Оглавление

Предисловие

Создание этой книги лично для меня стало увлекательным путешествием в мир, раскрывающий возможности языка Python в сегодняшнем процедурном и объектно ориентированном программировании, используемом для обеспечения функциональности при разработке онлайн-приложений. Примеры кода, представленные в этой книге, описывают, как за несколько простых шагов создавать программы на языке Python, а на скриншотах демонстрируются реальные результаты их работы. Я искренне надеюсь, что вам понравится открывать захватывающие возможности Python и вы получите при этом не меньше удовольствия, чем я во время написания этой книги.

Для того чтобы код, описанный в примерах, стал более наглядным, он отформатирован черным шрифтом, за исключением комментариев, выделенных серым шрифтом:

```
# Пишем традиционное приветствие

greeting = 'Hello World!'

print( greeting )
```

Кроме того, для идентификации исходных файлов, описываемых в пошаговых инструкциях, на полях рядом с каждым пунктом будут появляться значок и имя соответствующего файла:

script.py page.html image.gif

Для удобства файлы исходных кодов всех примеров, представленных в этой книге, помещены в один ZIP-архив. Вы можете получить его, выполнив следующие простые шаги.

1. Откройте браузер и загрузите архив по ссылке http://eksmo.ru/Python_examples.zip.
2. Извлеките из скачанного архива папки *MyScripts* и *MyProjects* в ваш домашний каталог (например, в *C:*) а также скопируйте содержимое папки *htdocs* в каталог документов вашего веб-сервера.
3. Теперь вы можете, используя пошаговые инструкции, выполнять примеры с помощью интерпретатора Python и видеть результаты его работы.

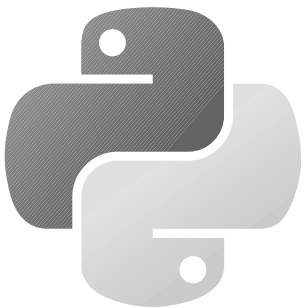
1 Приступаем к работе

Добро пожаловать в увлекательный мир языка программирования Python. В этой главе показывается, как установить Python и создать вашу первую программу.

- **Введение в язык Python**
- **Установка Python в среде Windows**
- **Установка Python в среде Linux**
- **Знакомство с интерпретатором**
- **Ваша первая программа**
- **Работа с переменными**
- **Получение введенных пользователем данных**
- **Исправление ошибок**
- **Заключение**

Введение в язык Python

Python является высокоуровневым («человекочитаемым») языком программирования, который для вывода результатов использует интерпретатор. Python содержит обширную стандартную библиотеку модулей протестированного кода, которые легко могут быть включены в ваши собственные программы.



Будьте в курсе последних новостей проекта Python на сайте**python.org**.

Язык Python, разработанный Гвидо ван Россумом (Guido van Rossum) в конце восьмидесятых — начале девяностых годов в Национальном научно-исследовательском институте математики и компьютерных наук в Нидерландах, является производным от многих других языков, в том числе C, C ++ и командной оболочки Unix. Сегодня Python поддерживается командой разработчиков ядра в институте, хотя Гвидо ван Россум по-прежнему играет важную роль в определении направления развития языка.

Читаемость кода, делающая язык Python особенно подходящим для новичков в программировании, — один из принципов философии Python, которую можно обобщить следующим образом.

- Красивое лучше, чем уродливое.
- Явное лучше, чем неявное.
- Простое лучше, чем сложное.
- Сложное лучше, чем запутанное.
- Читаемость имеет значение.

Совет

Так называемое правило офсайда, которое используют некоторые языки программирования, выделяя блоки кода при помощи отступов, заимствовано из футбола.

Поскольку Python ориентирован на читаемость кода, в нем часто используются ключевые слова на английском языке там, где другие языки программирования обычно используют знаки препинания. Особое его отличие состоит в том, что для группировки инструкций в блоке кода Python использует отступы, а не ключевые слова или знаки препинания. В языке Pascal, например, начало блоков обозначается ключевым словом `begin` и заканчивается ключевым словом `end`, в то время как программисты на C используют фигурные скобки для обозначения блоков кода. Очень часто такой подход группировки блоков отступами критикуется программистами, знакомыми с другими языками, но, несомненно, использование отступов в Python позволяет программам выглядеть менее нагроможденными.

Перечислим некоторые из важнейших отличительных особенностей языка Python, которые делают его привлекательным для начинающих программистов.

- **Python бесплатен** — это свободно распространяемое программное обеспечение с открытым исходным кодом.
- **Python легок в изучении** — он имеет простой синтаксис.
- **Python позволяет создавать легко читаемый код** — он не перегружен знаками препинания.
- **Python легок в обслуживании** — имеет модульную структуру.
- **Python располагает богатым «арсеналом»** — он предлагает большую стандартную библиотеку, которая легко интегрируется в ваши программы.
- **Python портируемый** — его можно запустить на обширном множестве различных платформ, и везде он будет иметь один и тот же интерфейс.
- **Python интерпретируемый** — компиляция не требуется.
- **Python является высокоуровневым языком** — он имеет статическое распределение памяти.
- **Python расширяемый** — позволяет добавлять низкоуровневые модули.
- **Python универсален** — поддерживает как процедурный, так и объектно ориентированный методы программирования.
- **Python гибок в использовании** — с его помощью можно создавать консольные программы, приложения графического интерфейса, а также сценарии для взаимодействия внешних программ с вебсерверами.

Как и любое другое программное обеспечение, Python продолжает развиваться, его новые версии выпускаются с определенной периодичностью. Объявлено, что версия 2.7 будет окончательной в ветке 2.x. Но ее поддержка будет продлена до 2020 года. Других больших релизов в данной ветке не ожидается.

Ветка версии 3.x находится в активной разработке и уже имеет несколько стабильных релизов. Это значит, что все последние улучшения стандартных библиотек, например, окажутся доступными только в версии Python 3.x. Описанные в нашей книге особенности языка будут относиться к версии 3.x.

На заметку

Название языку Python было дано в честь популярного британского комедийного шоу «Летающий цирк Монти Пайтона» (Monty Python's Flying Circus) — вы можете найти упоминание об этом в документации по языку.

Внимание

Python 3.x обратно несовместим с версией Python 2.7.

Установка Python в операционной системе Windows

Перед тем как вы начнете программировать на языке Python, необходимо установить на ваш компьютер интерпретатор Python, а также стандартную библиотеку модулей кода, поставляемую вместе с ним. Все это можно свободно загрузить на странице **python.org/download**. Для пользователей операционной системы Windows существуют две версии инсталлятора: для 32-битных и 64-битных систем.

1. Запустите веб-браузер, перейдите на страницу **python.org/download** и загрузите установщик, подходящий для вашей версии операционной системы — в нашем примере файл имеет имя *python-3.3.2.msi*.
2. После завершения загрузки запустите установщик, выберите режим установки для всех пользователей либо только для себя и нажмите кнопку **Next** (Далее) для продолжения.
3. Теперь подтвердите предлагаемое расположение установки, в название которого будет входить имя корневого диска, слово Python и номер версии — в данном примере установка произойдет в каталог *C:\Python33* для версии 3.3.2.

Установщики для операционной системы OS X 32-битной и 64-битной версий также доступны для загрузки на **python.org/download**.

Совет

Поддержка установщика MSI включена для всех версий Windows и доступна для свободной загрузки на **microsoft.com/downloads** — введите в строке поиска Windows Installer.

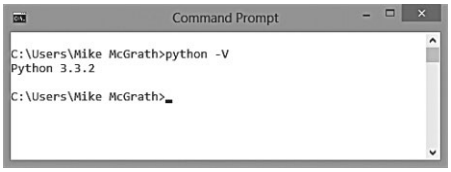
4. Нажмите кнопку **Next** (Далее) для продолжения и убедитесь, что выбран компонент **Add python.exe to Path** (Добавить путь в системную переменную Path).

На заметку

Убедитесь, что все компоненты установки включены, как показано в нашем примере.

5. Нажмите **Next** (Далее), чтобы начать копирование файлов на ваш компьютер, а затем — **Finish** (Готово) для завершения процесса установки.

6. Чтобы убедиться, что Python теперь доступен, перезагрузите компьютер, запустите командную строку (*cmd.exe*) и наберите команду `python -V` — в ответ интерпретатор Python выдаст номер установленной версии.

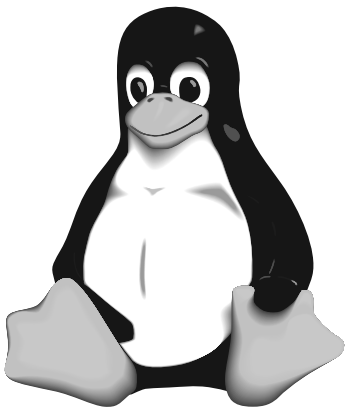


Внимание

Буква `V` в команде должна быть указана обязательно прописной. Перед тем как продолжать работать с примерами в книге, убедитесь, что данная команда выдает необходимый номер установленной версии.

Установка Python в операционной системе Linux

В дистрибутивы Linux обычно включен Python — по умолчанию там используется версия 2.7. Для работы с веткой 3.x вам, очевидно, предстоит установить нужный релиз дополнительно.



Обратитесь к документации по вашей операционной системе Linux для дальнейшей установки Python.

1. Запустите терминальное окно и наберите в точности, как указано, команду `python -V` для вывода информации об установленной версии по умолчанию.

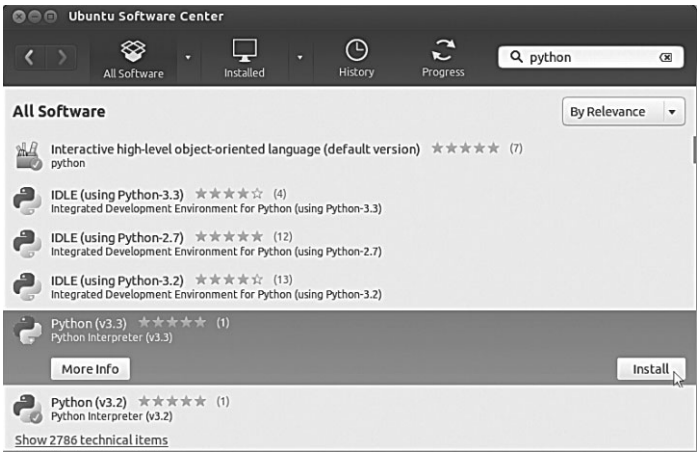
2. Затем наберите в точности команду `python3 -V` для того, чтобы увидеть информацию об установленной версии ветки 3.x, если таковая имеется.

Внимание

Не удаляйте версию 2.7 из вашей системы, так как во многих случаях существуют зависимые от нее приложения, работоспособность которых может нарушиться.

3. Теперь запустите на вашей Linux системе менеджер пакетов, чтобы посмотреть, какая из последних версий Python доступна для установки — например, на системах с Ubuntu вы можете использовать Центр приложений (Ubuntu Software Center).

4. Найдите в менеджере пакетов необходимое вам программное обеспечение, название которого содержит слово Python, чтобы посмотреть информацию, какие компоненты установлены или доступны для установки.



Совет

Вы можете по желанию установить среду разработки IDLE для Python3.3, но это совсем не обязательно, так как все примеры в книге созданы при помощи обычного текстового редактора, такого, как Nano.

5. Наконец установите последнюю версию из ветки Python3.x — в данном случае это Python3.3.

6. Для проверки доступности последней версии Python на вашем компьютере запустите терминальное окно и наберите команду `python3.3 -V`.

```
mike@ubuntu: ~
mike@ubuntu:~$ python3.3 -V
Python 3.3.2
mike@ubuntu:~$
```

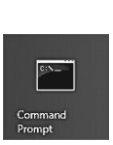
На заметку

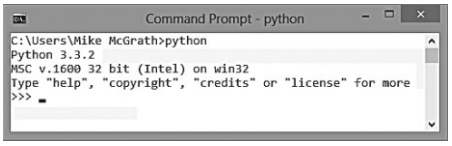
Теперь вы можете использовать команду `python3.3` для того, чтобы ваши программы отрабатывались интерпретатором именно этой версии.

Знакомство с интерпретатором

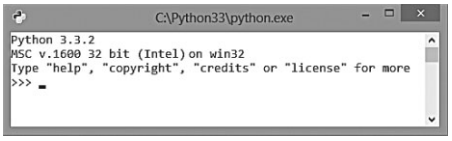
Интерпретатор Python обрабатывает текстовый код вашей программы, а также имеет интерактивный режим, полезный для отладки и тестирования фрагментов кода. В интерактивный режим Python можно попасть несколькими способами:

- из обычной командной строки — введите команду `python`, чтобы запустить начальную командную строку Python (символы >>>), в которой вы будете взаимодействовать с интерпретатором;

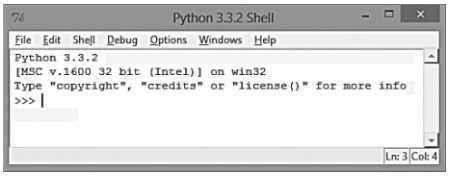




- из меню **Пуск** (Start) — выберите пункт **Python (command line)** — запустится окно, содержащее начальную командную строку интерпретатора Python с символами >>>;

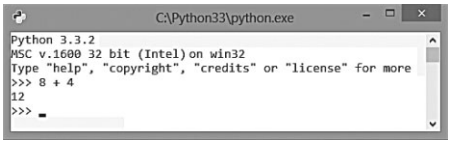


- из меню **Пуск** (Start) — выберите пункт **IDLE (Python GUI)**, чтобы запустить окно оболочки Python, содержащее командную строку с символами >>>.



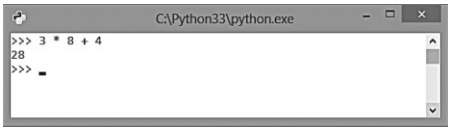
Независимо от того, какой способ вы выбрали для входа в интерактивный режим, интерпретатор Python будет одинаково реагировать на команды, вводимые в его командной строке после знаков >>>. В этом режиме его можно использовать в качестве калькулятора.

1. Войдите в интерактивный режим Python, используя любой из вышеперечисленных методов, затем наберите простое выражение с операцией сложения и нажмите кнопку **Enter**. Интерпретатор в ответ выдаст вам сумму.

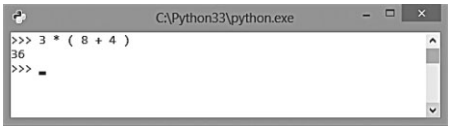


Интерпретатор Python понимает любые арифметические выражения, поэтому можно использовать скобки для указания порядка вычисления — часть выражения, заключенная в скобки, будет вычисляться первой.

2. Затем в командной строке Python наберите выражение с тремя операндами без указания порядка вычисления.



3. Теперь в командной строке Python наберите то же самое выражение, но добавьте скобки, определяющие порядок вычисления.



Совет

Пробелы в выражениях игнорируются, поэтому выражение 8+4, как показано здесь, можно записать с добавлением пробелов просто для красоты восприятия.

На заметку

Интерактивный режим используется в основном для тестирования и отладки фрагментов кода.

Внимание

IDLE расшифровывается как Integrated DeveLopment Environment — интегрированная среда разработки. Она имеет ограниченные функции и в данной книге не используется для демонстрации примеров.

Ваша первая программа

Кроме того, что интерактивный режим Python полезен в качестве простейшего калькулятора, его можно использовать для создания программ. Программа на языке Python — это обычный текстовый файл, созданный с помощью простого редактора, такого как Блокнот (Notepad), и сохраненный в файле с расширением *.py*. Запустить программу на Python можно, указав имя соответствующего файла после команды `python` в командной строке интерпретатора.

Внимание

Не используйте текстовые процессоры для создания исходного кода программ, поскольку они добавляют дополнительное форматирование.

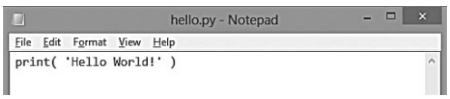
По традиции первая программа, которую создают при изучении языка программирования, просто выводит какое-либо сообщение с приветствием. На языке Python для этого используется функция `print()`, сообщение для вывода этой функции указывается в скобках. Это может быть строка символов, заключенная в кавычки. Кавычки могут быть как двойными (`""`), так и одинарными(`''`), но нельзя использовать одновременно и те, и другие.

`hello.py`

1. На компьютере под управлением операционной системы Windows запустите простой текстовый редактор, такой как, например, Блокнот (Notepad).
2. Затем наберите следующую инструкцию в пустой строке редактора:

```
print( 'Hello World!' )
```

3. Теперь создайте новый каталог *C:\MyScripts* и сохраните в нем файл под именем *hello.py*.



Совет

Созданный каталог *C:\MyScripts* будет использоваться во всех примерах этой книги для Windows.

4. Теперь запустите окно командной строки, перейдите в только что созданный каталог и наберите команду `python hello.py` — вы увидите, как интерпретатор Python запустит вашу программу и выведет приветственное сообщение.

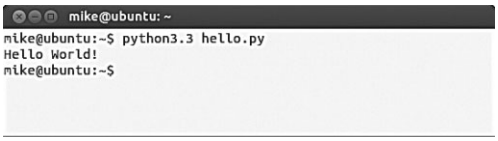
Процедура создания первой программы на Python в системе Linux ничем не отличается от той, которая делается в Windows. Однако, независимо от используемой платформы, всегда нужно помнить, что если установлены различные версии Python, то следует использовать корректную команду для вызова интерпретатора. Особенно это важно для системы Linux, которую обычно поставляют вместе с установленной версией Python 2.7, и набранная команда `python` по умолчанию вызывает именно этот интерпретатор. Если установлен Python 3.3 и вы хотите вызывать именно его для обработки вашей программы, то следует использовать команду `python3.3`, чтобы вызвать нужную версию интерпретатора.

1. В системе Linux запустите любой текстовый редактор, например Nano.
2. Затем наберите следующую инструкцию в пустом окне редактора:

```
print( 'Hello World!' )
```

3. Теперь сохраните файл в вашем домашнем каталоге под именем *hello.py*.

4. Наконец запустите терминальное окно и перейдите в ваш домашний каталог, а затем наберите команду `python3.3 hello.py` — вы увидите, как интерпретатор Python запустит вашу программу и выведет соответствующее сообщение.



`hello.py`

На заметку

Все дальнейшие примеры, описанные в этой книге, касаются операционной системы Windows (наиболее популярной среди пользователей). Но примеры также можно создать и выполнить на Linux.

Работа с переменными

В программировании переменная представляет собой некоторый контейнер в памяти компьютера, где хранятся данные. После того как данные сохранены, их можно вызвать, используя имя этой переменной. Программист может выбрать любое имя для переменной, за исключением ключевых слов языка Python. Лучше выбирать для переменных значащие имена, которые отражают их содержание.

На заметку

Строковые данные должны быть заключены в кавычки, обозначающие начало и конец строки.

В программах Python данные, которые нужно хранить в переменных, вносятся с помощью оператора присваивания `=`, например, чтобы сохранить числовое значение `8` в переменной с именем `a`, нужно написать:

```
a = 8
```

Затем можно обратиться к сохраненному значению переменной, используя ее имя. Таким образом, инструкция `print( a )` выведет сохраненное значение `8`. Переменным могут быть последовательно присвоены разные значения, и, следовательно, переменная способна принимать различные значения по мере работы программы — неслучайно она так и называется: переменная.

В языке Python переменной должно быть присвоено начальное значение (инициализация переменной) в инструкции, которая объявляет эту переменную в программе, — иначе интерпретатор вызовет сообщение об ошибке not defined (неопределенная переменная).

В одной инструкции разрешается инициализировать несколько переменных с одним значением. Это можно сделать, используя оператор присваивания `=`. Например, для инициализации переменных `a`, `b` и `c` и присваивания им значения `8` мы используем запись:

```
a = b = c = 8
```

Совет

Языки программирования, которые требуют определения типов переменных, известны как строго типизированные в отличие от нестрого типизированных.

Наоборот, несколько переменных можно проинициализировать с различными значениями и записать все это в одной инструкции, используя запятую в качестве разделителя. Например, в качестве инициализации переменных `a`, `b` и `c` с числовыми значениями `1`, `2`, `3` мы используем запись:

```
a, b, c = 1, 2, 3
```

Некоторые языки программирования, такие как Java, требуют указания типов переменных при их объявлении. При этом резервируется определенный объем памяти. Данный прием известен как статическая типизация. На переменные в языке Python такое ограничение не накладывается, и распределение памяти происходит в соответствии с при-

сваиваемыми переменным значениями (динамическая типизация). Это означает, что переменная может содержать как целые числа, так и числа с плавающей точкой, текстовые строки или логические значения.

Вы можете добавлять в свои программы на Python комментарии для описания инструкций или разделов кода. Для этого используется символ #. Все, что идет после этого символа до конца строки, игнорируется интерпретатором Python. Комментарии очень полезны — они помогают сделать ваш код понятным для других, а также для вас самих, когда вы позже к нему возвращаетесь.

1. Запустите текстовый редактор, в котором объявите и инициализируйте переменную, затем выведите хранящееся в ней значение.

var.py

```
# Инициализируем переменную целочисленным значением

var = 8

print( var )
```

2. Затем присвойте новое значение переменной и выведите его на экран.

```
# Присваиваем переменной значение числа с плавающей точкой

var = 3.142

print( var )
```

3. Теперь присваиваем другое значение и отображаем его опять.

```
# Присваиваем переменной строковое значение

var = 'Python in easy steps'

print( var )
```

4. Наконец присваиваем еще одно значение и снова выводим результат.

```
# Присваиваем переменной логическое значение

var = True

print( var )
```

5. Сохраните файл в вашем рабочем каталоге, затем откройте командную строку из этого каталога и запустите программу, чтобы посмотреть результат ее вывода.

Совет

Вы можете использовать в своих программах многострочные комментарии, которые заключаются в тройные кавычки, """…""".

Получение введенных пользователем данных

Значения переменным в языке Python можно присваивать не только с помощью программы, но и путем пользовательского ввода. Для этого используется функция `input()`. Она в качестве аргумента принимает строку, которая будет отображаться пользователю, приглашая его ввести данные, а затем читает строку, введенную пользователем.

Такие символы интерпретируются как текстовая строка, даже если на самом деле это числовые значения. Данная строка может быть присвоена любой переменной, используя оператор присваивания `=`. Впоследствии с этой переменной можно работать точно так же, как и с другими, например вывести ее значение, указав имя переменной в функции `print()`.

При помощи функции `print()` можно вывести и несколько значений переменных, указав их внутри скобок через запятую.

`input.py`

1. Запустите текстовый редактор, в котором объявите и проинициализируйте переменную с помощью запроса пользовательского ввода.

```
# Инициализируем переменную значением, введенным пользователем

user = input( 'I am Python. What is your name? : ' )
```

2. Затем отобразите ответное сообщение, подтверждая введенное пользователем значение.

```
# Выводим строку и значение переменной

print( 'Welcome' , user )
```

3. Сохраните файл в вашем рабочем каталоге, откройте командную строку и запустите программу — введите ваше имя, нажмите клавишу **Enter**, и вы увидите ответное сообщение, содержащее ваше имя.

Совет

Обратите внимание, что строка подсказки заканчивается пробелом, который отображается при выводе. Это простой способ немного отделить строку пользовательского ввода.

Когда вы выводите с помощью функции **print()** несколько значений, они по умолчанию отделяются единичным пробелом. Чтобы указать альтернативный разделитель, вы можете добавить параметр **sep** для функции **print()**. Например, при использовании **sep = '*'** вы получите в выводе величи́ны, разделенные символом *.

Также по умолчанию функция **print()** выводит в конце каждой строки неотображаемый символ новой строки (**\n**). Но существует возможность указать собственный символ, используя параметр **end**. Например, использование **end = '!'** выведет в конце каждой строки знак восклицания.

4. Отредактируйте вашу программу, объявив и проинициализировав вторую переменную с помощью еще одного запроса пользовательского ввода.

```
# Инициализируем еще одну переменную значением, введенным пользователем

lang = input( 'Favorite programming language? : ' )
```

5. Выведите ответное сообщение для подтверждения ввода пользователя, указав свой разделитель вывода и символ окончания строки.

```
# Выводим строку и значение переменной

print( lang , 'Is' , 'Fun' , sep = ' * ' , end = '!\n' )
```

6. Теперь сохраните файл, откройте командную строку и запустите программу заново — введите ваше имя и название языка программирования. Затем нажмите клавишу **Enter**, и вы увидите ответное сообщение, содержащее ваш ввод.

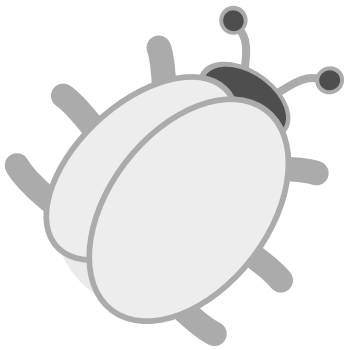
На заметку

Вы можете явно задать конец строки, добавив его в параметр **end**. Например, **end='!\n'** добавляет как восклицательный знак, так и символ новой строки.

Совет

Для большей наглядности и читаемости вы можете добавлять к разделителям пробелы, как показано в этом примере.

Исправление ошибок



Ошибки в программах часто называются багами (от англ. bug — жук), а процесс их отслеживания называется отладкой (англ. debugging, одно из значений — удаление насекомых с растений).

При выполнении написанных на языке Python программ могут происходить ошибки. Существуют три основных типа ошибок, которые следует различать, чтобы легче их потом было исправлять.

- **Синтаксическая ошибка** — происходит, когда интерпретатор обрабатывает код, который не соответствует правилам языка Python, например отсутствие кавычек вокруг строковой переменной. Интерпретатор останавливается и сообщает об ошибке без дальнейшего выполнения программы.
- **Ошибка исполнения —** происходит во время исполнения программы. Например, когда переменная не может быть распознана из-за несоответствия типов. Интерпретатор запускает программу, останавливается на ошибке и сообщает о природе этой ошибки как об исключении.
- **Логическая ошибка (смысловая)** — происходит, когда программа ведет себя не так, как было задумано. Например, когда не определен порядок действий в вычисляемом выражении. Интерпретатор запускает программу и не сообщает об ошибке.

С исправлением синтаксических и ошибок выполнения все достаточно очевидно, поскольку интерпретатор сообщает, где ошибка произошла, и указывает на природу ее происхождения. Но для логических ошибок требуется детальное изучение кода.

syntax.py

1. Запустите текстовый редактор и добавьте выражение для вывода строки, при этом опустив знак закрывающих кавычек.

```
print( 'Python in easy steps )
```

2. Сохраните файл в вашем рабочем каталоге, откройте командную строку и запустите программу — вы увидите, что интерпретатор сообщает о синтаксической ошибке и указывает ее позицию в коде.

Внимание

Обычно указатель на синтаксическую ошибку располагается не на месте ошибки, а на следующем символе.