

УДК 004.42
ББК 32.973.26-018.2
С17

Последнюю версию этой книги на английском языке можно скачать с сайта: <https://GoalKicker.com/PythonBook>. Пожалуйста, не стесняйтесь поделиться этим PDF-файлом с кем угодно бесплатно.

Книга Python® Notes for Professionals составлена на основе документации Stack Overflow (<https://archive.org/details/documentation-dump.7z>), содержание написано прекрасными людьми из Stack Overflow. **Текстовое содержимое представлено на условиях Creative Commons BY-SA.** В конце книги указаны авторы, внесшие вклад в создание различных глав. Авторские права на изображения принадлежат их соответствующим владельцам, если не указано иное.

С17 **Самое** полное руководство по разработке на Python в примерах от сообщества Stack Overflow. — Москва : Издательство АСТ, 2024. — 672 с. : ил. — (Программирование от экспертов). ISBN 978-5-17-160252-9.

Данное руководство по программированию на одном из широко распространенных языков — Python — основано на практических примерах кодов, написанных специалистами и экспертами сообщества Stack Overflow, в котором лучшие разработчики программного обеспечения со всего мира делятся своими знаниями и опытом, отвечая на многие технические вопросы. Опытные Python-программисты найдут в книге множество примеров кода с подробными комментариями, что поможет им усовершенствовать свои навыки и достичь новых высот в отрасли. Однако данное издание будет полезно и начинающим специалистам с минимальным опытом и уровнем знаний, так как содержит исчерпывающее объяснение важнейших концепций Python с примерами, которые позволят избежать погружения в сухую теорию и помогут быстро повысить уровень своих компетенций. Читатели найдут здесь мощный и универсальный инструментарий для профессиональной работы в самых разных областях применения: с базами данных, веб-фреймворком Flask, XML и JSON, звуковыми данными, синтаксическим анализатором Lex-Yacc, а также при сетевом программировании, визуализации данных, многопоточности и многопроцессорности, программировании «интернета вещей». Кроме того, в книге представлена информация о применении Python в сфере науки, например, в математике, химии и криптографии. Отдельные главы посвящены секретам повышения скорости работы Python-кода и оптимизирования его производительности.

УДК 004.42
ББК 32.973.26-018.2

ISBN 978-5-17-160252-9

Перевод на русский язык: ООО «Интеджер».
Издание на русском языке: ООО «Издательство АСТ».

Содержание

| | |
|--|----|
| Глава 1. Начало работы с языком Python | 27 |
| 1.1. Начало работы | 27 |
| 1.2. Создание переменных и присвоение им значений | 31 |
| 1.3. Отступы блоков | 34 |
| 1.4. Типы данных | 36 |
| 1.5. Типы коллекций | 39 |
| 1.6. IDLE – графический интерфейс Python | 43 |
| 1.7. Ввод данных пользователем | 44 |
| 1.8. Встроенные модули и функции | 45 |
| 1.9. Создание модуля | 47 |
| 1.10. Установка Python 2.7.x и 3.x | 48 |
| 1.11. Строковые функции – str() и repr() | 50 |
| 1.12. Установка внешних модулей с помощью pip | 51 |
| 1.13. Справочная утилита | 52 |
| Глава 2. Типы данных в Python | 53 |
| 2.1. Строковый тип данных | 53 |
| 2.2. Множества (set и frozenset) | 53 |
| 2.3. Числовые типы данных | 54 |
| 2.4. Тип данных “список” (list) | 54 |
| 2.5. Тип данных “словарь” (dic) | 54 |
| 2.6. Тип данных “кортеж” (tuple) | 54 |
| Глава 3. Отступы | 55 |
| 3.1. Простой пример | 55 |
| 3.2. Как происходит разбор отступов | 55 |
| 3.3. Ошибки отступа | 56 |
| Глава 4. Комментарии и документация | 56 |
| 4.1. Однострочные, строчные и многострочные комментарии | 56 |
| 4.2. Программный доступ к строкам документации (docstrings) | 57 |
| 4.3. Написание документации с использованием docstrings | 58 |
| Глава 5. Дата и время | 60 |
| 5.1. Разбор строки в объект даты и времени (datetime) с учетом часового пояса | 60 |
| 5.2. Построение временных интервалов с учетом временных зон | 60 |
| 5.3. Вычисление разницы во времени | 62 |
| 5.4. Использование базовых объектов datetime | 62 |
| 5.5. Переключение между часовыми поясами | 63 |
| 5.6. Простые арифметические действия с датами | 63 |
| 5.7. Преобразование временной метки (timestamp) в объект datetime | 64 |
| 5.8. Точное вычитание месяцев из даты | 64 |
| 5.9. Разбор (парсинг) произвольной временной метки ISO 8601 с минимальным использованием библиотек | 64 |
| 5.10. Получение временной метки ISO 8601 | 65 |
| 5.11. Разбор строки с коротким именем часового пояса в объект datetime с учетом часового пояса | 65 |
| 5.12. Нечеткий синтаксический анализ времени (извлечение даты и времени из текста) | 66 |
| 5.13. Итерация по датам | 66 |
| Глава 6. Форматирование даты | 67 |
| 6.1. Время между двумя датами | 67 |
| 6.2. Вывод объекта datetime в строку | 67 |
| 6.3. Парсинг строки в объект datetime | 67 |

| | |
|---|----|
| Глава 7. Модуль Enum | 68 |
| 7.1. Создание перечисления (Python 2.4–3.3) | 68 |
| 7.2. Итерация | 68 |
| Глава 8. Множества | 68 |
| 8.1. Операции над множествами | 68 |
| 8.2. Получение уникальных элементов списка | 69 |
| 8.3. Множество множеств | 70 |
| 8.4. Операции над множествами с использованием методов и встроенных модулей | 70 |
| 8.5. Множества и мультимножества | 72 |
| Глава 9. Простые математические операторы | 72 |
| 9.1. Деление | 73 |
| 9.2. Сложение | 74 |
| 9.3. Возведение в степень | 74 |
| 9.4. Тригонометрические функции | 75 |
| 9.5. Операторы присваивания (“операции на месте”) | 76 |
| 9.6. Вычитание | 76 |
| 9.7. Умножение | 77 |
| 9.8. Логарифмы | 77 |
| 9.9. Остаток от деления | 77 |
| Глава 10. Побитовые операторы | 78 |
| 10.1. Побитовое НЕ | 78 |
| 10.2. Побитовое XOR (исключающее ИЛИ) | 80 |
| 10.3. Побитовое И | 80 |
| 10.4. Побитовое ИЛИ | 80 |
| 10.5. Побитовый сдвиг влево | 81 |
| 10.6. Побитовый сдвиг вправо | 81 |
| 10.7. Операторы присваивания (“операции на месте”) | 81 |
| Глава 11. Логические операции | 82 |
| 11.1. “and” и “or” не гарантируют возврата булевого значения | 82 |
| 11.2. Простой пример | 82 |
| 11.3. Вычисления по короткой схеме (“короткое замыкание”) | 82 |
| 11.4. Оператор “and” (“и”) | 83 |
| 11.5. Оператор “or” (“или”) | 83 |
| 11.6. Оператор “not” (“не”) | 84 |
| Глава 12. Приоритет операторов | 84 |
| 12.1. Примеры приоритета операторов в Python | 84 |
| Глава 13. Область видимости и привязка переменных | 85 |
| 13.1. Нелокальные переменные | 85 |
| 13.2. Глобальные переменные | 86 |
| 13.3. Локальные переменные | 87 |
| 13.4. Команда “del” | 87 |
| 13.5. Функции пропускают область видимости класса при поиске имен | 88 |
| 13.6. Локальные и глобальные области | 89 |
| 13.7. Возникновение привязки | 92 |
| Глава 14. Условные выражения | 92 |
| 14.1. Условное выражение (“тернарный оператор”) | 92 |
| 14.2. if, elif и else | 92 |
| 14.3. Значения истинности | 93 |
| 14.4. Выражения булевой логики | 93 |
| 14.5. Использование функции “cmp” для получения результата сравнения двух объектов | 95 |

| | |
|--|-----|
| 14.6. Оператор "else" | 95 |
| 14.7. Проверка принадлежности объекта к None и его присвоение | 95 |
| 14.8. Оператор "if" | 96 |
| Глава 15. Сравнение..... | 96 |
| 15.1. Цепное сравнение..... | 96 |
| 15.2. Сравнение по принципу "is" и "==" | 97 |
| 15.3. Больше или меньше..... | 98 |
| 15.4. Не равно..... | 98 |
| 15.5. Равно | 99 |
| 15.6. Сравнение объектов | 99 |
| Глава 16. Циклы | 100 |
| 16.1. Перерыв и продолжение в циклах | 100 |
| 16.2. Циклы For | 102 |
| 16.3. Итерирование по спискам | 102 |
| 16.4. Циклы с "else" | 103 |
| 16.5. Заявление о прохождении | 105 |
| 16.6. Итерация в словарях | 105 |
| 16.7. "Полуцикл" do-while | 106 |
| 16.8. Циклирование и распаковка | 106 |
| 16.9. Итерирование части списка с разным размером шага | 107 |
| 16.10. Цикл While | 108 |
| Глава 17. Массивы..... | 108 |
| 17.1. Доступ к отдельным элементам через индексы..... | 109 |
| 17.2. Введение в массивы..... | 109 |
| 17.3. Добавление произвольного значения в массив с помощью метода append()..... | 110 |
| 17.4. Вставка значения в массив с помощью метода insert() | 110 |
| 17.5. Расширение массива с помощью метода extend() | 110 |
| 17.6. Добавление элементов из списка в массив с помощью метода fromlist() | 110 |
| 17.7. Удаление любого элемента массива с помощью метода remove()..... | 110 |
| 17.8. Удаление последнего элемента массива с помощью метода pop() | 110 |
| 17.9. Получение любого элемента по его индексу с помощью метода index() | 111 |
| 17.10. Обратное преобразование массива с помощью метода reverse() | 111 |
| 17.11. Получение информации о буфере массива с помощью метода buffer_info ()..... | 111 |
| 17.12. Проверка количества вхождений элемента с помощью метода count ()..... | 111 |
| 17.13. Преобразование массива в строку с помощью метода tostring ()..... | 111 |
| 17.14. Преобразование массива в список с одинаковыми элементами с использованием метода tolist () | 111 |
| 17.15. Добавление строки в массив char, используя метод fromstring () | 112 |
| Глава 18. Многомерные массивы..... | 112 |
| 18.1. Списки в списках..... | 112 |
| 18.2. Списки в списках в списках в | 113 |
| Глава 19. Словарь | 113 |
| 19.1. Введение в словарь | 113 |
| 19.2. Избегание исключений KeyError | 114 |
| 19.3. Итерация по словарю | 115 |
| 19.4. Словарь со значениями по умолчанию | 115 |
| 19.5. Слияние словарей..... | 116 |
| 19.6. Доступ к ключам и значениям..... | 116 |
| 19.7. Доступ к значениям словаря | 117 |
| 19.8. Создание словаря | 117 |
| 19.9. Создание упорядоченного словаря | 118 |
| 19.10. Распаковка словарей с помощью оператора **..... | 118 |

| | |
|---|-----|
| 19.11. Запятая в конце строки..... | 119 |
| 19.12. Конструктор dict() | 119 |
| 19.13. Пример словарей..... | 119 |
| 19.14. Все комбинации значений словаря..... | 119 |
| Глава 20. Список | 120 |
| 20.1. Методы перечисления и поддерживаемые операторы | 120 |
| 20.2. Доступ к значениям списка | 125 |
| 20.3. Проверка списка на пустоту..... | 126 |
| 20.4. Итерирование по списку | 126 |
| 20.5. Проверка наличия элемента в списке | 126 |
| 20.6. Функции “any” и “all” | 127 |
| 20.7. Реверсирование элементов списка | 127 |
| 20.8. Конкатенация и слияние списков | 128 |
| 20.9. Длина списка | 129 |
| 20.10. Удаление дублирующихся значений в списке | 129 |
| 20.11. Сравнение списков..... | 129 |
| 20.12. Доступ к значениям во вложенном списке | 129 |
| 20.13. Инициализация списка с фиксированным числом элементов | 130 |
| Глава 21. Генератор списков | 131 |
| 21.1. Списковые вычисления | 131 |
| 21.2. Условные генераторы списков..... | 133 |
| 21.3. Избегание повторных и ресурсоемких операций с помощью условного предложения | 134 |
| 21.4. Генератор словаря (словарь включений)..... | 136 |
| 21.5. Генераторы списков с вложенными циклами | 137 |
| 21.6. Генераторные выражения | 138 |
| 21.7. Генераторы наборов | 140 |
| 21.8. Рефакторинг функций filter и map в генераторы списков | 140 |
| 21.9. Генераторы с использованием кортежей | 141 |
| 21.10. Подсчет вхождений при использовании генераторов..... | 142 |
| 21.11. Изменение типов в списке | 142 |
| 21.12. Вложенные генераторы списков..... | 142 |
| 21.13. Итерация двух и более списков одновременно внутри генератора списка..... | 143 |
| Глава 22. Срезы списков (выделение частей списков)..... | 143 |
| 22.1. Использование третьего аргумента “step” | 143 |
| 22.2. Выбор подсписка из списка | 144 |
| 22.3. Реверсирование списка с помощью среза | 144 |
| 22.4. Смещение списка с помощью среза | 144 |
| Глава 23. Метод groupby()..... | 145 |
| 23.1. Пример | 145 |
| 23.2. Пример 2 | 146 |
| 23.3. Пример 3 | 146 |
| Глава 24. Связные списки..... | 147 |
| 24.1. Пример односвязного списка..... | 147 |
| Глава 25. Узел связного списка | 150 |
| 25.1. Написание простого узла связного списка на языке Python | 150 |
| Глава 26. Фильтр..... | 151 |
| 26.1. Основное использование фильтра | 151 |
| 26.2. Фильтр без функции..... | 151 |
| 26.3. Фильтр для проверки на “короткое замыкание” (вычисления по короткой схеме) | 152 |

| | |
|---|-----|
| 26.4. Дополняющая функция: <code>filterfalse</code> , <code>ifilterfalse</code> | 152 |
| Глава 27. Модуль “ <code>heapq</code> ” | 153 |
| 27.1. Самые большие и самые маленькие предметы в коллекции | 153 |
| 27.2. Наименьший элемент в коллекции | 154 |
| Глава 28. Кортеж (<code>tuple</code>) | 154 |
| 28.1. Кортеж | 154 |
| 28.2. Кортежи являются неизменяемыми | 155 |
| 28.3. Упаковка и распаковка кортежей | 155 |
| 28.4. Встроенные функции кортежей | 156 |
| 28.5. Кортежи являются поэлементно хешируемыми и сравниваемыми | 157 |
| 28.6. Индексирование кортежей | 157 |
| 28.7. Реверсирование элементов | 158 |
| Глава 29. Основы ввода и вывода данных | 158 |
| 29.1. Использование функции вывода “ <code>print</code> ” | 158 |
| 29.2. Ввод из файла | 158 |
| 29.3. Чтение из <code>stdin</code> | 160 |
| 29.4. Использование функций <code>input()</code> и <code>raw_input()</code> | 160 |
| 29.5. Функция запроса числа у пользователя | 161 |
| 29.6. Печать строки без новой строки в конце | 161 |
| Глава 30. Ввод/вывод файлов и папок | 162 |
| 30.1. Режимы работы с файлами | 162 |
| 30.2. Построчное чтение <code>file</code> | 164 |
| 30.3. Итерация файлов (рекурсивно) | 164 |
| 30.4. Получение полного содержимого файла | 165 |
| 30.5. Запись в файл | 165 |
| 30.6. Проверка наличия файла или пути | 166 |
| 30.7. Произвольный доступ к файлам с помощью <code>mmap</code> | 167 |
| 30.8. Замена текста в файле | 167 |
| 30.9. Проверка того, что файл пуст | 167 |
| 30.10. Чтение файла в диапазоне строк | 168 |
| 30.11. Копирование дерева каталогов | 168 |
| 30.12. Копирование содержимого файла в другой файл | 168 |
| Глава 31. Модуль <code>os.path</code> | 168 |
| 31.1. Объединение путей | 168 |
| 31.2. Управление компонентами пути | 169 |
| 31.3. Получение родительского каталога | 169 |
| 31.4. Проверка существования пути | 169 |
| 31.5. Проверка того, является ли данный путь каталогом, файлом, символической ссылкой, точкой монтирования | 169 |
| 31.6. Абсолютный путь из относительного пути | 169 |
| Глава 32. Итерируемые типы данных и итераторы | 170 |
| 32.1. Итератор (<code>iterator</code>), итерируемый объект (<code>iterable</code>) и генератор (<code>generator</code>) | 170 |
| 32.2. Извлечение значений по одному | 171 |
| 32.3. Итерирование по всему итерируемому | 171 |
| 32.4. Проверка только одного элемента в итерируемом | 171 |
| 32.5. Что может быть итерируемым | 171 |
| 32.6. В итератор нельзя входить повторно! | 172 |
| Глава 33. Функции | 172 |
| 33.1. Создание и вызов простых функций | 172 |
| 33.2. Определение функции с произвольным числом аргументов | 174 |

| | |
|--|-----|
| 33.3. Лямбда-функции (встроенные/анонимные) | 176 |
| 33.4. Создание функции с необязательными аргументами | 178 |
| 33.5. Определение функции с необязательными изменяемыми аргументами | 178 |
| 33.6. Передача аргументов и изменяемость | 179 |
| 33.7. Возврат значений из функций | 180 |
| 33.8. Замыкание (closure)..... | 181 |
| 33.9. Принудительное использование именованных параметров..... | 182 |
| 33.10. Вложенные функции..... | 182 |
| 33.11. Предел рекурсии | 183 |
| 33.12. Рекурсивная лямбда-функция с использованием присваиваемой переменной..... | 183 |
| 33.13. Рекурсивные функции..... | 184 |
| 33.14. Определение функции с аргументами..... | 185 |
| 33.15. Распаковка итерируемых объектов и словарей..... | 185 |
| 33.16. Определение функции с несколькими аргументами..... | 187 |
| Глава 34. Создание функций со списочными аргументами | 187 |
| 34.1. Функция и вызов | 187 |
| Глава 35. Функциональное программирование в Python | 188 |
| 35.1. Лямбда-функция..... | 188 |
| 35.2. Функция map | 188 |
| 35.3. Функция reduce | 188 |
| 35.4. Функция filter..... | 188 |
| Глава 36. Частичные функции..... | 189 |
| 36.1. Возведение в степень..... | 189 |
| Глава 37. Декораторы | 189 |
| 37.1. Функция-декоратор..... | 190 |
| 37.2. Класс декоратора..... | 191 |
| 37.3. Декоратор с аргументами (фабрика декораторов)..... | 192 |
| 37.4. Приведение декоратора к виду декорируемой функции | 193 |
| 37.5. Использование декоратора для определения времени выполнения функции | 194 |
| 37.6. Создание класса-синглтона (одиночки) с помощью декоратора | 194 |
| Глава 38. Классы..... | 195 |
| 38.1. Введение в классы..... | 195 |
| 38.2. Связанные, несвязанные и статические методы..... | 197 |
| 38.3. Базовое наследование..... | 199 |
| 38.4. Манкипатчинг (Monkey Patching)..... | 200 |
| 38.5. Классы нового и старого стиля | 201 |
| 38.6. Методы класса: альтернативные инициализаторы..... | 202 |
| 38.7. Множественное наследование..... | 203 |
| 38.8. Свойства | 204 |
| 38.9. Значения по умолчанию для переменных экземпляра | 206 |
| 38.10. Переменные класса и экземпляра | 207 |
| 38.11. Композиция классов | 208 |
| 38.12. Перечисление всех членов класса | 209 |
| 38.13. Класс-синглтон | 209 |
| 38.14. Дескрипторы и точечный поиск | 211 |
| Глава 39. Метаклассы..... | 211 |
| 39.1. Базовые метаклассы | 211 |
| 39.2. Синглтоны, использующие метаклассы | 212 |
| 39.3. Использование метакласса | 212 |
| 39.4. Введение в метаклассы..... | 213 |

| | |
|--|-----|
| 39.5. Пользовательская функциональность в метаклассах..... | 214 |
| 39.6. Метакласс по умолчанию..... | 214 |
| Глава 40. Форматирование строк..... | 215 |
| 40.1. Основы форматирования строк..... | 215 |
| 40.2. Выравнивание и заполнение..... | 216 |
| 40.3. Форматные литералы (f-строка)..... | 217 |
| 40.4. Форматирование чисел с плавающей точкой..... | 217 |
| 40.5. Именованные заполнители..... | 218 |
| 40.6. Форматирование строк с использованием типа <code>datetime</code> | 219 |
| 40.7. Форматирование числовых значений..... | 219 |
| 40.8. Вложенное форматирование..... | 219 |
| 40.9. Форматирование с помощью функций <code>Getitem</code> и <code>Getattr</code> | 220 |
| 40.10. Комбинированное заполнение и усечение строк..... | 220 |
| 40.11. Пользовательское форматирование класса..... | 221 |
| Глава 41. Строковые методы..... | 222 |
| 41.1. Изменение регистра букв в строке..... | 222 |
| 41.2. <code>str.translate</code> : замена символов в строке..... | 223 |
| 41.3. <code>str.format</code> и f-strings: форматирование значений в строку..... | 223 |
| 41.4. Полезные константы модуля <code>string</code> | 224 |
| 41.5. Удаление ненужных ведущих и завершающих символов из строки..... | 225 |
| 41.6. Реверсирование строки..... | 226 |
| 41.7. Разбиение строки по разделителю на список строк..... | 226 |
| 41.8. Замена всех вхождений одной подстроки на другую подстроку..... | 227 |
| 41.9. Проверка содержимого строк..... | 228 |
| 41.10. Проверка содержания подстроки в строке..... | 230 |
| 41.11. Объединение списка строк в одну строку..... | 230 |
| 41.12. Подсчет количества вхождений подстроки в строке..... | 230 |
| 41.13. Сравнение строк без учета регистра..... | 230 |
| 41.14. Выравнивание строк..... | 232 |
| 41.15. Проверка начальных и конечных символов строки..... | 232 |
| 41.16. Преобразование между строковыми или байтовыми данными и символами <code>Unicode</code> | 233 |
| Глава 42. Использование циклов внутри функций..... | 234 |
| 42.1. Оператор возврата внутри цикла в функции..... | 234 |
| Глава 43. Импорт модулей..... | 235 |
| 43.1. Импорт модуля..... | 235 |
| 43.2. Специальная переменная <code>__all__</code> | 236 |
| 43.3. Импорт модулей из произвольного места файловой системы..... | 237 |
| 43.4. Импорт всех имен из модуля..... | 237 |
| 43.5. Программный импорт..... | 237 |
| 43.6. Правила PEP8 для импорта..... | 238 |
| 43.7. Импорт конкретных имен из модуля..... | 238 |
| 43.8. Импорт submodule..... | 239 |
| 43.9. Повторный импорт модуля..... | 239 |
| 43.10. Функция <code>__import__()</code> | 240 |
| Глава 44. Разница между модулем и пакетом..... | 240 |
| 44.1. Модули..... | 240 |
| 44.2. Пакеты..... | 240 |
| Глава 45. Математический модуль <code>math</code> | 241 |
| 45.1. Округление: <code>round</code> , <code>floor</code> , <code>ceil</code> , <code>trunc</code> | 241 |
| 45.2. Тригонометрия..... | 242 |

| | |
|---|-----|
| 45.3. Функция pow для быстрого возведения в степень | 243 |
| 45.4. Бесконечность и NaN ("не число") | 244 |
| 45.5. Логарифмы | 246 |
| 45.6. Константы | 246 |
| 45.7. Мнимые числа | 247 |
| 45.8. Копирование знаков | 247 |
| 45.9. Комплексные числа и модуль cmath | 247 |
| Глава 46. Комплексная математика | 250 |
| 46.1. Расширенная комплексная арифметика | 250 |
| 46.2. Основы комплексной арифметики | 250 |
| Глава 47. Модуль collections | 251 |
| 47.1. collections.Counter | 251 |
| 47.2. collections.OrderedDict | 252 |
| 47.3. collections.defaultdict | 253 |
| 47.4. collections.namedtuple | 254 |
| 47.5. collections.deque | 254 |
| 47.6. collections.ChainMap | 256 |
| Глава 48. Модуль operator | 257 |
| 48.1. Функция Itemgetter | 257 |
| 48.2. operator как альтернатива инфиксному оператору | 257 |
| 48.3. Methodcaller | 257 |
| Глава 49. Модуль JSON | 258 |
| 49.1. Хранение данных в файле | 258 |
| 49.2. Получение данных из файла | 258 |
| 49.3. Форматирование вывода JSON | 258 |
| 49.4. "load" и "loads", "dump" и "dumps" | 259 |
| 49.5. Вызов "json.tool" из командной строки для эстетичного вывода JSON | 260 |
| 49.6. JSON-кодирование пользовательских объектов | 260 |
| 49.7. Создание JSON из словаря Python | 261 |
| 49.8. Создание словаря Python из JSON | 261 |
| Глава 50. Модуль Sqlite3 | 261 |
| 50.1. Sqlite3 не требует отдельного серверного процесса | 261 |
| 50.2. Получение значений из базы данных и обработка ошибок | 261 |
| Глава 51. Модуль os | 262 |
| 51.1. mkdir - recursive создание каталогов | 262 |
| 51.2. Создание каталога | 263 |
| 51.3. Получение текущего каталога | 263 |
| 51.4. Определение операционной системы | 263 |
| 51.5. Удаление каталога | 263 |
| 51.6. Следование по симлинку (POSIX) | 263 |
| 51.7. Изменение разрешений файла | 264 |
| Глава 52. Модуль locale | 264 |
| 52.1. Форматирование валюты в долларах США с использованием модуля locale | 264 |
| Глава 53. Модуль itertools | 264 |
| 53.1. Метод комбинаций в модуле itertools | 264 |
| 53.2. itertools.dropwhile | 265 |
| 53.3. Использование zip_longest для двух итераторов до тех пор, пока они оба не будут исчерпаны | 265 |
| 53.4. Получение среза генератора | 265 |

| | |
|---|-----|
| 53.5. Группировка элементов из итерируемого объекта с помощью функции | 266 |
| 53.6. <code>itertools takewhile</code> | 267 |
| 53.7. <code>itertools permutations</code> | 267 |
| 53.8. <code>itertools repeat</code> | 268 |
| 53.9. Получение накопленной суммы чисел в итерируемом объекте | 268 |
| 53.10. Циклический переход по элементам итератора | 268 |
| 53.11. <code>itertools product</code> | 268 |
| 53.12. <code>itertools count</code> | 269 |
| 53.13. Объединение нескольких итераторов в цепочку | 270 |
| Глава 54. Модуль <code>Asyncio</code> | 270 |
| 54.1. Синтаксис корутин (сопрограмм) и делегирования | 270 |
| 54.2. Асинхронные исполнители (<code>Executors</code>) | 271 |
| 54.3. Использование <code>UVLoop</code> | 272 |
| 54.4. Примитив синхронизации: <code>Event</code> | 272 |
| 54.5. Простой <code>Websocket</code> | 273 |
| 54.6. Распространенные заблуждения, связанные с модулем <code>asyncio</code> | 274 |
| Глава 55. Модуль <code>Random</code> | 274 |
| 55.1. Создание случайного пароля пользователя | 274 |
| 55.2. Создание криптографически защищенных случайных чисел | 275 |
| 55.3. Случайности и последовательности: перемешивание, выбор и выборка | 275 |
| 55.4. Создание чисел типов <code>int</code> и <code>float</code> : <code>randint</code> , <code>randrange</code> , <code>random</code> и <code>uniform</code> | 276 |
| 55.5. Воспроизводимые случайные числа: методы <code>Seed</code> и <code>State</code> | 277 |
| 55.6. Случайное двоичное решение | 277 |
| Глава 56. Модуль <code>Functools</code> | 278 |
| 56.1. Функция <code>partial</code> | 278 |
| 56.2. <code>cmp_to_key</code> | 278 |
| 56.3. <code>@lru_cache</code> | 278 |
| 56.4. <code>@total_ordering</code> | 279 |
| 56.5. <code>reduce</code> | 279 |
| Глава 57. Модуль <code>dis</code> | 280 |
| 57.1. Что такое байт-код Python? | 280 |
| 57.2. Константы в модуле <code>dis</code> | 280 |
| 57.3. Демонтаж модулей | 280 |
| Глава 58. Модуль <code>base64</code> | 281 |
| 58.1. Кодирование и декодирование <code>Base64</code> | 282 |
| 58.2. Кодирование и декодирование <code>Base32</code> | 283 |
| 58.3. Кодирование и декодирование <code>Base16</code> | 284 |
| 58.4. Кодирование и декодирование <code>ASCII85</code> | 284 |
| 58.5. Кодирование и декодирование <code>Base85</code> | 285 |
| Глава 59. Модуль <code>Queue</code> | 285 |
| 59.1. Простой пример | 285 |
| Глава 60. Модуль <code>Deque</code> | 286 |
| 60.1. Базовое использование <code>deque</code> | 286 |
| 60.2. Доступные методы в <code>deque</code> | 286 |
| 60.3. Ограничение размера <code>deque</code> | 287 |
| 60.4. Поиск по ширине (<code>Breadth First Search</code>) | 287 |
| Глава 61. Модуль <code>Webbrowser</code> | 288 |
| 61.1. Открытие URL-адреса браузером по умолчанию | 288 |
| 61.2. Открытие URL-адреса с помощью различных браузеров | 289 |

| | |
|---|-----|
| Глава 62. tkinter..... | 289 |
| 62.1. Менеджеры геометрии | 289 |
| 62.2. Минимальное приложение tkinter | 291 |
| Глава 63. Модуль ruautogui | 291 |
| 63.1. Функции мыши..... | 291 |
| 63.2. Функции клавиатуры | 292 |
| 63.3. Распознавание скриншотов и изображений..... | 292 |
| Глава 64. Индексация и нарезка | 292 |
| 64.1. Базовая нарезка | 292 |
| 64.2. Реверсирование объекта | 293 |
| 64.3. Назначение среза..... | 293 |
| 64.4. Создание неглубокой копии массива..... | 294 |
| 64.5. Индексация пользовательских классов: <code>__getitem__</code> , <code>__setitem__</code> и <code>__delitem__</code> | 294 |
| 64.6. Базовая индексация | 295 |
| Глава 65. Построение графиков с помощью Matplotlib | 296 |
| 65.1. Графики с общей осью x, но разными осями y: использование <code>twinx()</code> | 296 |
| 65.2. Графики с общей осью y, но разными осями x: использование функции <code>twiny()</code> ... | 297 |
| 65.3. Простой график в Matplotlib | 299 |
| 65.4. Добавление дополнительных функций к простому графику: названия осей, заголовков, метки осей, сетка и легенда | 300 |
| 65.5. Создание нескольких кривых на одном графике путем наложения, аналогично MATLAB..... | 301 |
| 65.6. Создание нескольких кривых на одном графике с помощью наложения с использованием отдельных команд <code>plot</code> | 302 |
| Глава 66. graph-tool | 303 |
| 66.1. PyDotPlus..... | 303 |
| 66.2. PyGraphviz | 303 |
| Глава 67. Генераторы | 304 |
| 67.1. Введение..... | 304 |
| 67.2. Бесконечные последовательности..... | 306 |
| 67.3. Отправка объектов генератору..... | 307 |
| 67.4. Предоставление всех значений из другого итерируемого объекта..... | 308 |
| 67.5. Итерация | 308 |
| 67.6. Функция <code>next()</code> | 308 |
| 67.7. Корутины (сопрограммы)..... | 309 |
| 67.8. Рефакторинг кода построения списков | 309 |
| 67.9. Использование оператора <code>yield</code> с рекурсией: рекурсивное перечисление всех файлов в каталоге | 310 |
| 67.10. Генераторные выражения | 310 |
| 67.11. Использование генератора для получения чисел Фибоначчи | 311 |
| 67.12. Поиск | 311 |
| 67.13. Параллельный итерационный перебор генераторов | 311 |
| Глава 68. Функция <code>reduce</code> | 312 |
| 68.1. Обзор | 312 |
| 68.2. Использование <code>reduce</code> | 312 |
| 68.3. Накопительный продукт | 313 |
| 68.4. Вариант без “короткого замыкания” функций <code>any/all</code> | 313 |
| Глава 69. Функция <code>map</code> | 313 |
| 69.1. Базовое использование <code>map</code> , <code>itertools.imap</code> и <code>future_builtins.map</code> | 313 |
| 69.2. Сопоставление каждого значения в итерируемом объекте | 314 |

| | |
|--|-----|
| 69.3. Отображение (mapping) значений различных итерируемых объектов | 315 |
| 69.4. Транспонирование с помощью map: использование "None" в качестве аргумента функции (только для Python 2.x) | 316 |
| 69.5. Последовательное и параллельное отображение | 317 |
| Глава 70. Возведение в степень | 318 |
| 70.1. Возведение в степень с использованием встроенных модулей: ** и pow() | 318 |
| 70.2. Квадратный корень: math.sqrt() и cmath.sqrt | 319 |
| 70.3. Возведение в степень по модулю: функция pow() с тремя аргументами | 319 |
| 70.4. Вычисление больших целых корней | 320 |
| 70.5. Возведение в степень с использованием модуля math: math.pow() | 321 |
| 70.6. Экспоненциальная функция: math.exp() и cmath.exp() | 321 |
| 70.7. Экспоненциальная функция -1: math.expm1() | 321 |
| 70.8. Магические методы и возведение в степень: builtin, math и cmath | 322 |
| 70.9. Корень n-й степени с дробным показателем степени | 323 |
| Глава 71. Поиск | 324 |
| 71.1. Поиск элемента | 324 |
| 71.2. Поиск в пользовательских классах: __contains__ и __iter__ | 324 |
| 71.3. Получение индекса для строк: str.index(), str.rindex() и str.find(), str.rfind() | 325 |
| 71.4. Получение индексов в списках и кортежах: list.index(), tuple.index() | 325 |
| 71.5. Поиск ключа или ключей для значения в словаре | 326 |
| 71.6. Получение индекса в отсортированных последовательностях: bisect.bisect_left() | 326 |
| 71.7. Поиск во вложенных последовательностях | 327 |
| Глава 72. Сортировка, минимум и максимум | 328 |
| 72.1. Упорядоченность в пользовательских классах | 328 |
| 72.2. Особый случай: словари | 330 |
| 72.3. Использование ключевого аргумента | 331 |
| 72.4. Аргумент по умолчанию для max, min | 331 |
| 72.5. Получение отсортированной последовательности | 331 |
| 72.6. Извлечение N наибольших или N наименьших элементов из итерируемого объекта | 332 |
| 72.7. Получение минимального или максимального из нескольких значений | 332 |
| 72.8. Минимум и максимум последовательности | 333 |
| Глава 73. Подсчет | 333 |
| 73.1. Подсчет всех вхождений всех элементов в итерируемом объекте: collections.Counter | 333 |
| 73.2. Получение наиболее часто встречающегося значения: collections.Counter.most_common() | 334 |
| 73.3. Подсчет количества вхождений элемента в последовательность: list.count() и tuple.count() | 334 |
| 73.4. Подсчет вхождений подстроки в строку: str.count() | 334 |
| 73.5. Подсчет вхождений в numpy-массиве | 335 |
| Глава 74. Функция print | 335 |
| 74.1. Основы вывода на экран | 335 |
| 74.2. Параметры функции print | 336 |
| Глава 75. Регулярные выражения (Regex) | 337 |
| 75.1. Сопоставление начала строки | 337 |
| 75.2. Поиск | 338 |
| 75.3. Предварительно скомпилированные шаблоны (precompiled patterns) | 339 |
| 75.4. Флаги | 339 |
| 75.5. Замена | 340 |

| | |
|--|-----|
| 75.6. Поиск всех непересекающихся совпадений..... | 340 |
| 75.7. Проверка на разрешенные символы..... | 341 |
| 75.8. Разбиение строки с помощью регулярных выражений..... | 341 |
| 75.9. Группировка | 341 |
| 75.10. Исключение специальных символов..... | 342 |
| 75.11. Сопоставление выражения только в определенных местах..... | 343 |
| 75.12. Итерация по совпадениям с помощью re.finditer..... | 343 |
| Глава 76. Копирование данных..... | 344 |
| 76.1. Копирование словаря..... | 344 |
| 76.2. Неглубокое копирование | 344 |
| 76.3. Глубокое копирование..... | 344 |
| 76.4. Неглубокое копирование списка..... | 345 |
| 76.5. Копирование набора..... | 345 |
| Глава 77. Менеджеры контекста (инструкция with) | 345 |
| 77.1. Введение в менеджеры контекста и инструкцию with..... | 345 |
| 77.2. Написание собственного менеджера контекста..... | 346 |
| 77.3. Написание собственного менеджера контекста с использованием синтаксиса генератора | 347 |
| 77.4. Использование нескольких менеджеров контекста | 348 |
| 77.5. Назначение цели | 348 |
| 77.6. Управление ресурсами..... | 348 |
| Глава 78. Специальная переменная __name__ | 349 |
| 78.1. Проверка __name__ == '__main__'..... | 349 |
| 78.2. Использование в протоколировании | 349 |
| 78.3. function_class_or_module.__name__..... | 349 |
| Глава 79. Проверка существования пути и прав доступа | 351 |
| 79.1. Выполнение проверки с помощью os.access | 351 |
| Глава 80. Создание пакетов Python..... | 351 |
| 80.1. Введение | 351 |
| 80.2. Загрузка в каталог пакетов PyPI | 352 |
| 80.3. Создание исполняемого пакета | 354 |
| Глава 81. Использование модуля “pip”: Менеджер пакетов PyPI..... | 354 |
| 81.1. Пример использования команды | 354 |
| 81.2. Обработка исключения ImportError..... | 355 |
| 81.3. Принудительная установка | 355 |
| Глава 82. pip: менеджер пакетов PyPI..... | 356 |
| 82.1. Установка пакетов | 356 |
| 82.2. Получение списка пакетов, установленных с помощью `pip` | 356 |
| 82.3. Обновление пакетов | 356 |
| 82.4. Удаление пакетов..... | 357 |
| 82.5. Обновление всех устаревших пакетов в Linux | 357 |
| 82.6. Обновление всех устаревших пакетов под Windows | 357 |
| 82.7. Создание файла requirements.txt file всех пакетов в системе | 357 |
| 82.8. Использование определенной версии Python с помощью pip | 357 |
| 82.9. Создание файла requirements.txt file пакетов только текущей virtualenv | 358 |
| 82.10. Установка пакетов, еще не включенных в pip в качестве wheels..... | 358 |
| Глава 83. Разбор аргументов командной строки | 361 |
| 83.1. Hello world в argparse | 361 |
| 83.2. Использование аргументов командной строки с помощью argv | 362 |

| | |
|--|-----|
| 83.3. Задание взаимоисключающих аргументов с помощью <code>argparse</code> | 363 |
| 83.4. Базовый пример с использованием <code>docopt</code> | 363 |
| 83.5. Пользовательское сообщение об ошибке синтаксического анализатора с помощью <code>argparse</code> | 364 |
| 83.6. Концептуальная группировка аргументов с помощью <code>argparse.add_argument_group()</code> | 364 |
| 83.7. Расширенный пример с <code>docopt</code> и <code>docopt_dispatch</code> | 365 |
| Глава 84. Библиотека подпроцессов | 366 |
| 84.1. Больше возможностей с помощью <code>Popen</code> | 366 |
| 84.2. Вызов внешних команд | 367 |
| 84.3. Как создать аргумент списка команд | 368 |
| Глава 85. Файл <code>setup.py</code> | 368 |
| 85.1. Назначение файла <code>setup.py</code> | 368 |
| 85.2. Использование метаданных системы управления версиями в файле <code>setup.py</code> | 369 |
| 85.3. Добавление скриптов командной строки в пакет Python | 369 |
| 85.4. Добавление параметров установки | 370 |
| Глава 86. Рекурсия | 370 |
| 86.1. Что, как и когда делать с рекурсией | 370 |
| 86.2. Задача “исследования дерева” с помощью рекурсии | 373 |
| 86.3. Сумма чисел от 1 до n | 374 |
| 86.4. Увеличение максимальной глубины рекурсии | 375 |
| 86.5. Хвостовая рекурсия – плохая практика | 375 |
| 86.6. Оптимизация хвостовой рекурсии с помощью интроспекции стека | 376 |
| Глава 87. Подсказки типов | 377 |
| 87.1. Добавление типов в функцию | 377 |
| 87.2. Функция <code>NamedTuple</code> | 378 |
| 87.3. Общие типы | 378 |
| 87.4. Переменные и атрибуты | 378 |
| 87.5. Члены и методы классов | 379 |
| 87.6. Подсказки типов для именованных аргументов | 379 |
| Глава 88. Исключения | 379 |
| 88.1. Перехват исключений | 379 |
| 88.2. Не перехватывайте все подряд! | 380 |
| 88.3. Повторный вызов исключений | 380 |
| 88.4. Перехват множественных исключений | 381 |
| 88.5. Иерархия исключений | 381 |
| 88.6. <code>Else</code> | 383 |
| 88.7. Вызов исключений | 384 |
| 88.8. Создание пользовательских типов исключений | 384 |
| 88.9. Практические примеры обработки исключений | 384 |
| 88.10. Исключения – это тоже объекты | 385 |
| 88.11. Выполнение кода очистки с помощью <code>finally</code> | 386 |
| 88.12. Создание цепочек исключений с использованием <code>raise from</code> | 386 |
| Глава 89. Вызов пользовательских ошибок и исключений | 387 |
| 89.1. Пользовательское исключение | 387 |
| 89.2. Перехват пользовательского исключения | 387 |
| Глава 90. Исключения сообщества | 387 |
| 90.1. Прочие ошибки | 387 |
| 90.2. Ошибка “ <code>NameError: name '??' is not defined</code> ” | 388 |
| 90.3. Ошибки типов данных | 389 |

| | |
|---|-----|
| 90.4. Синтаксическая ошибка в хорошем коде | 391 |
| 90.5. Ошибки отступа (или ошибки синтаксиса отступов) | 391 |
| Глава 91. urllib | 392 |
| 91.1. HTTP GET | 392 |
| 91.2. HTTP POST | 393 |
| 91.3. Декодирование полученных байтов в соответствии с кодировкой типа содержимого | 394 |
| Глава 92. Веб-скрапинг с помощью Python | 394 |
| 92.1. Скрапинг с использованием фреймворка Scrapy | 394 |
| 92.2. Скрапинг с использованием Selenium WebDriver | 395 |
| 92.3. Базовый пример использования запросов и lxml для скрапинга некоторых данных | 395 |
| 92.4. Поддержание сессии веб-скрапинга с помощью запросов | 396 |
| 92.5. Скрапинг с использованием BeautifulSoup4 | 396 |
| 92.6. Простое скачивание веб-контента с помощью urllib.request | 396 |
| 92.7. Модификация пользовательского агента Scrapy | 397 |
| 92.8. Скрапинг с использованием инструмента командной строки cURL | 397 |
| Глава 93. Парсинг HTML | 397 |
| 93.1. Использование CSS-селекторов в библиотеке BeautifulSoup | 397 |
| 93.2. Библиотека PyQuery | 398 |
| 93.3. Нахождение текста после элемента в библиотеке BeautifulSoup | 399 |
| Глава 94. Работа с XML | 399 |
| 94.1. Открытие и чтение с помощью библиотеки ElementTree | 399 |
| 94.2. Создание и построение XML-документов | 399 |
| 94.3. Изменение XML-файла | 400 |
| 94.4. Поиск в XML с помощью XPath | 400 |
| 94.5. Открытие и чтение больших XML-файлов с помощью инкрементного парсинга | 401 |
| Глава 95. Python Requests Post | 402 |
| 95.1. Простая операция Post | 402 |
| 95.2. Данные, закодированные в форме | 403 |
| 95.3. Загрузка файлов | 403 |
| 95.4. Ответы | 403 |
| 95.5. Аутентификация | 404 |
| 95.6. Прокси-серверы | 405 |
| Глава 96. Распространение кода | 405 |
| 96.1. py2app | 405 |
| 96.2. cx_Freeze | 406 |
| Глава 97. Объекты свойств | 407 |
| 97.1. Использование декоратора @property для свойств чтения и записи | 407 |
| 97.2. Еще об использовании декоратора @property | 407 |
| 97.3. Переопределение только функций getter, setter или deleter объекта свойства | 408 |
| 97.4. Использование свойств без декораторов | 408 |
| Глава 98. Перегрузка | 410 |
| 98.1. Перегрузка операторов | 410 |
| 98.2. "Магические" методы, или Dunder-методы | 412 |
| 98.3. Типы контейнеров и последовательностей | 413 |
| 98.4. Вызываемые типы | 413 |
| 98.5. Обработка нереализованного поведения | 414 |