Содержание

Об авторе 29
Об изображении на обложке 30

Предисловие 31
Введение 32
Для кого предназначена эта книга 32
Учет различий между версиями Python 3 и Python 2 32
Структура книги 33
Файлы примеров и дополнительные материалы 33
Ждем ваших отзывов! 34

Глава 1. Текст 35
1.1. `string`: текстовые константы и шаблоны 35
1.1.1. Функции 36
1.1.2. Шаблоны 36
1.1.3. Более сложные шаблоны 38
1.1.4. Класс `Formatter` 40
1.1.5. Константы 40
1.2. `textwrap`: форматирование текстовых абзацев 41
1.2.1. Данные, используемые в примерах 41
1.2.2. Заполнение абзацев с помощью функции `fill` 42
1.2.3. Удаление существующих отступов 42
1.2.4. Совместный эффект функций `dedent` и `fill` 43
1.2.5. Декорирование блоков текста с помощью функции `indent` 44
1.2.6. Висячие отступы 45
1.2.7. Усечение длинного текста 46
1.3. `re`: регулярные выражения 47
1.3.1. Поиск образцов текста 48
1.3.2. Компиляция выражений 48
1.3.3. Многократные совпадения 49
1.3.4. Синтаксис шаблонов регулярных выражений 50
1.3.5. Ограничение зоны поиска 61
1.3.6. Отделение совпадений с помощью групп 63
1.3.7. Опции поиска 69
1.3.8. Просмотр вперед или назад 76
1.3.9. Обратные ссылки 80
1.3.10. Изменение строк с помощью шаблонов 85
1.3.11. Разбиение текста с помощью шаблонов 87

1.4. difflib: сравнение последовательностей 90
1.4.1. Сравнение версий текста 90
1.4.2. Ненужные данные 93
1.4.3. Сравнение произвольных типов 94

Глава 2. Структуры данных 97

2.1. enum: перечисление 98
2.1.1. Создание перечислений 98
2.1.2. Итерирование по элементам 99
2.1.3. Сравнение элементов перечислений 100
2.1.4. Уникальность значений перечисления 101
2.1.5. Создание перечислений программным способом 103
2.1.6. Значения элементов, не являющиеся целыми числами 104
2.2. collections: контейнерные типы данных 107
2.2.1. ChainMap: поиск в нескольких словарях 107
2.2.2. Counter: подсчет хешируемых экземпляров 111
2.2.3. defaultdict: возврат значения по умолчанию для отсутствующего ключа 114
2.2.4. deque: двухсторонняя очередь 115
2.2.5. namedtuple: подкласс Tuple с именованными полями 120
2.2.6. OrderedDict: запоминание порядка добавляемых ключей 124
2.2.7. collections.abc: абстрактные базовые классы контейнеров 127
2.3. array: последовательность данных фиксированного типа 129
2.3.1. Инициализация 129
2.3.2. Манипулирование массивами 130
2.3.3. Массивы и файлы 130
2.3.4. Альтернативные варианты порядка байтов 132
2.4. heapq: алгоритм сортировки кучи 133
2.4.1. Данные для примеров 133
2.4.2. Создание кучи 134
2.4.3. Доступ к содержимому кучи 135
2.4.4. Получение наибольших и наименьших элементов кучи 137
2.4.5. Эффективное слияние отсортированных последовательностей 138
2.5. bisect: поддержание сортированного состояния списков 139
2.5.1. Сортировка при вставке 139
2.5.2. Обработка повторяющихся значений 140
2.6. queue: потокобезопасная реализация очереди FIFO 141
2.6.1. Базовая очередь FIFO 141
2.6.2. Очередь LIFO 142
2.6.3. Очередь с приоритетом 142
2.6.4. Создание многопоточного подкаст-клиента 144
2.7. struct: структуры двоичных данных 147
2.7.1. Функции уровня модуля и класс Struct 147
2.7.2. Упаковка и распаковка 147

2.7.3. Индикатор порядка байтов 148
2.7.4. Буферизация 150
2.8. weakref: слабые ссылки на объекты 151
2.8.1. Ссылки 151
2.8.2. Функции обратного вызова в слабых ссылках 152
2.8.3. Завершающие операции при удалении объектов 153
2.8.4. Прокси-объекты 156
2.8.5. Объекты кеширования 156
2.9. copy: создание дубликатов объектов 159
2.9.1. Мелкие копии 159
2.9.2. Глубокие копии 160
2.9.3. Настройка копирования 161
2.9.4. Рекурсия при глубоком копировании 162
2.10. pprint: "красивая печать" структур данных 164
2.10.1. Вывод на консоль 165
2.10.2. Форматирование 166
2.10.3. Произвольные классы 166
2.10.4. Рекурсия 167
2.10.5. Ограничение уровня выводимых вложенных структур 168
2.10.6. Управление шириной вывода 168

Глава 3. Алгоритмы 171

3.1. functools: инструменты для манипулирования функциями 171
3.1.1. Декораторы 171
3.1.2. Сравнение 179
3.1.3. Кеширование 182
3.1.4. Редукция набора данных 186
3.1.5. Обобщенные функции 188
3.2. itertools: функции-итераторы 190
3.2.1. Объединение и разделение итераторов 191
3.2.2. Преобразование входных данных 194
3.2.3. Создание новых значений 196
3.2.4. Фильтрация 198
3.2.5. Группирование данных 201
3.2.6. Комбинирование входных данных 203
3.3. operator: функциональный интерфейс встроенных операторов 209
3.3.1. Логические операции 209
3.3.2. Операторы сравнения 210
3.3.3. Арифметические операторы 211
3.3.4. Операторы для работы с последовательностями 212
3.3.5. Операторы, изменяющие операнды 213
3.3.6. Функции доступа к элементам и атрибутам 214
3.3.7. Сочетание операторов с пользовательскими классами 216
3.4. contextlib: утилиты менеджеров контекста 217

3.4.1. API менеджера контекста 217
3.4.2. Менеджеры контекста как декораторы функций 220
3.4.3. От генератора к менеджеру контекста 221
3.4.4. Закрытие открытых дескрипторов 224
3.4.5. Игнорирование исключений 225
3.4.6. Перенаправление выходных потоков 226
3.4.7. Стеки динамических менеджеров контекста 227

Глава 4. Дата и время 237

4.1. `time`: системное время 237
4.1.1. Сравнительные характеристики часов 238
4.1.2. Часы текущего времени 239
4.1.3. Монотонные часы 240
4.1.4. Процессорное время 240
4.1.5. Измерение производительности 242
4.1.6. Компоненты времени 243
4.1.7. Работа с часовыми поясами 244
4.1.8. Разбор и форматирование значений времени 245
4.2. `datetime`: манипулирование значениями даты и времени 247
4.2.1. Время 247
4.2.2. Даты 248
4.2.3. Промежутки времени 250
4.2.4. Арифметика дат 252
4.2.5. Сравнение значений 253
4.2.6. Объединение значений даты и времени 254
4.2.7. Форматирование и анализ значений 255
4.2.8. Часовые пояса 257
4.3. `calendar`: работа с датами 258
4.3.1. Примеры форматирования 258
4.3.2. Локали 261
4.3.3. Вычисление дат 262

Глава 5. Математика 265

5.1. `decimal`: математика чисел с фиксированной точностью и чисел с плавающей точкой 265
5.1.1. Класс `Decimal` 265
5.1.2. Форматирование 267
5.1.3. Арифметика 268
5.1.4. Специальные значения 269
5.1.5. Контекст 270
5.2. `fractions`: рациональные числа 275
5.2.1. Создание экземпляров `Fraction` 275
5.2.2. Арифметика 278
5.2.3. Аппроксимация значений 278

5.3. random: генератор псевдослучайных чисел 279
5.3.1. Генерация случайных чисел 279
5.3.2. Инициализация 280
5.3.3. Сохранение состояния 281
5.3.4. Случайные целые числа 282
5.3.5. Выбор случайных элементов 283
5.3.6. Перестановки 284
5.3.7. Выборки 286
5.3.8. Одновременное использование нескольких генераторов 286
5.3.9. Класс SystemRandom 287
5.3.10. Неоднородные распределения 288
5.4. math: математические функции 290
5.4.1. Специальные константы 290
5.4.2. Тестирование исключительных значений 291
5.4.3. Сравнение 293
5.4.4. Преобразование значений с плавающей точкой в целые числа 295
5.4.5. Альтернативные представления значений с плавающей точкой 296
5.4.6. Знак числа 298
5.4.7. Распространенные виды вычислений 299
5.4.8. Экспоненты и логарифмы 303
5.4.9. Углы 307
5.4.10. Тригонометрия 309
5.4.11. Гиперболические функции 312
5.4.12. Специальные функции 313
5.5. statistics: статистические расчеты 315
5.5.1. Средние значения 315
5.5.2. Дисперсия 317

Глава 6. Файловая система 319

6.1. os.path: платформонезависимое манипулирование именами файлов 320
6.1.1. Анализ путей 320
6.1.2. Создание путей 324
6.1.3. Нормализация путей 325
6.1.4. Временны*е* характеристики файлов 326
6.1.5. Тестирование файлов 327
6.2. pathlib: пути файловой системы как объекты 329
6.2.1. Представления пути 329
6.2.2. Создание путей 329
6.2.3. Анализ путей 331
6.2.4. Создание полных путей 333
6.2.5. Содержимое каталога 333
6.2.6. Чтение и запись файлов 336
6.2.7. Манипулирование каталогами и символическими ссылками 336
6.2.8. Типы файлов 337
6.2.9. Свойства файлов 338

6.2.10. Права доступа 341
6.2.11. Удаление объекта файловой системы 342
6.3. glob: шаблоны имен файлов 343
6.3.1. Данные для примеров 343
6.3.2. Групповые метасимволы 344
6.3.3. Метасимвол, соответствующий одиночному символу 345
6.3.4. Диапазоны символов 345
6.3.5. Экранирование метасимволов 346
6.4. fnmatch: шаблоны модуля Glob в стиле Unix 347
6.4.1. Простое сопоставление 347
6.4.2. Фильтрация 348
6.4.3. Трансляция шаблонов 349
6.5. linecache: эффективное чтение файлов 349
6.5.1. Тестовые данные 350
6.5.2. Чтение конкретных строк 350
6.5.3. Обработка пустых строк 351
6.5.4. Обработка ошибок 351
6.5.5. Чтение исходных файлов Python 352
6.6. tempfile: временные объекты файловой системы 353
6.6.1. Временные файлы 353
6.6.2. Именованные файлы 355
6.6.3. Буферизуемые файлы 356
6.6.4. Временные каталоги 357
6.6.5. Предсказуемые имена 358
6.6.6. Расположение временного файла 359
6.7. shutil: высокоуровневые файловые операции 360
6.7.1. Копирование файлов 360
6.7.2. Копирование метаданных файла 363
6.7.3. Работа с деревьями каталогов 364
6.7.4. Поиск файлов 367
6.7.5. Архивы 369
6.7.6. Размер файловой системы 372
6.8. filelecmp: сравнение файлов 373
6.8.1. Данные для примеров 373
6.8.2. Сравнение файлов 375
6.8.3. Сравнение каталогов 377
6.8.4. Использование различий в программах 378
6.9. mmap: файлы, отображаемые в памяти 382
6.9.1. Чтение 383
6.9.2. Запись 384
6.9.3. Регулярные выражения 385
6.10. codecs: кодирование и декодирование строк 386
6.10.1. Основы Unicode 386
6.10.2. Работа с файлами 389

6.10.3. Порядок байтов 391
6.10.4. Обработка ошибок 393
6.10.5. Преобразование кодировок 397
6.10.6. Другие кодировки 398
6.10.7. Инкрементное кодирование 400
6.10.8. Unicode и сетевой обмен данными 401
6.10.9. Определение нестандартной кодировки 404
6.11. `io`: инструменты для работы с текстовыми, двоичными и "сырыми" потоками ввода-вывода 411
6.11.1. Потоки, отображаемые в памяти 412
6.11.2. Обертывание байтовых потоков для текстовых данных 413

Глава 7. Постоянное хранение и обмен данными 415

7.1. `pickle`: сериализация объектов 416
7.1.1. Кодирование и декодирование строковых данных 416
7.1.2. Работа с потоками 418
7.1.3. Проблемы реконструирования объектов 419
7.1.4. Объекты, не сериализуемые с помощью модуля `pickle` 421
7.1.5. Циклические ссылки 422
7.2. `shelve`: постоянное хранение объектов 425
7.2.1. Создание нового хранилища 425
7.2.2. Обратная запись 426
7.2.3. Специализированные типы хранилищ 428
7.3. `dbm`: базы данных Unix с доступом по ключу 428
7.3.1. Типы баз данных 429
7.3.2. Создание новой базы данных 429
7.3.3. Открытие существующей базы данных 430
7.3.4. Примеры ошибок 431
7.4. `sqlite3`: встроенная реляционная база данных 432
7.4.1. Создание базы данных 432
7.4.2. Извлечение данных 435
7.4.3. Метаданные запроса 437
7.4.4. Объекты `Row` 437
7.4.5. Использование переменных в запросах 439
7.4.6. Групповая загрузка 441
7.4.7. Описание новых типов столбцов 442
7.4.8. Определение типов столбцов 445
7.4.9. Транзакции 447
7.4.10. Уровни изоляции 450
7.4.11. Базы данных в памяти 454
7.4.12. Экспорт содержимого базы данных 454
7.4.13. Использование функций Python в SQL 456
7.4.14. Использование регулярных выражений в запросах 458
7.4.15. Пользовательское агрегирование 459
7.4.16. Многопоточность и совместное использование соединений 460

7.4.17. Ограничение доступа к данным 461
7.5. xml.etree.ElementTree: API для манипулирования XML-элементами 464
7.5.1. Синтаксический анализ XML-документов 464
7.5.2. Обход дерева узлов 465
7.5.3. Поиск узлов в документе 466
7.5.4. Атрибуты узлов 468
7.5.5. Отслеживание событий в процессе анализа документа 469
7.5.6. Создание построителя пользовательского дерева 472
7.5.7. Синтаксический анализ строк 474
7.5.8. Создание документов с помощью узлов Element 475
7.5.9. Красивая печать XML 476
7.5.10. Установка свойств объектов Element 478
7.5.11. Создание деревьев из списков узлов 479
7.5.12. Сериализация XML-разметки в поток 482
7.6. csv: файлы с данными, разделенными запятыми 484
7.6.1. Чтение 485
7.6.2. Запись 486
7.6.3. Диалекты 487
7.6.4. Использование имен полей 492

Глава 8. Сжатие и архивирование данных 495

8.1. zlib: сжатие данных средствами библиотеки GNU zlib 495
8.1.1. Работа с данными в памяти 495
8.1.2. Инкрементное сжатие и восстановление данных 497
8.1.3. Потоки смешанного содержимого 498
8.1.4. Контрольные суммы 499
8.1.5. Сжатие сетевых данных 500
8.2. gzip: чтение и запись файлов GNU zip 504
8.2.1. Запись сжатых файлов 504
8.2.2. Чтение сжатых данных 506
8.2.3. Работа с потоками 507
8.3. bz2: формат сжатия bzip2 508
8.3.1. Обработка всего набора данных в памяти 509
8.3.2. Инкрементное сжатие и восстановление данных 510
8.3.3. Потоки смешанного содержимого 511
8.3.4. Запись сжатых файлов 512
8.3.5. Чтение сжатых файлов 514
8.3.6. Чтение и запись данных Unicode 515
8.3.7. Сжатие сетевых данных 516
8.4. tarfile: доступ к архивам Tar 520
8.4.1. Тестирование tar-файлов 520
8.4.2. Чтение метаданных из архива 520
8.4.3. Извлечение файлов из архива 522
8.4.4. Создание новых архивов 524

8.4.5. Использование альтернативных имен файлов в архиве 524
8.4.6. Запись данных из источников, отличных от файлов 525
8.4.7. Присоединение файла к архиву 525
8.4.8. Работа со сжатыми архивами 526
8.5. zipfile: доступ к ZIP-архивам 527
8.5.1. Тестирование ZIP-файлов 527
8.5.2. Чтение метаданных из архива 528
8.5.3. Извлечение файлов из архива 529
8.5.4. Создание новых архивов 530
8.5.5. Использование альтернативных имен файлов в архиве 532
8.5.6. Запись данных из источников, отличных от файлов 532
8.5.7. Запись с помощью экземпляра ZipInfo 533
8.5.8. Присоединение архива к файлу 534
8.5.9. ZIP-архивы Python 535
8.5.10. Ограничения 536

Глава 9. Криптография 537

9.1. hashlib: криптографическое хеширование 537
9.1.1. Алгоритмы хеширования 537
9.1.2. Пробный набор данных 538
9.1.3. Алгоритм MD5 538
9.1.4. Алгоритм SHA1 539
9.1.5. Создание хеш-кода с указанием имени алгоритма 539
9.1.6. Инкрементное обновление 540
9.2. hmac: криптографические цифровые подписи и верификация сообщений 541
9.2.1. Подписывание сообщений 541
9.2.2. Альтернативные типы дайджестов 542
9.2.3. Двоичные дайджесты 543
9.2.4. Применение цифровых подписей сообщений 544

Глава 10. Параллельные вычисления: процессы, потоки и сопрограммы 547

10.1. subprocess: порождение дополнительных процессов 548
10.1.1. Выполнение внешних команд 548
10.1.2. Непосредственная работа с каналами 554
10.1.3. Соединение сегментов канала 556
10.1.4. Взаимодействие с другой командой 558
10.1.5. Межпроцессный обмен сигналами 560
10.2. signal: асинхронные системные события 564
10.2.1. Получение сигналов 565
10.2.2. Получение информации о зарегистрированных обработчиках сигналов 566
10.2.3. Отправка сигналов 567

10.2.4. Сигналы таймера 567
10.2.5. Игнорирование сигналов 568
10.2.6. Сигналы и потоки 569
10.3. `threading`: управление параллельными вычислениями в рамках одного процесса 571
10.3.1. Объекты `Thread` 572
10.3.2. Определение текущего потока 573
10.3.3. Потоки, являющиеся и не являющиеся демонами 575
10.3.4. Перечисление всех потоков 577
10.3.5. Использование подклассов `Thread` 579
10.3.6. Потоки `Timer` 580
10.3.7. Обмен сигналами между потоками 581
10.3.8. Управление доступом к ресурсам 583
10.3.9. Синхронизация потоков 588
10.3.10. Ограничение одновременного доступа к ресурсам 592
10.3.11. Данные, специфичные для потока 593
10.4. `multiprocessing`: использование процессов вместо потоков 596
10.4.1. Основы многопроцессной обработки 596
10.4.2. Импортируемые целевые функции 597
10.4.3. Определение текущего процесса 598
10.4.4. Процессы-демоны 599
10.4.5. Ожидание завершения процессов 601
10.4.6. Прекращение работы процесса 603
10.4.7. Код завершения процесса 604
10.4.8. Протоколирование 605
10.4.9. Создание подклассов `Process` 607
10.4.10. Передача сообщений процессам 607
10.4.11. Обмен сигналами между процессами 611
10.4.12. Управление доступом к ресурсам 612
10.4.13. Синхронизация операций 613
10.4.14. Контроль одновременного доступа к ресурсам 614
10.4.15. Управление разделяемым состоянием 616
10.4.16. Разделяемые пространства имен 617
10.4.17. Пулы процессов 619
10.4.18. Реализация MapReduce 621
10.5. `asyncio`: асинхронные операции ввода-вывода, цикл событий и инструменты параллелизма 625
10.5.1. Принципы асинхронного параллелизма 626
10.5.2. Организация кооперативной многозадачности с помощью сопрограмм 627
10.5.3. Планирование вызовов обычных функций 631
10.5.4. Асинхронное получение результатов 633
10.5.5. Параллельное выполнение задач 636
10.5.6. Сочетание сопрограмм с управляющими конструкциями 640
10.5.7. Примитивы синхронизации 645

10.5.8. Асинхронный ввод-вывод с использованием абстракций класса `Protocol` 652
10.5.9. Асинхронные операции ввода-вывода с использованием сопрограмм и потоков 658
10.5.10. Использование SSL 663
10.5.11. Взаимодействие со службами DNS 666
10.5.12. Работа с подпроцессами 668
10.5.13. Получение сигналов Unix 675
10.5.14. Сочетание сопрограмм с потоками и процессами 677
10.5.15. Отладка с помощью модуля `asyncio` 681
10.6. `concurrent.futures`: управление пулами параллельных задач 684
10.6.1. Использование метода `map()` с базовым пулом потоков 685
10.6.2. Планирование индивидуальных задач 686
10.6.3. Ожидание завершения задач в произвольном порядке 687
10.6.4. Обратные вызовы с использованием экземпляров `Future` 688
10.6.5. Отмена выполнения задач 689
10.6.6. Возбуждение исключений в задачах 690
10.6.7. Менеджер контекста 691
10.6.8. Пулы процессов 691

Глава 11. Обмен данными по сети 695

11.1. `ipaddress`: интернет-адреса 695
11.1.1. Адреса 695
11.1.2. Сети 696
11.1.3. Интерфейсы 699
11.2. `socket`: сетевое взаимодействие 701
11.2.1. Адресация, семейства протоколов и типы сокетов 701
11.2.2. Клиент и сервер TCP/IP 711
11.2.3. Клиент и сервер UDP 719
11.2.4. Сокеты домена Unix 721
11.2.5. Многоадресатное вещание 724
11.2.6. Отправка двоичных данных 728
11.2.7. Неблокирующее взаимодействие и тайм-ауты 730
11.3. `selectors`: абстракции мультиплексирования ввода-вывода 731
11.3.1. Рабочая модель 731
11.3.2. Эхо-сервер 732
11.3.3. Эхо-клиент 733
11.3.4. Совместная работа сервера и клиента 735
11.4. `select`: эффективное ожидание завершения ввода-вывода 736
11.4.1. Использование функции `select()` 736
11.4.2. Неблокирующий ввод-вывод с тайм-аутами 742
11.4.3. Использование функции `poll()` 744
11.4.4. Опции, специфические для платформы 749
11.5. `socketserver`: создание сетевых серверов 749

11.5.1. Типы серверов 750
11.5.2. Объекты сервера 750
11.5.3. Реализация сервера 750
11.5.4. Обработчики запросов 751
11.5.5. Пример с эхо-сервером и эхо-клиентами 751
11.5.6. Создание потоков и порождение процессов 756

Глава 12. Интернет 761

12.1. `urllib.parse`: разбиение URL-адресов на отдельные элементы 762
12.1.1. Анализ URL-адресов 762
12.1.2. Конструирование строки URL-адреса из элементов 764
12.1.3. Объединение элементов 766
12.1.4. Кодирование параметров запроса 766
12.2. `urllib.request`: доступ к сетевым ресурсам 769
12.2.1. Метод HTTP GET 769
12.2.2. Кодирование параметров запроса 771
12.2.3. Метод HTTP POST 772
12.2.4. Добавление исходящих заголовков 772
12.2.5. Отправка формы данных на сервер 773
12.2.6. Выгрузка файлов 774
12.2.7. Создание пользовательских обработчиков протоколов 777
12.3. `urllib.robotparser`: управление действиями веб-роботов 780
12.3.1. Файл *robots.txt* 780
12.3.2. Тестирование прав доступа 781
12.3.3. Длительно выполняющиеся веб-роботы 782
12.4. `base64`: кодирование двоичных данных с помощью ASCII 783
12.4.1. Кодировка Base64 784
12.4.2. Декодирование формата Base64 784
12.4.3. Вариации, безопасные для использования в URL-адресах 785
12.4.4. Другие кодировки 786
12.5. `http.server`: базовые классы для реализации веб-серверов 788
12.5.1. HTTP GET 788
12.5.2. HTTP POST 790
12.5.3. Порождение потоков и процессов 792
12.5.4. Обработка ошибок 793
12.5.5. Настройка заголовков 794
12.5.6. Использование командной строки 795
12.6. `http.cookies`: cookie-файлы HTTP 796
12.6.1. Создание и настройка cookie-файлов 796
12.6.2. Атрибуты cookie-файлов 796
12.6.3. Кодированные значения 798
12.6.4. Получение и анализ заголовков cookie-файлов 799
12.6.5. Альтернативные выходные форматы 800
12.7. `webbrowser`: отображение веб-страниц 801

12.7.1. Простой пример 801
12.7.2. Окна и вкладки 801
12.7.3. Использование конкретного браузера 802
12.7.4. Переменная BROWSER 802
12.7.5. Интерфейс командной строки 802
12.8. uuid: универсальные уникальные идентификаторы 803
12.8.1. UUID 1: MAC-адрес (стандарт IEEE 802) 803
12.8.2. UUID версий 3 и 5: значения на основе заданного имени 805
12.8.3. UUID 4: случайные значения 807
12.8.4. Работа с объектами UUID 808
12.9. json: JavaScript Object Notation 809
12.9.1. Кодирование и декодирование простых типов данных 809
12.9.2. Удобочитаемость и компактность вывода 810
12.9.3. Кодирование словарей 812
12.9.4. Работа с пользовательскими типами 813
12.9.5. Классы кодировщиков и декодировщиков 815
12.9.6. Работа с потоками и файлами 818
12.9.7. Смешанные потоки данных 819
12.9.8. JSON и командная строка 820
12.10. xmlrpc.client: клиент XML-RPC 821
12.10.1. Подключение к серверу 822
12.10.2. Типы данных 824
12.10.3. Передача объектов 827
12.10.4. Двоичные данные 828
12.10.5. Обработка исключений 830
12.10.6. Комбинирование вызовов в одном сообщении 830
12.11. xmlrpc.server: сервер XML-RPC 832
12.11.1. Простой сервер 832
12.11.2. Альтернативные имена API 834
12.11.3. Имена API с точками 835
12.11.4. Произвольные имена API 836
12.11.5. Предоставление методов объектов 836
12.11.6. Диспетчеризация вызовов 838
12.11.7. API интроспекции 840

Глава 13. Электронная почта 843

13.1. smtplib: клиент SMTP 843
13.1.1. Отправка сообщений 843
13.1.2. Аутентификация и шифрование 845
13.1.3. Верификация адреса электронной почты 848
13.2. smtpd: примеры почтовых серверов 849
13.2.1. Базовый класс почтового сервера 849
13.2.2. Отладочный сервер 852
13.2.3. Прокси-сервер 852

13.3. mailbox: манипулирование архивами электронной почты 853
13.3.1. mbox 854
13.3.2. Maildir 856
13.3.3. Флаги сообщений 863
13.3.4. Другие форматы 865
13.4. imaplib: клиентская библиотека IMAP4 865
13.4.1. Разновидности класса IMAP4 865
13.4.2. Подключение к серверу 866
13.4.3. Конфигурационные данные для примера 867
13.4.4. Получение списка почтовых ящиков 868
13.4.5. Состояние почтового ящика 870
13.4.6. Выбор почтового ящика 871
13.4.7. Поиск сообщений 872
13.4.8. Критерии поиска 873
13.4.9. Извлечение сообщений 875
13.4.10. Извлечение всего сообщения 880
13.4.11. Выгрузка сообщений 881
13.4.12. Перемещение и копирование сообщений 883
13.4.13. Удаление сообщений 884

Глава 14. Строительные блоки приложений 887

14.1. argparse: анализ параметров и аргументов командной строки 888
14.1.1. Настройка синтаксического анализатора 888
14.1.2. Определение аргументов 889
14.1.3. Анализ командной строки 889
14.1.4. Простые примеры 889
14.1.5. Вывод справки 897
14.1.6. Организация работы анализатора 901
14.1.7. Дополнительная обработка аргументов 908
14.2. getopt: анализ параметров командной строки 915
14.2.1. Аргументы функции getopt() 916
14.2.2. Короткая форма параметров 916
14.2.3. Длинная форма параметров 917
14.2.4. Более полный пример 917
14.2.5. Сокращение длинной формы параметров 919
14.2.6. Анализ параметров в стиле GNU 919
14.2.7. Прекращение обработки аргументов 921
14.3. readline: библиотека GNU Readline 921
14.3.1. Конфигурирование библиотеки Readline 922
14.3.2. Автозавершение ввода 923
14.3.3. Доступ к буферу автозавершения ввода 926
14.3.4. История ввода 929
14.3.5. Функции-перехватчики 932
14.4. getpass: безопасный ввод пароля 933
14.4.1. Пример 934

14.4.2. Использование функции getpass() без терминала 935
14.5. cmd: построчные командные процессоры 936
14.5.1. Обработка команд 936
14.5.2. Аргументы команд 937
14.5.3. Активная справка 939
14.5.4. Автозавершение ввода 940
14.5.5. Переопределение методов базового класса 942
14.5.6. Конфигурирование класса Cmd с помощью атрибутов 944
14.5.7. Выполнение команд оболочки 945
14.5.8. Альтернативные варианты ввода 946
14.5.9. Извлечение команд из переменной sys.argv 947
14.6. shlex: лексический анализ синтаксисов в стиле командной оболочки Unix 949
14.6.1. Анализ строк, содержащих кавычки 949
14.6.2. Создание безопасных строк для командных оболочек 950
14.6.3. Встроенные комментарии 951
14.6.4. Разбиение строк на лексемы 952
14.6.5. Другие источники лексем 952
14.6.6. Управление анализатором 953
14.6.7. Обработка ошибок 955
14.6.8. Анализ в соответствии с требованиями стандарта POSIX 956
14.7. configparser: работа с конфигурационными файлами 958
14.7.1. Формат конфигурационных файлов 958
14.7.2. Чтение конфигурационных файлов 959
14.7.3. Доступ к конфигурационным параметрам 960
14.7.4. Изменение параметров 967
14.7.5. Сохранение конфигурационных файлов 968
14.7.6. Пути поиска параметров 969
14.7.7. Объединение значений с помощью интерполяции 971
14.8. logging: механизм структурированного журналирования 976
14.8.1. Журналирование событий компонентов 976
14.8.2. Отличия в журналировании событий приложений и библиотек 977
14.8.3. Запись журнала в файл 977
14.8.4. Циклическое создание файлов журнала 978
14.8.5. Уровни важности сообщений 979
14.8.6. Именование экземпляров регистратора 980
14.8.7. Иерархическое дерево регистраторов сообщений 981
14.8.8. Интеграция с модулем warnings 982
14.9. fileinput: библиотека фильтров для утилит командной строки 983
14.9.1. Преобразование M3U-файлов в каналы RSS 983
14.9.2. Метаданные хода выполнения процесса 985
14.9.3. Фильтрация на месте 987
14.10. atexit: вызов функций интерпретатором при завершении работы программы 989
14.10.1. Регистрация функций завершения 989

14.10.2. Синтаксис декораторов 990
14.10.3. Отмена регистрации функций обратного вызова 991
14.10.4. Случаи, когда функции обратного вызова модуля `atexit` не вызываются 992
14.10.5. Обработка исключений 994
14.11. `sched`: планирование запуска событий 995
14.11.1. Запуск событий с задержкой 995
14.11.2. Перекрывающиеся события 996
14.11.3. Приоритеты событий 997
14.11.4. Отмена событий 998

Глава 15. Интернационализация и локализация приложений 1001

15.1. `gettext`: каталоги сообщений 1001
15.1.1. Обзор процесса перевода сообщений 1001
15.1.2. Создание каталога сообщений на основе исходного кода 1002
15.1.3. Поиск каталогов сообщений во время выполнения 1005
15.1.4. Грамматические формы для множественного числа 1006
15.1.5. Локализация приложений и модулей 1009
15.1.6. Переключение вариантов перевода 1010
15.2. `locale`: API локализации 1010
15.2.1. Проверка региональных настроек 1011
15.2.2. Денежные единицы 1016
15.2.3. Форматирование чисел 1017
15.2.4. Анализ чисел 1018
15.2.5. Дата и время 1019

Глава 16. Инструменты разработки 1021

16.1. `pydoc`: оперативная справка для модулей 1022
16.1.1. Получение справки в формате простого текста 1023
16.1.2. Справка в формате HTML 1023
16.1.3. Интерактивная справка 1024
16.2. `doctest`: тестирование документации 1024
16.2.1. Начало работы 1025
16.2.2. Обработка непредсказуемого вывода 1026
16.2.3. Трассировочная информация 1030
16.2.4. Обработка пробелов 1032
16.2.5. Местонахождение тестов 1036
16.2.6. Внешняя документация 1040
16.2.7. Выполнение тестов 1042
16.2.8. Контекст тестирования 1045
16.3. `unittest`: фреймворк автоматизированного тестирования 1048
16.3.1. Базовая структура тестов 1048
16.3.2. Выполнение тестов 1049
16.3.3. Результаты тестов 1049

16.3.4. Подтверждение выполнения условия 1051
16.3.5. Тестирование равенства 1051
16.3.6. Приблизительное равенство 1053
16.3.7. Контейнеры 1053
16.3.8. Тестирование исключений 1058
16.3.9. Разделяемые контексты тестов 1059
16.3.10. Повторение тестов с различными входными данными 1062
16.3.11. Пропуск тестов 1063
16.3.12. Игнорирование неудачных тестов 1064
16.4. `trace`: трассировка выполнения программы 1065
16.4.1. Пример программы 1065
16.4.2. Трассировка выполнения 1066
16.4.3. Покрытие кода 1067
16.4.4. Взаимные вызовы функций 1069
16.4.5. Программный интерфейс 1070
16.4.6. Сохранение результатов 1072
16.4.7. Опции 1073
16.5. `traceback`: исключения и стек вызовов 1074
16.5.1. Вспомогательные функции 1075
16.5.2. Работа со стеком 1075
16.5.3. Исключение `TracebackException` 1077
16.5.4. Низкоуровневые программные интерфейсы исключений 1078
16.5.5. Низкоуровневые программные интерфейсы стека 1082
16.6. `cgitb`: подробные отчеты о необработанных исключениях 1084
16.6.1. Стандартные дампы трассировочной информации 1085
16.6.2. Активизация вывода подробной трассировочной информации 1085
16.6.3. Локальные переменные в трассировочных стеках вызовов 1088
16.6.4. Свойства объекта исключения 1091
16.6.5. Вывод в формате HTML 1093
16.6.6. Запись трассировочной информации в журнал 1093
16.7. `pdb`: интерактивный отладчик 1096
16.7.1. Запуск отладчика 1096
16.7.2. Управление отладчиком 1099
16.7.3. Точки останова 1111
16.7.4. Изменение потока управления 1123
16.7.5. Настройка отладчика с помощью псевдонимов 1129
16.7.6. Сохранение конфигурационных параметров 1131
16.8. `profile` и `pstats`: анализ производительности 1133
16.8.1. Запуск профилировщика 1133
16.8.2. Выполнение в контексте 1136
16.8.3. `pstats`: работа со статистиками 1137
16.8.4. Ограничение содержимого отчета 1138
16.8.5. Графы вызова функций 1139
16.9. `timeit`: замер времени выполнения небольших фрагментов кода Python 1141

16.9.1. Содержимое модуля 1141
16.9.2. Базовый пример 1141
16.9.3. Сохранение значений в словаре 1142
16.9.4. Тестирование из командной строки 1144
16.10. tabnanny: проверка отступов 1146
16.10.1. Запуск из командной строки 1146
16.11. compileall: файлы скомпилированного байт-кода 1147
16.11.1. Компиляция одного каталога 1147
16.11.2. Игнорирование файлов 1148
16.11.3. Компиляция sys.path 1149
16.11.4. Компиляция отдельных файлов 1150
16.11.5. Компиляция из командной строки 1151
16.12. pyclbr: обозреватель классов 1152
16.12.1. Поиск классов 1153
16.12.2. Поиск функций 1155
16.13. venv: создание виртуальных окружений 1155
16.13.1. Создание окружения 1155
16.13.2. Содержимое виртуального окружения 1156
16.13.3. Использование виртуальных окружений 1157
16.14. ensurepip: программа-установщик пакетов Python 1159
16.14.1. Установка pip 1159

Глава 17. Инструменты среды времени выполнения 1161

17.1. site: конфигурирование сайта 1162
17.1.1. Пути импорта модулей 1162
17.1.2. Пользовательские каталоги 1163
17.1.3. Конфигурационные файлы путей 1164
17.1.4. Настройка конфигурации сайта 1167
17.1.5. Настройка пользовательской конфигурации 1168
17.1.6. Отключение модуля site 1169
17.2. sys: настройка конфигурационных параметров, специфических для системы 1170
17.2.1. Параметры интерпретатора 1170
17.2.2. Среда времени выполнения 1177
17.2.3. Управление памятью и ограничения 1179
17.2.4. Обработка исключений 1185
17.2.5. Низкоуровневая поддержка потоков 1188
17.2.6. Модули и операции импорта 1191
17.2.7. Трассировка выполняющихся программ 1211
17.3. os: портируемый доступ к средствам, специфическим для операционных систем 1217
17.3.1. Исследование содержимого файловой системы 1217
17.3.2. Управление правами доступа к файловой системе 1220
17.3.3. Создание и удаление каталогов 1222

17.3.4. Работа с символическими ссылками 1223
17.3.5. Безопасная замена существующего файла 1224
17.3.6. Определение и изменение владельца процесса 1225
17.3.7. Управление окружением процесса 1227
17.3.8. Управление рабочим каталогом процесса 1228
17.3.9. Выполнение внешних команд 1228
17.3.10. Создание процессов с помощью вызова `os.fork()` 1230
17.3.11. Ожидание завершения дочерних процессов 1232
17.3.12. Создание новых процессов 1234
17.3.13. Коды ошибок операционной системы 1234
17.4. `platform`: информация о версии системы 1235
17.4.1. Интерпретатор 1236
17.4.2. Платформа 1237
17.4.3. Информация об операционной системе и оборудовании 1238
17.4.4. Архитектура исполняемой программы 1239
17.5. `resource`: управление системными ресурсами 1240
17.5.1. Текущее потребление ресурсов 1240
17.5.2. Лимитирование ресурсов 1241
17.6. `gc`: сборщик мусора 1244
17.6.1. Отслеживание ссылок 1244
17.6.2. Принудительная сборка мусора 1247
17.6.3. Обнаружение ссылок на объекты, которые не могут быть отобраны сборщиком мусора 1248
17.6.4. Пороги и поколения сборки мусора 1251
17.6.5. Отладка 1254
17.7. `sysconfig`: управление конфигурацией интерпретатора во время компиляции 1258
17.7.1. Конфигурационные переменные 1258
17.7.2. Пути к каталогам установки 1261
17.7.3. Информация о версии и платформе Python 1264

Глава 18. Инструменты языка 1267

18.1. `warnings`: предупреждения о потенциальных проблемах 1268
18.1.1. Категории предупреждений и фильтрация 1268
18.1.2. Генерация предупреждений 1268
18.1.3. Фильтрация с помощью шаблонов 1270
18.1.4. Повторные предупреждения 1272
18.1.5. Альтернативные функции доставки сообщений 1273
18.1.6. Форматирование 1273
18.1.7. Глубина просмотра стека модулем `warnings` 1274
18.2. `abc`: абстрактные базовые классы 1275
18.2.1. Как работают абстрактные классы 1275
18.2.2. Регистрация конкретного класса 1276
18.2.3. Реализация посредством создания подклассов 1277

18.2.4. Вспомогательный базовый класс 1278
18.2.5. Неполные реализации 1278
18.2.6. Конкретные методы в абстрактных базовых классах 1279
18.2.7. Абстрактные свойства 1280
18.2.8. Абстрактный класс и статические методы 1283
18.3. `dis`: дизассемблирование байт-кода Python 1284
18.3.1. Простой пример дизассемблирования 1284
18.3.2. Дизассемблирование функций 1285
18.3.3. Классы 1287
18.3.4. Исходный код 1288
18.3.5. Использование дизассемблирования в целях отладки 1289
18.3.6. Анализ производительности циклов 1291
18.3.7. Оптимизация, выполняемая компилятором 1297
18.4. `inspect`: инспектирование активных объектов 1298
18.4.1. Образец модуля для примеров 1299
18.4.2. Инспектирование модулей 1299
18.4.3. Инспектирование классов 1301
18.4.4. Инспектирование экземпляров 1302
18.4.5. Строки документирования 1303
18.4.6. Извлечение исходного кода 1304
18.4.7. Сигнатуры методов и функций 1305
18.4.8. Иерархии классов 1308
18.4.9. Порядок разрешения методов 1309
18.4.10. Стек и фреймы 1311
18.4.11. Интерфейс командной строки 1313

Глава 19. Модули и пакеты 1315

19.1. `importlib`: механизм импорта Python 1315
19.1.1. Пакет `example` 1315
19.1.2. Типы модулей 1316
19.1.3. Импортирование модулей 1317
19.1.4. Загрузчики 1318
19.2. `pkgutil`: вспомогательные функции для упаковывания программ 1319
19.2.1. Пути импорта пакетов 1320
19.2.2. Разработка версий пакетов 1322
19.2.3. Управление путями с помощью PKG-файлов 1323
19.2.4. Вложенные пакеты 1325
19.2.5. Пакетные данные 1326
19.3. `zipimport`: загрузка кода Python из ZIP-архивов 1329
19.3.1. Пример 1329
19.3.2. Поиск модуля 1330
19.3.3. Доступ к коду 1331
19.3.4. Исходный код 1331
19.3.5. Пакеты 1333
19.3.6. Данные 1333

Приложение А. Замечания относительно портирования программ 1337

А.1. Ссылки 1337
А.2. Новые модули 1337
А.3. Переименованные модули 1338
А.4. Удаленные модули 1340
А.4.1. bsddb 1340
А.4.2. commands 1340
А.4.3. compiler 1340
А.4.4. dircache 1340
А.4.5. EasyDialogs 1340
А.4.6. exceptions 1340
А.4.7. htmllib 1340
А.4.8. md5 1340
А.4.9. mimetools, MimeWriter, mimify, multifile и rfc822 1340
А.4.10. popen2 1340
А.4.11. posixfile 1341
А.4.12. sets 1341
А.4.13. sha 1341
А.4.14. sre 1341
А.4.15. statvfs 1341
А.4.16. thread 1341
А.4.17. user 1341
А.5. Устаревшие модули 1341
А.5.1. asyncore и asynchat 1341
А.5.2. formatter 1341
А.5.3. imp 1342
А.5.4. optparse 1342
А.6. Сводка изменений, внесенных в модули 1342
А.6.1. abc 1342
А.6.2. anydbm 1342
А.6.3. argparse 1342
А.6.4. array 1343
А.6.5. atexit 1343
А.6.6. base64 1343
А.6.7. bz2 1343
А.6.8. collections 1343
А.6.9. comands 1343
А.6.10. configparser 1344
А.6.11. contextlib 1344
А.6.12. csv 1344
А.6.13. datetime 1344
А.6.14. decimal 1344
А.6.15. fractions 1344

A.6.16. gc 1345
A.6.17. gettext 1345
A.6.18. glob 1345
A.6.19. http.cookies 1345
A.6.20. imaplib 1345
A.6.21. inspect 1345
A.6.22. itertools 1345
A.6.23. json 1345
A.6.24. locale 1346
A.6.25. logging 1346
A.6.26. mailbox 1346
A.6.27. mmap 1346
A.6.28. operator 1346
A.6.29. os 1347
A.6.30. os.path 1347
A.6.31. pdb 1347
A.6.32. pickle 1347
A.6.33. pipes 1348
A.6.34. platform 1348
A.6.35. random 1348
A.6.36. re 1349
A.6.37. shelve 1349
A.6.38. signal 1349
A.6.39. socket 1349
A.6.40. socketserver 1349
A.6.41. string 1349
A.6.42. struct 1349
A.6.43. subprocess 1350
A.6.44. sys 1350
A.6.45. threading 1350
A.6.46. time 1351
A.6.47. unittest 1351
A.6.48. Классы UserDict, UserList и UserString 1351
A.6.49. uuid 1352
A.6.50. whichdb 1352
A.6.51. xml.etree.ElementTree 1352
A.6.52. zipimport 1352

Приложение Б. Внешние ресурсы, дополняющие стандартную библиотеку 1353

Б.1. Текст 1353
Б.2. Алгоритмы 1354
Б.3. Дата и время 1354
Б.4. Математика 1354
Б.5. Постоянное хранение и обмен данными 1354

Б.6. Криптография 1355
Б.7. Параллельные вычисления: процессы, потоки и сопрограммы 1355
Б.8. Интернет 1356
Б.9. Электронная почта 1356
Б.10. Строительные блоки приложений 1357
Б.11. Инструменты разработки 1357

Указатель модулей Python 1359
Предметный указатель 1361