

Марте, с любовью

Оглавление

Предисловие от издательства	19
Отзывы и пожелания.....	19
Список опечаток	19
Нарушение авторских прав	19
Об авторе	20
Колофон	20
Предисловие.....	21
На кого рассчитана эта книга	21
На кого эта книга не рассчитана	22
Пять книг в одной	22
Как организована эта книга.....	22
Практикум.....	24
Поговорим: мое личное мнение.....	25
Сопроводительный сайт: fluentpython.com	25
Графические выделения	25
О примерах кода	26
Как с нами связаться	26
Благодарности	27
Благодарности к первому изданию.....	28
ЧАСТЬ I. СТРУКТУРЫ ДАННЫХ	31
Глава 1. Модель данных в языке Python	32
Что нового в этой главе	33
Колода карт на Python	33
Как используются специальные методы	36
Эмуляция числовых типов	37
Строковое представление	40
Булево значение пользовательского типа	41
API коллекций.....	41
Сводка специальных методов	43
Почему len – не метод	45
Резюме.....	45
Дополнительная литература.....	46

Глава 2. Массив последовательностей	48
Что нового в этой главе	49
Общие сведения о встроенных последовательностях	49
Списковое включение и генераторные выражения	51
Списковое включение и удобочитаемость	52
Сравнение спискового включения с <code>map</code> и <code>filter</code>	53
Декартовы произведения	54
Генераторные выражения	55
Кортеж – не просто неизменяемый список	57
Кортежи как записи	57
Кортежи как неизменяемые списки	58
Сравнение методов кортежа и списка	60
Распаковка последовательностей и итерируемых объектов	61
Распаковка с помощью * в вызовах функций и литеральных последовательностях	63
Распаковка вложенных объектов	63
Сопоставление с последовательностями-образцами	64
Сопоставление с последовательностями-образцами в интерпретаторе	69
Получение среза	72
Почему в срезы и диапазоны не включается последний элемент	73
Объекты среза	73
Многомерные срезы и многоточие	74
Присваивание срезу	75
Использование + и * для последовательностей	76
Построение списка списков	76
Составное присваивание последовательностей	78
Головоломка: присваивание <code>A +=</code>	79
Метод <code>list.sort</code> и встроенная функция <code>sorted</code>	81
Когда список не подходит	83
Массивы	83
Представления областей памяти	86
<code>NumPy</code>	88
Двусторонние и другие очереди	90
Резюме	93
Дополнительная литература	94
Глава 3. Словари и множества	99
Что нового в этой главе	99
Современный синтаксис словарей	100
Словарные включения	100
Распаковка отображений	101
Объединение отображений оператором <code> </code>	102
Сопоставление с отображением-образцом	102
Стандартный API типов отображений	105
Что значит «хешируемый»?	105
Обзор наиболее употребительных методов отображений	106

Вставка и обновление изменяемых значений	108
Автоматическая обработка отсутствующих ключей	111
defaultdict: еще один подход к обработке отсутствия ключа	111
Метод <code>_missing_</code>	112
Несогласованное использование <code>_missing_</code> в стандартной библиотеке	114
Вариации на тему dict	115
collections.OrderedDict	115
collections.ChainMap	116
collections.Counter	117
shelve.Shelf	117
Создание подкласса UserDict вместо dict	118
Неизменяемые отображения	120
Представления словаря	121
Практические последствия внутреннего устройства класса dict	122
Теория множеств	123
Литеральные множества	125
Множественное включение	126
Практические последствия внутреннего устройства класса set	126
Операции над множествами	127
Теоретико-множественные операции над представлениями словарей	129
Резюме	131
Дополнительная литература	132

Глава 4. Unicode-текст и байты 135

Что нового в этой главе	136
О символах, и не только	136
Все, что нужно знать о байтах	137
Базовые кодировщики и декодировщики	140
Проблемы кодирования и декодирования	141
Обработка UnicodeEncodeError	142
Обработка UnicodeDecodeError	143
Исключение SyntaxError при загрузке модулей с неожиданной кодировкой	144
Как определить кодировку последовательности байтов	145
ВОМ: полезный крокозябр	146
Обработка текстовых файлов	147
Остерегайтесь кодировок по умолчанию	150
Нормализация Unicode для надежного сравнения	155
Сворачивание регистра	158
Служебные функции для сравнения нормализованного текста	158
Экстремальная «нормализация»: удаление диакритических знаков	159
Сортировка Unicode-текстов	162
Сортировка с помощью алгоритма упорядочивания Unicode	164
База данных Unicode	165
Поиск символов по имени	165
Символы, связанные с числами	167

Двухрежимный API.....	168
str и bytes в регулярных выражениях.....	168
str и bytes в функциях из модуля os	170
Резюме.....	170
Дополнительная литература.....	171
Глава 5. Построители классов данных.....	176
Что нового в этой главе	177
Обзор построителей классов данных.....	177
Основные возможности	179
Классические именованные кортежи	181
Типизированные именованные кортежи	184
Краткое введение в аннотации типов.....	185
Никаких последствий во время выполнения	185
Синтаксис аннотаций переменных	186
Семантика аннотаций переменных.....	186
Инспекция typing.NamedTuple	187
Инспектирование класса с декоратором dataclass.....	188
Еще о @dataclass	190
Опции полей	191
Постинициализация.....	194
Типизированные атрибуты класса.....	196
Инициализируемые переменные, не являющиеся полями	196
Пример использования @dataclass: запись о ресурсе из дублинского ядра	197
Класс данных как признак кода с душком.....	199
Класс данных как временная конструкция	201
Класс данных как промежуточное представление	201
Сопоставление с экземплярами классов – образцами	201
Простые классы-образцы.....	202
Именованные классы-образцы	202
Позиционные классы-образцы	204
Резюме.....	205
Дополнительная литература.....	205
Глава 6. Ссылки на объекты, изменяемость и повторное использование.....	209
Что нового в этой главе	210
Переменные – не ящики	210
Тождественность, равенство и псевдонимы.....	212
Выбор между == и is.....	213
Относительная неизменяемость кортежей	214
По умолчанию копирование поверхностное.....	215
Глубокое и поверхностное копирование произвольных объектов	218
Параметры функций как ссылки	219
Значения по умолчанию изменяемого типа: неудачная мысль	220
Защитное программирование при наличии изменяемых параметров.....	222

del и сборка мусора.....	224
Как Python хитрит с неизменяемыми объектами.....	226
Резюме.....	228
Дополнительная литература.....	229

ЧАСТЬ II. ФУНКЦИИ КАК ОБЪЕКТЫ233

Глава 7. Функции как полноправные объекты..... 234

Что нового в этой главе.....	235
Обращение с функцией как с объектом.....	235
Функции высшего порядка.....	236
Современные альтернативы функциям map, filter и reduce.....	237
Анонимные функции.....	239
Девять видов вызываемых объектов.....	240
Пользовательские вызываемые типы.....	241
От позиционных к чисто именованным параметрам.....	242
Чисто позиционные параметры.....	244
Пакеты для функционального программирования.....	245
Модуль operator.....	245
Фиксация аргументов с помощью functools.partial.....	248
Резюме.....	250
Дополнительная литература.....	250

Глава 8. Аннотации типов в функциях..... 254

Что нового в этой главе.....	255
О постепенной типизации.....	255
Постепенная типизация на практике.....	256
Начинаем работать с Муру.....	257
А теперь построче.....	258
Значение параметра по умолчанию.....	258
None в качестве значения по умолчанию.....	260
Типы определяются тем, какие операции они поддерживают.....	261
Типы, пригодные для использования в аннотациях.....	266
Тип Any.....	266
«Является подтипом» и «совместим с».....	267
Простые типы и классы.....	269
Типы Optional и Union.....	269
Обобщенные коллекции.....	270
Типы кортежей.....	273
Обобщенные отображения.....	275
Абстрактные базовые классы.....	276
Тип Iterable.....	278
Параметризованные обобщенные типы и TypeVar.....	280
Статические протоколы.....	284
Тип Callable.....	288
Тип NoReturn.....	291
Аннотирование чисто позиционных и вариadicеских параметров.....	291

Несовершенная типизация и строгое тестирование	292
Резюме	293
Дополнительная литература	294
Глава 9. Декораторы и замыкания	300
Что нового в этой главе	301
Краткое введение в декораторы	301
Когда Python выполняет декораторы	302
Регистрационные декораторы	304
Правила видимости переменных	304
Замыкания	307
Объявление nonlocal	310
Логика поиска переменных	311
Реализация простого декоратора	312
Как это работает	313
Декораторы в стандартной библиотеке	314
Запоминание с помощью functools.cache	315
Использование lru_cache	317
Обобщенные функции с одиночной диспетчеризацией	318
Параметризованные декораторы	322
Параметризованный регистрационный декоратор	323
Параметризованный декоратор clock	324
Декоратор clock на основе класса	327
Резюме	328
Дополнительная литература	328
Глава 10. Реализация паттернов проектирования с помощью полноправных функций	333
Что нового в этой главе	334
Практический пример: переработка паттерна Стратегия	334
Классическая Стратегия	334
Функционально-ориентированная стратегия	338
Выбор наилучшей стратегии: простой подход	341
Поиск стратегий в модуле	342
Паттерн Стратегия, дополненный декоратором	343
Паттерн Команда	345
Резюме	346
Дополнительная литература	347
ЧАСТЬ III. КЛАССЫ И ПРОТОКОЛЫ	351
Глава 11. Объект в духе Python	352
Что нового в этой главе	353
Представления объекта	353
И снова класс вектора	354
Альтернативный конструктор	356
Декораторы classmethod и staticmethod	357

Форматирование при выводе	358
Хешируемый класс Vector2d	361
Поддержка позиционного сопоставления с образцом	363
Полный код класса Vector2d, версия 3.....	365
Закрытые и «защищенные» атрибуты в Python	368
Экономия памяти с помощью атрибута класса <code>_slots_</code>	370
Простое измерение экономии, достигаемой за счет <code>_slot_</code>	372
Проблемы при использовании <code>_slots_</code>	373
Переопределение атрибутов класса	374
Резюме.....	376
Дополнительная литература.....	377

Глава 12. Специальные методы для последовательностей 381

Что нового в этой главе	381
Vector: пользовательский тип последовательности.....	382
Vector, попытка № 1: совместимость с Vector2d.....	382
Протоколы и утиная типизация	385
Vector, попытка № 2: последовательность, допускающая срез.....	386
Как работает срезка	387
Метод <code>_getitem_</code> с учетом срезов	388
Vector, попытка № 3: доступ к динамическим атрибутам	390
Vector, попытка № 4: хеширование и ускорение оператора <code>==</code>	393
Vector, попытка № 5: форматирование	399
Резюме.....	406
Дополнительная литература.....	407

Глава 13. Интерфейсы, протоколы и ABC..... 411

Карта типизации.....	412
Что нового в этой главе	413
Два вида протоколов	413
Программирование уток.....	415
Python в поисках следов последовательностей.....	415
Партизанское латание как средство реализации протокола во время выполнения.....	417
Защитное программирование и принцип быстрого отказа	419
Гусиная типизация	421
Создание подкласса ABC.....	426
ABC в стандартной библиотеке	427
Определение и использование ABC	430
Синтаксические детали ABC.....	435
Создание подклассов ABC.....	435
Виртуальный подкласс <code>Tombole</code>	438
Использование функции <code>register</code> на практике	440
ABC и структурная типизация	440
Статические протоколы	442
Типизированная функция <code>double</code>	443
Статические протоколы, допускающие проверку во время выполнения.....	444

Ограничения протоколов, допускающих проверку во время выполнения.....	447
Поддержка статического протокола	448
Проектирование статического протокола	450
Рекомендации по проектированию протоколов.....	451
Расширение протокола	452
ABC из пакета numbers и числовые протоколы.....	453
Резюме.....	456
Дополнительная литература.....	457
Глава 14. Наследование: к добру или к худу.....	462
Что нового в этой главе	463
Функция super()	463
Сложности наследования встроенным типам.....	465
Множественное наследование и порядок разрешения методов	468
Классы-примеси	473
Отображения, не зависящие от регистра.....	473
Множественное наследование в реальном мире	475
ABC – тоже примеси	475
ThreadingMixIn и ForkingMixIn	475
Множественное наследование в Tkinter	480
Жизнь с множественным наследованием	482
Предпочитайте композицию наследованию класса	483
Разберитесь, зачем наследование используется в каждом конкретном случае.....	483
Определяйте интерфейсы явно с помощью ABC	483
Используйте примеси для повторного использования кода	484
Предоставляйте пользователям агрегатные классы	484
Наследуйте только классам, предназначенным для наследования	484
Воздерживайтесь от наследования конкретным классам	485
Tkinter: хороший, плохой, злой	485
Резюме.....	487
Дополнительная литература.....	488
Глава 15. Еще об аннотациях типов.....	492
Что нового в этой главе	492
Перегруженные сигнатуры	492
Перегрузка max.....	494
Уроки перегрузки max.....	498
TypedDict	498
Приведение типов	505
Чтение аннотаций типов во время выполнения.....	508
Проблемы с аннотациями во время выполнения	508
Как решать проблему	511
Реализация обобщенного класса	511
Основы терминологии, относящейся к обобщенным типам	513
Вариантность	514

Инвариантный разливающий автомат	514
Ковариантный разливающий автомат.....	516
Контравариантная урна	516
Обзор вариантности.....	518
Реализация обобщенного статического протокола	520
Резюме.....	522
Дополнительная литература.....	523

Глава 16. Перегрузка операторов 528

Что нового в этой главе	529
Основы перегрузки операторов	529
Унарные операторы	530
Перегрузка оператора сложения векторов +	533
Перегрузка оператора умножения на скаляр *	538
Использование @ как инфиксного оператора	540
Арифметические операторы – итоги.....	541
Операторы сравнения.....	542
Операторы составного присваивания	545
Резюме.....	549
Дополнительная литература.....	550

ЧАСТЬ IV. ПОТОК УПРАВЛЕНИЯ 555

Глава 17. Итераторы, генераторы и классические сопрограммы..... 556

Что нового в этой главе	557
Последовательность слов	557
Почему последовательности итерируемы: функция iter.....	558
Использование iter в сочетании с Callable.....	560
Итерируемые объекты и итераторы	561
Классы Sentence с методом <code>__iter__</code>	564
Класс Sentence, попытка № 2: классический итератор	565
Не делайте итерируемый объект итератором для самого себя.....	566
Класс Sentence, попытка № 3: генераторная функция	567
Как работает генератор	568
Ленивые классы Sentence.....	570
Класс Sentence, попытка № 4: ленивый генератор	570
Класс Sentence, попытка № 5: генераторное выражение	571
Генераторные выражения: когда использовать	573
Генератор арифметической прогрессии.....	575
Построение арифметической прогрессии с помощью <code>itertools</code>	577
Генераторные функции в стандартной библиотеке.....	578
Функции редуцирования итерируемого объекта.....	588
<code>yield from</code> и субгенераторы	590
Изобретаем <code>chain</code> заново	591
Обход дерева	592
Обобщенные итерируемые типы	596

Классические сопрограммы.....	597
Пример: сопрограмма для вычисления накопительного среднего.....	599
Возврат значения из сопрограммы.....	601
Аннотации обобщенных типов для классических сопрограмм.....	605
Резюме.....	607
Дополнительная литература.....	607
Глава 18. Блоки with, match и else	612
Что нового в этой главе	613
Контекстные менеджеры и блоки with	613
Утилиты contextlib.....	617
Использование @contextmanager	618
Сопоставление с образцом в lis.py: развернутый пример.....	622
Синтаксис Scheme	622
Предложения импорта и типы	623
Синтаксический анализатор	624
Класс Environment	626
Цикл REPL	628
Вычислитель	629
Procedure: класс, реализующий замыкание	636
Использование OR-образцов.....	637
Делай то, потом это: блоки else вне if	638
Резюме.....	640
Дополнительная литература.....	641
Глава 19. Модели конкурентности в Python.....	646
Что нового в этой главе	647
Общая картина.....	647
Немного терминологии.....	648
Процессы, потоки и знаменитая блокировка GIL в Python	650
Конкурентная программа Hello World	652
Анимированный индикатор с потоками.....	652
Индикатор с процессами	655
Индикатор с сопрограммами	656
Сравнение супервизоров	660
Истинное влияние GIL	662
Проверка знаний	662
Доморощенный пул процессов	665
Решение на основе процессов	666
Интерпретация времени работы.....	667
Код проверки на простоту для многоядерной машины	668
Эксперименты с большим и меньшим числом процессов	671
Не решение на основе потоков.....	672
Python в многоядерном мире.....	673
Системное администрирование.....	674
Наука о данных.....	675
Веб-разработка на стороне сервера и на мобильных устройствах.....	676

WSGI-серверы приложений.....	678
Распределенные очереди задач.....	680
Резюме.....	681
Дополнительная литература.....	682
Конкурентность с применением потоков и процессов	682
GIL.....	684
Конкурентность за пределами стандартной библиотеки.....	684
Конкурентность и масштабируемость за пределами Python	686
Глава 20. Конкурентные исполнители.....	691
Что нового в этой главе	691
Конкурентная загрузка из веба	692
Скрипт последовательной загрузки.....	694
Загрузка с применением библиотеки <code>concurrent.futures</code>	696
Где находятся будущие объекты?	698
Запуск процессов с помощью <code>concurrent.futures</code>	701
И снова о проверке на простоту на многоядерной машине	701
Эксперименты с <code>Executor.map</code>	704
Загрузка с индикацией хода выполнения и обработкой ошибок	707
Обработка ошибок во <code>flags2</code> -примерах.....	711
Использование <code>futures.as_completed</code>	713
Резюме.....	716
Дополнительная литература.....	716
Глава 21. Асинхронное программирование	719
Что нового в этой главе	720
Несколько определений	720
Пример использования <code>asyncio</code> : проверка доменных имен	721
Предложенный Гвидо способ чтения асинхронного кода.....	723
Новая концепция: объекты, допускающие ожидание	724
Загрузка файлов с помощью <code>asyncio</code> и <code>HTTPX</code>	725
Секрет платформенных сопрограмм: скромные генераторы.....	727
Проблема «все или ничего»	728
Асинхронные контекстные менеджеры.....	729
Улучшение асинхронного загрузчика	730
Использование <code>asyncio.as_completed</code> и потока.....	731
Регулирование темпа запросов с помощью семафора	733
Отправка нескольких запросов при каждой загрузке	736
Делегирование задач исполнителям.....	739
Написание асинхронных серверов.....	740
Веб-служба <code>FastAPI</code>	742
Асинхронный <code>TCP</code> -сервер.....	746
Асинхронные итераторы и итерируемые объекты.....	751
Асинхронные генераторные функции.....	752
Асинхронные включения и асинхронные генераторные выражения.....	758
<code>async</code> за пределами <code>asyncio</code> : <code>Curio</code>	760
Аннотации типов для асинхронных объектов	763

Как работает и как не работает асинхронность	764
Круги, разбегающиеся вокруг блокирующих вызовов	764
Миф о системах, ограниченных вводом-выводом	765
Как не попасть в ловушку счетных функций.....	765
Резюме.....	766
Дополнительная литература.....	767

ЧАСТЬ V. МЕТАПРОГРАММИРОВАНИЕ.....771

Глава 22. Динамические атрибуты и свойства772

Что нового в этой главе	772
Применение динамических атрибутов для обработки данных.....	773
Исследование JSON-подобных данных с динамическими атрибутами	775
Проблема недопустимого имени атрибута	778
Гибкое создание объектов с помощью метода <code>_new_</code>	779
Вычисляемые свойства	781
Шаг 1: создание управляемого данными атрибута.....	782
Шаг 2: выборка связанных записей с помощью свойств.....	784
Шаг 3: переопределение существующего атрибута свойством.....	787
Шаг 4: кеширование свойств на заказ.....	788
Шаг 5: кеширование свойств с помощью <code>functools</code>	789
Использование свойств для контроля атрибутов.....	791
<code>LineItem</code> , попытка № 1: класс строки заказа	791
<code>LineItem</code> , попытка № 2: контролирующее свойство	792
Правильный взгляд на свойства.....	794
Свойства переопределяют атрибуты экземпляра.....	795
Документирование свойств.....	797
Программирование фабрики свойств.....	798
Удаление атрибутов.....	800
Важные атрибуты и функции для работы с атрибутами	802
Специальные атрибуты, влияющие на обработку атрибутов	802
Встроенные функции для работы с атрибутами	803
Специальные методы для работы с атрибутами.....	804
Резюме.....	805
Дополнительная литература.....	806

Глава 23. Дескрипторы атрибутов810

Что нового в этой главе	810
Пример дескриптора: проверка значений атрибутов	811
<code>LineItem</code> попытка № 3: простой дескриптор.....	811
<code>LineItem</code> попытка № 4: автоматическое генерирование имен атрибутов хранения.....	816
<code>LineItem</code> попытка № 5: новый тип дескриптора.....	818
Переопределяющие и непереопределяющие дескрипторы.....	820
Переопределяющие дескрипторы.....	822
Переопределяющий дескриптор без <code>_get_</code>	823
Непереопределяющий дескриптор.....	824
Перезаписывание дескриптора в классе	825

Методы являются дескрипторами	826
Советы по использованию дескрипторов.....	828
Строка документации дескриптора и перехват удаления.....	829
Резюме.....	831
Дополнительная литература.....	831
Глава 24. Метaproграммирование классов.....	834
Что нового в этой главе	835
Классы как объекты	835
type: встроенная фабрика классов	836
Функция-фабрика классов	837
Введение в <code>_init_subclass_</code>	840
Почему <code>_init_subclass_</code> не может конфигурировать <code>_slots_</code>	846
Дополнение класса с помощью декоратора класса.....	847
Что когда происходит: этап импорта и этап выполнения.....	849
Демонстрация работы интерпретатора.....	850
Основы метаклассов.....	854
Как метакласс настраивает класс	856
Элегантный пример метакласса.....	857
Демонстрация работы метакласса	860
Реализация <code>Checked</code> с помощью метакласса	864
Метаклассы на практике	868
Современные средства позволяют упростить или заменить метаклассы.....	868
Метаклассы – стабильное языковое средство	869
У класса может быть только один метакласс.....	869
Метаклассы должны быть деталью реализации.....	870
Метаклассный трюк с <code>_prepare_</code>	870
Заключение	872
Резюме.....	873
Дополнительная литература.....	874
Послесловие	878
Предметный указатель	881

Предисловие от издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в основном тексте или программном коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Об авторе

Лусиану Рамальо был веб-разработчиком до выхода компании Netscape на IPO в 1995 году, а в 1998 году перешел с Perl на Java, а затем на Python. В 2015 году пришел в компанию Thoughtworks, где работает главным консультантом в отделении в Сан-Паулу. Он выступал с основными докладами, презентациями и пособиями на различных мероприятиях, связанных с Python, в обеих Америках, Европе и Азии. Выступал также на конференциях по Go и Elixir по вопросам проектирования языков. Рамальо – член фонда Python Software Foundation и сооснователь клуба Garoa Hacker Clube, первого места для общения хакеров в Бразилии.

Колофон

На обложке изображена намакская песчаная ящерица (*Pedioplanis namaquensis*), встречающаяся в засушливых саваннах и полупустынях Намибии.

Внешние признаки: туловище черное с четырьмя белыми полосками на спине, лапы коричневые с белыми пятнышками, брюшко белое, длинный розовато-коричневый хвост. Одна из самых быстрых ящериц, активна в течение дня, питается мелкими насекомыми. Обитает на бедных растительностью песчано-каменистых равнинах. Самка откладывает от трех до пяти яиц в ноябре. Остаток зимы ящерицы спят в норах, которые роют в корнях кустов.

В настоящее время охранный статус намакской песчаной ящерицы – «пониженная уязвимость». Многие животные, изображенные на обложках книг O'Reilly, находятся под угрозой вымирания; все они важны для мира.

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>.

– Тим Питерс, легендарный разработчик ядра
и автор сборника поучений «The Zen of Python»¹

«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 слышали 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 пакетах, 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. Я тестировал примеры на Python 3.10,

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>).

а большую их часть также на Python 3.9 и 3.8. Если какой-то пример требует версии 3.10, то это явно оговаривается.

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.

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.

ПЯТЬ КНИГ В ОДНОЙ

Я рекомендую всем прочитать главу 1 «Модель данных в языке Python». Читатели этой книги в большинстве своем после ознакомления с главой 1, скорее всего, смогут легко перейти к любой части, но я зачастую предполагаю, что главы каждой части читаются по порядку. Части I–V можно рассматривать как отдельные книги внутри книги.

Я старался сначала рассказывать о том, что уже есть, а лишь затем о том, как создавать что-то свое. Например, в главе 2 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 III, 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, что теперь, когда вы узнали, почему я избрал такой путь, вам будет проще смириться с этим.

КАК ОРГАНИЗОВАНА ЭТА КНИГА

Ниже описаны основные темы, рассматриваемые в каждой части книги.

Часть I «Структуры данных»

В главе I, посвященной модели данных в Python, объясняется ключевая роль специальных методов (например, `__repr__`) для обеспечения единообразного поведения объектов любого типа. Специальные методы более подробно обсуждаются на протяжении всей книги. В остальных главах этой части рассматривается использование типов коллекций: последовательностей, отображений и множеств, а также различие между типами `str` и `bytes` – то, что радостно приветствовали пользователи Python 3 и чего отчаянно не хватает пользователям Python 2, еще не модернизовавшим свой код. Также рассматриваются высокоуровневые строители классов, имеющиеся в стандартной библиотеке: фабрики именованных кортежей и декоратор `@dataclass`. Сопоставление с образцом – новая возможность, появившаяся в Python 3.10, – рассматривается в разделах глав 2, 3 и 5, где обсуждаются паттерны последовательностей, отображений и классов. Последняя глава части I посвящена жизненному циклу объектов: ссылкам, изменению и сборке мусора.

Часть II «Функции как объекты»

Здесь речь пойдет о функциях как полноправных объектах языка: что под этим понимается, как это отражается на некоторых популярных паттернах проектирования и как реализовать декораторы функций с помощью замыканий. Рассматриваются также следующие вопросы: общая идея вызываемых объектов, атрибуты функций, интроспекция, аннотации параметров и появившееся в Python 3 объявление `nonlocal`. Глава 8 содержит введение в новую важную тему – аннотации типов в сигнатурах функций.

Часть III «Классы и протоколы»

Теперь наше внимание перемещается на создание классов «вручную» – в отличие от использования строителей классов, рассмотренных в главе 5. Как и в любом объектно-ориентированном (ОО) языке, в Python имеется свой набор средств; какие-то из них, возможно, присутствовали в языке, с которого вы и я начинали изучение программирования на основе классов, а какие-то – нет. В главах из этой части объясняется, как создать свою коллекцию, абстрактный базовый класс (ABC) и протокол, как работать со множественным наследованием и как реализовать перегрузку операторов (если это имеет смысл). В главе 15 мы продолжим обсуждать аннотации типов.

Часть IV «Поток управления»

Эта часть посвящена языковым конструкциям и библиотекам, выходящим за рамки последовательного потока управления с его условными выражениями, циклами и подпрограммами. Сначала мы рассматриваем генераторы, затем – контекстные менеджеры и сопрограммы, в том числе трудную для понимания, но исключительно полезную новую конструкцию `yield from`. В главу 18 включен важный пример использования сопоставления с образцом в простом, но функциональном интерпретаторе языка. Глава 19 «Модели конкурентности в Python» новая, она посвящена обзору различных

видов конкурентной и параллельной обработки в Python, их ограничений и вопросу о том, как архитектура программы позволяет использовать Python в приложениях масштаба веба. Я переписал главу об *асинхронном программировании*, стремясь уделить больше внимания базовым средствам языка – `await`, `async dev`, `async for` и `async with` – и показать, как они используются совместно с библиотекой *asyncio* и другими каркасами.

Часть V «Метапрограммирование»

Эта часть начинается с обзора способов построения классов с динамически создаваемыми атрибутами для обработки слабоструктурированных данных, например в формате JSON. Затем мы рассматриваем знакомый механизм свойств, после чего переходим к низкоуровневым деталям доступа к атрибутам объекта с помощью дескрипторов. Объясняется связь между функциями, методами и дескрипторами. На примере приведенной здесь пошаговой реализации библиотеки контроля полей мы вскрываем тонкие нюансы, которые делают необходимым применение рассмотренных в этой главе продвинутых инструментов: декораторов классов и метаклассов.

ПРАКТИКУМ

Часто для исследования языка и библиотек мы будем пользоваться интерактивной оболочкой Python. Я считаю важным всячески подчеркивать удобство этого средства для обучения. Особенно это относится к читателям, привыкшим к статическим компилируемым языкам, в которых нет цикла чтения-вычисления-печати (`read-eval-print#loop` – REPL).

Один из стандартных пакетов тестирования для Python, `doctest` (<https://docs.python.org/3/library/doctest.html>), работает следующим образом: имитирует сеансы оболочки и проверяет, что результат вычисления выражения совпадает с заданным. Я использовал `doctest` для проверки большей части приведенного в книге кода, включая листинги сеансов оболочки. Для чтения книги ни применять, ни даже знать о пакете `doctest` не обязательно: основная характеристика `doctest`-скриптов (или просто тестов) состоит в том, что они выглядят как копии интерактивных сеансов оболочки Python, поэтому вы можете сами выполнить весь демонстрационный код.

Иногда я буду объяснять, чего мы хотим добиться, демонстрируя тест раньше кода, который заставляет его выполниться успешно. Если сначала отчетливо представить себе, что необходимо сделать, а только потом задумываться о том, как это сделать, то структура кода заметно улучшится. Написание тестов раньше кода – основа методологии разработки через тестирование (TDD); мне кажется, что и для преподавания это полезно. Если вы незнакомы с `doctest`, загляните в документацию (<https://docs.python.org/3/library/doctest.html>) и в репозиторий исходного кода к этой книге (<https://github.com/fluentpython/example-code-2e>).

Я также написал автономные тесты для нескольких крупных примеров, воспользовавшись модулем `pytest`, который, как мне кажется, проще и имеет больше возможностей, чем модуль `unittest` из стандартной библиотеки. Вы увидите, что для проверки правильности большей части кода в книге до-

Конец ознакомительного фрагмента.

Приобрести книгу можно

в интернет-магазине

«Электронный универс»

e-Univers.ru