Николай Прохоренок, Владимир Дронов

Python 3 и PyQt 5

Разработка приложений

- Описание языка Python 3
- Использование библиотеки PyQt 5
- Объектно-ориентированное программирование
- Работа с файлами и каталогами
- Получение данных из Интернета
- Создание оконных приложений
- Работа с базами данных
- Мультимедиа
- Печать и экспорт в формат PDF

Материалы на www.bhv.ru

bhv®

УДК 004.43
ББК 32.973.26-018.1
П84

Прохоренок, Н. А.

П84 Python 3 и PyQt 5. Разработка приложений / Н. А. Прохоренок, В. А. Дронов. — СПб.: БХВ-Петербург, 2016. — 832 с.: ил.

ISBN 978-5-9775-3648-6

Описан язык Python 3: типы данных, операторы, условия, циклы, регулярные выражения, функции, инструменты объектно-ориентированного программирования, работа с файлами и каталогами, часто используемые модули стандартной библиотеки. Приведены основы базы данных SQLite, интерфейс доступа к базе и способы получения данных из Интернета. Особое внимание уделено библиотеке PyQt 5, позволяющей создавать приложения с графическим интерфейсом на языке Python. Рассмотрены средства для обработки сигналов и событий, управления свойствами окна, разработки многопоточных приложений, описаны основные компоненты (кнопки, текстовые поля, списки, таблицы, меню, панели инструментов и др.), варианты их размещения внутри окна, инструменты для работы с базами данных, мультимедиа, печати документов и экспорта их в формате Adobe PDF. На сайте издательства приведены все примеры из книги.

Для программистов

УДК 004.43
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Григорий Добин*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Марины Дамбиевой*

Подписано в печать 30.11.15.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 67,08.
Тираж 1000 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-3648-6

Оглавление

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!

1.1. Установка Python

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

1. Для загрузки дистрибутива переходим на страницу **https://www.python.org/downloads/** и в списке доступных версий щелкаем на гиперссылке **Python 3.4.3** (эта версия является самой актуальной из стабильных версий на момент подготовки книги). На открывшейся странице находим раздел **Files** и щелкаем на гиперссылке **Windows x86 MSI installer** (32-разрядная версия интерпретатора) или **Windows x86-64 MSI installer** (его 64-разрядная версия). В результате на наш компьютер будет загружен файл python-3.4.3.msi или python-3.4.3.amd64.msi соответственно. Затем запускаем загруженный файл двойным щелчком на нем.

2. В открывшемся окне (рис. 1.1) устанавливаем переключатель **Install for all users** (Установить для всех пользователей) и нажимаем кнопку **Next**.

3. На следующем шаге (рис. 1.2) нам предлагается выбрать каталог для установки. Оставляем каталог по умолчанию (C:\Python34\) и нажимаем кнопку **Next**.

4. В следующем диалоговом окне (рис. 1.3) выбираем компоненты, которые необходимо установить. По умолчанию устанавливаются все компоненты и прописывается ассоциация с файловыми расширениями py, pyw и др. В этом случае запускать Python-программы можно будет с помощью двойного щелчка мышью на значке файла. Оставляем выбранными все компоненты и нажимаем кнопку **Next**.

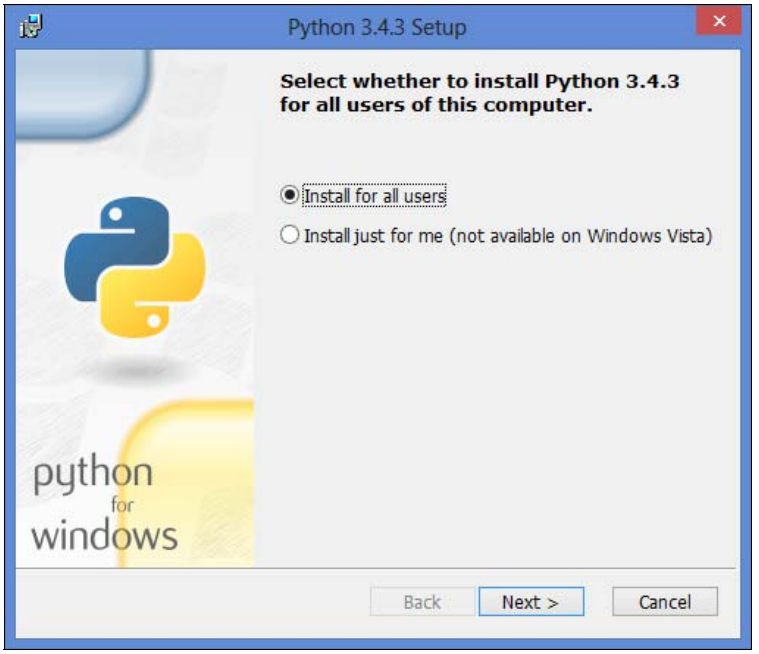

Рис. 1.1. Установка Python. Шаг 1

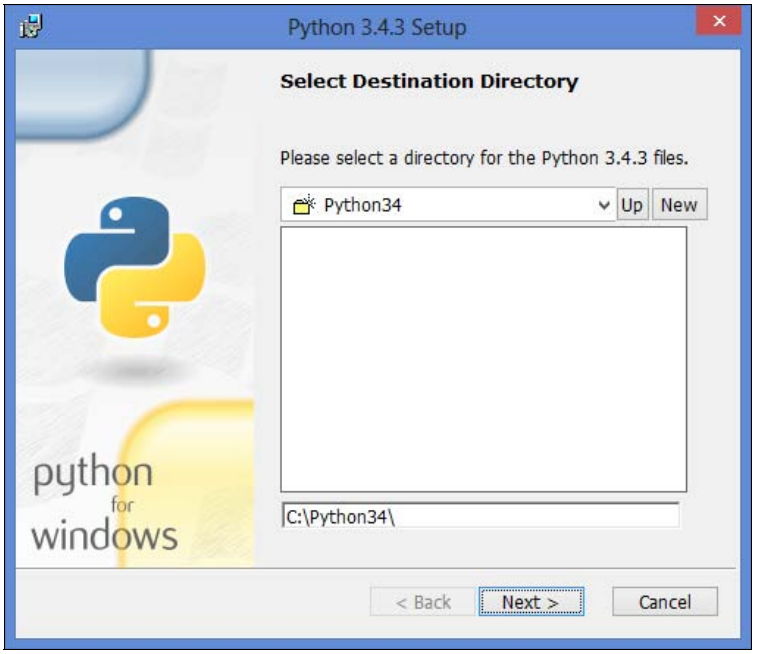

Рис. 1.2. Установка Python. Шаг 2

Рис. 1.3. Установка Python. Шаг 3

Примечание

Пользователям Windows Vista и более поздних версий этой системы следует положительно ответить на появившееся на экране предупреждение системы UAC (Контроль учетных записей). Если этого не сделать, Python установлен не будет.

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

В результате установки исходные файлы интерпретатора будут скопированы в папку C:\Python34. 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.

Открыть такое же окно можно с помощью пункта **Python 3.4 (command line — 32 bit)** или **Python 3.4 (command line — 64 bit)** в меню **Пуск | Программы (Все программы) | Python 3.4**.

Вместо такой интерактивной оболочки для изучения языка, а также для создания и редактирования файлов с программой лучше воспользоваться редактором IDLE, который входит

Рис. 1.4. Установка Python. Шаг 4

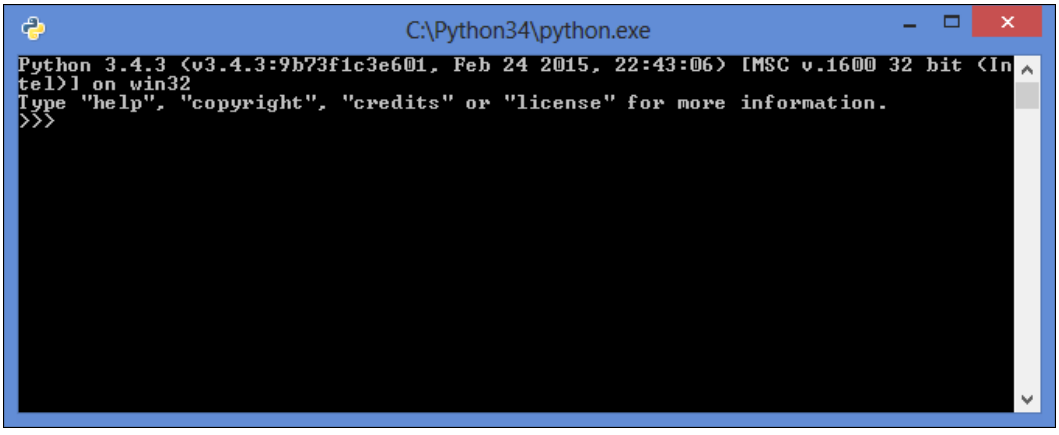

Рис. 1.5. Интерактивная оболочка

в состав установленных компонентов. Для запуска редактора в меню **Пуск | Программы (Все программы) | Python 3.4** выбираем пункт **IDLE (Python 3.4 GUI — 32 bit)** или **IDLE (Python 3.4 GUI — 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.

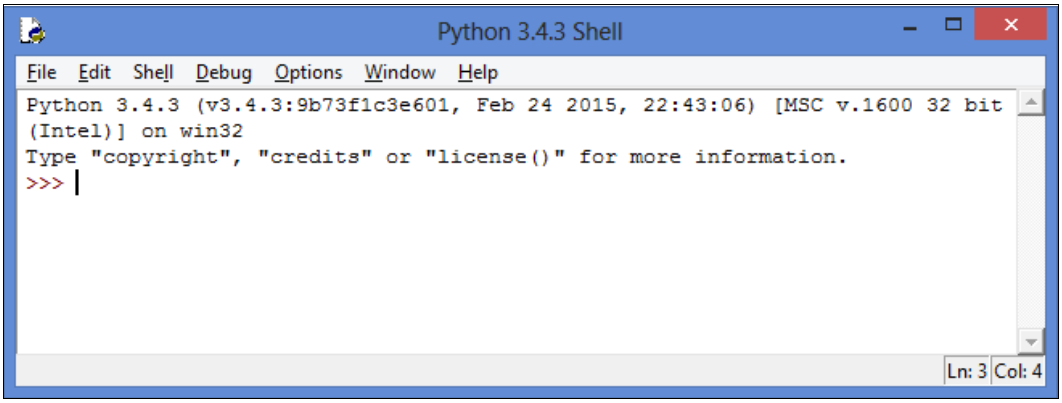

Рис. 1.6. Окно **Python Shell** редактора IDLE

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 приходится при наличии версии Python 3 использовать на практике также и версию Python 2. Как же быть, если установлена версия 3.4, а необходимо запустить модуль для версии 2.7? В этом случае удалять версию 3.4 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 при установке просто отключить.

В качестве примера мы дополнительно установим на компьютер версию 2.7.8.10, 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, которая была установлена последней. Установку программы производим в каталог по умолчанию (C:\Python27\).

> ***Внимание!***
>
> При установке в окне **Custom Setup** (рис. 1.7) необходимо отключить компонент **Register as Default Python** (рис. 1.8). Не забудьте это сделать, иначе Python 3.4.3 перестанет быть текущей версией.

В состав ActivePython, кроме редактора PythonWin, входит также редактор IDLE. Однако ни в одном меню 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.

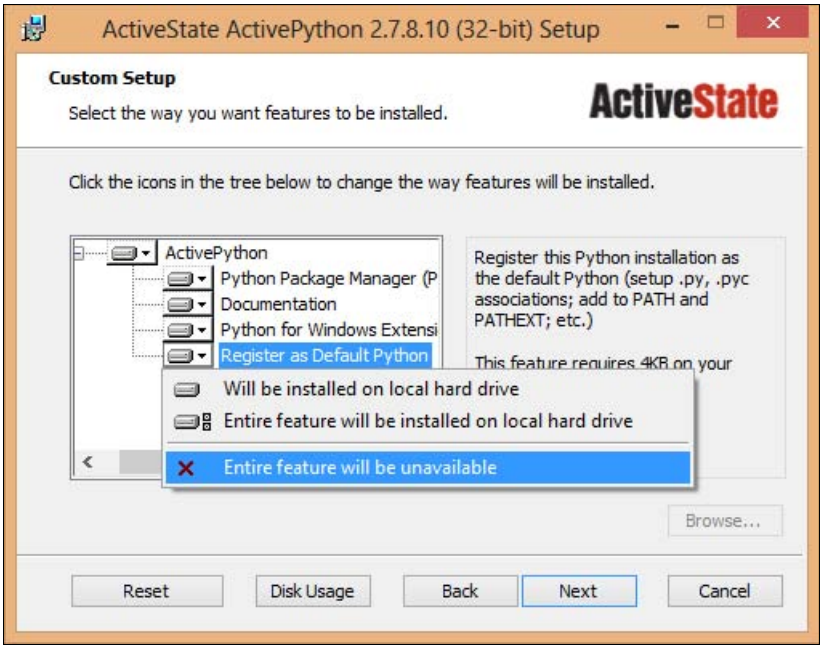

Рис. 1.7. Окно **Custom Setup**

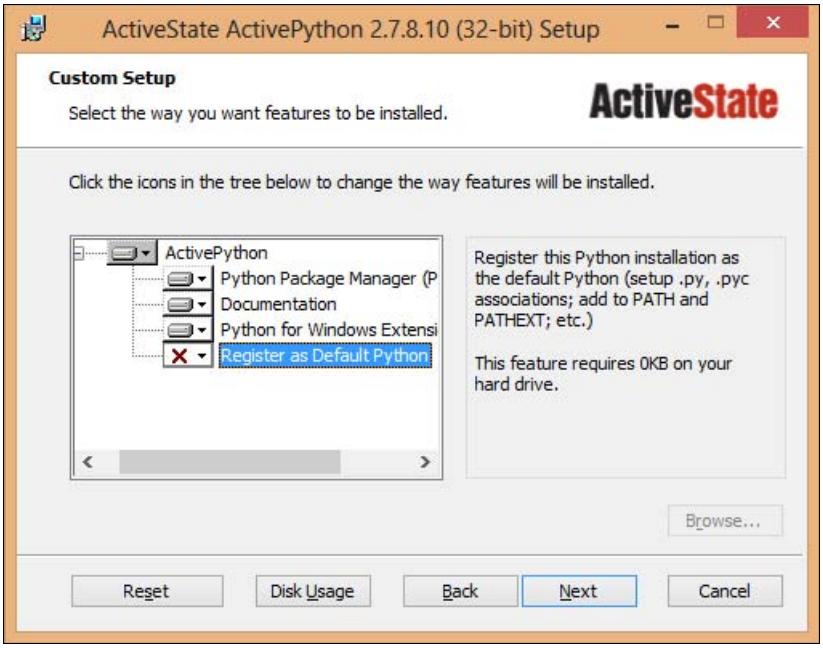

Рис. 1.8. Компонент **Register as Default Python** отключен

Ну, а запуск IDLE для версии Python 3.4 будет по-прежнему осуществляться так же, как и предлагалось ранее, — выбором в меню **Пуск | Программы (Все программы) | Python 3.4** пункта **IDLE (Python 3.4 GUI — 32 bit)** или **IDLE (Python 3.4 GUI — 64 bit)**.

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 3.4. 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**.

В Windows XP при выборе этого пункта появится подменю, в котором изначально будет присутствовать только программа python.exe. Чтобы добавить другую версию, щелкаем на пункте **Выбрать программу**, в открывшемся окне нажимаем кнопку **Обзор** и выбираем программу python2.7.exe из папки C:\Python27. Выбранная нами программа будет добавлена в подменю, открывающееся при выборе пункта **Открыть с помощью**. Впоследствии для выполнения файла под управлением Python 2.7 мы просто выберем этот пункт.

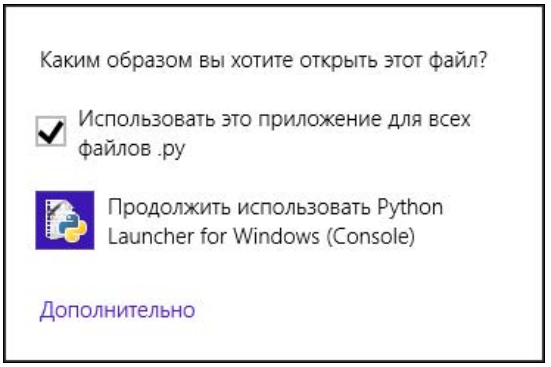

Рис. 1.9. Диалоговое окно выбора альтернативной программы для запуска файла

В Windows Vista и более поздних версиях этой системы при выборе упомянутого пункта изначально не будет открываться никакого подменю. Вместо этого на экране появится небольшое окно выбора альтернативной программы для запуска файла (рис. 1.9). Сразу же сбросим флажок **Использовать это приложение для всех файлов .py** и нажмем ссылку **Дополнительно**. В окне появится список установленных на нашем компьютере программ, но нужного нам приложения python2.7.exe в нем не будет. Поэтому щелкнем на ссылке **Найти другое приложение на этом компьютере**, находящейся под списком. На экране появится стандартное диалоговое окно открытия файла, в котором мы выберем программу python2.7.exe из папки C:\Python27. Теперь эта 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)**.

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
```

Затем запустите программу с помощью двойного щелчка на значке файла. Если результат выполнения: `(3, 4, 3, 'final', 0)`, то установка прошла нормально, а если `(2, 7, 8, 'final', 0)`, то вы не отключили компонент **Register as Default Python**.

Для изучения материала этой книги по умолчанию должна запускаться версия Python 3.4.

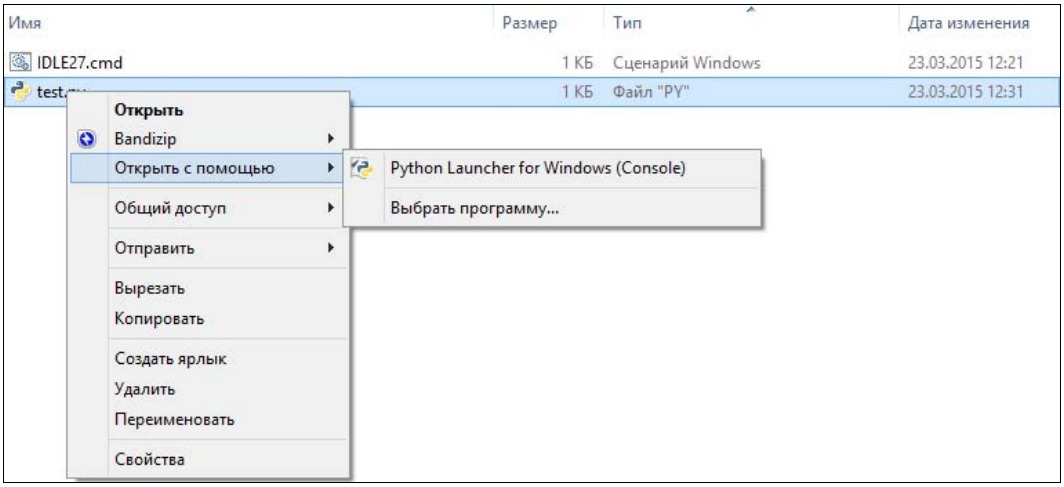

Рис. 1.10. Варианты запуска программы разными версиями Python

1.2. Первая программа на Python

Изучение языков программирования принято начинать с программы, выводящей надпись "Привет, мир!" Не будем нарушать традицию 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 3.4** выбираем пункт **IDLE (Python 3.4 GUI — 32 bit)** или **IDLE (Python 3.4 GUI — 64 bit)**. В результате откроется окно **Python Shell**, в котором символы >>> означают приглашение ввести команду. 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 показана в листинге 1.3.

Листинг 1.3. Последовательность выполнения программы в окне *Python Shell*

```
>>> # 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**. В открывшемся окне набираем код из листинга 1.2, а затем сохраняем его под именем hello_world.py, выбрав пункт меню **File | Save As**. 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 можно также 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 1.4.

Листинг 1.4. Программа для запуска с помощью двойного щелчка мыши

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

Примечание

Если до 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** и укажем 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 программ — 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-34.pyc.

При последующем запуске того же файла на выполнение 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, то она должна начинаться с начала строки, иначе будет выведено сообщение об ошибке (листинг 1.5).

Листинг 1.5. Ошибка `SyntaxError`

```
>>>  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 дополнительно 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 двух символов: `\r` (перевод каретки) и `\n` (перевод строки). В операционной системе UNIX перевод строки осуществляется 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 1.6.

Листинг 1.6. 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* (листинг 1.7).

Листинг 1.7. Несколько инструкций на одной строке

```
>>> 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 1.8).

Листинг 1.8. 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 разместить символ \. После этого символа должен следовать символ перевода строки. Другие символы (в том числе и комментарии) недопустимы.