

Николай Прохоренко
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	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Екатерина Капалыгина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Дизайн обложки	<i>Марины Дамбиевой</i>

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

ISBN 978-5-9775-3978-4

© ООО "БХВ", 2018
© Оформление. ООО "БХВ-Петербург", 2018

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. Дополнительные возможности IDLE.....	31
1.6. Вывод результатов работы программы	33
1.7. Ввод данных.....	35
1.8. Доступ к документации	36
Глава 2. Переменные	40
2.1. Именованние переменных	40
2.2. Типы данных	41
2.3. Присваивание значения переменным	44
2.4. Проверка типа данных.....	47
2.5. Преобразование типов данных	48
2.6. Удаление переменных	50
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 <i>if...else</i>	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> : генерирование случайных чисел.....	79
Глава 6. Строки и двоичные данные	82
6.1. Создание строки.....	83
6.2. Специальные символы	86
6.3. Операции над строками.....	87
6.4. Форматирование строк.....	90
6.5. Метод <i>format()</i>	95
6.5.1. Форматируемые строки	99
6.6. Функции и методы для работы со строками	100
6.7. Настройка локали	103
6.8. Изменение регистра символов.....	104
6.9. Функции для работы с символами	105
6.10. Поиск и замена в строке.....	105
6.11. Проверка типа содержимого строки	109
6.12. Тип данных <i>bytes</i>	111
6.13. Тип данных <i>bytearray</i>	116
6.14. Преобразование объекта в последовательность байтов	119
6.15. Шифрование строк	120
Глава 7. Регулярные выражения	122
7.1. Синтаксис регулярных выражений	122
7.2. Поиск первого совпадения с шаблоном.....	131
7.3. Поиск всех совпадений с шаблоном	136
7.4. Замена в строке	137
7.5. Прочие функции и методы.....	139
Глава 8. Списки, кортежи, множества и диапазоны	141
8.1. Создание списка.....	142
8.2. Операции над списками	145
8.3. Многомерные списки	148
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	156
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>	168
8.17.1. Генерирование неопределенного количества значений	168
8.17.2. Генерирование комбинаций значений	169
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	177
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	188
10.4.1. Класс <i>timedelta</i>	189
10.4.2. Класс <i>date</i>	191
10.4.3. Класс <i>time</i>	194
10.4.4. Класс <i>datetime</i>	196
10.5. Модуль <i>calendar</i> : вывод календаря	201
10.5.1. Методы классов <i>TextCalendar</i> и <i>LocaleTextCalendar</i>	202
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	210
11.1. Определение функции и ее вызов	210
11.2. Расположение определений функций	213
11.3. Необязательные параметры и сопоставление по ключам	214
11.4. Переменное число параметров в функции	216
11.5. Анонимные функции	218
11.6. Функции-генераторы	219
11.7. Декораторы функций	221
11.8. Рекурсия. Вычисление факториала	223
11.9. Глобальные и локальные переменные	224
11.10. Вложенные функции	228
11.11. Аннотации функций	229
Глава 12. Модули и пакеты	231
12.1. Инструкция <i>import</i>	231
12.2. Инструкция <i>from</i>	234
12.3. Пути поиска модулей	237
12.4. Повторная загрузка модулей	238
12.5. Пакеты	239

Глава 13. Объектно-ориентированное программирование	244
13.1. Определение класса и создание экземпляра класса.....	244
13.2. Методы <code>__init__()</code> и <code>__del__()</code>	247
13.3. Наследование	248
13.4. Множественное наследование.....	250
13.4.1. Примеси и их использование.....	252
13.5. Специальные методы.....	253
13.6. Перегрузка операторов.....	255
13.7. Статические методы и методы класса	258
13.8. Абстрактные методы	259
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.....	293
16.3. Доступ к файлам с помощью модуля <code>os</code>	299
16.4. Классы <code>StringIO</code> и <code>BytesIO</code>	301
16.5. Права доступа к файлам и каталогам.....	305
16.6. Функции для манипулирования файлами	307
16.7. Преобразование пути к файлу или каталогу.....	310
16.8. Перенаправление ввода/вывода.....	312
16.9. Сохранение объектов в файл	315
16.10. Функции для работы с каталогами.....	319
16.10.1. Функция <code>scandir()</code>	322
16.11. Исключения, возбуждаемые файловыми операциями	324
ЧАСТЬ II. БИБЛИОТЕКА PYQT 5.....	325
Глава 17. Знакомство с PyQt 5.....	327
17.1. Установка PyQt 5	327
17.2. Первая программа.....	328
17.3. Структура PyQt-программы.....	329
17.4. ООП-стиль создания окна.....	331
17.5. Создание окна с помощью программы Qt Designer.....	335
17.5.1. Создание формы	335

17.5.2. Использование UI-файла в программе	337
17.5.3. Преобразование UI-файла в PY-файл	339
17.6. Модули PyQt 5	340
17.7. Типы данных в PyQt	341
17.8. Управление основным циклом приложения.....	342
17.9. Многопоточные приложения.....	344
17.9.1. Класс <i>QThread</i> : создание потока.....	344
17.9.2. Управление циклом внутри потока.....	347
17.9.3. Модуль <i>queue</i> : создание очереди заданий.....	351
17.9.4. Классы <i>QMutex</i> и <i>QMutexLocker</i>	354
17.10. Вывод заставки	359
17.11. Доступ к документации.....	361
Глава 18. Управление окном приложения	362
18.1. Создание и отображение окна	362
18.2. Указание типа окна.....	363
18.3. Изменение и получение размеров окна	365
18.4. Местоположение окна на экране и управление им.....	367
18.5. Указание координат и размеров.....	370
18.5.1. Класс <i>QPoint</i> : координаты точки	370
18.5.2. Класс <i>QSize</i> : размеры прямоугольной области.....	372
18.5.3. Класс <i>QRect</i> : координаты и размеры прямоугольной области.....	374
18.6. Разворачивание и сворачивание окна	379
18.7. Управление прозрачностью окна	381
18.8. Модальные окна.....	381
18.9. Смена значка в заголовке окна.....	383
18.10. Изменение цвета фона окна.....	384
18.11. Вывод изображения в качестве фона.....	385
18.12. Создание окна произвольной формы.....	387
18.13. Всплывающие подсказки	388
18.14. Программное закрытие окна.....	390
18.15. Использование таблиц стилей CSS для оформления окон.....	390
Глава 19. Обработка сигналов и событий.....	395
19.1. Назначение обработчиков сигналов.....	395
19.2. Блокировка и удаление обработчика	399
19.3. Генерация сигналов	401
19.4. Передача данных в обработчик	403
19.5. Использование таймеров.....	404
19.6. Перехват всех событий.....	407
19.7. События окна	410
19.7.1. Изменение состояния окна	410
19.7.2. Изменение положения и размеров окна	411
19.7.3. Перерисовка окна или его части	412
19.7.4. Предотвращение закрытия окна.....	413
19.8. События клавиатуры	414
19.8.1. Установка фокуса ввода.....	414
19.8.2. Назначение клавиш быстрого доступа	417
19.8.3. Нажатие и отпускание клавиши на клавиатуре.....	419

19.9. События мыши.....	420
19.9.1. Нажатие и отпускание кнопки мыши	420
19.9.2. Перемещение указателя мыши.....	422
19.9.3. Наведение и увод указателя.....	423
19.9.4. Прокрутка колесика мыши	423
19.9.5. Изменение внешнего вида указателя мыши.....	424
19.10. Технология drag & drop.....	425
19.10.1. Запуск перетаскивания.....	425
19.10.2. Класс <i>QMimeData</i>	427
19.10.3. Обработка сброса	429
19.11. Работа с буфером обмена.....	430
19.12. Фильтрация событий	431
19.13. Искусственные события.....	432
Глава 20. Размещение компонентов в окнах	434
20.1. Абсолютное позиционирование	434
20.2. Горизонтальное и вертикальное выравнивание	435
20.3. Выравнивание по сетке	438
20.4. Выравнивание компонентов формы	441
20.5. Классы <i>QStackedLayout</i> и <i>QStackedWidget</i>	443
20.6. Класс <i>QSizePolicy</i>	444
20.7. Объединение компонентов в группу.....	445
20.8. Панель с рамкой.....	447
20.9. Панель с вкладками	448
20.10. Компонент «аккордеон».....	452
20.11. Панели с изменяемым размером	454
20.12. Область с полосами прокрутки	456
Глава 21. Основные компоненты	458
21.1. Надпись.....	458
21.2. Командная кнопка	461
21.3. Переключатель.....	463
21.4. Флажок	463
21.5. Однострочное текстовое поле	464
21.5.1. Основные методы и сигналы	464
21.5.2. Ввод данных по маске.....	467
21.5.3. Контроль ввода	468
21.6. Многострочное текстовое поле	469
21.6.1. Основные методы и сигналы.....	469
21.6.2. Изменение параметров поля.....	471
21.6.3. Указание параметров текста и фона	473
21.6.4. Класс <i>QTextDocument</i>	474
21.6.5. Класс <i>QTextCursor</i>	477
21.7. Текстовый браузер.....	480
21.8. Поля для ввода целых и вещественных чисел.....	481
21.9. Поля для ввода даты и времени.....	483
21.10. Календарь	486
21.11. Электронный индикатор	488
21.12. Индикатор хода процесса.....	489

21.13. Шкала с ползунком.....	490
21.14. Круговая шкала с ползунком	492
21.15. Полоса прокрутки	492
21.16. Веб-браузер	493
Глава 22. Списки и таблицы.....	498
22.1. Раскрывающийся список.....	498
22.1.1. Добавление, изменение и удаление элементов	498
22.1.2. Изменение параметров списка	499
22.1.3. Поиск элементов.....	500
22.1.4. Сигналы	501
22.2. Список для выбора шрифта	501
22.3. Роли элементов	502
22.4. Модели.....	503
22.4.1. Доступ к данным внутри модели	503
22.4.2. Класс <i>QStringListModel</i>	504
22.4.3. Класс <i>QStandardItemModel</i>	506
22.4.4. Класс <i>QStandardItem</i>	509
22.5. Представления	513
22.5.1. Класс <i>QAbstractItemView</i>	513
22.5.2. Простой список.....	516
22.5.3. Таблица.....	518
22.5.4. Иерархический список	520
22.5.5. Управление заголовками строк и столбцов.....	522
22.6. Управление выделением элементов.....	525
22.7. Промежуточные модели.....	526
22.8. Использование делегатов.....	528
Глава 23. Работа с базами данных	532
23.1. Соединение с базой данных.....	532
23.2. Получение сведений о структуре таблицы	535
23.2.1. Получение сведений о таблице	535
23.2.2. Получение сведений об отдельном поле	536
23.2.3. Получение сведений об индексе	537
23.2.4. Получение сведений об ошибке	537
23.3. Выполнение SQL-запросов и получение их результатов	538
23.3.1. Выполнение запросов	538
23.3.2. Обработка результатов выполнения запросов	541
23.3.3. Очистка запроса.....	543
23.3.4. Получение служебных сведений о запросе	543
23.4. Модели, связанные с данными	544
23.4.1. Модель, связанная с SQL-запросом.....	544
23.4.2. Модель, связанная с таблицей.....	545
23.4.3. Модель, поддерживающая межтабличные связи.....	551
23.4.4. Использование связанных делегатов	553
Глава 24. Работа с графикой	555
24.1. Вспомогательные классы.....	555
24.1.1. Класс <i>QColor</i> : цвет	556
24.1.2. Класс <i>QPen</i> : перо.....	559

24.1.3. Класс <i>QBrush</i> : кисть	561
24.1.4. Класс <i>QLine</i> : линия	562
24.1.5. Класс <i>QPolygon</i> : многоугольник	562
24.1.6. Класс <i>QFont</i> : шрифт	564
24.2. Класс <i>QPainter</i>	566
24.2.1. Рисование линий и фигур	567
24.2.2. Вывод текста	570
24.2.3. Вывод изображения	571
24.2.4. Преобразование систем координат	572
24.2.5. Сохранение команд рисования в файл	573
24.3. Работа с изображениями	574
24.3.1. Класс <i>QPixmap</i>	575
24.3.2. Класс <i>QBitmap</i>	577
24.3.3. Класс <i>QImage</i>	578
24.3.4. Класс <i>QIcon</i>	581
Глава 25. Графическая сцена.....	583
25.1. Класс <i>QGraphicsScene</i> : сцена	583
25.1.1. Настройка сцены	584
25.1.2. Добавление и удаление графических объектов	584
25.1.3. Добавление компонентов на сцену	585
25.1.4. Поиск объектов	586
25.1.5. Управление фокусом ввода	587
25.1.6. Управление выделением объектов	588
25.1.7. Прочие методы и сигналы	588
25.2. Класс <i>QGraphicsView</i> : представление	590
25.2.1. Настройка представления	590
25.2.2. Преобразования между координатами представления и сцены	592
25.2.3. Поиск объектов	592
25.2.4. Преобразование системы координат	593
25.2.5. Прочие методы	593
25.3. Класс <i>QGraphicsItem</i> : базовый класс для графических объектов	594
25.3.1. Настройка объекта	595
25.3.2. Выполнение преобразований	597
25.3.3. Прочие методы	597
25.4. Графические объекты	598
25.4.1. Линия	598
25.4.2. Класс <i>QAbstractGraphicsShapeltem</i>	599
25.4.3. Прямоугольник	599
25.4.4. Многоугольник	599
25.4.5. Эллипс	600
25.4.6. Изображение	600
25.4.7. Простой текст	601
25.4.8. Форматированный текст	602
25.5. Группировка объектов	603
25.6. Эффекты	603
25.6.1. Класс <i>QGraphicsEffect</i>	604
25.6.2. Тень	604

25.6.3. Размытие	605
25.6.4. Изменение цвета	605
25.6.5. Изменение прозрачности	606
25.7. Обработка событий.....	606
25.7.1. События клавиатуры	607
25.7.2. События мыши	607
25.7.3. Обработка перетаскивания и сброса	610
25.7.4. Фильтрация событий.....	611
25.7.5. Обработка изменения состояния объекта.....	612
Глава 26. Диалоговые окна	614
26.1. Пользовательские диалоговые окна	614
26.2. Класс <i>QDialogButtonBox</i>	616
26.3. Класс <i>QMessageBox</i>	619
26.3.1. Основные методы и сигналы	620
26.3.2. Окно информационного сообщения	622
26.3.3. Окно подтверждения	623
26.3.4. Окно предупреждающего сообщения	624
26.3.5. Окно критического сообщения	624
26.3.6. Окно сведений о программе	625
26.3.7. Окно сведений о библиотеке Qt	625
26.4. Класс <i>QInputDialog</i>	626
26.4.1. Основные методы и сигналы	626
26.4.2. Окно для ввода строки	628
26.4.3. Окно для ввода целого числа.....	629
26.4.4. Окно для ввода вещественного числа.....	630
26.4.5. Окно для выбора пункта из списка	630
26.4.6. Окно для ввода большого текста.....	631
26.5. Класс <i>QFileDialog</i>	632
26.5.1. Основные методы и сигналы	632
26.5.2. Окно для выбора каталога	634
26.5.3. Окно для открытия файлов	635
26.5.4. Окно для сохранения файла.....	637
26.6. Окно для выбора цвета	638
26.7. Окно для выбора шрифта.....	639
26.8. Окно для вывода сообщения об ошибке.....	640
26.9. Окно с индикатором хода процесса	641
26.10. Создание многостраничного мастера	642
26.10.1. Класс <i>QWizard</i>	642
26.10.2. Класс <i>QWizardPage</i>	646
Глава 27. Создание SDI- и MDI-приложений.....	648
27.1. Главное окно приложения.....	648
27.2. Меню.....	653
27.2.1. Класс <i>QMenuBar</i>	653
27.2.2. Класс <i>QMenu</i>	654
27.2.3. Контекстное меню компонента	657
27.2.4. Класс <i>QAction</i>	658
27.2.5. Объединение переключателей в группу	661

27.3. Панели инструментов.....	662
27.3.1. Класс <i>QToolBar</i>	662
27.3.2. Класс <i>QToolButton</i>	664
27.4. Прикрепляемые панели.....	665
27.5. Управление строкой состояния.....	667
27.6. MDI-приложения.....	668
27.6.1. Класс <i>QMdiArea</i>	668
27.6.2. Класс <i>QMdiSubWindow</i>	671
27.7. Добавление значка приложения в область уведомлений.....	672
Глава 28. Мультимедиа.....	674
28.1. Класс <i>QMediaPlayer</i>	674
28.2. Класс <i>QVideoWidget</i>	683
28.3. Класс <i>QMediaPlaylist</i>	686
28.4. Запись звука.....	689
28.4.1. Класс <i>QAudioRecorder</i>	689
28.4.2. Класс <i>QAudioEncoderSettings</i>	692
28.5. Класс <i>QSoundEffect</i>	695
Глава 29. Печать документов.....	699
29.1. Основные средства печати.....	699
29.1.1. Класс <i>QPrinter</i>	699
29.1.2. Вывод на печать.....	703
29.1.3. Служебные классы.....	708
29.1.3.1. Класс <i>QPageSize</i>	708
29.1.3.2. Класс <i>QPageLayout</i>	710
29.2. Задание параметров принтера и страницы.....	712
29.2.1. Класс <i>QPrintDialog</i>	712
29.2.2. Класс <i>QPageSetupDialog</i>	714
29.3. Предварительный просмотр документов перед печатью.....	716
29.3.1. Класс <i>QPrintPreviewDialog</i>	716
29.3.2. Класс <i>QPrintPreviewWidget</i>	719
29.4. Класс <i>QPrinterInfo</i> : получение сведений о принтере.....	721
29.5. Класс <i>QPdfWriter</i> : экспорт в формат PDF.....	723
Глава 30. Взаимодействие с Windows.....	725
30.1. Управление кнопкой в панели задач.....	725
30.1.1. Класс <i>QWinTaskbarButton</i>	725
30.1.2. Класс <i>QWinTaskbarProgress</i>	727
30.2. Списки быстрого доступа.....	730
30.2.1. Класс <i>QWinJumpList</i>	730
30.2.2. Класс <i>QWinJumpListCategory</i>	731
30.2.3. Класс <i>QWinJumpListItem</i>	732
30.3. Панели инструментов, выводющиеся на миниатюрах.....	735
30.3.1. Класс <i>QWinThumbnailToolBar</i>	735
30.3.2. Класс <i>QWinThumbnailToolButton</i>	736
30.4. Дополнительные инструменты по управлению окнами.....	739
30.5. Получение сведений об операционной системе.....	741
30.6. Получение путей к системным каталогам.....	742

Глава 31. Сохранение настроек приложений	743
31.1. Создание экземпляра класса <i>QSettings</i>	743
31.2. Запись и чтение данных	745
31.2.1. Базовые средства записи и чтения данных.....	745
31.2.2. Группировка сохраняемых значений. Ключи	746
31.2.3. Запись списков.....	748
31.3. Вспомогательные методы класса <i>QSettings</i>	750
31.4. Где хранятся настройки?.....	750
Глава 32. Приложение «Судоку».....	752
32.1. Правила судоку	752
32.2. Описание приложения «Судоку».....	753
32.3. Программирование приложения	755
32.3.1. Подготовительные действия.....	755
32.3.2. Класс <i>MyLabel</i> : ячейка поля судоку	755
32.3.3. Класс <i>Widget</i> : поле судоку	759
32.3.3.1. Конструктор класса <i>Widget</i>	760
32.3.3.2. Прочие методы класса <i>Widget</i>	762
32.3.4. Класс <i>MainWindow</i> : основное окно приложения	766
32.3.4.1. Конструктор класса <i>MainWindow</i>	767
32.3.4.2. Остальные методы класса <i>MainWindow</i>	770
32.3.5. Запускающий модуль	770
32.3.6. Копирование и вставка головоломок.....	771
32.3.6.1. Форматы данных	771
32.3.6.2. Реализация копирования и вставки в классе <i>Widget</i>	772
32.3.6.3. Реализация копирования и вставки в классе <i>MainWindow</i>	774
32.3.7. Сохранение и загрузка данных.....	778
32.3.8. Печать и предварительный просмотр	781
32.3.8.1. Реализация печати в классе <i>Widget</i>	781
32.3.8.2. Класс <i>PreviewDialog</i> : диалоговое окно предварительного просмотра	782
32.3.8.3. Реализация печати в классе <i>MainWindow</i>	785
Заключение.....	787
Приложение. Описание электронного архива.....	789
Предметный указатель	791

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 кросс-платформенный, 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 двойного 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 будут смотреться 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
- Глава 2.** Переменные
- Глава 3.** Операторы
- Глава 4.** Условные операторы и циклы
- Глава 5.** Числа
- Глава 6.** Строки и двоичные данные
- Глава 7.** Регулярные выражения
- Глава 8.** Списки, кортежи, множества и диапазоны
- Глава 9.** Словари
- Глава 10.** Работа с датой и временем
- Глава 11.** Пользовательские функции
- Глава 12.** Модули и пакеты
- Глава 13.** Объектно-ориентированное программирование
- Глава 14.** Обработка исключений
- Глава 15.** Итераторы, контейнеры и перечисления
- Глава 16.** Работа с файлами и каталогами



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**.

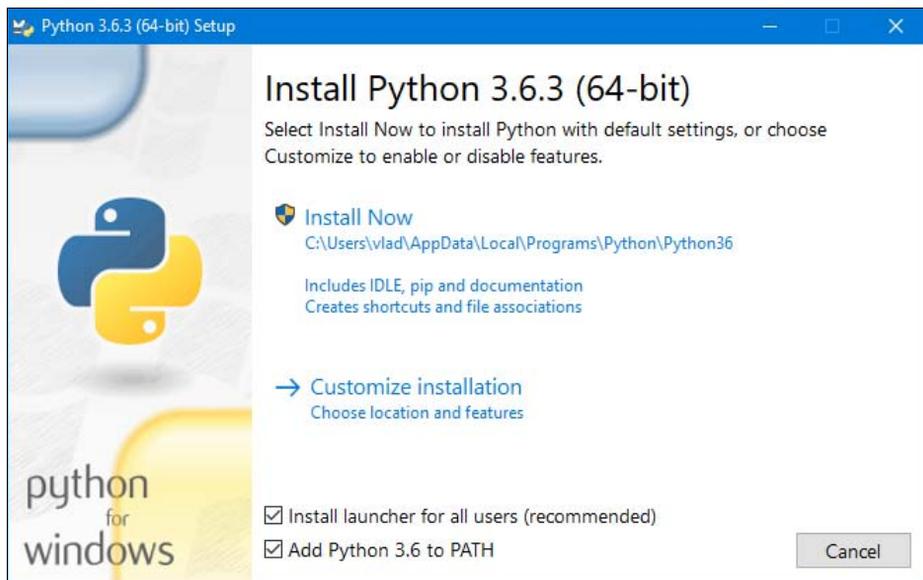


Рис. 1.1. Установка Python. Шаг 1

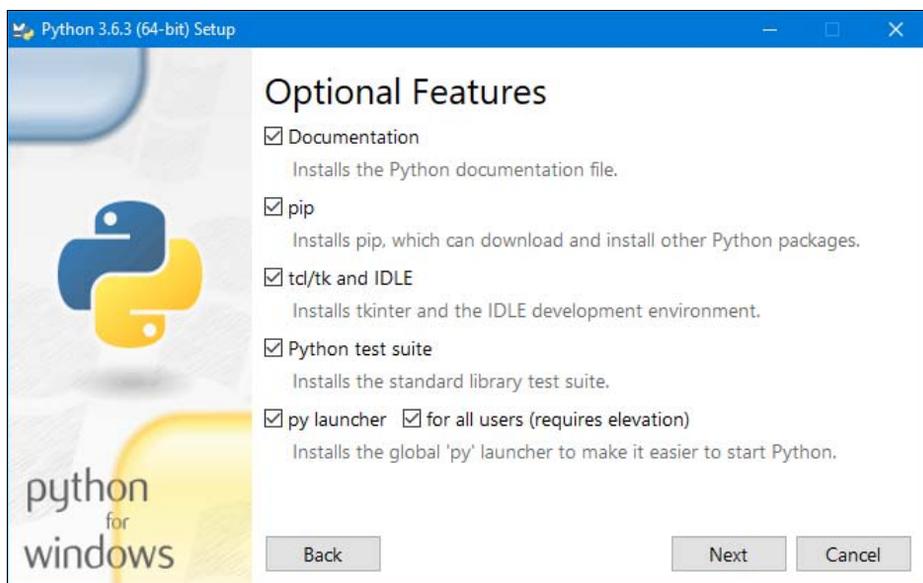


Рис. 1.2. Установка Python. Шаг 2

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).

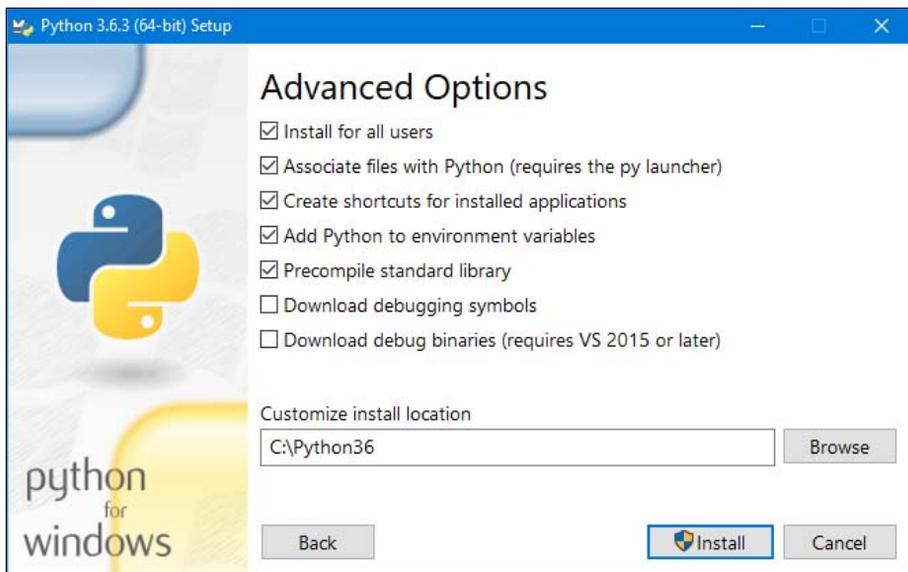


Рис. 1.3. Установка Python. Шаг 3

ВНИМАНИЕ!

Некоторые параметры при установке Python приходится задавать по несколько 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.

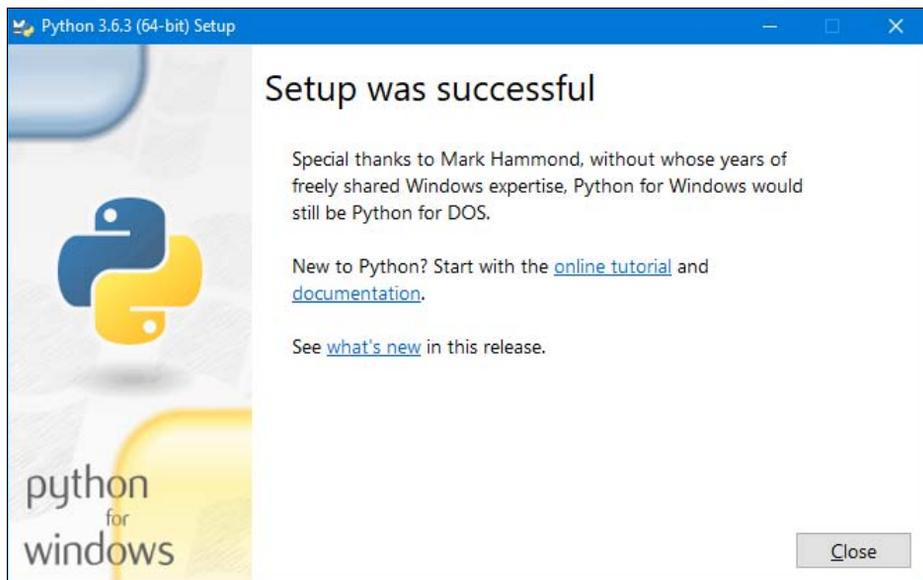


Рис. 1.4. Установка Python. Шаг 4

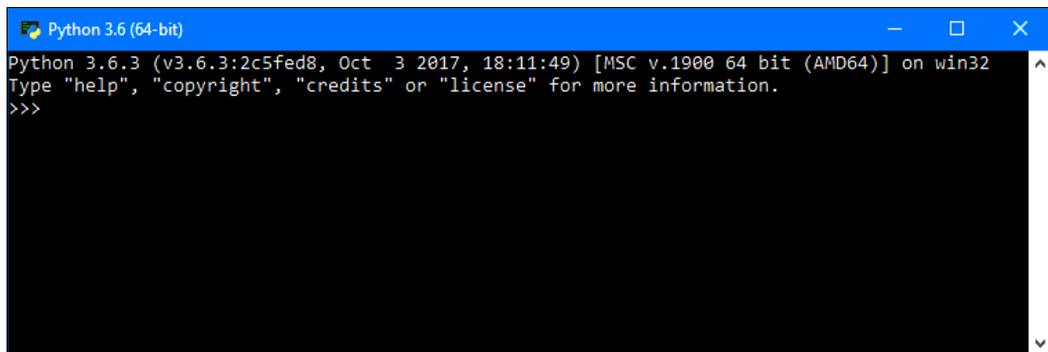


Рис. 1.5. Интерактивная оболочка

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.

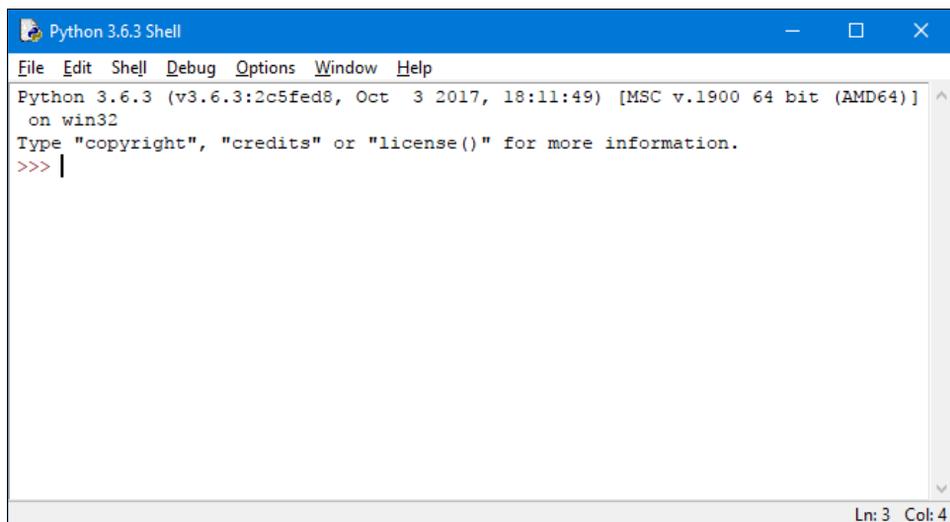


Рис. 1.6. Окно Python Shell редактора IDLE

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)**.

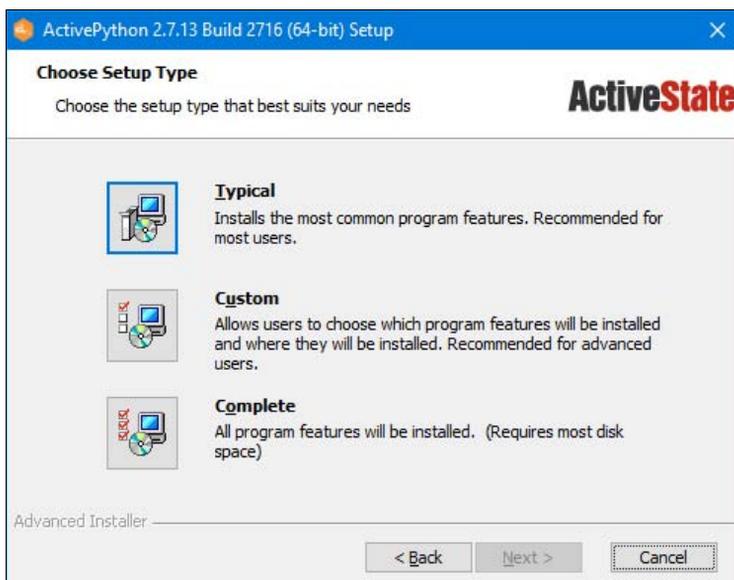


Рис. 1.7. Окно Choose Setup Type

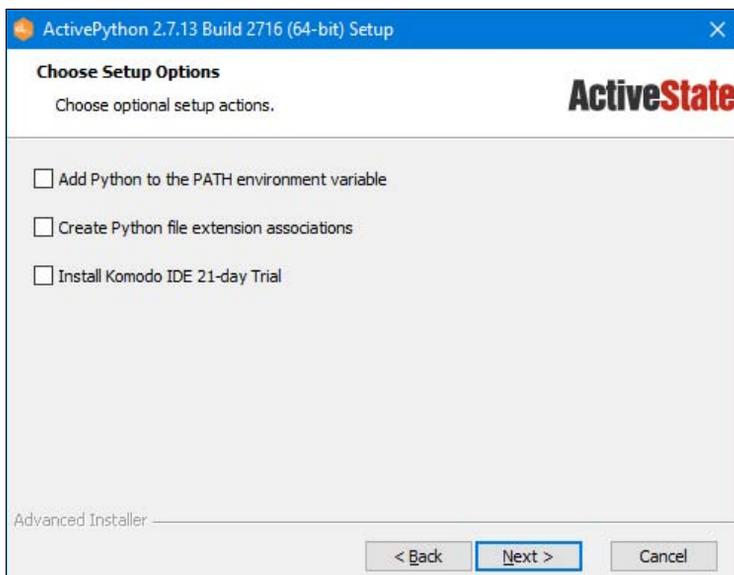


Рис. 1.8. Окно Choose Setup Options

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`.

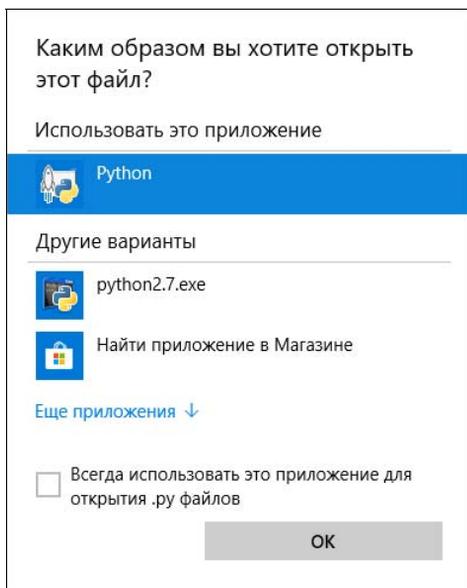


Рис. 1.9. Диалоговое окно выбора альтернативной программы для запуска файла

В Windows Vista, 7, 8 и 8.1 выбранная нами программа появится в подменю, открываемом 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
```

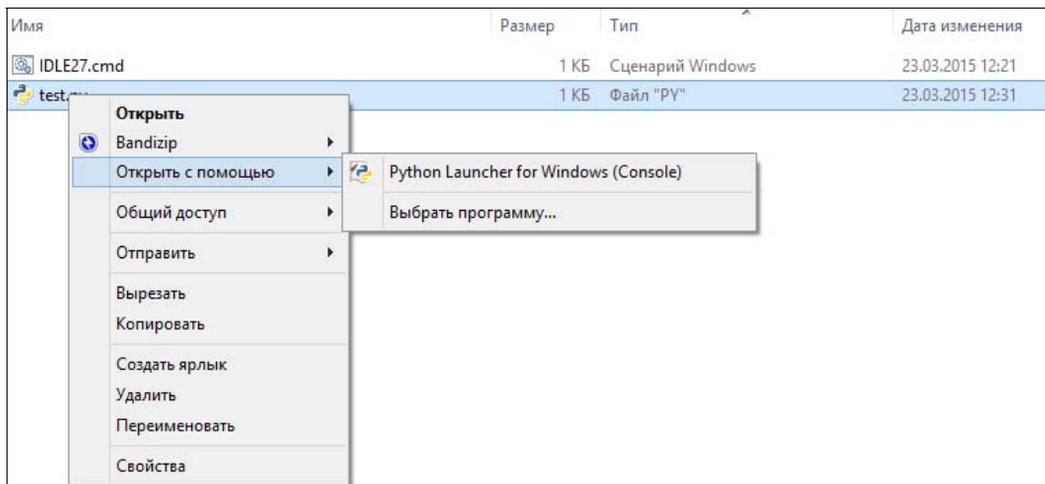


Рис. 1.10. Варианты запуска программы разными версиями Python

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)
```