

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



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	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Екатерина Капалыгина</i>
Редактор	<i>Григорий Добин</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн обложки	<i>Марины Дамбиевой</i>

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

© Прохоренок Н. А., Дронов В. А., 2016
© Оформление, издательство "БХВ-Петербург", 2016

Оглавление

- Введение 15**
- ЧАСТЬ I. ОСНОВЫ ЯЗЫКА PYTHON..... 17**
- Глава 1. Первые шаги 19**
 - 1.1. Установка Python 19
 - 1.1.1. Установка нескольких интерпретаторов Python 23
 - 1.1.2. Запуск программы с помощью разных версий Python 25
 - 1.2. Первая программа на Python..... 26
 - 1.3. Структура программы 28
 - 1.4. Комментарии..... 31
 - 1.5. Скрытые возможности IDLE 32
 - 1.6. Вывод результатов работы программы 33
 - 1.7. Ввод данных..... 35
 - 1.8. Доступ к документации..... 37
- Глава 2. Переменные 40**
 - 2.1. Именованние переменных 40
 - 2.2. Типы данных 42
 - 2.3. Присваивание значения переменным 45
 - 2.4. Проверка типа данных..... 47
 - 2.5. Преобразование типов данных 48
 - 2.6. Удаление переменной 51
- Глава 3. Операторы 52**
 - 3.1. Математические операторы..... 52
 - 3.2. Двоичные операторы..... 54
 - 3.3. Операторы для работы с последовательностями..... 55
 - 3.4. Операторы присваивания..... 56
 - 3.5. Приоритет выполнения операторов 57
- Глава 4. Условные операторы и циклы 59**
 - 4.1. Операторы сравнения..... 60
 - 4.2. Оператор ветвления *if...else* 62

4.3. Цикл <i>for</i>	65
4.4. Функции <i>range()</i> и <i>enumerate()</i>	67
4.5. Цикл <i>while</i>	70
4.6. Оператор <i>continue</i> . Переход на следующую итерацию цикла	71
4.7. Оператор <i>break</i> . Прерывание цикла	71
Глава 5. Числа.....	73
5.1. Встроенные функции и методы для работы с числами	75
5.2. Модуль <i>math</i> . Математические функции.....	77
5.3. Модуль <i>random</i> . Генерация случайных чисел	78
Глава 6. Строки и двоичные данные	81
6.1. Создание строки.....	82
6.2. Специальные символы	86
6.3. Операции над строками.....	86
6.4. Форматирование строк.....	89
6.5. Метод <i>format()</i>	95
6.6. Функции и методы для работы со строками	99
6.7. Настройка локали	103
6.8. Изменение регистра символов.....	104
6.9. Функции для работы с символами	104
6.10. Поиск и замена в строке.....	105
6.11. Проверка типа содержимого строки	108
6.12. Тип данных <i>bytes</i>	111
6.13. Тип данных <i>bytearray</i>	115
6.14. Преобразование объекта в последовательность байтов	118
6.15. Шифрование строк	119
Глава 7. Регулярные выражения	121
7.1. Синтаксис регулярных выражений	121
7.2. Поиск первого совпадения с шаблоном.....	130
7.3. Поиск всех совпадений с шаблоном	135
7.4. Замена в строке	137
7.5. Прочие функции и методы.....	139
Глава 8. Списки, кортежи, множества и диапазоны	140
8.1. Создание списка.....	141
8.2. Операции над списками	144
8.3. Многомерные списки.....	147
8.4. Перебор элементов списка.....	148
8.5. Генераторы списков и выражения-генераторы.....	149
8.6. Функции <i>map()</i> , <i>zip()</i> , <i>filter()</i> и <i>reduce()</i>	150
8.7. Добавление и удаление элементов списка.....	153
8.8. Поиск элемента в списке и получение сведений о значениях, входящих в список	155
8.9. Переворачивание и перемешивание списка	157
8.10. Выбор элементов случайным образом.....	157
8.11. Сортировка списка.....	158
8.12. Заполнение списка числами.....	159
8.13. Преобразование списка в строку	160

8.14. Кортежи	160
8.15. Множества	162
8.16. Диапазоны	167
8.17. Модуль <i>itertools</i>	169
8.17.1. Генерация неопределенного количества значений	169
8.17.2. Генерация комбинаций значений	170
8.17.3. Фильтрация элементов последовательности	171
8.17.4. Прочие функции	172
Глава 9. Словари	175
9.1. Создание словаря	175
9.2. Операции над словарями	178
9.3. Перебор элементов словаря	179
9.4. Методы для работы со словарями	180
9.5. Генераторы словарей	183
Глава 10. Работа с датой и временем	184
10.1. Получение текущих даты и времени	184
10.2. Форматирование даты и времени	186
10.3. «Засыпание» скрипта	188
10.4. Модуль <i>datetime</i> . Манипуляции датой и временем	189
10.4.1. Класс <i>timedelta</i>	189
10.4.2. Класс <i>date</i>	191
10.4.3. Класс <i>time</i>	195
10.4.4. Класс <i>datetime</i>	197
10.5. Модуль <i>calendar</i> . Вывод календаря	201
10.5.1. Методы классов <i>TextCalendar</i> и <i>LocaleTextCalendar</i>	203
10.5.2. Методы классов <i>HTMLCalendar</i> и <i>LocaleHTMLCalendar</i>	204
10.5.3. Другие полезные функции	205
10.6. Измерение времени выполнения фрагментов кода	208
Глава 11. Пользовательские функции	211
11.1. Определение функции и ее вызов	211
11.2. Расположение определений функций	214
11.3. Необязательные параметры и сопоставление по ключам	215
11.4. Переменное число параметров в функции	218
11.5. Анонимные функции	220
11.6. Функции-генераторы	221
11.7. Декораторы функций	222
11.8. Рекурсия. Вычисление факториала	224
11.9. Глобальные и локальные переменные	225
11.10. Вложенные функции	228
11.11. Аннотации функций	230
Глава 12. Модули и пакеты	231
12.1. Инструкция <i>import</i>	231
12.2. Инструкция <i>from</i>	235
12.3. Пути поиска модулей	237
12.4. Повторная загрузка модулей	238
12.5. Пакеты	239

Глава 13. Объектно-ориентированное программирование	243
13.1. Определение класса и создание экземпляра класса.....	243
13.2. Методы <code>__init__()</code> и <code>__del__()</code>	247
13.3. Наследование	247
13.4. Множественное наследование	249
13.4.1. Примеси и их использование.....	251
13.5. Специальные методы.....	252
13.6. Перегрузка операторов.....	255
13.7. Статические методы и методы класса	257
13.8. Абстрактные методы	258
13.9. Ограничение доступа к идентификаторам внутри класса	260
13.10. Свойства класса	261
13.11. Декораторы классов	263
Глава 14. Обработка исключений.....	264
14.1. Инструкция <code>try...except...else...finally</code>	265
14.2. Инструкция <code>with...as</code>	269
14.3. Классы встроенных исключений.....	271
14.4. Пользовательские исключения	273
Глава 15. Итераторы, контейнеры и перечисления	277
15.1. Итераторы	278
15.2. Контейнеры	279
15.2.1. Контейнеры-последовательности.....	279
15.2.2. Контейнеры-словари	281
15.3. Перечисления	282
Глава 16. Работа с файлами и каталогами.....	287
16.1. Открытие файла	287
16.2. Методы для работы с файлами.....	294
16.3. Доступ к файлам с помощью модуля <code>os</code>	300
16.4. Классы <code>StringIO</code> и <code>BytesIO</code>	302
16.5. Права доступа к файлам и каталогам	306
16.6. Функции для манипулирования файлами	308
16.7. Преобразование пути к файлу или каталогу.....	311
16.8. Перенаправление ввода/вывода.....	313
16.9. Сохранение объектов в файл	316
16.10. Функции для работы с каталогами.....	320
16.11. Исключения, возбуждаемые файловыми операциями	323
Глава 17. Основы SQLite	325
17.1. Создание базы данных	325
17.2. Создание таблицы	327
17.3. Вставка записей	333
17.4. Обновление и удаление записей.....	336
17.5. Изменение структуры таблицы.....	336
17.6. Выбор записей	337
17.7. Выбор записей из нескольких таблиц	340
17.8. Условия в инструкциях <code>WHERE</code> и <code>HAVING</code>	342

17.9. Индексы	345
17.10. Вложенные запросы	347
17.11. Транзакции	348
17.12. Удаление таблицы и базы данных.....	351

Глава 18. Доступ к базе данных SQLite из Python352

18.1. Создание и открытие базы данных	353
18.2. Выполнение запросов.....	354
18.3. Обработка результата запроса	358
18.4. Управление транзакциями	362
18.5. Создание пользовательской сортировки.....	364
18.6. Поиск без учета регистра символов	365
18.7. Создание агрегатных функций	366
18.8. Преобразование типов данных	367
18.9. Сохранение в таблице даты и времени	371
18.10. Обработка исключений	372
18.11. Трассировка выполняемых запросов	375

Глава 19. Взаимодействие с Интернетом376

19.1. Разбор URL-адреса	376
19.2. Кодирование и декодирование строки запроса.....	379
19.3. Преобразование относительного URL-адреса в абсолютный.....	383
19.4. Разбор HTML-эквивалентов	383
19.5. Обмен данными по протоколу HTTP.....	385
19.6. Обмен данными с помощью модуля <i>urllib.request</i>	390
19.7. Определение кодировки.....	393

ЧАСТЬ II. БИБЛИОТЕКА PYQT 5.....395

Глава 20. Знакомство с PyQt 5.....397

20.1. Установка PyQt 5	397
20.2. Первая программа.....	400
20.3. Структура PyQt-программы.....	401
20.4. ООП-стиль создания окна.....	403
20.5. Создание окна с помощью программы Qt Designer.....	407
20.5.1. Создание формы	407
20.5.2. Загрузка ui-файла в программе.....	409
20.5.3. Преобразование ui-файла в ru-файл	411
20.6. Модули PyQt 5	413
20.7. Типы данных в PyQt.....	414
20.8. Управление основным циклом приложения.....	415
20.9. Многопоточные приложения.....	416
20.9.1. Класс <i>QThread</i> : создание потока.....	416
20.9.2. Управление циклом внутри потока.....	420
20.9.3. Модуль <i>queue</i> : создание очереди заданий.....	424
20.9.4. Классы <i>QMutex</i> и <i>QMutexLocker</i>	427
20.10. Вывод заставки	431
20.11. Доступ к документации.....	433

Глава 21. Управление окном приложения	435
21.1. Создание и отображение окна	435
21.2. Указание типа окна	436
21.3. Изменение и получение размеров окна	438
21.4. Местоположение окна на экране и управление им	441
21.5. Указание координат и размеров	444
21.5.1. Класс <i>QPoint</i> : координаты точки	444
21.5.2. Класс <i>QSize</i> : размеры прямоугольной области	445
21.5.3. Класс <i>QRect</i> : координаты и размеры прямоугольной области	447
21.6. Разворачивание и сворачивание окна	452
21.7. Управление прозрачностью окна	454
21.8. Модальные окна	455
21.9. Смена значка в заголовке окна	456
21.10. Изменение цвета фона окна	457
21.11. Вывод изображения в качестве фона	459
21.12. Создание окна произвольной формы	460
21.13. Всплывающие подсказки	462
21.14. Закрытие окна из программы	463
 Глава 22. Обработка сигналов и событий	 464
22.1. Назначение обработчиков сигналов	464
22.2. Блокировка и удаление обработчика	468
22.3. Генерация сигналов	470
22.4. Передача данных в обработчик	472
22.5. Использование таймеров	473
22.6. Перехват всех событий	476
22.7. События окна	479
22.7.1. Изменение состояния окна	479
22.7.2. Изменение положения и размеров окна	480
22.7.3. Перерисовка окна или его части	481
22.7.4. Предотвращение закрытия окна	482
22.8. События клавиатуры	483
22.8.1. Установка фокуса ввода	483
22.8.2. Назначение клавиш быстрого доступа	486
22.8.3. Нажатие и отпускание клавиши на клавиатуре	488
22.9. События мыши	489
22.9.1. Нажатие и отпускание кнопки мыши	489
22.9.2. Перемещение указателя мыши	491
22.9.3. Наведение и увод указателя	492
22.9.4. Прокрутка колесика мыши	492
22.9.5. Изменение внешнего вида указателя мыши	493
22.10. Технология drag & drop	495
22.10.1. Запуск перетаскивания	495
22.10.2. Класс <i>QMimeData</i>	497
22.10.3. Обработка сброса	498
22.11. Работа с буфером обмена	500
22.12. Фильтрация событий	501
22.13. Искусственные события	501

Глава 23. Размещение компонентов в окнах	503
23.1. Абсолютное позиционирование	503
23.2. Горизонтальное и вертикальное выравнивание	504
23.3. Выравнивание по сетке	507
23.4. Выравнивание компонентов формы	510
23.5. Классы <i>QStackedLayout</i> и <i>QStackedWidget</i>	512
23.6. Класс <i>QSizePolicy</i>	513
23.7. Объединение компонентов в группу.....	514
23.8. Панель с рамкой.....	516
23.9. Панель с вкладками	517
23.10. Компонент «аккордеон»	521
23.11. Панели с изменяемым размером	523
23.12. Область с полосами прокрутки	525
 Глава 24. Основные компоненты	526
24.1. Надпись.....	526
24.2. Командная кнопка	529
24.3. Переключатель.....	531
24.4. Флажок	531
24.5. Однострочное текстовое поле	532
24.5.1. Основные методы и сигналы	532
24.5.2. Ввод данных по маске.....	535
24.5.3. Контроль ввода	536
24.6. Многострочное текстовое поле	537
24.6.1. Основные методы и сигналы	538
24.6.2. Изменение параметров поля.....	540
24.6.3. Указание параметров текста и фона	541
24.6.4. Класс <i>QTextDocument</i>	542
24.6.5. Класс <i>QTextCursor</i>	545
24.7. Текстовый браузер.....	548
24.8. Поля для ввода целых и вещественных чисел.....	550
24.9. Поля для ввода даты и времени.....	551
24.10. Календарь	554
24.11. Электронный индикатор	556
24.12. Индикатор хода процесса.....	557
24.13. Шкала с ползунком.....	558
24.14. Круговая шкала с ползунком	560
24.15. Полоса прокрутки.....	561
24.16. Web-браузер	561
 Глава 25. Списки и таблицы.....	565
25.1. Раскрывающийся список.....	565
25.1.1. Добавление, изменение и удаление элементов	565
25.1.2. Изменение параметров списка	566
25.1.3. Поиск элементов.....	567
25.1.4. Сигналы	568
25.2. Список для выбора шрифта	568
25.3. Роли элементов	569

25.4. Модели.....	570
25.4.1. Доступ к данным внутри модели	570
25.4.2. Класс <i>QStringListModel</i>	571
25.4.3. Класс <i>QStandardItemModel</i>	573
25.4.4. Класс <i>QStandardItem</i>	576
25.5. Представления	579
25.5.1. Класс <i>QAbstractItemView</i>	580
25.5.2. Простой список.....	583
25.5.3. Таблица.....	585
25.5.4. Иерархический список	587
25.5.5. Управление заголовками строк и столбцов.....	589
25.6. Управление выделением элементов.....	591
25.7. Промежуточные модели.....	593
25.8. Использование делегатов.....	595
Глава 26. Работа с базами данных	599
26.1. Соединение с базой данных.....	599
26.2. Получение сведений о структуре таблицы	602
26.2.1. Получение сведений о таблице	602
26.2.2. Получение сведений об отдельном поле	603
26.2.3. Получение сведений об индексе	603
26.2.4. Получение сведений об ошибке	604
26.3. Выполнение SQL-запросов и получение их результатов	604
26.3.1. Выполнение запросов	605
26.3.2. Обработка результатов выполнения запросов	607
26.3.3. Очистка запроса.....	608
26.3.4. Получение служебных сведений о запросе	609
26.4. Модели, связанные с данными	609
26.4.1. Модель, связанная с SQL-запросом	609
26.4.2. Модель, связанная с таблицей.....	611
26.4.3. Модель, поддерживающая межтабличные связи.....	616
26.4.4. Использование связанных делегатов	619
Глава 27. Работа с графикой	621
27.1. Вспомогательные классы	621
27.1.1. Класс <i>QColor</i> : цвет	622
27.1.2. Класс <i>QPen</i> : перо.....	625
27.1.3. Класс <i>QBrush</i> : кисть	627
27.1.4. Класс <i>QLine</i> : линия.....	627
27.1.5. Класс <i>QPolygon</i> : многоугольник.....	628
27.1.6. Класс <i>QFont</i> : шрифт.....	630
27.2. Класс <i>QPainter</i>	632
27.2.1. Рисование линий и фигур	633
27.2.2. Вывод текста.....	636
27.2.3. Вывод изображения.....	637
27.2.4. Преобразование систем координат	638
27.2.5. Сохранение команд рисования в файл.....	639
27.3. Работа с изображениями	640
27.3.1. Класс <i>QPixmap</i>	641

27.3.2. Класс <i>QBitmap</i>	643
27.3.3. Класс <i>QImage</i>	644
27.3.4. Класс <i>QIcon</i>	647
Глава 28. Графическая сцена.....	649
28.1. Класс <i>QGraphicsScene</i> : сцена.....	649
28.1.1. Настройка сцены	650
28.1.2. Добавление и удаление графических объектов	650
28.1.3. Добавление компонентов на сцену	651
28.1.4. Поиск объектов.....	652
28.1.5. Управление фокусом ввода	653
28.1.6. Управление выделением объектов.....	654
28.1.7. Прочие методы и сигналы	654
28.2. Класс <i>QGraphicsView</i> : представление.....	656
28.2.1. Настройка представления	656
28.2.2. Преобразования между координатами представления и сцены	657
28.2.3. Поиск объектов.....	658
28.2.4. Преобразование системы координат	658
28.2.5. Прочие методы	659
28.3. Класс <i>QGraphicsItem</i> : базовый класс для графических объектов.....	660
28.3.1. Настройка объекта.....	660
28.3.2. Выполнение преобразований	662
28.3.3. Прочие методы	663
28.4. Графические объекты.....	664
28.4.1. Линия.....	664
28.4.2. Класс <i>QAbstractGraphicsShapeItem</i>	664
28.4.3. Прямоугольник	665
28.4.4. Многоугольник	665
28.4.5. Эллипс	665
28.4.6. Изображение	666
28.4.7. Простой текст	667
28.4.8. Форматированный текст	667
28.5. Группировка объектов.....	668
28.6. Эффекты	669
28.6.1. Класс <i>QGraphicsEffect</i>	669
28.6.2. Тень.....	669
28.6.3. Размытие	670
28.6.4. Изменение цвета	671
28.6.5. Изменение прозрачности	671
28.7. Обработка событий.....	672
28.7.1. События клавиатуры	672
28.7.2. События мыши	673
28.7.3. Обработка перетаскивания и сброса.....	675
28.7.4. Фильтрация событий.....	677
28.7.5. Обработка изменения состояния объекта.....	677
Глава 29. Диалоговые окна	679
29.1. Пользовательские диалоговые окна.....	679
29.2. Класс <i>QDialogButtonBox</i>	681

29.3. Класс <i>QMessageBox</i>	684
29.3.1. Основные методы и сигналы	685
29.3.2. Окно информационного сообщения	688
29.3.3. Окно подтверждения	688
29.3.4. Окно предупреждающего сообщения	689
29.3.5. Окно критического сообщения	689
29.3.6. Окно сведений о программе	690
29.3.7. Окно сведений о библиотеке Qt	690
29.4. Класс <i>QInputDialog</i>	691
29.4.1. Основные методы и сигналы	692
29.4.2. Окно для ввода строки	694
29.4.3. Окно для ввода целого числа	694
29.4.4. Окно для ввода вещественного числа	695
29.4.5. Окно для выбора пункта из списка	696
29.4.6. Окно для ввода большого текста	696
29.5. Класс <i>QFileDialog</i>	697
29.5.1. Основные методы и сигналы	698
29.5.2. Окно для выбора каталога	700
29.5.3. Окно для открытия файлов	701
29.5.4. Окно для сохранения файла	702
29.6. Окно для выбора цвета	704
29.7. Окно для выбора шрифта	705
29.8. Окно для вывода сообщения об ошибке	706
29.9. Окно с индикатором хода процесса	706
29.10. Создание многостраничного мастера	708
29.10.1. Класс <i>QWizard</i>	708
29.10.2. Класс <i>QWizardPage</i>	711
Глава 30. Создание SDI- и MDI-приложений	714
30.1. Создание главного окна приложения	714
30.2. Меню	718
30.2.1. Класс <i>QMenuBar</i>	719
30.2.2. Класс <i>QMenu</i>	720
30.2.3. Контекстное меню компонента	722
30.2.4. Класс <i>QAction</i>	723
30.2.5. Объединение переключателей в группу	726
30.3. Панели инструментов	727
30.3.1. Класс <i>QToolBar</i>	728
30.3.2. Класс <i>QToolButton</i>	729
30.4. Прикрепляемые панели	730
30.5. Управление строкой состояния	732
30.6. MDI-приложения	733
30.6.1. Класс <i>QMdiArea</i>	733
30.6.2. Класс <i>QMdiSubWindow</i>	736
30.7. Добавление значка приложения в область уведомлений	737
Глава 31. Мультимедиа	739
31.1. Класс <i>QMediaPlayer</i>	739
31.2. Класс <i>QVideoWidget</i>	748

31.3. Класс <i>QMediaPlaylist</i>	751
31.4. Запись звука	754
31.4.1. Класс <i>QAudioRecorder</i>	755
31.4.2. Класс <i>QAudioEncoderSettings</i>	757
31.5. Класс <i>QSoundEffect</i>	761
Глава 32. Печать документов	764
32.1. Основные средства печати	764
32.1.1. Класс <i>QPrinter</i>	764
32.1.2. Вывод на печать	768
32.1.3. Служебные классы	773
32.1.3.1. Класс <i>QPageSize</i>	774
32.1.3.2. Класс <i>QPageLayout</i>	776
32.2. Задание параметров принтера и страницы	777
32.2.1. Класс <i>QPrintDialog</i>	778
32.2.2. Класс <i>QPageSetupDialog</i>	779
32.3. Предварительный просмотр документов перед печатью	781
32.3.1. Класс <i>QPrintPreviewDialog</i>	781
32.3.2. Класс <i>QPrintPreviewWidget</i>	784
32.4. Получение сведений о принтере. Класс <i>QPrinterInfo</i>	786
32.5. Экспорт в формат PDF. Класс <i>QPdfWriter</i>	788
Заключение	791
Приложение. Описание электронного архива	792
Предметный указатель	793



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**.

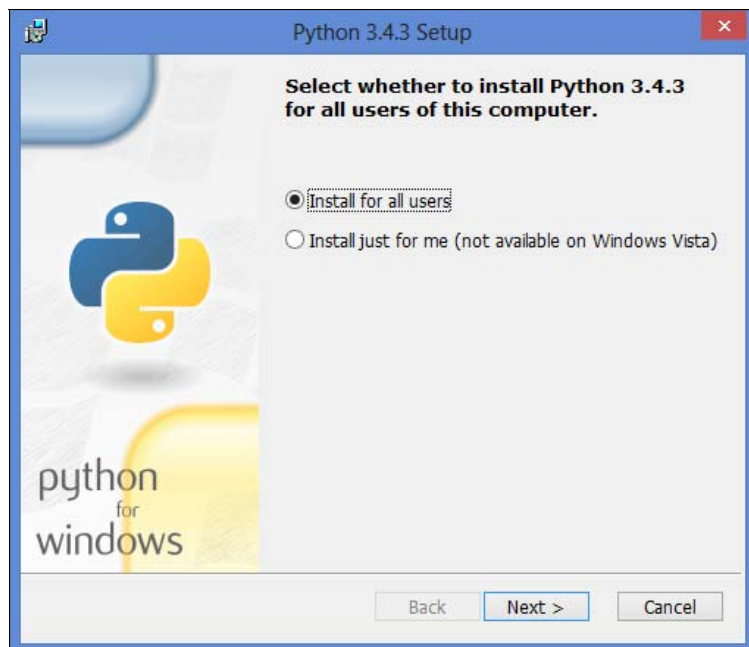


Рис. 1.1. Установка Python. Шаг 1

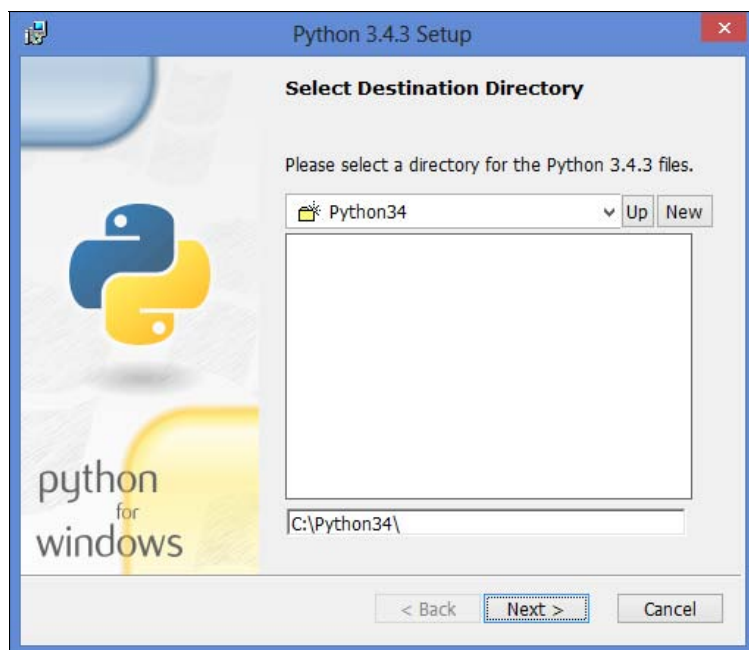


Рис. 1.2. Установка Python. Шаг 2



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

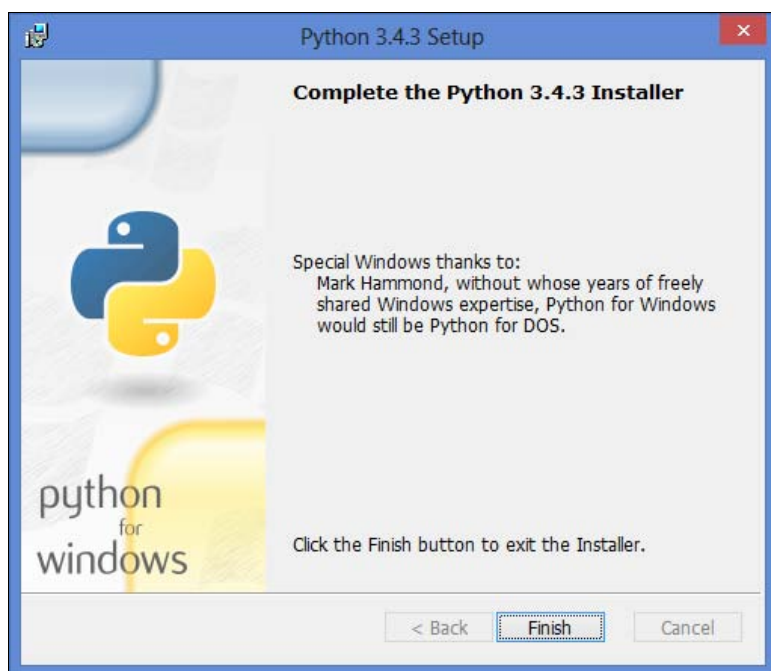


Рис. 1.4. Установка Python. Шаг 4

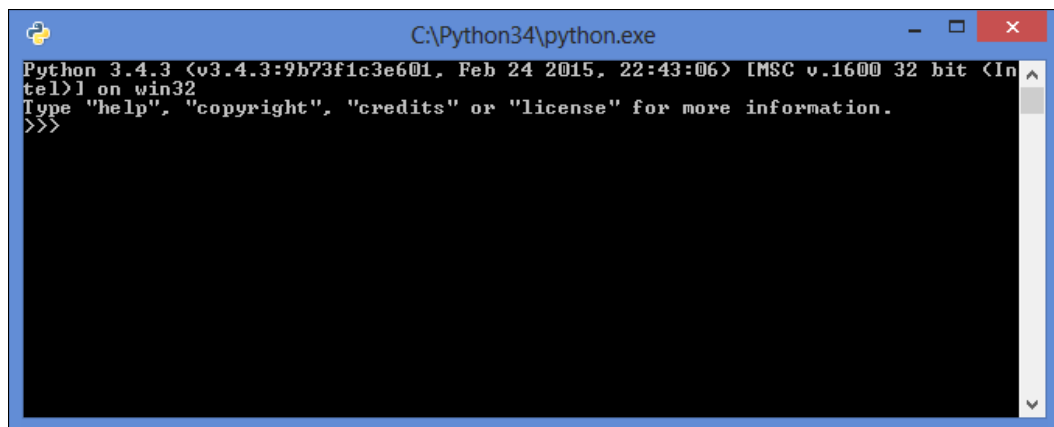
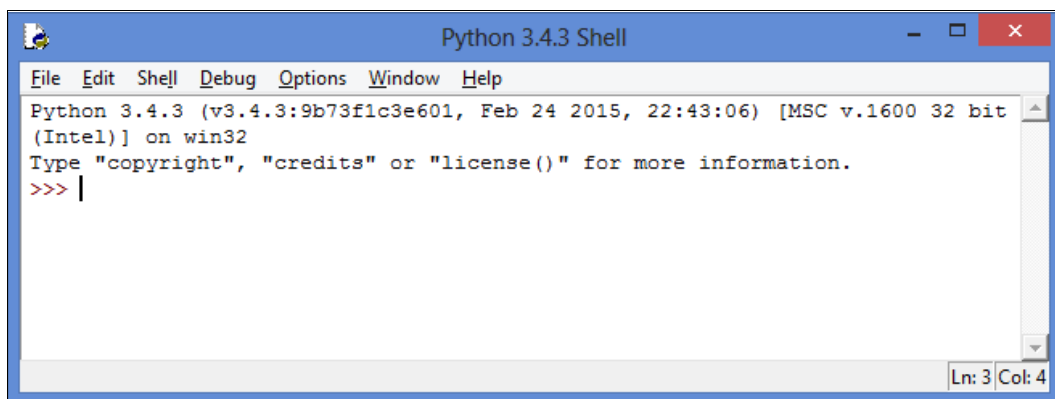


Рис. 1.5. Интерактивная оболочка

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.

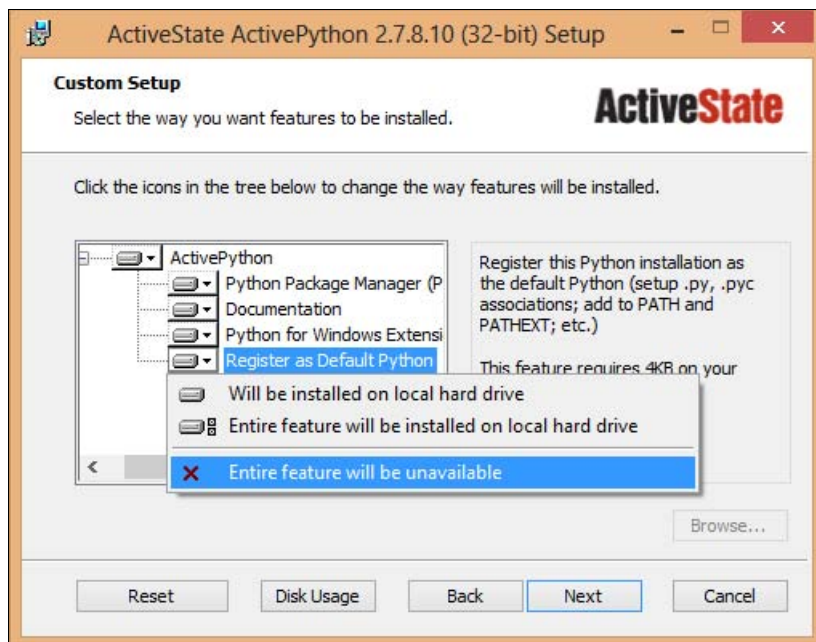


Рис. 1.7. Окно Custom Setup

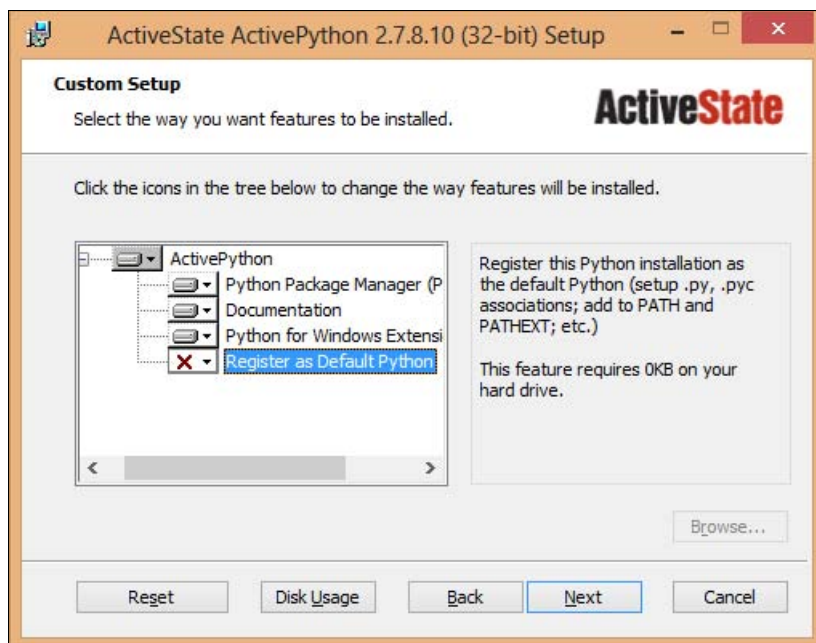


Рис. 1.8. Компонент Register as Default Python отключен

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.

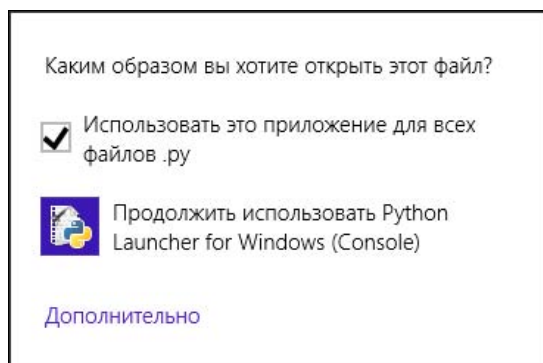


Рис. 1.9. Диалоговое окно выбора альтернативной программы для запуска файла

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.

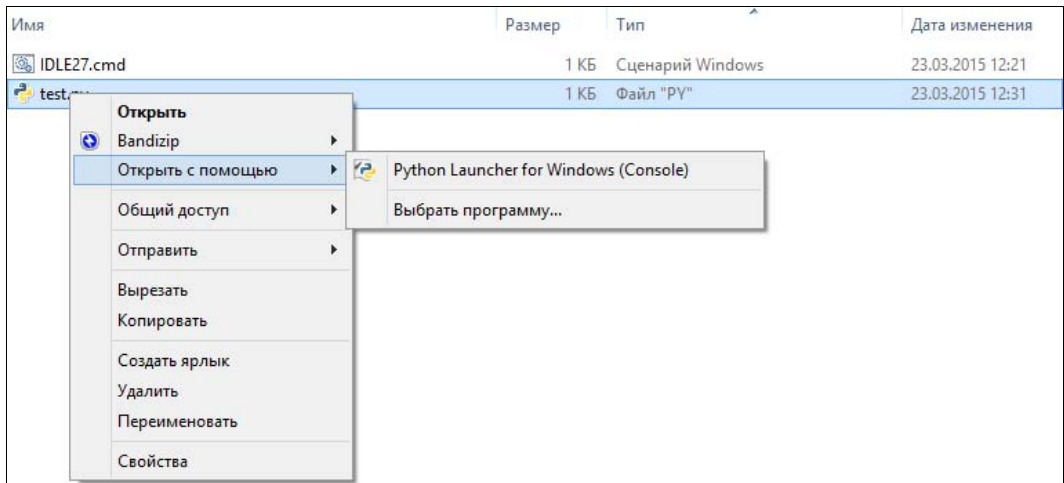


Рис. 1.10. Варианты запуска программы разными версиями Python

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 программы `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.