

Оглавление (кратко)

	Введение	23
1	Приступим	37
2	Сначала было значение	69
3	Принятие решений	109
4	Структуры данных	161
5	Функциональный код	215
4a	Наведение порядка в данных	261
6	Сведем все воедино	281
7	Модульное программирование	327
8	И снова об индексах и циклах	377
9	Длительное хранение данных	429
10	Так хочется большего	471
11	Интерактивные возможности	503
12	Путешествие в страну объектов	559
	Приложение: Десять ключевых тем (которые не были рассмотрены)	611

Содержание (подробно)

Введение

Настройте свой мозг на изучение программирования.

Конечно, вам хочется чему-то научиться, но у мозга могут быть свои планы на этот счет. Он контролирует, чтобы вы не забивали себе голову ненужной информацией. Есть ведь вещи и поважнее, например какой фильм пойти посмотреть на выходных или что надеть на вечеринку. Как же убедить мозг, что книга по программированию содержит нужную информацию?



Для кого эта книга	24
Мы знаем, о чем вы думаете	25
Мы считаем читателей книги учениками	26
Метапознание: осознание знаний	27
Вот что сделали мы	28
Вот что можете сделать вы, чтобы настроить свой мозг на обучение	29
Это важно	30
Необходимо установить Python	32
Как работать с исходными кодами	34
Благодарности	35
Команда рецензентов	36

учимся думать как программист

1 Приступим

Чтобы научиться писать программы, нужно думать как программист. В мире современных технологий все вокруг нас становится взаимосвязанным, настраиваемым, программируемым и в каком-то смысле **компьютерным**. Можно оставаться пассивным наблюдателем, а можно *научиться программировать*. Умея писать программный код, вы становитесь творцом, устанавливающим правила игры, ведь именно вы отдаете команды компьютеру. Но как освоить искусство программирования? Самое главное — это начать **думать как программист**. Далее нужно определиться с **языком программирования** — написанные на нем программы должны выполняться на всех целевых устройствах (компьютерах, смартфонах и любых других электронных гаджетах, снабженных процессором). Что это вам даст? Больше свободного времени, больше ресурсов и больше творческих возможностей для воплощения в жизнь задуманного.



Шаг за шагом	38
Как пишут программы	42
Понимают ли нас компьютеры?	43
Мир языков программирования	44
Ввод и выполнение кода Python	49
Краткая история Python	51
Тестируем среду	54
Сохранение кода	56
Первая программа готова!	57
Phrase-O-Matic	61
Запуск кода на выполнение	62

значения, переменные и типы данных

2 Сначала было значение

В реальности компьютеры умеют делать всего две вещи: **сохранять значения и выполнять над ними операции**. А как же редактирование текста, ретуширование фотографий и онлайн-покупки, спросите вы? Удивительно, но решение любых подобных задач в конечном счете сводится к выполнению **простых операций** над не менее **простыми значениями**. Человек, обладающий вычислительным мышлением, понимает, что это за операции и значения и как с их помощью создать что-то более сложное. В этой главе мы познакомимся с типами значений, узнаем, какие операции можно выполнять над ними и какую роль во всем этом играют переменные.

Калькулятор возраста собаки	70
От псевдокода к программному коду	72
Этап 1: запрос входных данных	73
Как работает функция <code>input()</code>	74
Сохранение значений в переменных	74
Сохранение вводимых данных в переменной	75
Этап 2: еще один запрос	75
Пора выполнить код	76
Ввод кода в редакторе	79
Подробнее о переменных	80
Добавляем выражения	81
Изменение переменных	82
Приоритет операторов	83
Вычисление выражений согласно приоритету операторов	84
Руки прочь от клавиатуры!	87
Этап 3: вычисление возраста собаки	88
У нас проблема!	89
Неизбежность ошибок	90
Еще немного отладки	92
Типы данных Python	94
Исправление ошибки	95
Ура, заработало!	96
Этап 4: вывод результата	97

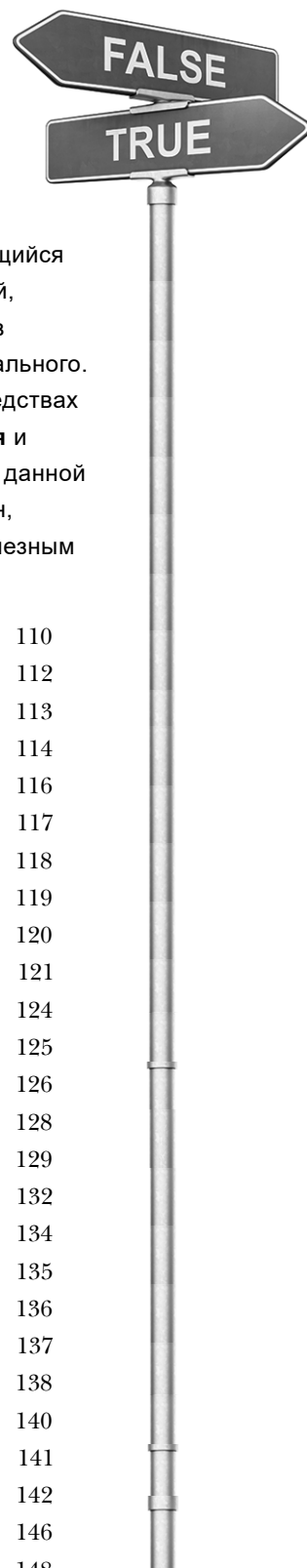


булевы значения, условия и циклы

3 Принятие решений

Думаю, вы заметили, что наши предыдущие программы были не особо интересными. Содержащийся в них код представлял собой простую последовательность инструкций, выполняемых интерпретатором **сверху вниз**. Никаких тебе поворотов сюжета, внезапных изменений, сюрпризов — в общем, ничего оригинального. Чтобы программа могла стать более интересной, она нуждается в средствах **принятия решений**, позволяющих **управлять ходом ее выполнения** и многократно повторять одни и те же действия. Этим мы и займемся в данной главе. По ходу дела вы научитесь играть в загадочную игру шоуилин, познакомитесь с персонажем по имени Буль и увидите, насколько полезным может быть тип данных, содержащий всего два возможных значения.

Сыграем в игру?	110
Общий алгоритм игры	112
Компьютер делает свой выбор	113
Применение случайных чисел	114
Истина или ложь?	116
Булевы значения	117
Принимаем решения	118
Дополнительные условия	119
Вернемся к игре	120
Пользователь делает свой выбор	121
Проверяем, что выбрал пользователь	124
Код определения ничьей	125
Кто выиграл?	126
Реализация логики игры	128
Подробнее о булевых операторах	129
Вывод имени победителя	132
А где документация?	134
Добавление комментариев в код	135
Завершение игры	136
Проверка правильности введенных данных	137
Проверка и уточнение выражения	138
Повторный вывод запроса	140
Многократное повторение действий	141
Как работает цикл <code>while</code>	142
Как использовать цикл <code>while</code> для повторного вывода запроса	146
Итак, наша первая игра готова!	148



списки и циклы

4 Структуры данных

Числа, строки и булевы значения — не единственные

типы данных в программировании. Пока что мы использовали только **примитивные типы**: числа с плавающей точкой, целые числа, строки и булевы значения (например, 3.14, 42, "имя собаки" и True). И хотя их возможности достаточно велики, время от времени в программах приходится иметь дело с большими наборами данных, такими как все добавленные в корзину покупки, названия всех известных звезд или содержимое всего каталога товаров. Для эффективной работы с подобными наборами нужны **структуры данных**. В этой главе вы познакомитесь с новым типом данных — **списком**, хранящим коллекцию значений. Списки позволяют структурировать данные, вместо того чтобы хранить каждый элемент в отдельной переменной. Вы узнаете о том, как работать со списком в целом и как поочередно перебирать его элементы в цикле `for`. Все это расширит ваши возможности по работе с данными.

Поможете разобраться?	162
Сохранение нескольких значений в одной переменной	163
Работа со списками	164
Размер списка	167
Извлечение последнего элемента списка	168
В Python все намного проще	168
Отрицательные индексы	169
Продолжим подсчет мыльных пузырьков	171
Просмотр элементов списка	174
Устранение ошибки вывода	175
Правильный способ вывода результата	176
Цикл <code>for</code> предпочтителен при работе со списками	178
Обработка диапазона чисел в цикле <code>for</code>	181
Другие операции с диапазонами	182
Составление отчета	184
Построение списка с нуля	192
Другие операции со списками	193
Тест-драйв готового отчета	197
И победителями становятся...	197
Тестирование самого выгодного раствора	201



функции и абстракции

5 Функциональный код

Вы уже многому научились. Переменные и типы данных, условные конструкции и циклы — этих инструментов вполне достаточно, чтобы написать почти любую программу. Но зачем останавливаться на достигнутом? Следующий шаг — научиться **создавать абстракции** в коде. Звучит пугающе, хотя ничего страшного в этом нет. Абстракции — ваш спасательный круг. Они упрощают процесс разработки, давая возможность создавать более эффективные и функциональные программы. Абстракции позволяют упаковать код в небольшие блоки, удобные для повторного использования. С ними вы перестанете концентрироваться на низкоуровневых деталях и сможете программировать на высоком уровне.

Что не так с кодом?	217
Превращение блока кода в функцию	219
Функция создана — можно использовать	220
Как работает функция	220
Функции могут возвращать значения	228
Вызов функции, возвращающей значение	229
Улучшение имеющегося кода	231
Выполнение кода	232
Создание абстракций в коде выбора аватарки	233
Тело функции <code>get_attribute()</code>	234
Вызов функции <code>get_attribute()</code>	235
И снова о переменных	237
Область видимости переменной	238
Передача переменных в функцию	239
Вызов функции <code>drink_me()</code>	240
Использование глобальных переменных в функциях	243
Размышления о параметрах: значения по умолчанию и ключевые слова	246
Параметры по умолчанию	246
Первыми всегда указываются обязательные параметры	247
Аргументы с ключевыми словами	248
Как правильно использовать все доступные возможности	248



сортировка и Вложенные списки

4

(продолжение)

Наведение порядка в данных

Иногда стандартный порядок сортировки данных нас не устраивает. Например, у вас есть список лучших результатов в аркадных играх, и вы хотите упорядочить его по названиям игр. Или же это может быть список сотрудников, регулярно опаздывающих на работу, и вы хотите узнать, кто из них проштрафился чаще остальных. Для решения подобных задач нужно знать, как сортировать данные и как самостоятельно настраивать порядок сортировки. В этой главе мы также изучим вложенные циклы и выясним, насколько эффективен написанный нами код.

Пузырьковый метод сортировки	264
Псевдокод алгоритма пузырьковой сортировки	267
Реализация пузырьковой сортировки на Python	270
Правильное определение номеров растворов	272



текст, строки и эвристические алгоритмы

6

Сведем все воедино

Вы уже многое знаете. Пора задействовать знания по максимуму. В этой главе мы сведем все воедино и начнем писать **сложный код**. В то же время мы продолжим пополнять арсенал знаний и навыков программирования. В частности, будет показано, как написать код, который **загружает текст**, обрабатывает его и выполняет **анализ текстовых данных**. Вы также узнаете, что такое **эвристический алгоритм** и как его реализовать. Приготовьтесь — вас ждет по-настоящему сложная и насыщенная глава.

Добро пожаловать в науку о данных	282
Как вычисляется индекс удобочитаемости	283
План действий	284
Общий псевдокод	285
Нужен текст для анализа	286
Создание основной функции	288
Подсчет числа слов в тексте	289
Подсчет общего числа предложений	293
Функция <code>count_sentences()</code>	294
Подсчет слогов: знакомство с эвристическими алгоритмами	300
Создание эвристического алгоритма	303
Начинаем кодировать эвристический алгоритм	304
Подсчет гласных	305
Игнорирование последовательных гласных	305
Код игнорирования последовательных гласных	306
Учет конечной 'е', а также знаков препинания	308
Создание срезов (подстрок)	310
Завершение эвристического алгоритма	312
Код вычисления индекса удобочитаемости	314
Дальнейшие улучшения	319

Вот это сложные
темы пошли!



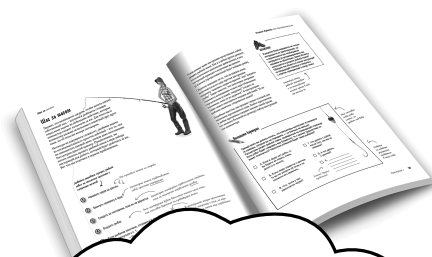
Классы, объекты, Методы и Модули

7 Модульное программирование

Написанные вами программы становятся все более сложными.

Это заставляет тщательнее продумывать структуру и организацию программ. Вы уже знаете, что функции позволяют группировать строки кода в компактные блоки, удобные для повторного использования. Вы также знаете, что наборы функций и переменных можно объединять в модули, подключаемые к программам. В этой главе мы вернемся к теме модулей и рассмотрим, как работать с ними эффективнее (и даже делиться ими с другими пользователями). Кроме того, вас ждет знакомство с ключевыми строительными блоками программ — *объектами*. Вы увидите, что в Python они встречаются повсеместно, даже там, где вы не ожидаете их встретить.

Знакомство с модулями	330
Глобальная переменная <code>__name__</code>	332
Обновление файла <i>analyze.py</i>	333
Использование файла <i>analyze.py</i> в качестве модуля	335
Добавление документирующих строк в файл <i>analyze.py</i>	337
Знакомство с другими модулями Python	341
Что еще за черепашки?	342
Создание собственной черепашки	344
Изучение черепашек	345
Еще одна черепашка	347
Кто вы, черепашки?	350
Что такое объект?	351
А что тогда класс?	352
Использование объектов и классов	354
Что такое атрибуты и методы?	355
Объекты и классы повсюду	356
Черепашки бега	358
Планирование игры	359
Приступаем к написанию кода игры	360
Код настройки игры	361
Спешка ни к чему!	362
Начало забега	364



Классная работа! Я смог
быстро воспользоваться
модулем, особенно благодаря
прекрасной документации!



рекурсия и словари

8 И снова об индексах и циклах

Настало время перейти на новый уровень. Пока что мы придерживались итеративного стиля программирования, создавая структуры данных наподобие списков, строк и диапазонов чисел и обрабатывая их в циклах поэлементно. В этой главе мы пересмотрим подход к программированию и выбору структур данных. Наш новый стиль программирования будет требовать написания *рекурсивного*, т.е. вызывающего самого себя, кода. Кроме того, вы познакомитесь с новым типом данных — *словарем*, который больше напоминает ассоциативный массив, а не список. Воспользовавшись новыми знаниями, мы научимся эффективно решать множество задач. Заранее предупредим: это достаточно сложные темы. Пройдет какое-то время, прежде чем вы освоите все нюансы, но поверьте, оно того стоит.



Иной стиль программирования	378
Мы пойдем другим путем	379
Код для обоих случаев	380
Давайте попрактикуемся	383
Нахождение палиндромов рекурсивным способом	384
Код рекурсивной функции распознавания палиндромов	385
Антисоциальная сеть	396
Знакомство со словарем	398
Просмотр словаря	400
Словари в антисоциальной сети	402
Добавление новых атрибутов	404
Ключевая особенность антисоциальной сети	406
Определение самого антисоциального пользователя	407
Слово за вами!	408
Можно ли запомнить результаты работы функции?	412
Мемоизация	413
Анализ функции <code>koch()</code>	416
Фрактал Коха	418

Чтение и запись файлов

9 Длительное хранение данных

Вы умеете хранить данные в переменных, но как только программа завершается — бах! — они исчезают навсегда. Вот для чего нужно постоянное хранилище данных. Большинство компьютерных устройств оснащено такими хранилищами, в частности, жесткими дисками и флеш-накопителями. Кроме того, данные можно хранить в облачной службе. В этой главе вы узнаете, как писать программы, умеющие работать с файлами. Зачем это нужно? Вариантов множество: сохранение конфигурационных настроек, создание файлов отчетов, обработка графических файлов, поиск сообщений электронной почты — список можно продолжать еще долго.

Игра в слова	430
Логика игры	432
Считывание истории из файла	435
Путь к файлу	436
Не забудьте все закрыть в конце	438
Чтение файла в коде	439
Пора прекратить...	442
Вернемся к игре	443
Как узнать, что мы достигли конца?	445
Чтение шаблона игры	446
Обработка текста шаблона	447
Исправление ошибки с помощью метода <code>strip()</code>	449
Окончательное исправление ошибки	450
Остались проблемы посерьезнее	451
Обработка исключений	453
Явная обработка исключений	454
Обработка исключений в игре в слова	456
Последний этап: сохранение результата	457
Обновление остального кода	457



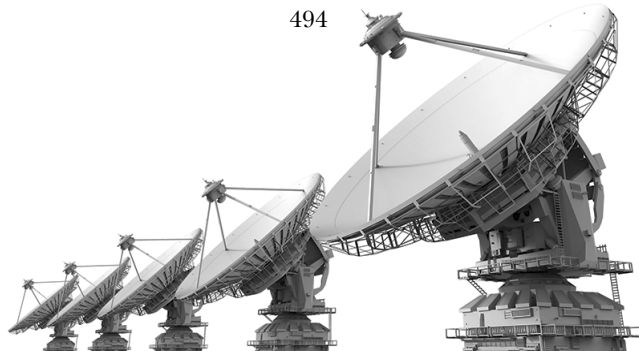
Веб-службы

10 Так хочется большего

Вы научились писать неплохие программы, но всегда

хочется большего. В Интернете хранится просто невероятное количество **данных**, которые так и ждут, чтобы с ними поработали. Нужен прогноз погоды? Как насчет базы данных рецептов? Или вас интересуют спортивные результаты? А может, ваш интерес — музыкальная база данных с информацией о музыкантах, альбомах и композициях? Вся эту информацию можно получить с помощью веб-служб. Для работы с ними нужно знать, как организуется передача данных в Интернете, а также познакомиться с парочкой модулей Python: `requests` и `json`. В этой главе вы изучите возможности **веб-служб** и повысите свой уровень знаний Python до заоблачных высот. Так что приготовьтесь к путешествию в космос.

Расширение возможностей благодаря веб-службам	472
Как работает веб-служба	473
Адрес веб-службы	474
Небольшое обновление Python	477
Установка обновления	478
Нам нужна интересная веб-служба	479
Возможности Open Notify	480
Передача данных в формате JSON	481
Вернемся к модулю <code>requests</code>	483
Собираем все вместе: создание запроса к службе Open Notify	485
Работа с данными JSON в Python	486
Применение модуля <code>json</code>	487
Представим результат в графическом виде	488
Знакомство с объектом <code>screen</code>	489
Черепашка-космонавт	491
Черепашка может выглядеть как космическая станция	492
Забудьте об МКС. Где находимся мы?	493
Завершение программы	494



Виджеты, события и непредсказуемое поведение

11 Интерактивные возможности

Вам уже приходилось писать простые графические приложения, но у них не было полноценного графического интерфейса. Другими словами, приложения не позволяли пользователям взаимодействовать с экранными элементами. Чтобы это стало возможным, необходимо применить иную модель выполнения программы, в рамках которой приложение **реагирует** на действия пользователя. Пользователь щелкнул на кнопке? Программа должна знать, как реагировать на такое событие, и быть готовым к нему. Процесс создания графического интерфейса сильно отличается от привычного процедурного стиля программирования, который мы применяли до сих пор. Нам придется совершенно по-иному подойти к решению задач. В этой главе вы создадите свой первый графический интерфейс, причем не какое-то примитивное экранное приложение, а нечто намного более интересное. Мы напомним игровой симулятор искусственной жизни с непредсказуемым поведением.

Откройте для себя удивительный мир игры “Искусственная жизнь”	504
Правила игры “Жизнь”	505
Что нам предстоит создать	508
Дизайнерские решения	509
Создание симулятора игры “Жизнь”	512
Построение модели данных	513
Вычисление поколений клеток	514
Завершение кода модели	518
Экранное представление	521
Создание первого виджета	522
Добавление остальных виджетов	523
Исправление макета	524
Размещение виджетов по сетке	525
Переходим к написанию контроллера	527
Новый способ вычислений	530
Обработка щелчков мыши	531
Обработка событий, связанных с кнопкой Start/Pause	533
Реализация кнопки Start/Pause	534
Другой тип событий	535
А как же метод <code>after()</code> ?	537
Непосредственное редактирование клеток	540
Обработчик <code>grid_handler()</code>	541
Добавление шаблонов	542
Обработчик событий для объекта <code>OptionMenu</code>	543
Создание собственных фигур	545
Загрузчик шаблонов	546
Улучшаем симулятор игры “Жизнь”	553



12 Путешествие в страну объектов

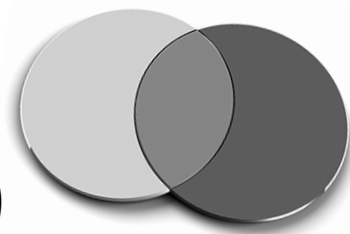
В предыдущих главах для создания абстракций в коде применялись **функции**. Мы всячески придерживались **процедурного** подхода, используя в функциях простые выражения, условные конструкции и циклы `for/while` — ничего такого, что относилось бы к **объектно-ориентированному** программированию. Конечно, вы познакомились с объектами и научились применять их, но вам еще ни разу не приходилось создавать классы и писать полностью объектно-ориентированный код. Пришло время попрощаться со скучным процедурным прошлым. Нас ожидает новый мир, полный объектов и надежд на лучшую жизнь.

Другой подход к структурированию программ	560
Преимущества объектно-ориентированного программирования	561
Разработка своего первого класса	563
Написание кода класса	564
Как работает конструктор	564
Метод <code>bark()</code>	567
Как работает метод	568
Немного о наследовании	570
Реализация класса <code>ServiceDog</code>	571
Подклассы	572
<code>ServiceDog</code> — это потомок класса <code>Dog</code>	573
Проверка принадлежности к классу	574
Переопределение и расширение поведения	578
Учим термины	580
Объекты внутри объектов	582
Создание отеля для собак	585
Реализация отеля для собак	586
Усовершенствование отеля для собак	589
Расширение функций отеля	590
Мы с тобой одной крови: полиморфизм	591
Учимся выгуливать собак	592
Сила наследования	594
Служба выгула собак	595
Как нанять человека для выгула собак	596



приложение: не попавшее в избранное

Десять ключевых тем (которые не были рассмотрены)



Вы прошли длинный путь и находитесь у финишной черты. Нам искренне жаль расставаться с вами, и, прежде чем отправить вас в самостоятельное плавание, нам бы хотелось дать последние наставления. Конечно, в одном маленьком приложении невозможно охватить все необходимое, хотя мы попытались это сделать, уменьшив размер шрифта до 0,00004 пункта. К сожалению, никому в издательстве так и не удалось прочитать столь микроскопический текст, поэтому нам пришлось выкинуть все лишнее, оставив только самое важное.

1. Списковые включения	612
2. Работа с датой и временем	613
3. Регулярные выражения	614
4. Другие типы данных: кортежи	615
5. Другие типы данных: множества	616
6. Написание серверного кода	617
7. Отложенные вычисления	618
8. Декораторы	619
9. Объекты первого класса и функции высшего порядка	620
10. Важные библиотеки	621



Ждем ваших отзывов!

Вы, читатель этой книги, и есть главный ее критик. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересны любые ваши замечания в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам электронное письмо либо просто посетить наш сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится ли вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Отправляя письмо или сообщение, не забудьте указать название книги и ее авторов, а также свой обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию новых книг.

Наши электронные адреса:

E-mail: info@dialektika.com

WWW: <http://www.dialektika.com>