



Python К вершинам мастерства

Лучано Рамальо

Лучано Рамальо

Python.

К вершинам мастерства

Fluent Python

Luciano Ramalho

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'REILLY®

Python. К вершинам мастерства

Лучано Рамальо

Москва, 2016



УДК 004.438Python:004.6
ББК 32.973.22
P21

P21 Лучано Рамальо

Python. К вершинам мастерства / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2016. – 768 с.: ил.

ISBN 978-5-97060-384-0

Язык Python настолько прост, что научиться продуктивно писать на нем программы можно быстро, но зачастую вы при этом используете не все имеющиеся в нем возможности. Данная книга покажет, как создавать эффективный идиоматичный код на Python, задействуя его лучшие – и иногда несправедливо игнорируемые – черты. Автор, Лучано Рамальо, рассказывает о базовых средствах и библиотеках Python и демонстрирует, как сделать код одновременно короче, быстрее и понятнее. Многие опытные программисты стараются подогнать Python под приемы, знакомые им по работе с другими языками. Эта книга покажет, как достичь истинного профессионализма в программировании на Python 3.

Издание предназначено для программистов, уже работающих на Python, но также может быть полезно и начинающим пользователям языка.

УДК 004.438Python:004.6
ББК 32.973.22

Original English language edition published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. Copyright © 2015 O'Reilly Media, Inc. Russian-language edition copyright © 2015 by DMK Press. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но, поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 978-1-491-94600-8 (англ.)
ISBN 978-5-97060-384-0 (рус.)

© Luciano Gama de Sousa Ramalho, 2015.
© Оформление, перевод на русский язык,
издание, ДМК Пресс, 2016

Para Marta, com todo o meu amor.



ОГЛАВЛЕНИЕ

Предисловие	17
На кого рассчитана эта книга	18
На кого эта книга не рассчитана	18
Как организована эта книга	18
Практикум	20
Как производился хронометраж	21
Поговорим: мое личное мнение	21
Терминология Python	22
Использованная версия Python	22
Графические выделения	22
О примерах кода	23
Как с нами связаться	23
Благодарности	24
ЧАСТЬ I. Пролог	27
Глава 1. Модель данных в языке Python	28
Колода карт на Python	29
Как используются специальные методы	33
Эмуляция числовых типов	33
Строковое представление	35
Арифметические операторы	36
Булево значение пользовательского типа	37
Сводка специальных методов	37
Почему len – не метод	39
Резюме	40
Дополнительная литература	40
ЧАСТЬ II. Структуры данных	43
Глава 2. Массив последовательностей	44
Общие сведения о встроенных последовательностях	45
Списковое включение и генераторные выражения	46
Списковое включение и удобочитаемость	46
Сравнение спискового включения с map и filter	48
Декартовы произведения	49

Генераторные выражения.....	50
Кортеж – не просто неизменяемый список	52
Кортежи как записи	52
Распаковка кортежа	53
Использование * для выборки лишних элементов	54
Распаковка вложенного кортежа	55
Именованные кортежи	56
Кортежи как неизменяемые списки	57
Получение среза	59
Почему в срезы и диапазоны не включается последний элемент	59
Объекты среза	59
Многомерные срезы и многоточие	61
Присваивание срезу	61
Использование + и * для последовательностей	62
Построение списка списков	63
Составное присваивание последовательностей	64
Головоломка: присваивание A +=	66
Метод list.sort и встроенная функция sorted	68
Средства работы с упорядоченными последовательностями в модуле bisect	70
Поиск средствами bisect	70
Вставка с помощью функции bisect.insort	73
Когда список не подходит	74
Массивы	74
Представления областей памяти	78
Библиотеки NumPy и SciPy	79
Двусторонние и другие очереди	81
Резюме	85
Дополнительная литература	86
Глава 3. Словари и множества	91
Общие типы отображений	91
Словарное включение	94
Обзор наиболее употребительных методов отображений	94
Обработка отсутствия ключей с помощью.setdefault	97
Отображения с гибким поиском по ключу	99
defaultdict: еще один подход к обработке отсутствия ключа	99
Метод __missing__	101
Вариации на тему dict	103
Создание подкласса UserDict	105
Неизменяемые отображения	106
Теория множеств	108
Литеральные множества	109
Множественное включение	111

Операции над множествами.....	111
Под капотом dict и set	114
Экспериментальная демонстрация производительности	115
Хэш-таблицы в словарях	117
Практические последствия механизма работы dict	120
Как работают множества – практические следствия.....	123
Резюме.....	123
Дополнительная литература	124
Поговорим.....	124
Глава 4. Текст и байты.....	126
О символах и не только	127
Все, что нужно знать о байтах	128
Структуры и представления областей памяти	131
Базовые кодировщики и декодировщики.....	132
Проблемы кодирования и декодирования.....	134
Обработка UnicodeEncodeError	134
Обработка UnicodeDecodeError	135
Исключение SyntaxError при загрузке модулей и неожиданной кодировкой	136
Как определить кодировку последовательности байтов	138
ВОМ: полезный крокозябр	139
Обработка текстовых файлов.....	140
Кодировки по умолчанию: сумасшедший дом	143
Нормализация Unicode для правильного сравнения	146
Сворачивание регистра	149
Служебные функции для сравнения нормализованного текста.....	150
Экстремальная «нормализация»: удаление диакритических знаков	151
Сортировка Unicode-текстов	154
Сортировка с помощью алгоритма упорядочивания Unicode.....	156
База данных Unicode.....	157
Двухрежимный API.....	159
str и bytes в регулярных выражениях	159
str и bytes в функциях из модуля os.....	160
Резюме	162
Дополнительная литература	164
Поговорим.....	166
ЧАСТЬ III. Функции как объекты.....	169
Глава 5. Полноправные функции	170
Обращение с функцией как с объектом.....	171
Функции высшего порядка.....	172
Современные альтернативы функциям map, filter и reduce	173

Анонимные функции	175
Семь видов вызываемых объектов.....	176
Пользовательские вызываемые типы.....	177
Интроспекция функций.....	178
От позиционных к чисто именованным параметрам.....	180
Получение информации о параметрах.....	182
Аннотации функций	186
Пакеты для функционального программирования	188
Модуль operator	188
Фиксация аргументов с помощью functools.partial.....	191
Резюме.....	193
Дополнительная литература	194
Поговорим.....	195

Глава 6. Реализация паттернов проектирования с помощью полноправных функций 198

Практический пример: переработка паттерна Стратегия	199
Классическая Стратегия	199
Функционально-ориентированная стратегия.....	203
Выбор наилучшей стратегии: простой подход.....	206
Поиск стратегий в модуле	207
Паттерн Команда	208
Резюме.....	210
Дополнительная литература	211
Поговорим.....	212

Глава 7. Декораторы функций и замыкания 214

Краткое введение в декораторы	215
Когда Python выполняет декораторы.....	216
Паттерн Стратегия, дополненный декоратором.....	218
Правила видимости переменных	219
Замыкания.....	222
Объявление nonlocal.....	225
Реализация простого декоратора	227
Как это работает	228
Декораторы в стандартной библиотеке	230
Кэширование с помощью functools.lru_cache	230
Одиночная диспетчеризация и обобщенные функции	233
Композиции декораторов	236
Параметризованные декораторы.....	236
Параметризованный регистрационный декоратор.....	237
Параметризованный декоратор clock.....	239

Резюме	242
Дополнительная литература	242
Поговорим	243

ЧАСТЬ IV. Объектно-ориентированные идиомы..... 247

Глава 8. Ссылки на объекты, изменяемость и повторное использование..... 248

Переменные – не ящики	249
Тождественность, равенство и синонимы	250
Выбор между == и is	252
Относительная неизменяемость кортежей.....	253
По умолчанию копирование поверхностное.....	254
Глубокое и поверхностное копирование произвольных объектов	256
Параметры функций как ссылки	258
Значения по умолчанию изменяемого типа: неудачная мысль.....	259
Защитное программирование при наличии изменяемых параметров	261
del и сборка мусора	263
Слабые ссылки	265
Коллекция WeakValueDictionary	266
Ограничения слабых ссылок.....	268
Как Python хитрит с неизменяемыми объектами	269
Резюме	270
Дополнительная литература	271
Поговорим.....	272

Глава 9. Объект в духе Python 276

Представления объекта	277
И снова класс вектора	277
Альтернативный конструктор	280
Декораторы classmethod и staticmethod.....	281
Форматирование при выводе	282
Хэшируемый класс Vector2d	286
Закрытые и «защищенные» атрибуты в Python	291
Экономия памяти с помощью атрибута класса __slots__	293
Проблемы при использовании __slots__	296
Переопределение атрибутов класса	296
Резюме.....	299
Дополнительная литература	300
Поговорим.....	301

Глава 10. Рубим, перемешиваем и нарезаем последовательности 305

Vector: пользовательский тип последовательности.....	306
Vector, попытка № 1: совместимость с Vector2d	306
Протоколы и динамическая типизация.....	309
Vector, попытка № 2: последовательность, допускающая срезку	310
Как работает срезка	311
Метод <code>__getitem__</code> с учетом срезов	313
Vector, попытка № 3: доступ к динамическим атрибутам	315
Vector, попытка № 4: хэширование и ускорение оператора <code>==</code>	319
Vector, попытка № 5:	
форматирование	324
Резюме.....	331
Дополнительная литература	332
Поговорим.....	333

Глава 11. Интерфейсы: от протоколов до абстрактных

базовых классов..... 338

Интерфейсы и протоколы в культуре Python.....	339
Python в поисках следов последовательностей.....	341
Партизанское латание как средство реализации протокола во время выполнения.....	343
Алекс Мартелли о водоплавающих	345
Создание подкласса ABC.....	350
ABC в стандартной библиотеке.....	352
ABC в модуле <code>collections.abc</code>	352
Числовая башня ABC.....	354
Определение и использование ABC.....	355
Синтаксические детали ABC.....	359
Создание подклассов ABC <code>Tombola</code>	360
Виртуальный подкласс <code>Tombola</code>	363
Как тестировались подклассы <code>Tombola</code>	365
Использование метода <code>register</code> на практике	368
Гуси могут вести себя как утки	369
Резюме.....	371
Дополнительная литература	373
Поговорим.....	374

Глава 12. Наследование: хорошо или плохо 380

Сложности наследования встроенным типам	380
Множественное наследование и порядок разрешения методов	384
Множественное наследование в реальном мире	388
Жизнь с множественным наследованием	391
Tkinter: хороший, плохой, злой	393

Современный пример: примеси в обобщенных представлениях	
Django	395
Резюме.....	398
Дополнительная литература	399
Поговорим.....	400

Глава 13. Перегрузка операторов: как правильно? 403

Основы перегрузки операторов	404
Унарные операторы	404
Перегрузка оператора сложения векторов +	407
Перегрузка оператора умножения на скаляр *	412
Операторы сравнения	416
Операторы составного присваивания	420
Резюме	425
Дополнительная литература	426
Поговорим.....	427

ЧАСТЬ V. Поток управления 431

Глава 14. Итерируемые объекты, итераторы и генераторы .. 432

Класс Sentence, попытка № 1: последовательность слов	433
Почему последовательности итерируемы: функция iter.....	435
Итерируемые объекты и итераторы	436
Класс Sentence, попытка № 2: классический вариант	440
Почему идея сделать Sentence итератором плоха	442
Класс Sentence, попытка № 3: генераторная функция.....	443
Как работает генераторная функция	444
Класс Sentence, попытка № 4: ленивая реализация	447
Класс Sentence, попытка № 5: генераторное выражение	448
Генераторные выражения: когда использовать	450
Другой пример: генератор арифметической прогрессии.....	451
Построение арифметической прогрессии с помощью itertools	453
Генераторные функции в стандартной библиотеке.....	454
yield from – новая конструкция в Python 3.3	465
Функции редуцирования итерируемого объекта	466
Более пристальный взгляд на функцию iter	468
Пример: генераторы в утилите преобразования базы данных.....	469
Генераторы как сопрограммы	471
Резюме	472
Дополнительная литература	472
Поговорим.....	473

Глава 15. Контекстные менеджеры и блоки else	479
Делай то, потом это: блоки else вне if	480
Контекстные менеджеры и блоки with	482
Утилиты contextlib	486
Использование @contextmanager	487
Резюме	490
Дополнительная литература	491
Поговорим	492
Глава 16. Сопрограммы	494
Эволюция: от генераторов к сопрограммам	495
Базовое поведение генератора, используемого в качестве сопрограммы	496
Пример: сопрограмма для вычисления накопительного среднего	499
Декораторы для инициализации сопрограмм	501
Завершение сопрограммы и обработка исключений	502
Возврат значения из сопрограммы	506
Использование yield from	508
Семантика yield from	514
Пример: применение сопрограмм для моделирования дискретных событий	520
О моделировании дискретных событий	521
Моделирование работы таксопарка	522
Резюме	529
Дополнительная литература	531
Поговорим	533
Глава 17. Параллелизм и будущие объекты	536
Пример: три способа загрузки из веба	536
Скрипт последовательной загрузки	538
Загрузка с применением библиотеки concurrent.futures	540
Где находятся будущие объекты?	542
Блокирующий ввод-вывод и GiL	545
Запуск процессов с помощью concurrent.futures	546
Эксперименты с Executor.map	548
Загрузка с индикацией хода выполнения и обработкой ошибок	551
Обработка ошибок во flags2-примерах	556
Использование futures.as_completed	558
Альтернативы: многопоточная и многопроцессная обработка	561
Резюме	561
Дополнительная литература	562
Поговорим	564

Глава 18. Применение пакета <code>asyncio</code> для организации конкурентной работы	567
Сравнение потока и сопрограммы	569
<code>asyncio.Future</code> : не блокирует умышленно	575
<code>Yield from</code> из будущих объектов, задач и сопрограмм	576
Загрузка с применением <code>asyncio</code> и <code>aiohttp</code>	578
Объезд блокирующих вызовов	582
Улучшение скрипта загрузки на основе <code>asyncio</code>	585
Использование <code>asyncio.as_completed</code>	585
Использование исполнителя для предотвращения блокировки цикла обработки событий	591
От обратных вызовов к будущим объектам и сопрограммам	592
Выполнение нескольких запросов для каждой операции загрузки	595
Разработка серверов с помощью пакета <code>asyncio</code>	597
TCP-сервер на основе <code>asyncio</code>	598
Веб-сервер на основе библиотеки <code>aiohttp</code>	602
Повышение степени параллелизма за счет более интеллектуальных клиентов	606
Резюме	607
Дополнительная литература	608
Поговорим	610
ЧАСТЬ VI. Метапрограммирование	613
Глава 19. Динамические атрибуты и свойства	614
Применение динамических атрибутов для обработки данных	615
Исследование JSON-подобных данных с динамическими атрибутами	617
Проблема недопустимого имени атрибута	620
Гибкое создание объектов с помощью метода <code>__new__</code>	622
Изменение структуры набора данных OSCON с помощью модуля <code>shelve</code>	624
Выборка связанных записей с помощью свойств	627
Использование свойств для контроля атрибутов	633
<code>Linelltem</code> , попытка № 1: класс строки заказа	633
<code>Linelltem</code> , попытка № 2: контролирующее свойство	634
Правильный взгляд на свойства	636
Свойства переопределяют атрибуты экземпляра	637
Документирование свойств	639
Программирование фабрики свойств	640
Удаление атрибутов	643
Важные атрибуты и функции для работы с атрибутами	644
Специальные атрибуты, влияющие на обработку атрибутов	645
Встроенные функции для работы с атрибутами	645
Специальные методы для работы с атрибутами	646
Резюме	648

Дополнительная литература	648
Поговорим	649
Глава 20. Дескрипторы атрибутов	653
Пример дескриптора: проверка значений атрибутов	653
Lineltm попытка № 3: простой дескриптор	654
Lineltm попытка № 4: автоматическая генерация имен атрибутов хранения	659
Lineltm попытка № 5: новый тип дескриптора	665
Переопределяющие и непереопределяющие дескрипторы	668
Переопределяющий дескриптор	669
Переопределяющий дескриптор без <code>__get__</code>	670
Непереопределяющий дескриптор	671
Перезаписывание дескриптора в классе	673
Методы являются дескрипторами	673
Советы по использованию дескрипторов	676
Строка документации дескриптора и перехват удаления	677
Резюме	678
Дополнительная литература	679
Поговорим	680
Глава 21. Метапрограммирование классов	682
Фабрика классов	683
Декоратор класса для настройки дескрипторов	686
Что когда происходит: этап импорта и этап выполнения	688
Демонстрация работы интерпретатора	689
Основы метаклассов	693
Демонстрация работы метакласса	695
Метакласс для настройки дескрипторов	699
Специальный метод метакласса <code>__prepare__</code>	701
Классы как объекты	703
Резюме	704
Дополнительная литература	705
Поговорим	707
Послесловие	709
Дополнительная литература	710
Приложение А. Основы языка Python	713
Глава 3: тест производительности оператора <code>in</code>	713
Глава 3: сравнение битовых представлений хэшей	715
Глава 9. Потребление оперативной памяти при наличии и отсутствии <code>__slots__</code>	716

Глава 14: скрипт преобразования базы данных isis2json.py	717
Глава 16: моделирование дискретных событий таксопарка	722
Глава 17: примеры, относящиеся к криптографии	726
Глава 17: примеры HTTP-клиентов из серии flags2	729
Глава 19: скрипты и тесты для обработки набора данных OSCON	734
Терминология Python	739
Предметный указатель	754



ПРЕДИСЛОВИЕ

План такой: если кто-то пользуется средством, которое вы не понимаете, просто пристрелите его. Это проще, чем учить что-то новое, и очень скоро в мире останутся только кодировщики, которые используют только всем понятное крохотное подмножество Python 0.9.6 <смешок>.¹

– Тим Питерс,
легендарный разработчик ядра и автор сборника поучений «The Zen of Python»

«Python – простой для изучения и мощный язык программирования». Это первые слова в официальном «Пособии по Python» (<https://docs.python.org/3/tutorial/>). И это правда, но не вся правда: поскольку язык так просто выучить и начать применять на деле, многие практикующие программисты используют лишь малую часть его обширных возможностей.

Опытный программист может написать полезный код на Python уже через несколько часов изучения. Но вот проходят недели, месяцы – и многие разработчики так и продолжают писать на Python код, в котором отчетливо видно влияние языков, которые они учили раньше. И даже если Python – ваш первый язык, все равно авторы академических и вводных учебников зачастую излагают его, тщательно избегая особенностей, характерных только для этого языка.

Будучи преподавателем, который знакомит с Python программистов, знающих другие языки, я нередко сталкиваюсь еще с одной проблемой, которую пытаюсь решить в этой книге: нас интересует только то, о чем мы уже знаем. Любой программист, знакомый с каким-то другим языком, догадывается, что Python поддерживает регулярные выражения, и начинает смотреть, что про них написано в документации. Но если вы никогда раньше не слыхали о распаковке кортежей или о дескрипторах, то, скорее всего, и искать сведения о них не станете, а в результате не будете использовать эти средства лишь потому, что они специфичны для Python.

Эта книга не является полным справочным руководством по Python. Упор в ней сделан на языковые средства, которые либо уникальны для Python, либо отсутствуют во многих других популярных языках. Кроме того, в книге рассматривается в основном ядро языка и немногие библиотеки. Я редко упоминаю о паке-

¹ Сообщение в группе Usenet comp.lang.python от 23 декабря 2002: «Acrimony in c.l.p.» (<https://mail.python.org/pipermail/python-list/2002-December/147293.html>).

тах, не включенных в стандартную библиотеку, хотя нынче количество пакетов для Python уже перевалило за 60 000, и многие из них исключительно полезны.

На кого рассчитана эта книга

Эта книга написана для практикующих программистов на Python, которые хотят усовершенствоваться в Python 3. Если вы уже знакомы с Python и хотели бы перейти на версию Python 3.4 или старше, эта книга для вас. Когда я писал ее, большинство профессиональных программистов работали с Python 2, поэтому я специально выделял особенности Python 3, которые для этой аудитории могли оказаться внове.

Однако поскольку книга посвящена, главным образом, тому, как получить максимум от Python 3.4, я не останавливаюсь на исправлениях, которые нужно внести в старый код, чтобы он продолжал работать. Большинство примеров будут работать в Python 2.7 с минимальными изменениями или вообще без оных, но иногда обратный перенос требует значительных усилий.

И все же я полагаю, что эта книга может быть полезна и тем, кто вынужден продолжать писать на Python 2.7, поскольку базовые концепции остались теми же самыми. Python 3 – не новый язык, и большинство различий можно изучить за полдня. Желаящие узнать, что нового появилось в Python 3.0, могут начать со страницы <https://docs.python.org/3.0/whatsnew/3.0.html>. Разумеется, с момента выхода версии 3.0 в 2009 году Python не стоял на месте, но все последующие изменения не так существенны, как внесенные в 3.0.

Если вы не уверены в том, достаточно ли хорошо знаете Python, чтобы читать эту книгу, загляните в оглавление официального «Пособия по Python» (<https://docs.python.org/3/tutorial/>). Темы, рассмотренные в пособии, в этой книге не затрагиваются, за исключением некоторых новых средств, появившихся в Python 3.

На кого эта книга не рассчитана

Если вы только начинаете изучать Python, эта книга покажется вам сложноватой. Более того, если вы откроете ее на слишком раннем этапе путешествия в мир Python, то может сложиться впечатление, будто в каждом Python-скрипте следует использовать специальные методы и приемы метапрограммирования. Преждевременное абстрагирование ничем не лучше преждевременной оптимизации.

Как организована эта книга

Читатели, на которых рассчитана эта книга, без труда смогут начать чтение с любой главы. Но каждая из шести частей образует книгу в книге. Я предполагал, что главы, составляющие одну часть, будут читаться по порядку.

Я старался сначала рассказывать о том, что уже есть, а лишь затем – о том, как создавать что-то свое. Например, в главе 2 из части II рассматриваются готовые типы последовательностей, в том числе не слишком хорошо известные, например

`collections.deque`. О создании пользовательских последовательностей речь пойдет только в части IV, где мы также узнаем об использовании абстрактных базовых классов (`abstract base classes` – ABC) из модуля `collections.abc`. Создание собственного ABC обсуждается еще позже, поскольку я считаю, что сначала нужно освоиться с использованием ABC, а уж потом писать свои.

У такого подхода несколько достоинств. Прежде всего, зная, что есть в вашем распоряжении, вы не станете заново изобретать велосипед. Мы пользуемся готовыми классами коллекций чаще, чем реализуем собственные, и можем уделить больше внимания нетривиальным способам работы с имеющимися средствами, отложив на потом разговор о разработке новых. И мы скорее унаследуем существующему абстрактному базовому классу, чем будем создавать новый с нуля. Наконец, я полагаю, что понять абстракцию проще после того, как видел ее в действии.

Недостаток же такой стратегии в том, что главы изобилуют ссылками на более поздние материалы. Надеюсь, узнав, почему я выбрал такой путь, вам будет проще с этим смириться.

Ниже описаны основные темы, рассматриваемые в каждой части книги.

Часть I

Содержит всего одну главу, посвященную модели данных в Python, где объясняется ключевая роль специальных методов (например, `__repr__`) для обеспечения единообразного поведения объектов любого типа – в языке, заслуженно считающемся образцом единообразия. Осмысление различных граней модели данных – сквозная тема книги, но именно в главе 1 дается общий обзор.

Часть II

В главах из этой части рассматриваются типы коллекций: последовательности, отображения и множества, а также сравниваются типы `str` и `bytes`. Это вещи, которые радостно приветствовали пользователи Python 3 и которых отчаянно не хватает пользователям Python 2, еще не модернизировавшим свой код. Основная цель – напомнить, что уже имеется, и объяснить некоторые особенности поведения, которые могут оказаться неожиданными, например, изменение порядка ключей словаря `dict` в то время, когда в нем никто ничего не ищет, или подводные камни, связанные с зависящей от локали сортировкой строки `Unicode`. Во имя достижения этой цели изложение временами становится широким и высокоуровневым (например, во время знакомства с многочисленными типами последовательностей и отображений), а временами – углубленным (например, при описании деталей хэш-таблиц, лежащих в основе типов `dict` и `set`).

Часть III

Здесь речь пойдет о функциях, как полноправных объектах языка: что под этим понимается, как это отражается на некоторых популярных паттернах

КОНЕЦ ОЗНАКОМИТЕЛЬНОГО
ФРАГМЕНТА.

"

"

www.e-univers.ru