются отдельные дистрибутивы, рассчитанные на 64- и 32-разрядные версии, поэтому предварительно определитесь, какой именно вариант вам нужен. Если компьютер был куплен после 2007 года, то, скорее всего, на нем установлена 64-разрядная операционная система. Чтобы убедиться в этом наверняка, выполните следующие действия.

- В Windows выберите Пуск⇨Панель управления⇨Система⇨О программе и проверьте значение поля Тип системы.
- В macOS перейдите в меню Apple, выберите About This Mac⇨More Info⇨System Report⇨Hardware и проверьте значение поля Processor Name. Если там указано "Intel Core Solo" или "Intel Core Duo", то у вас 32-разрядный компьютер. Если же указано что-то другое (включая "Intel Core 2 Duo"), то у вас 64-разрядный компьютер.
- В Ubuntu Linux откройте приложение Terminal и введите команду **`uname -m`**. Ответ `i686` означает, что у вас 32-разрядный компьютер, ответ `x86_64` — 64-разрядный.

В Windows загрузите установщик Python (файл с расширением *.msi*) и дважды щелкните на нем. Далее следуйте инструкциям, отображаемым на экране.

1. Выберите вариант Install for All Users (Установить для всех пользователей) и щелкните на кнопке Next.
2. В следующих окнах примите параметры, заданные по умолчанию, щелкая на кнопке Next.

В macOS загрузите файл с расширением *.dmg*, соответствующий вашей версии macOS, и дважды щелкните на нем. Далее следуйте инструкциям, отображаемым на экране.

1. Когда в новом окне откроется пакет DMG, дважды щелкните на файле *Python.mpkg*. Возможно, вам придется ввести пароль администратора.
2. В следующих окнах щелкайте на кнопках Continue, чтобы принять параметры, заданные по умолчанию, а затем щелкните на кнопке Agree для принятия условий лицензии.
3. В последнем окне щелкните на кнопке Install.

В Ubuntu можно установить Python из окна программы Terminal, выполнив следующие действия.

1. Откройте окно Terminal.
2. Введите команду **`sudo apt-get install python3`**.
3. Введите команду **`sudo apt-get install idle3`**.
4. Введите команду **`sudo apt-get install python3-pip`**.

Загрузка и установка Mu

Интерпретатор Python — это программа, которая выполняет код Python. Редактор Mu — это программа, позволяющая вводить код подобно тому, как вы набираете текст в Word. Редактор Mu доступен для загрузки на сайте `https://codewith.mu/`.

В Windows и macOS загрузите соответствующий инсталлятор и запустите его, дважды щелкнув на файле установки. В macOS при запуске установщика открывается окно, в котором нужно перетащить значок Mu на значок папки *Программы*, чтобы продолжить установку. В Ubuntu необходимо установить Mu как пакет Python. В этом случае щелкните на кнопке Instructions, находящейся в разделе Python Package на странице загрузки.

Запуск Mu

Вот как запустить редактор Mu.

- В Windows щелкните на значке Пуск в левом нижнем углу экрана, введите **`Mu`** в поле поиска и выберите Mu.
- В macOS откройте окно Finder, щелкните на значке Applications, а затем щелкните на значке mu-editor.
- В Ubuntu выберите Applications⇨Accessories⇨Terminal и введите команду **`python3 -m mu`**.

При первом запуске Mu появится окно Select Mode (Выбрать режим), в котором доступны варианты Adafruit CircuitPython, BBC micro:bit, Pygame Zero и Python 3. Выберите Python 3. В дальнейшем вы сможете изменить режим редактора, щелкнув на кнопке Mode в верхней части окна.

Примечание

Чтобы иметь возможность устанавливать сторонние модули, рассматриваемые в книге, загрузите Mu версии 1.1.0 или выше.

Запуск IDLE

В этой книге Mu применяется и как редактор, и как интерактивная оболочка. В то же время для написания кода Python доступно множество других редакторов. IDLE (Integrated Development and Learning Environment) — это интегрированная среда разработки, входящая в состав Python. Она послужит запасным вариантом, если по какой-то причине вам не удастся установить Mu. Вот как запустить IDLE.

- В Windows щелкните на кнопке Пуск в левом нижнем углу экрана, введите **`IDLE`** в поле поиска и выберите IDLE.
- В macOS откройте окно Finder, щелкните последовательно на значках Applications и Python 3.8, а затем щелкните на значке IDLE.
- В Ubuntu выберите Applications⇨Accessories⇨Terminal и введите команду **`idle3`**. (Можете также щелкнуть на кнопке Applications в верхней части экрана, выбрать раздел Programming и щелкнуть на значке IDLE 3.)

Интерактивная оболочка

После запуска Mu появится окно *редактора файла*. Чтобы открыть *интерактивную оболочку*, щелкните на кнопке REPL. Оболочка — это программа, которая позволяет вводить инструкции аналогично тому, как это делается в окне терминала или в командной строке Windows. Команды, вводимые в интерактивной оболочке, тут же выполняются интерпретатором Python.

В Mu интерактивная оболочка представляет собой панель в нижней части окна, где отображается следующий текст.

```
Jupyter QtConsole 4.3.1
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 18:11:49) [MSC v.1900 64
 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information.
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

В случае IDLE интерактивная оболочка — это окно, которое появляется первым. Оно в основном пустое, за исключением текста в верхней части окна.

```
Python 3.8.0b1 (tags/v3.8.0b1:3b5deb0116, Jun 4 2019, 19:52:55)
 [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Обозначения `In [1]:` и `>>>` называются *приглашениями командной строки*. В примерах книги будет использоваться приглашение `>>>`, поскольку оно более распространенное (вы увидите его при запуске Python из окна Terminal или командной строки). Приглашение `In [1]:` появилось в Jupyter Notebook — другом популярном редакторе Python.

Например, введите в интерактивной оболочке следующую команду:

```
>>> print('Hello world!')
```

Как только вы нажмете клавишу <Enter>, оболочка выдаст результат.

```
>>> print('Hello world!')
Hello world!
```

Вы только что передали компьютеру инструкцию, и он выполнил то, о чем вы его попросили!

Установка сторонних модулей

Иногда в программе требуется импортировать дополнительные модули. Некоторые из них поставляются вместе с Python, но есть и сторонние модули, созданные независимыми разработчиками. В приложении А даны инструкции, как использовать утилиту `pip` (в Windows) или `pip3` (в macOS и Linux) для установки сторонних модулей. Обратитесь к нему, когда в книге встретится указание установить тот или иной сторонний модуль.

Как получить справку

Программисты часто получают новые знания, находя в Интернете ответы на свои вопросы. Это сильно отличается от привычного способа обучения через занятия с преподавателем, который читает лекции и может отвечать на вопросы. Преимущество Интернета как обучающей платформы состоит в том, что есть целые сообщества людей, готовых ответить на ваши вопросы. Более того, на многие вопросы наверняка уже были даны ответы, которые остается лишь найти в Интернете. Если вы получили сообщение

об ошибке в программе, то вряд ли вы первый, кто столкнулся с подобной проблемой, и найти решение будет проще, чем вы думаете.

Например, давайте намеренно сгенерируем ошибку следующим образом: введите в интерактивной оболочке выражение `'42' + 3`. Вам необязательно сейчас знать, что оно означает, но результат будет таким, как показано ниже.

```
  >>> '42' + 3
❶ Traceback (most recent call last):
    File "<pyshell#0>", line 1, in <module>
      '42' + 3
❷ TypeError: Can't convert 'int' object to str implicitly
  >>>
```

Появление сообщения об ошибке ❷ обусловлено тем, что смысл введенной вами инструкции оказался непонятным для Python. В той части сообщения, которая касается текущего стека вызовов (`Traceback`) ❶, отображаются конкретная инструкция и номер строки, где интерпретатор столкнулся с проблемой. Если сообщение об ошибке ни о чем вам не говорит, выполните поиск в Интернете по точному тексту сообщения. Введите текст “TypeError: Can’t convert ‘int’ object to str implicitly” (включая внешние кавычки) в поисковой системе, и вы получите тысячи ссылок, по которым можно узнать о том, что означает данное сообщение и что породило ошибку (рис. 2).

Зачастую оказывается, что у кого-то уже возникал аналогичный вопрос и на него уже был дан ответ. Никто не может знать абсолютно все о программировании, поэтому повседневная практика любого разработчика — поиск ответов на различные вопросы технического характера.

Правильно формулируйте вопросы, ответы на которые ищете

Если поиск в Интернете не дал результатов, то попробуйте задать вопрос на таких форумах, как Stack Overflow (`https://ru.stackoverflow.com`) или Reddit (`https://reddit.com/r/learnprogramming`). Но имейте в виду, что при обращении за помощью очень важно правильно формулировать свои вопросы. Обязательно прочитайте разделы “Frequently Asked Questions” (часто задаваемые вопросы) на этих сайтах, где объясняется, как правильно задавать вопросы.

Задавая вопросы, касающиеся программирования, старайтесь придерживаться следующих рекомендаций.

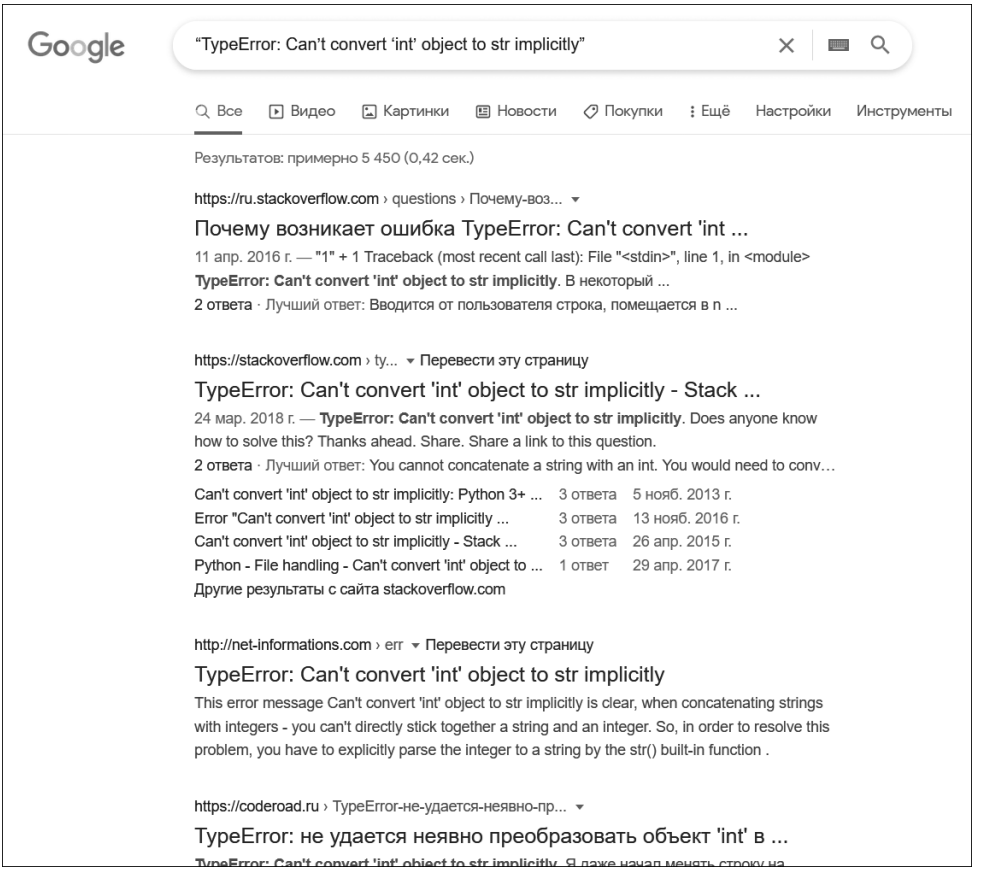


Рис. 2. Получение дополнительной информации о причине ошибки путем поиска в Google с использованием текста сообщения об ошибке в качестве строки поиска

- Объясните, что именно вы *пытаетесь* сделать, а не только то, что вы уже сделали. Это позволит другим понять, где именно вы сбились с пути.
- Укажите, когда именно возникает ошибка: сразу после запуска программы или после того, как выполняются определенные действия.
- Скопируйте и вставьте *полный* текст сообщения об ошибке вместе с программным кодом в хранилище на сайте `https://pastebin.com/` или `https://gist.github.com/`.

  На этих сайтах можно делиться большими объемами кода с другими пользователями, сохраняя форматирование текста. URL-адрес помещенного в хранилище кода можно переслать нужному человеку по электронной почте или опубликовать на форуме. Чтобы увидеть,

как это работает, просмотрите код, который я опубликовал по адресу https://gist.github.com/asweigart/6912168/.

- Объясните, какие меры вы предпринимали для решения возникшей проблемы. Тем самым другие пользователи поймут, что определенная часть работы уже была сделана.
- Укажите используемую версию Python (между интерпретаторами Python версий 2 и 3 имеются существенные различия). Также укажите используемую вами операционную систему и ее версию.
- Если ошибка появилась после того, как вы внесли изменения в код, детально опишите, что именно вы поменяли.
- Укажите, повторяется ли ошибка при каждом запуске программы или же она возникает лишь после того, как вы совершаете определенные действия. Во втором случае опишите, в чем именно заключаются эти действия.

Кроме того, всегда следуйте правилам сетевого этикета. В частности, задавая на форуме вопросы, не набирайте весь текст прописными буквами, пытаясь сделать его более заметным, и не требуйте слишком многого от людей, которые добровольно пытаются вам помочь.

Статья о том, как получить помощь по вопросам программирования, доступна по адресу https://autbor.com/help/. Список часто задаваемых вопросов, связанных с программированием, опубликован по адресу https://www.reddit.com/r/learnprogramming/wiki/faq/. Аналогичный список вопросов, касающихся карьеры программиста, доступен по адресу https://www.reddit.com/r/cscareerquestions/wiki/index/.

Мне нравится помогать людям осваивать Python. Я регулярно публикую соответствующие статьи в своем блоге на сайте https://inventwithpython.com/blog/. Кроме того, мне можно задать вопрос по адресу al@inventwithpython.com. Но если хотите получить более быстрый ответ, опубликуйте вопрос по адресу https://reddit.com/r/inventwithpython/.

Файлы примеров

Архив с файлами Python и дополнительными файлами, которые используются в примерах книги, доступен на сайте издательства No Starch Press:

```
https://nostarch.com/download/
Automate_the_Boring_Stuff_2e_onlinematerials.zip
```

Также архив примеров книги доступен на сайте издательства "Диалектика":

http://go.dialektika.com/automate2

Резюме

Для большинства людей компьютер — всего лишь полезное устройство, а не рабочий инструмент. Но научившись программировать, вы получите доступ к одному из наиболее мощных инструментов в современном мире, работа с которым к тому же доставит вам немалое удовольствие. Программирование — не настолько сложное занятие, и даже любители способны освоить его. Главное — не бояться экспериментировать и совершать ошибки.

Книга подойдет даже для тех, у кого нулевой опыт программирования. Вы многое узнаете из книги, но не рассчитывайте найти в ней ответы на все вопросы. Не забывайте о том, что умение задавать правильные вопросы и находить ответы на них пригодится вам в путешествии в мир программирования.

Итак, приступим!

Ждем ваших отзывов!

Вы, читатель этой книги, и есть главный ее критик. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересны любые ваши замечания в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам электронное письмо либо просто посетить наш веб-сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится ли вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Отправляя письмо или сообщение, не забудьте указать название книги и ее авторов, а также свой обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию новых книг.

Наши электронные адреса:

E-mail: info.dialektika@gmail.com

WWW: http://www.dialektika.com