

Содержание

Предисловие	17
Часть VI. Классы и объектно-ориентированное программирование	19
ГЛАВА 26. Объектно-ориентированное программирование: общая картина	20
Для чего используются классы?	21
Объектно-ориентированное программирование с высоты птичьего полета	22
Поиск в иерархии наследования	23
Классы и экземпляры	25
Вызовы методов	25
Создание деревьев классов	26
Перегрузка операций	28
Объектно-ориентированное программирование – это многократное использование кода	29
Резюме	32
Проверьте свои знания: контрольные вопросы	32
Проверьте свои знания: ответы	33
ГЛАВА 27. Основы написания классов	34
Классы генерируют множество объектов экземпляров	34
Объекты классов обеспечивают стандартное поведение	35
Объекты экземпляров являются конкретными элементами	35
Первый пример	36
Классы настраиваются через наследование	38
Второй пример	39
Классы являются атрибутами в модулях	41
Классы могут перехватывать операции Python	42
Третий пример	43
Для чего используется перегрузка операций?	45
Простейший в мире класс Python	46
Снова о записях: классы или словари	49
Резюме	51
Проверьте свои знания: контрольные вопросы	51
Проверьте свои знания: ответы	52
ГЛАВА 28. Более реалистичный пример	54
Шаг 1: создание экземпляров	55
Написание кода конструкторов	55
Тестирование в ходе дела	56
Использование кода двумя способами	58
Шаг 2: добавление методов, реализующих поведение	59
Написание кода методов	61
Шаг 3: перегрузка операций	63
Реализация отображения	63

Шаг 4: настройка поведения за счет создания подклассов	65
Написание кода подклассов	66
Расширение методов: плохой способ	66
Расширение методов: хороший способ	67
Полиморфизм в действии	69
Наследование, настройка и расширение	70
Объектно-ориентированное программирование: основная идея	71
Шаг 5: настройка конструкторов	72
Объектно-ориентированное программирование проще, чем может казаться	73
Другие способы комбинирования классов	74
Шаг 6: использование инструментов интроспекции	77
Специальные атрибуты класса	78
Обобщенный инструмент отображения	79
Атрибуты экземпляра или атрибуты класса	81
Размышления относительно имен в классах инструментов	82
Финальная форма классов	83
Шаг 7 (последний): сохранение объектов в базе данных	84
Модули pickle, dbm и shelve	85
Сохранение объектов в базе данных shelve	86
Исследование хранилища shelve в интерактивной подсказке	87
Обновление объектов в хранилище shelve	89
Указания на будущее	91
Резюме	93
Проверьте свои знания: контрольные вопросы	93
Проверьте свои знания: ответы	94
ГЛАВА 29. Детали реализации классов	96
Оператор class	96
Общая форма	97
Пример	97
Методы	99
Пример метода	100
Вызов конструкторов суперклассов	101
Другие возможности вызова методов	101
Наследование	102
Построение дерева атрибутов	102
Специализации унаследованных методов	104
Методики связывания классов	104
Абстрактные суперклассы	106
Пространства имен: заключение	108
Простые имена: глобальные, если не выполнено их присваивание	109
Имена атрибутов: пространства имен объектов	109
“Дзен” пространств имен: присваивания классифицируют имена	110
Вложенные классы: снова о правиле областей видимости LEGB	112
Словари пространств имен: обзор	114
Связи между пространствами имен: инструмент подъема по дереву	117
Снова о строках документации	119
Классы или модули	120

Резюме	121
Проверьте свои знания: контрольные вопросы	121
Проверьте свои знания: ответы	122
ГЛАВА 30. Перегрузка операций	123
Основы	123
Конструкторы и выражения: <code>__init__</code> и <code>__sub__</code>	124
Распространенные методы перегрузки операций	124
Индексирование и нарезание: <code>__getitem__</code> и <code>__setitem__</code>	127
Перехват срезов	127
Нарезание и индексирование в Python 2.X	129
Но метод <code>__index__</code> в Python 3.X не имеет отношения к индексированию!	130
Итерация по индексам: <code>__getitem__</code>	130
Итерируемые объекты: <code>__iter__</code> и <code>__next__</code>	131
Итерируемые объекты, определяемые пользователем	132
Множество итераторов в одном объекте	135
Альтернативная реализация: <code>__iter__</code> плюс <code>yield</code>	138
Членство: <code>__contains__</code> , <code>__iter__</code> и <code>__getitem__</code>	142
Доступ к атрибутам: <code>__getattr__</code> и <code>__setattr__</code>	145
Ссылка на атрибуты	146
Присваивание и удаление атрибутов	147
Другие инструменты управления атрибутами	148
Эмуляция защиты атрибутов экземпляра: часть 1	149
Строковое представление: <code>__repr__</code> и <code>__str__</code>	150
Для чего используются два метода отображения?	151
Замечания по использованию отображения	152
Использование с правой стороны и на месте: <code>__radd__</code> и <code>__iadd__</code>	153
Правостороннее сложение	154
Сложение на месте	157
Выражения вызовов: <code>__call__</code>	158
Функциональные интерфейсы и код, основанный на обратных вызовах	160
Сравнения: <code>__lt__</code> , <code>__gt__</code> и другие	162
Метод <code>__cmp__</code> в Python 2.X	163
Булевские проверки: <code>__bool__</code> и <code>__len__</code>	163
Булевские методы в Python 2.X	164
Уничтожение объектов: <code>__del__</code>	166
Замечания относительно использования деструкторов	166
Резюме	167
Проверьте свои знания: контрольные вопросы	168
Проверьте свои знания: ответы	168
ГЛАВА 31. Проектирование с использованием классов	169
Python и объектно-ориентированное программирование	169
Полиморфизм означает интерфейсы, а не сигнатуры вызовов	170
Объектно-ориентированное программирование и наследование: отношения “является”	171
Объектно-ориентированное программирование и композиция: отношения “имеет”	173

Снова об обработчиках потоков данных	174
Объектно-ориентированное программирование и делегирование:	
промежуточные объекты-оболочки	178
Псевдозакрытые атрибуты классов	180
Обзор корректировки имен	180
Для чего используются псевдозакрытые атрибуты?	181
Методы являются объектами: связанные или несвязанные методы	183
Несвязанные методы являются функциями в Python 3.X	185
Связанные методы и другие вызываемые объекты	187
Классы являются объектами: обобщенные фабрики объектов	190
Для чего используются фабрики?	191
Множественное наследование: “подмешиваемые” классы	192
Реализация подмешиваемых классов отображения	193
Другие темы, связанные с проектированием	214
Резюме	214
Проверьте свои знания: контрольные вопросы	215
Проверьте свои знания: ответы	215
ГЛАВА 32. Расширенные возможности классов	216
Расширение встроенных типов	217
Расширение типов путем внедрения	217
Расширение типов путем создания подклассов	218
Модель классов “нового стиля”	220
Что нового в новом стиле?	221
Изменения в классах нового стиля	222
Процедура извлечения атрибутов для встроенных операций пропускает экземпляры	224
Изменения модели типов	229
Все классы являются производными от object	232
Изменение ромбовидного наследования	234
Дополнительные сведения о MRO: порядок распознавания методов	238
Пример: отображение атрибутов на источники наследования	241
Расширения в классах нового стиля	246
Слоты: объявления атрибутов	247
Свойства: средства доступа к атрибутам	256
Метод <code>__getattr__</code> и дескрипторы: инструменты для работы с атрибутами	259
Другие изменения и расширения классов	260
Статические методы и методы классов	261
Для чего используются специальные методы?	261
Статические методы в Python 2.X и 3.X	262
Альтернативы для статических методов	264
Использование статических методов и методов класса	265
Подсчет экземпляров с помощью статических методов	267
Подсчет экземпляров с помощью методов классов	268
Декораторы и метаклассы: часть 1	271
Основы декораторов функций	272
Первый взгляд на декораторы функций, определяемые пользователем	273
Первый взгляд на декораторы классов и метаклассы	275

Дополнительные сведения	277
Встроенная функция <code>super</code> : для лучшего или для худшего?	277
Продолжительные дебаты относительно <code>super</code>	277
Традиционная форма вызова методов суперкласса: переносимая, универсальная	279
Базовое использование встроенной функции <code>super</code> и связанные с ней компромиссы	279
Положительные стороны <code>SUPER</code> : изменения деревьев и координирование	285
Изменения классов во время выполнения и <code>super</code>	286
Кооперативная координация вызовов методов при множественном наследовании	287
Сводка по <code>super</code>	299
Затруднения, связанные с классами	300
Изменение атрибутов классов может иметь побочные эффекты	301
Модификация изменяемых атрибутов классов тоже может иметь побочные эффекты	302
Множественное наследование: порядок имеет значение	303
Области видимости в методах и классах	304
Другие затруднения, связанные с классами	305
Еще раз о KISS: чрезмерно большое количество уровней	306
Резюме	307
Проверьте свои знания: контрольные вопросы	307
Проверьте свои знания: ответы	307
Проверьте свои знания: упражнения для части VI	309
Часть VII. Исключения и инструменты	315
ГЛАВА 33. Основы исключений	316
Для чего используются исключения?	316
Роли, исполняемые исключениями	317
Исключения: краткая история	318
Стандартный обработчик исключений	318
Перехват исключений	320
Генерация исключений	321
Исключения, определяемые пользователем	321
Действия при завершении	322
Резюме	325
Проверьте свои знания: контрольные вопросы	326
Проверьте свои знания: ответы	326
ГЛАВА 34. Детали обработки исключений	327
Оператор <code>try/except/else</code>	327
Как работают операторы <code>try</code>	328
Конструкции оператора <code>try</code>	329
Конструкция <code>else</code> оператора <code>try</code>	332
Пример: стандартное поведение	333
Пример: перехват встроенных исключений	334

Оператор <code>try/finally</code>	335
Пример: написание кода действий при завершении с помощью <code>try/finally</code>	336
Унифицированный оператор <code>try/except/finally</code>	337
Унифицированный синтаксис оператора <code>try</code>	338
Комбинирование <code>finally</code> и <code>except</code> за счет вложения	339
Пример унифицированного оператора <code>try</code>	339
Оператор <code>raise</code>	341
Генерация исключений	342
Области видимости и переменные <code>except</code> в <code>try</code>	343
Распространение исключений с помощью <code>raise</code>	344
Сцепление исключений в Python 3.X: <code>raise from</code>	345
Оператор <code>assert</code>	347
Пример: улавливание нарушений ограничений (но не ошибок!)	348
Диспетчеры контекстов <code>with/as</code>	349
Базовое использование	350
Протокол управления контекстами	351
Множество диспетчеров контекстов в Python 3.1, 2.7 и последующих версиях	353
Резюме	355
Проверьте свои знания: контрольные вопросы	355
Проверьте свои знания: ответы	355
ГЛАВА 35. Объекты исключений	357
Исключения: назад в будущее	358
Строковые исключения канули в лету!	358
Исключения на основе классов	359
Реализация классов исключений	360
Для чего используются иерархии исключений?	362
Встроенные классы исключений	365
Категории встроенных исключений	366
Стандартный вывод и состояние	367
Специальное отображение при выводе	369
Специальные данные и поведение	371
Предоставление деталей исключения	371
Предоставление методов исключений	372
Резюме	373
Проверьте свои знания: контрольные вопросы	373
Проверьте свои знания: ответы	374
ГЛАВА 36. Проектирование с использованием исключений	375
Вложение обработчиков исключений	375
Пример: вложение в потоке управления	377
Пример: синтаксическое вложение	377
Идиомы исключений	379
Прерывание множества вложенных циклов: “безусловный переход”	379
Исключения не всегда являются ошибками	380
Функции могут сигнализировать об условиях с помощью <code>raise</code>	381
Закрытие файлов и серверных подключений	382
Отладка с помощью внешних операторов <code>try</code>	383
Выполнение внутрипроцессных тестов	383

Дополнительные сведения о функции <code>sys.exc_info</code>	384
Отображение сообщений об ошибках и трассировок	385
Советы по проектированию с использованием исключений и связанные с ними затруднения	386
Что должно быть помещено внутрь операторов <code>try</code> ?	386
Перехват слишком многого: избегайте использования пустой конструкции <code>except</code> и конструкции <code>except Exception</code>	387
Перехват чересчур малого: используйте категории на основе классов	389
Сводка по базовому языку	390
Комплект инструментов Python	390
Инструменты для разработки, ориентированные на более крупные проекты	391
Резюме	395
Проверьте свои знания: контрольные вопросы	396
Проверьте свои знания: ответы	396
Проверьте свои знания: упражнения для части VII	396
Часть VIII. Более сложные темы	399
ГЛАВА 37. Unicode и байтовые строки	400
Изменения строк в Python 3.X	401
Основы строк	402
Схемы кодирования символов	402
Хранение строк Python в памяти	405
Типы строк Python	406
Текстовые и двоичные файлы	408
Написание базовых строк	409
Строковые литералы Python 3.X	410
Строковые литералы Python 2.X	412
Преобразования строковых типов	412
Написание строк Unicode	414
Написание текста ASCII	414
Написание текста, отличающегося от ASCII	415
Кодирование и декодирование текста, отличающегося от ASCII	416
Другие схемы кодирования	417
Байтовые строковые литералы: закодированный текст	418
Преобразования между кодировками	420
Кодирование строк Unicode в Python 2.X	420
Объявления кодировок в файлах исходного кода	424
Использование объектов <code>bytes</code> в Python 3.X	425
Вызовы методов	425
Операции над последовательностями	426
Другие способы создания объектов <code>bytes</code>	427
Смешивание строковых типов	428
Использование объектов <code>bytearray</code> в Python 3.X/2.6+	428
Объекты <code>bytearray</code> в действии	429
Сводка по строковым типам Python 3.X	431
Использование текстовых и двоичных файлов	431
Основы текстовых файлов	432
Текстовый и двоичный режимы в Python 2.X и 3.X	433

Несоответствия типов и содержимого в Python 3.X	434
Использование файлов Unicode	435
Чтение и запись данных Unicode в Python 3.X	436
Обработка маркера BOM в Python 3.X	437
Файлы Unicode в Python 2.X	440
Имена файлов и потоки данных Unicode	441
Другие изменения инструментов для обработки строк в Python 3.X	442
Модуль re для сопоставления с образцом	442
Модуль struct для работы с двоичными данными	444
Модуль pickle для сериализации объектов	446
Инструменты для разбора XML	447
Резюме	451
Проверьте свои знания: контрольные вопросы	452
Проверьте свои знания: ответы	452
ГЛАВА 38. Управляемые атрибуты	455
Для чего используются управляемые атрибуты?	455
Вставка кода для запуска при доступе к атрибутам	456
Свойства	457
Основы	458
Первый пример	458
Вычисляемые атрибуты	459
Реализация свойств с помощью декораторов	460
Дескрипторы	462
Основы	462
Первый пример	465
Вычисляемые атрибуты	467
Использование информации состояния в дескрипторах	468
Связь между свойствами и дескрипторами	471
__getattr__ и __getattribute__	473
Основы	474
Первый пример	477
Вычисляемые атрибуты	478
Сравнение __getattr__ и __getattribute__	480
Сравнение методик управления	481
Перехват атрибутов для встроенных операций	484
Пример: проверка достоверности атрибутов	491
Использование свойств для проверки достоверности	492
Использование дескрипторов для проверки достоверности	494
Использование __getattr__ для проверки достоверности	498
Использование __getattribute__ для проверки достоверности	500
Резюме	501
Проверьте свои знания: контрольные вопросы	502
Проверьте свои знания: ответы	502
ГЛАВА 39. Декораторы	504
Что такое декоратор?	504
Управление вызовами и экземплярами	505
Управление функциями и классами	505

Использование и определение декораторов	506
Для чего используются декораторы?	506
Основы	508
Декораторы функций	508
Декораторы классов	512
Вложение декораторов	514
Аргументы декораторов	516
Декораторы одновременно управляют функциями и классами	517
Реализация декораторов функций	518
Отслеживание вызовов	518
Варианты предохранения состояния для декораторов	519
Грубые ошибки, связанные с классами, часть I: декорирование методов	524
Измерение времени вызовов	530
Добавление аргументов к декоратору	533
Реализация декораторов классов	536
Классы-одиночки	536
Отслеживание объектных интерфейсов	538
Грубые ошибки, связанные с классами, часть II: предохранение множества экземпляров	542
Декораторы или управляющие функции	543
Для чего используются декораторы? (Еще раз)	545
Управление функциями и классами напрямую	547
Пример: “закрытые” и “открытые” атрибуты	549
Реализация закрытых атрибутов	549
Детали реализации, часть I	551
Обобщение также для открытых объявлений	553
Детали реализации, часть II	555
Нерешенные проблемы	556
Python не поощряет контроль доступа	564
Пример: проверка допустимости аргументов функций	565
Цель	565
Базовый декоратор проверки вхождения значений в диапазон для позиционных аргументов	566
Обобщение для поддержки также ключевых аргументов и стандартных значений	568
Детали реализации	571
Нерешенные проблемы	574
Аргументы декоратора или аннотации функций	576
Другие приложения: проверка типов (если вы настаиваете!)	578
Резюме	579
Проверьте свои знания: контрольные вопросы	579
Проверьте свои знания: ответы	580
ГЛАВА 40. Метаклассы	590
Нужно ли иметь дело с метаклассами?	591
Повышение уровней “магии”	592
Язык привязок	593
Недостаток “вспомогательных” функций	594
Метаклассы против декораторов классов: раунд 1	596

Модель метаклассов	599
Классы являются экземплярами <code>type</code>	599
Метаклассы являются подклассами <code>type</code>	601
Протокол оператора <code>class</code>	602
Объявление метаклассов	603
Объявление в Python 3.X	603
Объявление в Python 2.X	604
Координирование метаклассов в Python 3.X и 2.X	605
Реализация метаклассов	605
Базовый метакласс	606
Настройка создания и инициализации	607
Другие методики реализации метаклассов	608
Наследование и экземпляр	613
Метакласс или суперкласс	615
Наследование: вся история	617
Методы метаклассов	623
Методы метаклассов или методы классов	624
Перегрузка операций в методах метакласса	624
Пример: добавление методов в классы	626
Ручное дополнение	626
Дополнение на основе метаклассов	628
Метаклассы против декораторов классов: раунд 2	629
Пример: применение декораторов к методам	634
Трассировка с помощью декорирования вручную	635
Трассировка с помощью метаклассов и декораторов	636
Применение любого декоратора к методам	637
Метаклассы против декораторов классов: раунд 3 (и последний)	639
Резюме	641
Проверьте свои знания: контрольные вопросы	642
Проверьте свои знания: ответы	642
ГЛАВА 41. Все хорошее когда-нибудь заканчивается	644
Парадокс Python	644
О “необязательных” языковых средствах	645
Против тревожных усовершенствований	646
Сложность или мощь	647
Простота или элитарность	648
Заключительные размышления	648
Куда двигаться дальше?	649
На бис: распечатайте собственный сертификат об окончании!	649
Часть IX. Приложения	653
Приложение А. Установка и конфигурирование	654
Установка интерпретатора Python	654
Присутствует ли Python на компьютере?	654
Где взять интерпретатор Python	655
Шаги установки	656

Конфигурирование интерпретатора Python	658
Переменные среды Python	658
Способы установки конфигурационных параметров	660
Аргументы командной строки интерпретатора Python	663
Командные строки запускающего модуля, появившегося в Python 3.3	666
Дополнительная помощь	667
Приложение Б. Запускающий модуль Windows для Python	668
Наследие Unix	668
Наследие Windows	669
Введение в запускающий модуль Windows	670
Учебное руководство по запускающему модулю Windows	672
Шаг 1: использование директив версий в файлах	672
Шаг 2: использование переключателей версий командной строки	675
Выводы: чистый выигрыш для Windows	676
Приложение В. Изменения в Python и настоящая книга	677
Главные отличия между Python 2.X и Python 3.X	677
Отличия Python 3.X	678
Расширения, доступные только в Python 3.X	679
Общие замечания: изменения в Python 3.X	680
Изменения в библиотеках и инструментах	681
Переход на Python 3.X	682
Изменения в Python, относящиеся к пятому изданию: Python 2.7, 3.2, 3.3	682
Изменения в Python 2.7	683
Изменения в Python 3.8	683
Изменения в Python 3.7	683
Изменения в Python 3.3	685
Изменения в Python 3.2	686
Изменения в Python, относящиеся к четвертому изданию: Python 2.6, 3.0, 3.1	686
Изменения в Python 3.1	686
Изменения в Python 3.0 и 2.6	687
Удаления в языке Python 3.0	688
Изменения в Python, относящиеся к третьему изданию: Python 2.3, 2.4, 2.5	691
Более ранние и более поздние изменения в Python	691
Приложение Г. Решения упражнений, приводимых в конце частей	692
Часть VI, “Классы и объектно-ориентированное программирование”	692
Часть VII, “Исключения и инструменты”	700
Предметный указатель	709