Николай Прохоренок
Владимир Дронов

PRO
ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Python 3 и PyQt 5
Разработка приложений
2-е издание

Описание языка Python
Объектно-ориентированное программирование
Работа с файлами и каталогами
Создание оконных приложений
Работа с базами данных
Мультимедиа
Печать и экспорт в формат PDF
Взаимодействие с Windows
Сохранение настроек приложений
Работающий пример: приложение «Судоку»

Материалы на www.bhv.ru

bhv®

Николай Прохоренок
Владимир Дронов

Python 3 и PyQt 5
Разработка приложений
2-е издание

Санкт-Петербург
«БХВ-Петербург»
2018

УДК 004.43
ББК 32.973.26-018.1
П84

Прохоренок, Н. А.

П84 Python 3 и PyQt 5. Разработка приложений. — 2-е изд., перераб. и доп. / Н. А. Прохоренок, В. А. Дронов. — СПб.: БХВ-Петербург, 2018. — 832 с.: ил. — (Профессиональное программирование)

ISBN 978-5-9775-3978-4

Описан язык Python 3: типы данных, операторы, условия, циклы, регулярные выражения, функции, инструменты объектно-ориентированного программирования, работа с файлами и каталогами, модули стандартной библиотеки. Особое внимание уделено библиотеке PyQt, позволяющей создавать приложения с графическим интерфейсом. Рассмотрены средства для обработки сигналов и событий, управления свойствами окна, разработки многопоточных приложений, описаны основные компоненты (кнопки, поля и др.), инструменты для работы с базами данных, мультимедиа, печати документов и их экспорта. На сайте издательства приведены примеры из книги.

Во втором издании описаны актуальные версии Python 3.6.3 и PyQt 5.9.2, средства взаимодействия с Windows и сохранения настроек приложений, рассмотрен процесс разработки полнофункционального приложения.

Для программистов

УДК 004.43
ББК 32.973.26-018.1

Группа подготовки издания:

Руководитель проекта	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Капалыгина*
Компьютерная верстка	*Ольги Сергиенко*
Дизайн обложки	*Марины Дамбиевой*

"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

ISBN 978-5-9775-3978-4

Оглавление

Введение

Добро пожаловать в мир Python!

В *первой части* книги мы рассмотрим собственно *Python* — высокоуровневый, объектно-ориентированный, тьюринг-полный, интерпретируемый язык программирования, предназначенный для решения самого широкого круга задач. С его помощью можно обрабатывать числовую и текстовую информацию, создавать изображения, работать с базами данных, разрабатывать веб-сайты и приложения с графическим интерфейсом. Python — язык кроссплатформенный, он позволяет создавать программы, которые будут работать во всех операционных системах. В этой книге мы изучим базовые возможности Windows-редакции Python версии 3.6.3.

Согласно официальной версии, название языка произошло вовсе не от змеи. Создатель языка, Гвидо ван Россум (Guido van Rossum), назвал свое творение в честь британского комедийного телешоу Би-Би-Си «Летающий цирк Монти Пайтона» (Monty Python's Flying Circus). Поэтому правильно название этого замечательного языка должно звучать как «Пайтон».

Программа на языке Python представляет собой обычный текстовый файл с расширением py (консольная программа) или pyw (программа с графическим интерфейсом). Все инструкции из этого файла выполняются интерпретатором построчно. Для ускорения работы при первом импорте модуля создается промежуточный байт-код, который сохраняется в одноименном файле с расширением pyc. При последующих запусках, если модуль не был изменен, исполняется именно байт-код. Для выполнения низкоуровневых операций и задач, требующих высокой скорости работы, можно написать модуль на языке C или C++, скомпилировать его, а затем подключить к основной программе.

Поскольку Python, как было только что отмечено, является языком объектно-ориентированным, практически все данные в нем представляются объектами — даже значения, относящиеся к элементарным типам данных, наподобие чисел и строк, а также сами типы данных. При этом в переменной всегда сохраняется только ссылка на объект, а не сам объект. Например, можно создать функцию, сохранить ссылку на нее в переменной, а затем вызвать функцию через эту переменную. Такое обстоятельство делает язык Python идеальным инструментом для создания программ, использующих функции обратного вызова, — например, при разработке графического интерфейса. Тот факт, что Python относится к категории языков объектно-ориентированных, отнюдь не означает, что и объектно-ориентированный стиль программирования (ООП) является при его использовании обязательным: на языке Python можно писать программы как в стиле ООП, так и в процедурном стиле, — как того требует конкретная ситуация или как предпочитает программист.

Python — самый стильный язык программирования в мире, он не допускает двоякого написания кода. Так, языку Perl, например, присущи зависимость от контекста и множественность синтаксиса, и часто два программиста, пишущие на Perl, просто не понимают код друг друга. В Python же отсутствуют лишние конструкции, и код можно написать только одним способом. Все программисты, работающие с языком Python, должны придерживаться стандарта PEP-8, описанного в документе **https://www.python.org/dev/peps/pep-0008/**. Соответственно, более читаемого кода нет ни в одном ином языке программирования.

Синтаксис языка Python вызывает много нареканий у программистов, знакомых с другими языками программирования. На первый взгляд может показаться, что отсутствие ограничительных символов для выделения блоков (фигурных скобок или конструкции `begin...end`) и использование для их формирования пробелов могут приводить к ошибкам. Однако это только первое и неправильное впечатление. Хороший стиль программирования в любом языке обязывает выделять инструкции внутри блока одинаковым количеством пробелов. В этой ситуации ограничительные символы просто ни к чему. Бытует мнение, что программа будет по-разному смотреться в разных редакторах. И это неверно — согласно стандарту, для выделения блоков необходимо использовать *четыре пробела*, а четыре пробела в любом редакторе смотрятся одинаково. При этом, если количество пробелов внутри блока окажется разным, интерпретатор выведет сообщение о фатальной ошибке, и программа будет остановлена. Таким образом язык Python приучает программистов писать красивый и понятный код, и, если в другом языке вас не приучили к хорошему стилю программирования, язык Python быстро это исправит.

Поскольку программа на языке Python представляет собой обычный текстовый файл, его можно редактировать с помощью любого текстового редактора — например, того же Notepad++. Можно использовать и другие, более специализированные программы такого рода: PyScripter, PythonWin, UliPad, Eclipse с установленным модулем PyDev, Netbeans и др. (полный список приемлемых редакторов можно найти на странице **https://wiki.python.org/moin/PythonEditors**). Мы же в процессе изложения материала этой книги будем пользоваться интерактивным интерпретатором IDLE, который входит в состав стандартной поставки Python в Windows, — он идеально подходит для изучения Python.

Вторая часть книги посвящена рассмотрению версии 5.9.2 библиотеки PyQt, позволяющей создавать кроссплатформенные приложения с графическим интерфейсом. Библиотека очень проста в использовании и идеально подходит для разработки весьма серьезных оконных приложений. Пользуясь исключительно ее средствами, мы можем выводить на экран графику практически любой сложности, работать с базами данных наиболее распространенных форматов, воспроизводить мультимедийные файлы, выводить документы на печать и экспортировать их в популярный формат Adobe PDF.

А в самом конце мы с вами самостоятельно напишем на языке Python с применением библиотеки PyQt полнофункциональное приложение «Судоку», предназначенное для создания и решения одноименных головоломок.

Все листинги из этой книги вы найдете в файлах Listings.doc (листинги из глав книги), PyQt.doc (дополнительные листинги с примерами на PyQt) и в папке sudoku (листинги приложения «Судоку»), электронный архив с которыми можно загрузить с FTP-сервера издательства «БХВ-Петербург» по ссылке: **ftp://ftp.bhv.ru/9785977539784.zip** или со страницы книги на сайте **www.bhv.ru** (см. *приложение*).

Авторы книги желают вам приятного чтения и надеются, что она станет верным спутником в вашей грядущей карьере программиста! Тем не менее, не забывайте, что книги по программированию нужно не только читать, — весьма желательно выполнять все имеющиеся в них примеры, а также экспериментировать, что-либо в этих примерах изменяя.

ЧАСТЬ I

Основы языка Python

ГЛАВА 1

Первые шаги

Прежде чем мы начнем рассматривать синтаксис языка, необходимо сделать два замечания. Во-первых, как уже было отмечено во *введении*, не забывайте, что книги по программированию нужно не только читать, — весьма желательно выполнять все имеющиеся в них примеры, а также экспериментировать, что-нибудь в этих примерах изменяя. Поэтому, если вы удобно устроились на диване и настроились просто читать, у вас практически нет шансов изучить язык. Чем больше вы будете делать самостоятельно, тем большему научитесь.

Ну что, приступим к изучению языка? Python достоин того, чтобы его знал каждый программист!

> ***Внимание!***
>
> Начиная с версии 3.5, Python более не поддерживает Windows XP. В связи с этим в книге не будут описываться моменты, касающиеся его применения под этой версией операционной системы.

1.1. Установка Python

Вначале необходимо установить на компьютер *интерпретатор* Python (его также называют *исполняющей средой*).

1. Для загрузки дистрибутива заходим на страницу **https://www.python.org/downloads/** и в списке доступных версий щелкаем на гиперссылке **Python 3.6.3** (эта версия является самой актуальной из стабильных версий на момент подготовки книги). На открывшейся странице находим раздел **Files** и щелкаем на гиперссылке **Windows x86 executable installer** (32-разрядная редакция интерпретатора) или **Windows x86-64 executable installer** (его 64-разрядная редакция) — в зависимости от версии вашей операционной системы. В результате на наш компьютер будет загружен файл python-3.6.3.exe или python-3.6.3-amd64.exe соответственно. Затем запускаем загруженный файл двойным щелчком на нем.

2. В открывшемся окне (рис. 1.1) проверяем, установлен ли флажок **Install launcher for all users (recommended)** (Установить исполняющую среду для всех пользователей), устанавливаем флажок **Add Python 3.6 to PATH** (Добавить Python 3.6 в список путей переменной PATH) и нажимаем кнопку **Customize installation** (Настроить установку).

3. В следующем диалоговом окне (рис. 1.2) нам предлагается выбрать устанавливаемые компоненты. Оставляем установленными все флажки, представляющие эти компоненты, и нажимаем кнопку **Next**.

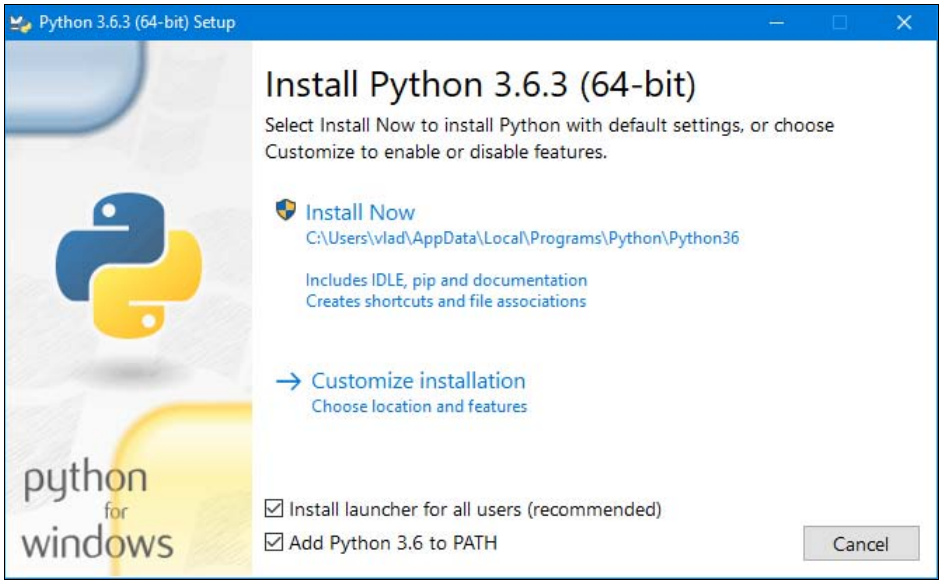

Рис. 1.1. Установка Python. Шаг 1

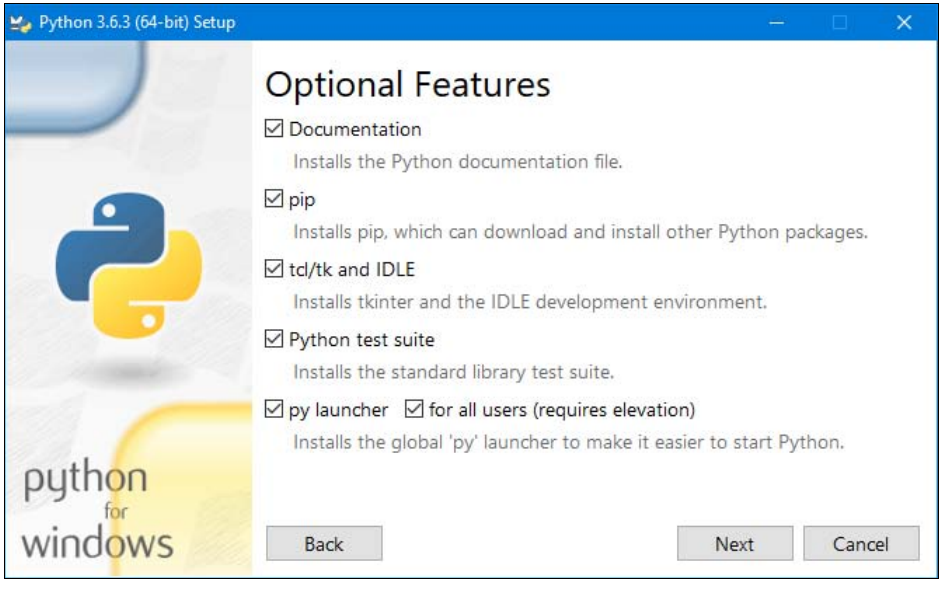

Рис. 1.2. Установка Python. Шаг 2

4. На следующем шаге (рис. 1.3) мы зададим некоторые дополнительные настройки и выберем путь установки. Проверим, установлены ли флажки **Associate files with Python (requires the py launcher)** (Ассоциировать файлы с Python), **Create shortcuts for installed applications** (Создать ярлыки для установленных приложений), **Add Python to environment variables** (Добавить Python в переменные окружения) и **Precompile standard library** (Предварительно откомпилировать стандартную библиотеку), и установим флажок **Install for all users** (Установить для всех пользователей).

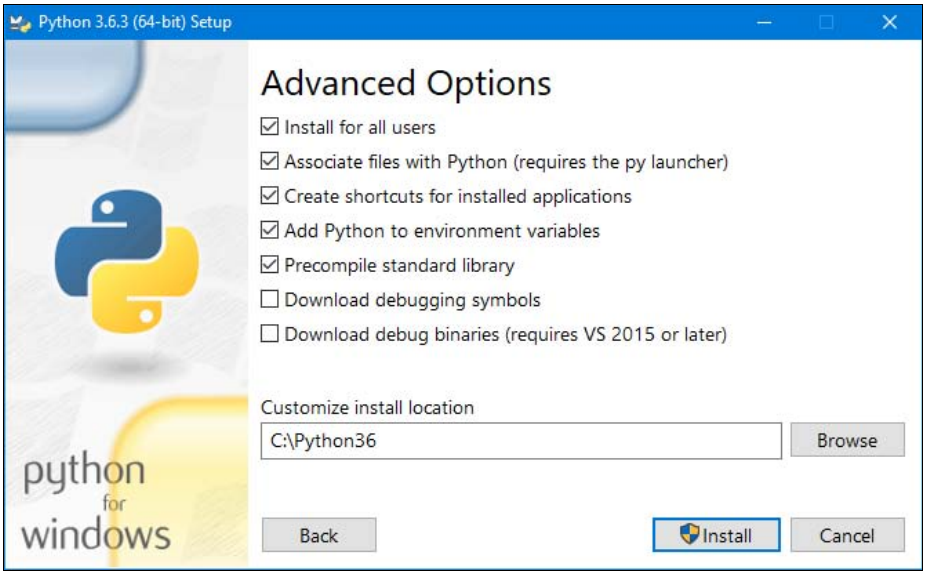

Рис. 1.3. Установка Python. Шаг 3

Внимание!

Некоторые параметры при установке Python приходится задавать по нескольку раз на разных шагах. Вероятно, это недоработка разработчиков инсталлятора.

Теперь уточним путь, по которому будет установлен Python. Изначально нам предлагается установить интерпретатор по пути c:\Program Files\Python36. Можно сделать и так, но тогда при установке любой дополнительной библиотеки понадобится запускать командную строку с повышенными правами, иначе библиотека не установится.

Авторы книги рекомендуют установить Python по пути c:\Python36, т. е. непосредственно в корень диска (см. рис. 1.3). В этом случае мы избежим проблем при установке дополнительных библиотек.

Задав все необходимые параметры, нажимаем кнопку **Install** и положительно отвечаем на появившееся на экране предупреждение UAC.

5. После завершения установки откроется окно, изображенное на рис. 1.4. Нажимаем в нем кнопку **Close** для выхода из программы установки.

В результате установки исходные файлы интерпретатора будут скопированы в папку C:\Python36. В этой папке вы найдете два исполняемых файла: python.exe и pythonw.exe. Файл python.exe предназначен для выполнения консольных приложений. Именно эта программа запускается при двойном щелчке на файле с расширением py. Файл pythonw.exe служит для запуска оконных приложений (при двойном щелчке на файле с расширением pyw) — в этом случае окно консоли выводиться не будет.

Итак, если выполнить двойной щелчок на файле python.exe, то интерактивная оболочка запустится в окне консоли (рис. 1.5). Символы `>>>` в этом окне означают приглашение для ввода инструкций на языке Python. Если после этих символов ввести, например, `2 + 2` и нажать клавишу <Enter>, то на следующей строке сразу будет выведен результат выполнения, а затем опять приглашение для ввода новой инструкции. Таким образом, это окно можно использовать в качестве калькулятора, а также для изучения языка.

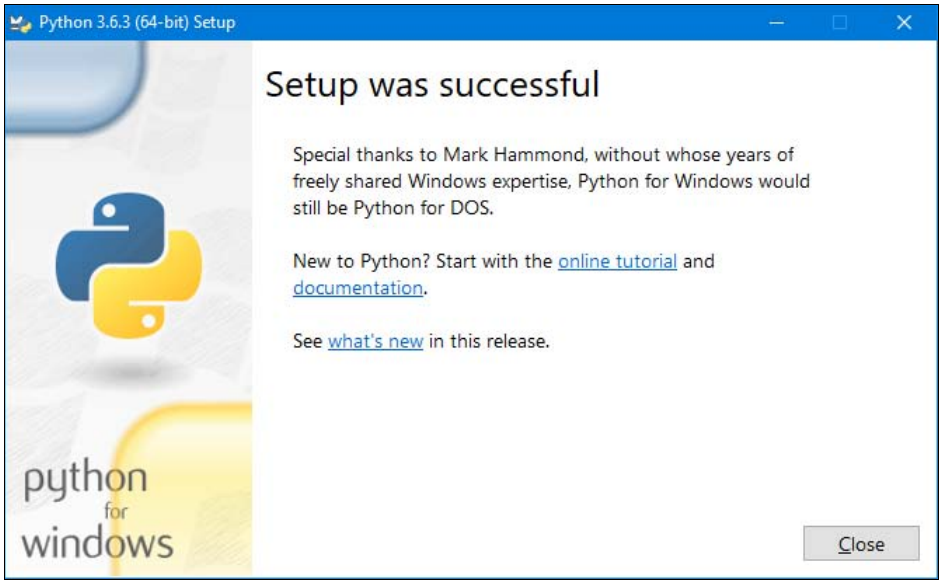

Рис. 1.4. Установка Python. Шаг 4

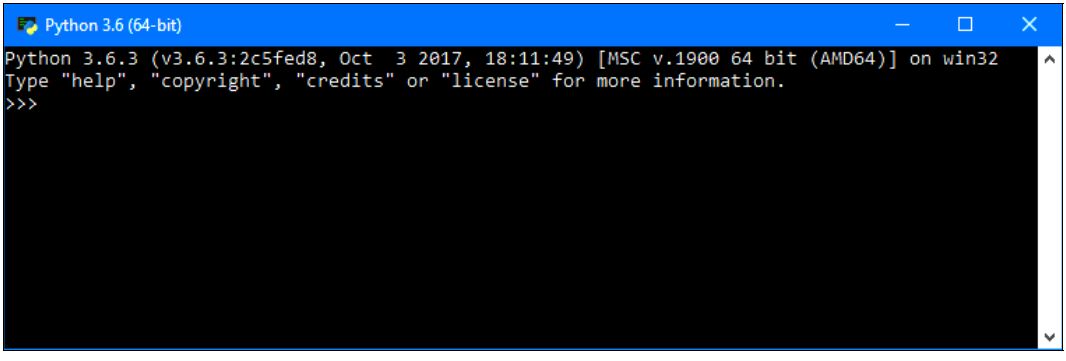

Рис. 1.5. Интерактивная оболочка

Открыть это же окно можно, выбрав пункт **Python 3.6 (32-bit)** или **Python 3.6 (64-bit)** в меню **Пуск | Программы (Все программы) | Python 3.6**.

Однако для изучения языка, а также для создания и редактирования файлов с программами, лучше пользоваться редактором IDLE, который входит в состав установленных компонентов. Для запуска этого редактора в меню **Пуск | Программы (Все программы) | Python 3.6** выбираем пункт **IDLE (Python 3.6 32-bit)** или **IDLE (Python 3.6 64-bit)**. В результате откроется окно **Python Shell** (рис. 1.6), которое выполняет все функции интерактивной оболочки, но дополнительно производит подсветку синтаксиса, выводит подсказки и др. Именно этим редактором мы будем пользоваться в процессе изучения материала книги. Более подробно редактор IDLE мы рассмотрим немного позже.

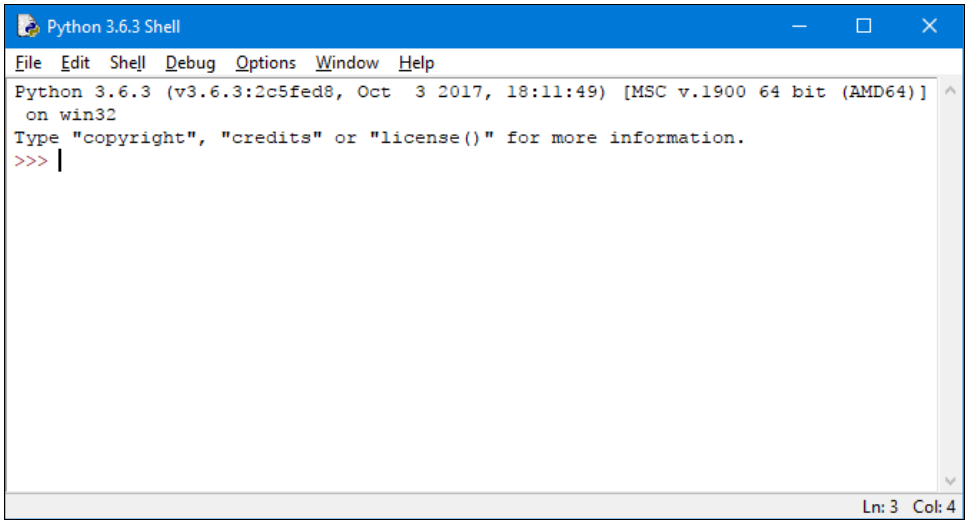

Рис. 1.6. Окно **Python Shell** редактора IDLE

1.1.1. Установка нескольких интерпретаторов Python

Версии языка Python выпускаются с завидной регулярностью, но, к сожалению, сторонние разработчики не успевают за такой скоростью и не столь часто обновляют свои модули. Поэтому иногда приходится при наличии версии Python 3 использовать на практике также и версию Python 2. Как же быть, если установлена версия 3.6, а необходимо запустить модуль для версии 2.7? В этом случае удалять версию 3.6 с компьютера не нужно. Все программы установки позволяют выбрать устанавливаемые компоненты. Существует также возможность задать ассоциацию запускаемой версии с файловым расширением — так вот эту возможность необходимо отключить при установке.

В качестве примера мы дополнительно установим на компьютер версию 2.7.13.2716, но вместо программы установки с сайта **https://www.python.org/** выберем альтернативный дистрибутив от компании ActiveState.

Итак, переходим на страницу **http://www.activestate.com/activepython/downloads/** и скачиваем дистрибутив. Последовательность запуска нескольких программ установки от компании ActiveState имеет значение, поскольку в контекстное меню добавляется пункт **Edit with Pythonwin**. С помощью этого пункта запускается редактор PythonWin, который можно использовать вместо IDLE. Соответственно, из контекстного меню будет открываться версия PythonWin, установленная последней. Установку программы производим в каталог по умолчанию (C:\Python27).

> ***Внимание!***
>
> При установке в окне **Choose Setup Type** (рис. 1.7) необходимо нажать кнопку **Custom**, а в окне **Choose Setup Options** (рис. 1.8) — сбросить флажки **Add Python to the PATH environment variable** и **Create Python file extension associations**. Не забудьте это сделать, иначе Python 3.6.3 перестанет быть текущей версией.

В состав ActivePython, кроме редактора PythonWin, входит также редактор IDLE. Однако в меню **Пуск** нет пункта, с помощью которого можно его запустить. Чтобы это исправить, создадим файл IDLE27.cmd со следующим содержимым:

```
@echo off
start C:\Python27\pythonw.exe C:\Python27\Lib\idlelib\idle.pyw
```

С помощью двойного щелчка на этом файле можно будет запускать редактор IDLE для версии Python 2.7.

Ну, а запуск IDLE для версии Python 3.6 будет по-прежнему осуществляться так же, как и предлагалось ранее, — выбором в меню **Пуск | Программы (Все программы) | Python 3.6** пункта **IDLE (Python 3.6 32-bit)** или **IDLE (Python 3.6 64-bit)**.

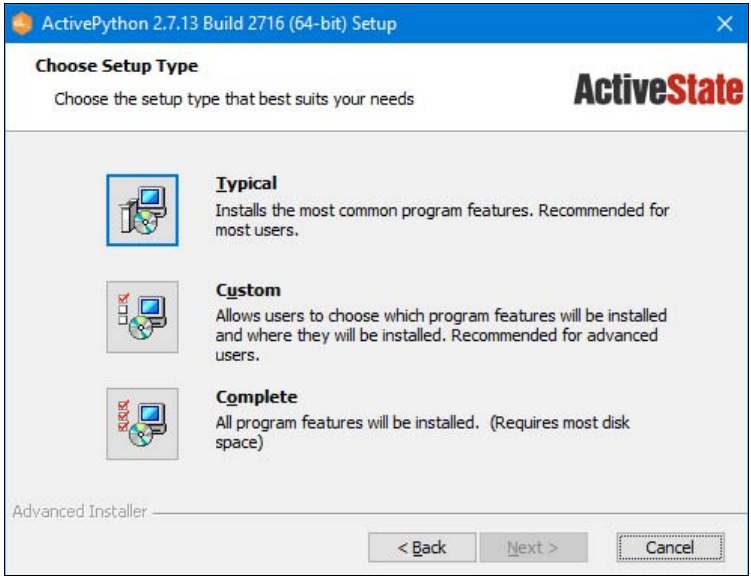

Рис. 1.7. Окно **Choose Setup Type**

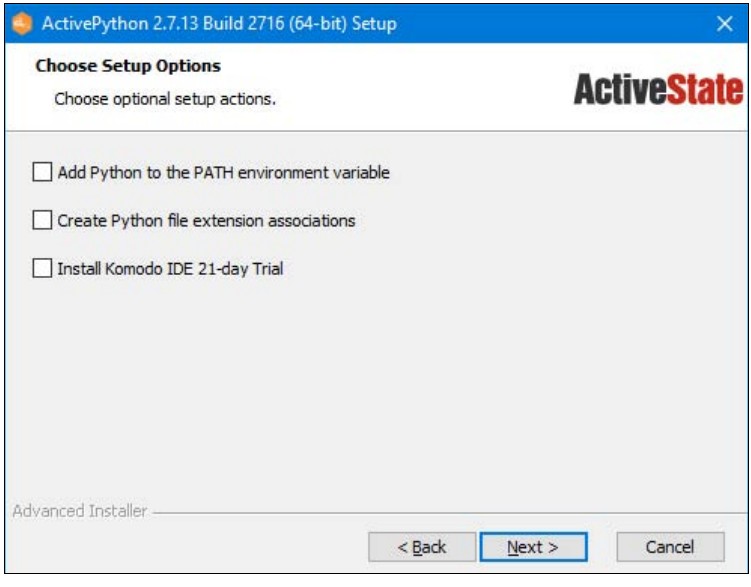

Рис. 1.8. Окно **Choose Setup Options**

1.1.2. Запуск программы с помощью разных версий Python

Теперь рассмотрим запуск программы с помощью разных версий Python. По умолчанию при двойном щелчке на значке файла запускается Python 3.6. Чтобы запустить Python-программу с помощью другой версии этого языка, щелкаем правой кнопкой мыши на значке файла с программой и в контекстном меню находим пункт **Открыть с помощью**.

На экране откроется небольшое окно выбора альтернативной программы для запуска файла (рис. 1.9). Сразу же сбросим флажок **Всегда использовать это приложение для открытия .py файлов** (подпись у этого флажка различается в разных версиях Windows) и щелкнем на гиперссылке **Еще приложения**. В окне появится список установленных на вашем компьютере программ, но нужного нам приложения Python 2.7 в нем не будет. Поэтому щелкнем на ссылке **Найти другое приложение на этом компьютере**, находящейся под списком. На экране откроется стандартное диалоговое окно открытия файла, в котором мы выберем программу python.exe, python2.exe или python2.7.exe из папки C:\Python27.

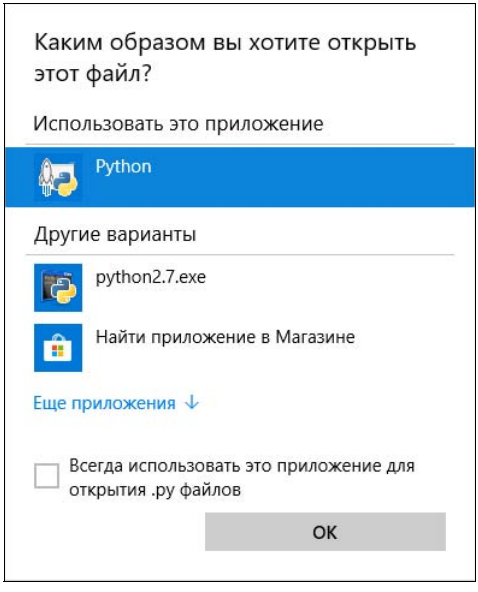

Рис. 1.9. Диалоговое окно выбора альтернативной программы для запуска файла

В Windows Vista, 7, 8 и 8.1 выбранная нами программа появится в подменю, открывающемся при выборе пункта **Открыть с помощью** (рис. 1.10), — здесь Python 2.7 представлен как **Python Launcher for Windows (Console)**. А в Windows 10 она будет присутствовать в списке, что выводится в диалоговом окне выбора альтернативной программы (см. рис. 1.9).

Для проверки установки создайте файл test.py с помощью любого текстового редактора — например, Блокнота. Содержимое файла приведено в листинге 1.1.

Листинг 1.1. Проверка установки

```
import sys
print (tuple(sys.version_info))
try:
    raw_input()         # Python 2
except NameError:
    input()             # Python 3
```

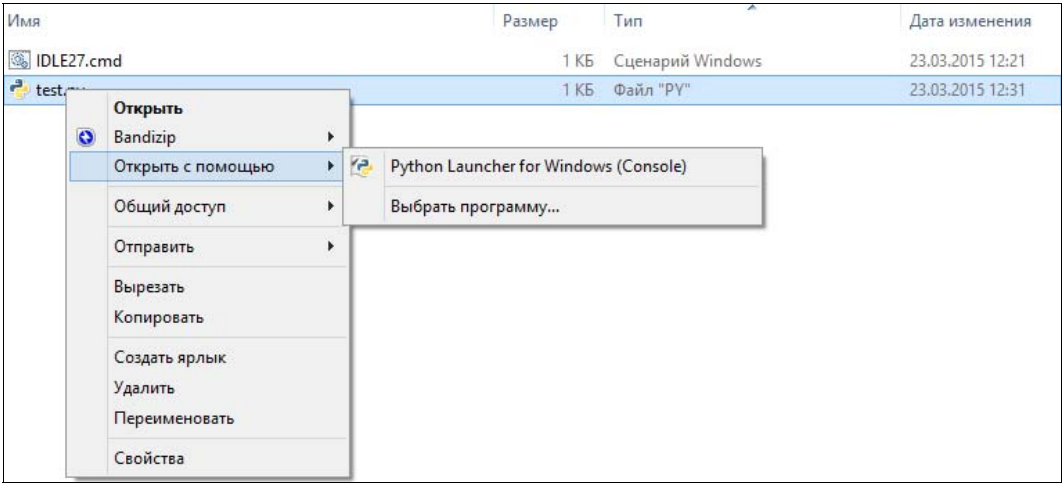

Рис. 1.10. Варианты запуска программы разными версиями Python

Затем запустите программу с помощью двойного щелчка на значке файла. Если результат выполнения: `(3, 6, 3, 'final', 0)`, то установка прошла нормально, а если `(2, 7, 13, 'final', 0)`, то вы не сбросили флажки **Add Python to the PATH environment variable** и **Create Python file extension associations** в окне **Choose Setup Options** (см. рис. 1.8).

Для изучения материала этой книги по умолчанию должна запускаться версия Python 3.6.

1.2. Первая программа на Python

Изучение языков программирования принято начинать с программы, выводящей надпись «Привет, мир!» Не будем нарушать традиции и продемонстрируем, как это будет выглядеть на Python (листинг 1.2).

Листинг 1.2. Первая программа на Python

```
# Выводим надпись с помощью функции print()
print("Привет, мир!")
```

Для запуска программы в меню **Пуск | Программы (Все программы) | Python 3.6** выбираем пункт **IDLE (Python 3.6 32-bit)** или **IDLE (Python 3.6 64-bit)**. В результате откроется окно **Python Shell**, в котором символы `>>>` означают приглашение ввести команду (см. рис. 1.6). Вводим сначала первую строку из листинга 1.2, а затем вторую. После ввода каждой строки нажимаем клавишу <Enter>. На следующей строке сразу отобразится результат, а далее — приглашение для ввода новой команды. Последовательность выполнения нашей программы такова:

```
>>> # Выводим надпись с помощью функции print()
>>> print("Привет, мир!")
Привет, мир!
>>>
```

Примечание

Символы `>>>` вводить не нужно, они вставляются автоматически.

Для создания файла с программой в меню **File** выбираем пункт **New File** или нажимаем комбинацию клавиш <Ctrl>+<N>. В открывшемся окне набираем код из листинга 1.2, а затем сохраняем его под именем hello_world.py, выбрав пункт меню **File | Save** (комбинация клавиш <Ctrl>+<S>). При этом редактор сохранит файл в кодировке UTF-8 без BOM (Byte Order Mark, метка порядка байтов). Именно кодировка UTF-8 является кодировкой по умолчанию в Python 3. Если файл содержит инструкции в другой кодировке, то необходимо в первой или второй строке указать кодировку с помощью инструкции:

```
# -*- coding: <Кодировка> -*-
```

Например, для кодировки Windows-1251 инструкция будет выглядеть так:

```
# -*- coding: cp1251 -*-
```

Редактор IDLE учитывает указанную кодировку и автоматически производит перекодирование при сохранении файла. При использовании других редакторов следует проконтролировать соответствие указанной кодировки и реальной кодировки файла. Если кодировки не совпадают, то данные будут преобразованы некорректно, или во время преобразования произойдет ошибка.

Запустить программу на выполнение можно, выбрав пункт меню **Run | Run Module** или нажав клавишу <F5>. Результат выполнения программы будет отображен в окне **Python Shell**.

Запустить программу также можно с помощью двойного щелчка мыши на значке файла. В этом случае результат выполнения будет отображен в консоли Windows. Следует учитывать, что после вывода результата окно консоли сразу закрывается. Чтобы предотвратить закрытие окна, необходимо добавить вызов функции `input()`, которая станет ожидать нажатия клавиши <Enter> и не позволит окну сразу закрыться. С учетом сказанного наша программа будет выглядеть так, как показано в листинге 1.3.

Листинг 1.3. Программа для запуска с помощью двойного щелчка мыши

```
# -*- coding: utf-8 -*-
print("Привет, мир!")          # Выводим строку
input()                        # Ожидаем нажатия клавиши <Enter>
```

Примечание

Если до выполнения функции `input()` возникнет ошибка, то сообщение о ней будет выведено в консоль, но сама консоль после этого сразу закроется, и вы не сможете прочитать сообщение об ошибке. Попав в подобную ситуацию, запустите программу из командной строки или с помощью редактора IDLE, и вы сможете прочитать сообщение об ошибке.

В языке Python 3 строки по умолчанию хранятся в кодировке Unicode. При выводе кодировка Unicode автоматически преобразуется в кодировку терминала. Поэтому русские буквы отображаются корректно, хотя в окне консоли в Windows по умолчанию используется кодировка cp866, а файл с программой у нас в кодировке UTF-8.

Чтобы отредактировать уже созданный файл, запустим IDLE, выполним команду меню **File | Open** (комбинация клавиш <Ctrl>+<O>) и выберем нужный файл, который будет открыт в другом окне.

Напоминание

Поскольку программа на языке Python представляет собой обычный текстовый файл, сохраненный с расширением *py* или *pyw*, его можно редактировать с помощью других про-

грамм, например Notepad++. Можно также воспользоваться специализированными редакторами, скажем, PyScripter.

Когда интерпретатор Python начинает выполнение программы, хранящейся в файле, он сначала компилирует ее в особое внутреннее представление, — это делается с целью увеличить производительность кода. Файл с откомпилированным кодом хранится в папке __pycache__, вложенной в папку, где хранится сам файл программы, а его имя имеет следующий вид:

<имя файла с исходным, неоткомпилированным кодом>.cpython-<первые две цифры номера версии Python>.pyc

Так, при запуске на исполнение файла test4.py будет создан файл откомпилированного кода с именем test4.cpython-36.pyc.

При последующем запуске на выполнение того же файла будет исполняться именно откомпилированный код. Если же мы исправим исходный код, программа его автоматически перекомпилирует. При необходимости мы можем удалить файлы с откомпилированным кодом или даже саму папку __pycache__ — впоследствии интерпретатор сформирует их заново.

1.3. Структура программы

Как вы уже знаете, программа на языке Python представляет собой обычный текстовый файл с инструкциями. Каждая инструкция располагается на отдельной строке. Если инструкция не является вложенной, она должна начинаться с начала строки, иначе будет выведено сообщение об ошибке:

```
>>>  import sys


SyntaxError: unexpected indent
>>>
```

В этом случае перед инструкцией `import` расположен один лишний пробел, который привел к выводу сообщения об ошибке.

Если программа предназначена для исполнения в операционной системе UNIX, то в первой строке необходимо указать путь к интерпретатору Python:

```
#!/usr/bin/python
```

В некоторых операционных системах путь к интерпретатору выглядит по-другому:

```
#!/usr/local/bin/python
```

Иногда можно не указывать точный путь к интерпретатору, а передать название языка программе `env`:

```
#!/usr/bin/env python
```

В этом случае программа `env` произведет поиск интерпретатора Python в соответствии с настройками путей поиска.

Помимо указания пути к интерпретатору Python, необходимо, чтобы в правах доступа к файлу был установлен бит на выполнение. Кроме того, следует помнить, что перевод строки в операционной системе Windows состоит из последовательности символов `\r` (перевод каретки) и `\n` (перевод строки). В операционной системе UNIX перевод строки осу-

ществляется только одним символом `\n`. Если загрузить файл программы по протоколу FTP в бинарном режиме, то символ `\r` вызовет фатальную ошибку. По этой причине файлы по протоколу FTP следует загружать только в текстовом режиме (режим ASCII). В этом режиме символ `\r` будет удален автоматически.

После загрузки файла следует установить права на выполнение. Для исполнения скриптов на Python устанавливаем права в 755 (`-rwxr-xr-x`).

Во второй строке (для ОС Windows в первой строке) следует указать кодировку. Если кодировка не указана, то предполагается, что файл сохранен в кодировке UTF-8. Для кодировки Windows-1251 строка будет выглядеть так:

```
# -*- coding: cp1251 -*-
```

Редактор IDLE учитывает указанную кодировку и автоматически производит перекодирование при сохранении файла. Получить полный список поддерживаемых кодировок и их псевдонимы позволяет код, приведенный в листинге 1.4.

Листинг 1.4. Вывод списка поддерживаемых кодировок

```
# -*- coding: utf-8 -*-
import encodings.aliases
arr = encodings.aliases.aliases
keys = list( arr.keys() )
keys.sort()
for key in keys:
    print("%s => %s" % (key, arr[key]))
```

Во многих языках программирования (например, в PHP, Perl и др.) каждая инструкция должна завершаться точкой с запятой. В языке Python в конце инструкции также можно поставить точку с запятой, но это не обязательно. Более того, в отличие от языка JavaScript, где рекомендуется завершать инструкции точкой с запятой, в языке Python точку с запятой ставить *не рекомендуется*. Концом инструкции является конец строки. Тем не менее, если необходимо разместить несколько инструкций на одной строке, точку с запятой *следует указать*:

```
>>> x = 5; y = 10; z = x + y # Три инструкции на одной строке
>>> print(z)
15
```

Еще одной отличительной особенностью языка Python является отсутствие ограничительных символов для выделения инструкций внутри блока. Например, в языке PHP инструкции внутри цикла `while` выделяются фигурными скобками:

```
$i = 1;
while ($i < 11) {
   echo $i . "\n";
   $i++;
}
echo "Конец программы";
```

В языке Python тот же код будет выглядеть по-другому (листинг 1.5).

Листинг 1.5. Выделение инструкций внутри блока

```
i = 1
while i < 11:
    print(i)
    i += 1
print("Конец программы")
```

Обратите внимание, что перед всеми инструкциями внутри блока расположено одинаковое количество пробелов. Именно так в языке Python выделяются *блоки*. Инструкции, перед которыми расположено одинаковое количество пробелов, являются *телом блока*. В нашем примере две инструкции выполняются десять раз. Концом блока является инструкция, перед которой расположено меньшее количество пробелов. В нашем случае это функция `print()`, которая выводит строку `"Конец программы"`. Если количество пробелов внутри блока окажется разным, то интерпретатор выведет сообщение о фатальной ошибке, и программа будет остановлена. Так язык Python приучает программистов писать красивый и понятный код.

> ***Примечание***
>
> В языке Python принято использовать четыре пробела для выделения инструкций внутри блока.

Если блок состоит из одной инструкции, то допустимо разместить ее на одной строке с основной инструкцией. Например, код:

```
for i in range(1, 11):
    print(i)
print("Конец программы")
```

можно записать так:

```
for i in range(1, 11): print(i)
print("Конец программы")
```

Если инструкция является слишком длинной, то ее можно перенести на следующую строку, например, так:

- в конце строки поставить символ \, после которого должен следовать перевод строки. Другие символы (в том числе и комментарии) недопустимы. Пример:

```
x = 15 + 20 \
    + 30
print(x)
```

- поместить выражение внутри круглых скобок. Этот способ лучше, т. к. внутри круглых скобок можно разместить любое выражение. Пример:

```
x = (15 + 20                # Это комментарий
    + 30)
print(x)
```

- определение списка и словаря можно разместить на нескольких строках, т. к. при этом используются квадратные и фигурные скобки соответственно. Пример определения списка:

```
arr = [15, 20,              # Это комментарий
       30]
print(arr)
```