Use a Cabeça! Aprenda a Programar

Não seria ótimo se existisse um livro para aprender a programar que fosse mais divertido do que ir ao dentista e mais revelador do que uma declaração do imposto de renda? Provavelmente é só um sonho...



Eric Freeman



pensando computacionalmente

Começando

Saber como pensar computacionalmente o deixa no controle. Não é segredo que o mundo à nossa volta está cada vez mais conectado, configurável, programável, e mais computacional. Você pode continuar como um participantes passivo ou pode aprender a programar. Quando sabemos programar, somos os diretores, os criadores — dizemos a todos os computadores o que deveriam fazer para nós. Quando sabemos programar, podemos controlar nosso próprio destino (ou, pelo menos, conseguiremos programar o sistema de irrigação do jardim conectado à internet)

Mas como aprendemos a programar? Primeiro, devemos aprender a pensar computacionalmente. Depois, pegamos uma linguagem de programação para falar a mesma linguagem do computador, dispositivo móvel ou qualquer coisa com uma CPU. O que ganhamos com isso? Mais tempo, poder e possibilidades criativas para fazer o que realmente queremos. Vamos lá, vamos começar...

Separando em partes	2
Como funciona a programação	6
Estamos mesmo falando a mesma língua?	7
O mundo das linguagens de programação	8
Como escrever e rodar código com Python	13
Uma história muito breve sobre o Python	15
Testando o Python	18
Salvando seu trabalho	20
Parabéns por programar seu primeiro código Python!	21
Phrase-O-Matic	25
Colocando o código na máquina	26

valores simples, variáveis e tipos

Conheça Seu Valor Os computadores só fazem duas coisas direito:

armazenam valores e executam operações sobre eles. Você pode achar que estão fazendo muito mais coisas quando envia mensagens, faz compras online, usa o Photoshop ou depende do seu telefone para usar o GPS; no entanto, tudo o que os computadores fazem pode ser dividido em operações simples que são executadas sobre valores simples. Agora, parte do pensamento computacional é aprender a usar essas operações e valores para criar algo muito mais sofisticado, complexo e significativo — e vamos chegar a isso. Mas primeiro vamos dar uma olhada no que são esses valores, nas operações que podemos executar sobre eles e qual é o papel das variáveis em tudo isso.

Programando a Calculadora da Idade Ca	anina	
Passando do pseudocódigo para o código		
Passo 1: Recebendo input		
Como funciona a função input		
Usando variáveis para lembrar e armazer	nar val	lores
Atribuindo a entrada do usuário à variáv	el	
Passo 2: Obtendo mais input		
É hora de rodar o código		
Inserindo código		
Mergulhando mais fundo nas variáveis		
Adicionando expressões	45	
Variáveis são chamadas de VARI-áveis		
por uma razão	46	
Vivendo melhor com a precedência		
de operadores	47	
Calculando com a precedência de	40	
operadores	48	
Afaste-se desse teclado!	51	
Passo 3: Calculando a idade do cachorro	52	A
Houston, temos um problema!	53	
Errar é humano programar	54	
Mais um pouco de depuração	56	
E quais são os tipos em Python?	58	
Corrigindo nosso código	59	
Houston, temos um plano	60	- 1
Passo 4: Output amigável	61	`



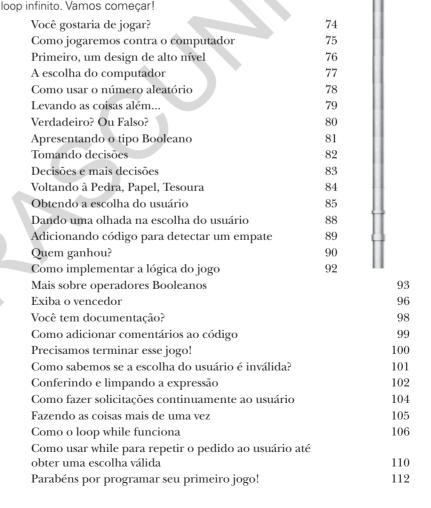
3

booleanoos, decisões e loops

Código Decisivo Até agora, você já percebeu o quanto nossos programas não são muito

VERDADEIRO

interessantes? Isto é, todo o nosso código tem sido um conjunto de declarações que o interpretador avalia do início ao fim — sem reviravoltas, sem voltas repentinas, sem surpresas, sem pensamento independente. Para que o código seja mais interessante, ele precisa tomar decisões, controlar seu próprio destino, e fazer coisas mais de uma vez só. E neste capítulo é exatamente isso que aprenderemos a fazer. No caminho aprenderemos sobre o jogo misterioso chamado shoushiling, conheceremos um personagem chamado Boole e veremos como um tipo de dado com apenas dois valores pode valer nosso tempo. Também aprenderemos a lidar com o temido





listas e iteração

Fornecendo Estrutura

Há mais tipos de dados do que apenas números, strings e

Booleanos. Até agora escrevemos código Python usando tipos primitivos — floats, integers, strings e Booleanos, é claro — com valores como 3.14, 42, "hey, it's my turn" e True. E você pode fazer muita coisa com os primitivos, mas em algum ponto vai querer escrever um código que lide com muitos dados — digamos, todos os itens em um carrinho de compras, os nomes de todas as estrelas notáveis ou um catálogo inteiro de produtos. Para isso, precisamos de um pouco mais de emoção. Neste capítulo, veremos um novo tipo, chamado lista, que pode armazenar uma coleção de valores. Com as listas, você será capaz de fornecer estrutura para seus dados, em vez de ter zilhões de variáveis flutuando pelo código com valores. Você também aprenderá a tratar todos esses valores como um todo, bem como a iterar sobre cada item em uma lista usando o loop for mencionado no último capítulo. Depois deste capítulo, sua habilidade de lidar com dados se desenvolverá.

Você pode ajudar a Bubbles-R-Us?			126
Como representar vários valores em Py	thon		127
Como as listas funcionam			128
Como acessar um item da lista			129
Atualizando um valor na lista	129		
Afinal, qual é o tamanho dessa lista?	131		
Acessando o último item da lista	132		
Usando os índices negativos			
do Python	133		
Enquanto isso, na Bubbles-R-Us	135		
Como iterar sobre uma lista	138	Same a	
Corrigindo a falha técnica da output	139	THE COLUMN TO SELECT	- 70
Corrigindo de verdade a falha			
écnica da output	140		
O loop for, o modo preferido de iterar	sobre uma l	ista	142
Como o loop for funciona em um inter	valo de nún	neros	145
Fazendo mais com as ranges			146
untando tudo			148
Test drive do relatório de bolhas			149
Conversa de cubículo, continuação			153
Criando sua própria lista, do zero			156
Fazendo ainda mais com listas as			157
Test drive do relatório final			161
E os vencedores são			161





funções e abstração

Funcionalizando

Você já sabe muito. Variáveis, tipos de dados, condicionais e iterações — isso é o suficiente para escrever basicamente qualquer programa que você queira. Na verdade, um cientista da computação diria que é o bastante para criar qualquer programa que qualquer pessoa possa imaginar. Mas você não quer parar agora, porque o próximo passo no pensamento computacional é aprender a criar abstrações em seu código. Isso pode parecer complicado, mas, na verdade, facilitará sua vida de programador. Criar abstrações lhe dá vantagens; com a abstração, é possível criar programas exponencialmente mais complexos e robustos de modo mais fácil. Você pode colocar seu código em pacotinhos organizados que podem ser reutilizados continuamente. E você pode esquecer de todos os detalhes essenciais do seu código e começar a pensar grande.

Mas qual é o problema do código?	181
Transformando um bloco de código em uma FUNÇÃO	183
Criamos uma função, e agora, como usá-la?	184
Mas como tudo isso realmente funciona?	184
Funções também podem RETORNAR coisas	192
Como chamar uma função que tem um valor return	193
Aprendendo um pouco de refatoração	195
Rodando o código	196
Como abstrair o código do avatar	197
Escrevendo o corpo da função get_attribute	198
Chamando a get_attribute	199
Vamos falar mais um pouco sobre variáveis	201
Entendendo o escopo da variável	202
Quando as variáveis são passadas para funções	203
Chamando a função drink me	204
E quanto a usar variáveis globais em funções?	207
Indo além com parâmetros: valores padrões e	
palavras-chave	210
Sempre liste primeiro seus parâmetros requeridos	211
Usando argumentos com palavras-chave	212
Como pensar em todas essas opções	212



4 parte 2

ordenação e iteração aninhada

Colocando Ordem nos Seus Dados

Às vezes, a ordem padrão dos seus dados não serve. Existe aquela lista de high scores em fliperamas dos anos 80, mas precisamos que ela seja classificada em ordem alfabética pelo nome do jogo. E existe aquela lista de número de vezes que seus colegas de trabalho passaram a perna em você — seria legal saber quem está no topo dessa lista. Mas, para isso, precisamos aprender a ordenar os dados, e para fazer isso precisamos explorar alguns algoritmos um pouco mais complicados do que os que vimos até aqui. Precisaremos também explorar como esses loops aninhados funcionam, bem como, pensar um pouco mais sobre a eficácia do código que estamos escrevendo. Vamos lá, vamos subir de nível em em nosso pensamento computacional!

Entendendo o bubble sort	228
Um pouco de pseudocódigo de bubble sort	231
Implementando bubble sort em Python	234
Computando os números das soluções de bolhas	236

Classificação
integrada! Talvez
você pudesse ter
dito isso 10 páginas
antes?







texto, strings e heuristicas

Juntando Tudo

Você já tem muitos superpoderes. Agora é hora de usá-los. Neste capítulo, integraremos o que aprendemos até agora, juntando tudo em um tipo de código cada vez mais legal. Continuaremos também a agregar conhecimento e técnicas de programação. Mais especificamente, neste capítulo exploraremos como escrever código que pega um pouco de texto, divide-o e, então, faz um pouco de análise de dados nele. Descobriremos, também, o que é uma heurística, e implementaremos uma. Prepare-se — este é um capítulo seríssimo, completo, focado e dinâmico de programação!

Seja bem-vindo as ciencias de dados	246
Como calcular algo como a legibilidade?	247
O plano de jogo	248
Escrevendo o pseudocódigo	249
Precisamos de um texto para analisar	250
Configurando a função	252
Primeiro: precisamos do total de palavras no texto	253
Calculando o número total de frases	257
Escrevendo a função count_sentences	258
Calculando o número de sílabas ou aprendendo a amar	
as heurísticas	264
Configurando a heurística	267
Escrevendo a heurística	268
Como contar vogais	269
Ignorando as vogais consecutivas	269
Escrevendo o código para ignorar vogais consecutivas	270
Removendo o e final, y e pontuações	272
Trabalhando com fatias (substrings)	274
Terminando o código heurístico	276
Implementando a fórmula de fácil leitura	278
Indo além	283

Definitivamente uma linguagem sofisticada neste livro.





módulos, métodos, classes e objetos

Modulando

Seu código está crescendo em tamanho e complexidade. Como podemos ver, precisamos de maneiras melhores de abstrair, modularizar e organizar o código. Vimos que as funções podem ser usadas para agrupar

organizar o codigo. Vimos que as runções podem ser usadas para agrupar linhas de código em pacotes reutilizáveis. E também vimos que coleções de funções e variáveis podem ser colocadas em módulos para que possam ser compartilhadas e reutilizadas mais facilmente. Neste capítulo, revisitaremos os módulos e aprenderemos a usá-los com mais eficácia ainda (estaremos prontos para compartilhar código com os outros) e, então, veremos o máximo do reuso de código: *objetos*. Veremos que os objetos Python estão por todos os lugares, apenas esperando para serem usados.

Uma revisão rápida de módulos	294
A variável globalname	296
Atualizando o analyze.py	297
Usando o analyze.py como um módulo	299
Incluindo as docstrings em analyze.py	301
Explorando outros módulos Python	305
Espera, alguém disse "tartarugas"?!	306
Criando seu próprio turtle	308
Laboratório de tartarugas	309

Adicionando um segundo turtle 311 Afinal, o que são turtles? 314 O que são objetos? 315 316 Tudo bem, então o que é uma classe? Uma classe nos diz o que um objeto sabe e o que ele pode fazer 317 Como usar os objetos e as classes 318 E quanto aos métodos e atributos? 319 Vendo classes e objetos por todas as partes 320 399 Prepare-se para uma corrida de tartarugas 323 Planejando o jogo Vamos começar a programar 324

Bom trabalho, fui capaz de usar o módulo de análise rapidamente, especialmente com a ajuda da ótima documentação!



CRIME SCENE DO NOT ENTER

Configurando o jogo

Começando a corrida

Não tão rápido!

Escrevendo o código de configuração



324 325

326

328

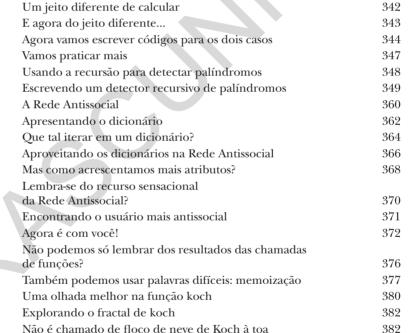


recursão e dicionários

Além de Iteração e Índices

É hora de elevar o nível do nosso pensamento computacional.

E este é o capítulo para fazer isso: estivemos felizes da vida programando com um estilo iterativo de programação — criamos estruturas de dados como listas e strings, e intervalos de números, e escrevemos código para calcular iterando sobre elas. Neste capítulo, veremos o mundo de maneira diferente, primeiro em termos de computação, e depois, em termos de estruturas de dados. Computacionalmente, veremos um estilo de computação que implica em escrever um código que retorna, ou chama a si mesmo. Ampliaremos os tipos de estruturas de dados com as quais conseguimos trabalhar observando um tipo de dado similar a um dicionário, mais parecido com um *mapa associativo* do que uma lista. Vamos juntá-los e causar todos os tipos de problemas. Fique esperto: esses tópicos demoram um pouco para serem absorvidos, mas o esforço vale a pena.







salvando e recuperando arquivos

Persistência

Você sabe que pode salvar valores em variáveis, mas uma vez que seu programa é finalizado, puf! — eles somem para

sempre. É aí que o armazenamento persistente entra em ação — ele permite que seus valores e dados durem um pouco mais. A maioria dos dispositivos que executam Python também tem armazenamento persistente, como hard drives e flash cards, ou podem ter acesso ao armazenamento na nuvem também. Neste capítulo, veremos como escrever o código para armazenar e recuperar dados de arquivos. Para que serve isso? AH, sempre que você quiser salvar a configuração de um usuário, os resultados da sua análise enorme para o chefe, ler uma imagem em um código para processá-la, escrever o código para pesquisar por uma década de e-mails, reformatar alguns dados para que sejam usados em um app de planilhas — essa lista não tem fim.

Pronto para um Crazy Libs?	394
Como o Crazy Libs funcionará	396
Passo 1: Leia o texto da história a partir de um arquivo	399
Como usar os caminhos	400
Caminhos relativos	400
Caminhos absolutos	401
Ah, e não se esqueça de limpar tudo quando acabar!	402
Lendo um arquivo em seu código Python	403
Usando o objeto de arquivo para ler um arquivo	403
Ah, me dá um tempo	406
Ei, temos um jogo Crazy Lib para terminar!	407
Como sabemos que lemos a última linha?	409
Lendo em um template Crazy Lib	410
Processando o template text	411
Usando um novo método string para corrigir o bug	413
Corrigindo o bug	414
Alguns códigos têm problemas sérios	415
Lidando com exceções	417
Lidando explicitamente com as exceções	418
Atualizando Crazy Libs para lidar com exceções	420
Nosso último passo: armazenando o Crazy Lib	421
Atualizando o resto do código	421



usando web apis

Você Realmente Deveria Sair Mais

Você tem escrito códigos ótimos, mas realmente deveria sair

mais. Há um mundo inteiro de dados esperando por você na web: precisa de dados sobre o tempo? Ou que tal acessar uma base de dados enorme de receitas? Ou você prefere pontuações sobre esportes? Talvez uma base de dados musical de artistas, álbuns e canções? Eles estão todos disponíveis por aí em Web APIs. Para usá-los, tudo o que você precisa é aprender um pouco mais sobre como a web funciona, como falar o dialeto local da web e como usar alguns módulos Python novos: requests e json. Neste capítulo, exploraremos as Web APIs e levaremos nossas habilidades Python a novos patamares; na verdade, as levaremos em uma viagem de ida e volta do espaço sideral.

436
437
438
441
442
443
444
445
447
449
450
451
453
455
456
457
458



468

469



Widgets, eventos e comportamento emergente Interagindo

Vida artificial

Mais detalhes sobre o Jogo da Vida

Escrevendo o carregador de padrão

Mais Além!

Você certamente já escreveu aplicações gráficas, mas ainda não criou uma interface do usuário de verdade. Ou seja, você ainda não escreveu nada que permita que o usuário interaja com uma interface gráfica de usuário (também conhecida como GUI). Para isso, você precisa adotar uma nova maneira de pensar sobre como um programa é executado, uma que seja mais reativa. Espera, o usuário acabou de clicar naquele botão? É bom que seu código saiba como reagir e o que fazer em seguida. Programar interfaces é bem diferente do método processual típico que usamos até agora, e isso exige uma maneira diferente de pensar o problema. Neste capítulo, você escreverá sua primeira GUI de verdade, e não, não vamos escrever uma lista de afazeres simples de gestão ou uma calculadora de altura/ peso, vamos fazer algo muito mais interessante. Vamos escrever um simulador de vida artificial com comportamento emergente.





510

517

12

programação orientada a objetos

Uma viagem à Objetolândia

Neste livro você usou funções para abstrair seu código. E

abordou a programação de maneira processual usando declarações simples, condicionais e loops for/while com funções — nada disso é exatamente orientado a objeto. Na verdade, não é nada orientado a objeto! Observamos os objetos e como usá-los em nosso código, mas não criamos nenhum objeto próprio ainda, e não abordamos a criação do código de maneira orientada a objeto. Então chegou a hora de deixar para trás a cidade processual chata. Neste capítulo, você descobrirá por que usar objetos facilitará a sua vida — bem, pelo menos, no sentido da programação (não podemos ajudá-lo em outras áreas da sua vida e com suas habilidades de programação em um livro só).

Dividindo, de um jeito diferente	524
Afinal de contas, qual é o propósito da programação	
orientada a objetos?	525
Projetando sua primeira classe	527
Escrevendo sua primeira classe	528
Escrevendo o método bark	531
Como os métodos funcionam	532
Incluindo heranças	534
Implementando a classe ServiceDog	535
Olhando atentamente as subclasses	536
Um ServiceDog É-UMA Dog	537
Testando É-UMA no código	538
Como você se descreveria?	541
Sobrepondo e estendendo o comportamento	542
Seja bem-vindo a Campos do Jargão	544
O objeto pode TER-UM outro objeto	546
Projetando um Hotel para Cachorros	549
Implementando o Hotel para Cachorros	550
Renovando o Hotel para Cachorros	553
Incluindo algumas atividades no hotel	554
Eu posso fazer tudo o que você pode, ou Polimorfismo	555
Já é hora de ensinar os outros cachorros a andar	556
O poder da herança	558

apendice: remanescente





Nós tratamos de vários assuntos e o livro está quase no fim.

Vamos sentir saudades, porém antes de ir embora, não nos sentiríamos bem em deixá-lo se aventurar no mundo real sem um pouco mais de preparação. É impossível colocar tudo o que você precisa saber neste capítulo relativamente pequeno. Na verdade, originalmente, incluímos tudo o que você precisa saber sobre programação Python (ainda não tratado nos outros capítulos), reduzindo o tamanho da fonte para 0,00004. Coube tudo, mas ninguém conseguiria ler. Então, jogamos boa parte fora e mantivemos as melhores partes para este apêndice de Dez Assuntos Principais.

Este é realmente o fim do livro. Sem contar o índice, é claro (leitura obrigatória!).

Nº. 1 Compreensão de lista	576
Nº. 2 Datas e horas	577
Nº. 3 Expressões regulares	578
Nº. 4 Outros tipos de dados: tuples	579
Nº. 5 Outros tipos de dados: conjuntos	580
Nº. 6 Código de servidor	581
Nº. 7 Avaliação preguiçosa	582
N°. 8 Decoradores	583
Nº, 9 Funções de ordem superior e de primeira classe	584
Nº. 10 Um monte de bibliotecas	585

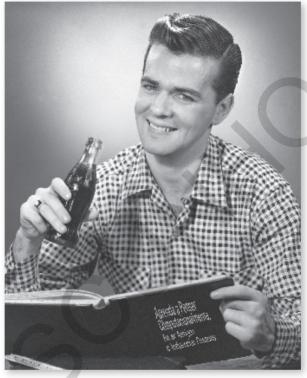


Código do lado do cliente é executado no cliente — ou seja, no seu computador.

1 pensando computacionalmente

Começando





Saber como pensar computacionalmente o deixa no controle. Não é segredo que o mundo à nossa volta está cada vez mais conectado, configurável, programável, e mais computacional. Você pode continuar como um participantes passivo ou pode aprender a programar. Quando sabemos programar, somos os diretores, os criadores — dizemos a todos os computadores o que deveriam fazer para nós. Quando sabemos programar, podemos controlar nosso próprio destino (ou, pelo menos, conseguiremos programar o sistema de irrigação do jardim conectado à internet) Mas como aprendemos a programar? Primeiro, devemos aprender a pensar computacionalmente. Depois, pegamos uma linguagem de programação para falar a mesma linguagem do computador, dispositivo móvel ou qualquer coisa com uma CPU. O que ganhamos com isso? Mais tempo, poder e possibilidades criativas para fazer o que realmente queremos. Vamos lá, vamos começar...

Separando em partes

O primeiro obstáculo entre você e escrever seu primeiro pedaço de código de verdade é aprender a habilidade de separar os problemas em pequenas ações executáveis que um computador possa fazer por você. É claro que você e o computador também precisarão falar a mesma língua, mas daqui a pouco entramos nesse assunto.

Separar os problemas em vários passos pode até parecer uma habilidade nova, mas, na verdade, é algo que você faz todos os dias. Vejamos um exemplo simples: digamos que você queira dividir a atividade de pescar em um conjunto de instruções simples para que um robô pesque para você. Veja sua primeira tentativa de fazer isso:



Vamos dividir o processo de pescar um peixe em passos facilmente compreensíveis.

,)

Seguimos os passos na ordem.

- Coloque a minhoca no anzol.
- Arremesse a linha no lago. u

Alguns passos são instruções simples, ou declarações se preferir, como "arremesse a linha no lago."

Observe a boia até que ela seja puxada para dentro da água.

Uma declaração pode esperar condicionalmente antes de continuar.

Puxe a linha e o peixe.

Esta declaração só acontece depois que a boia ficou submersa na declaração anterior.

Se acabou de pescar, volte para casa; se não, volte ao passo 1.

Declarações também podem tomar decisões, como: é hora de ir para casa ou deveríamos continuar pescando?

A boia

2 0 anzol

A minhoca

Note que as declarações se repetem com frequência, como aqui: se não formos para casa, em vez disso, voltamos ao início e repetimos as instruções para pegar outro peixe.

Você pode pensar nessas declarações como uma receita divertida para pescar. Como qualquer receita, esta fornece um conjunto de passos que, quando seguidos em ordem, produzirão um resultado (no nosso caso, esperamos pegar alguns peixes).

Note que a maioria dos passos consiste em uma instrução simples, como "arremesse a linha no lago" ou "puxe o peixe". Mas veja também que outras instruções são um pouco diferentes, porque dependem de uma condição, como "a boia está acima ou dentro da água?" Instruções também podem direcionar o andamento da receita, como "se não acabou de pescar, então volte ao começo e coloque outra minhoca no anzol". E que tal uma condição para parar de pescar, como em "se terminou, vá para casa"?

Você verá que essas declarações ou instruções simples são a base da programação. Na verdade, todo app ou software que você já usou não são nada mais, nada menos, do que um conjunto (às vezes grande) de instruções que dizem ao computador o que fazer.



Exercício

As receitas não dizem só o que fazer, mas incluem objetos usados para fazer um prato específico (como medidores, batedores, processadores de alimentos e, é claro, ingredientes). Quais objetos são usados em nossa receita de pescaria? Circule todos os objetos na receita da página anterior e verifique sua resposta no final do capítulo antes de continuar.

Isto é um livro de exercícios. Nós o incentivamos a escrever nele

Caso não esteja familiarizado com a pesea, isto é uma boia. Quando um peixe é fisqado, ela afunda.

Aponte o seu lápis

Uma coisa que você deve entender de imediato é que os computadores fazem **exatamente** o que dizemos a eles — nem mais, nem menos. Veja nossa receita de pescaria na página anterior. Se você fosse um robô e seguisse essas instruções precisamente, quais problemas poderia encontrar? Você acha que teríamos sucesso usando essa receita?

Você conseque

pensar em mais problemas?

- A. Se não houver peixes, ficaremos pescando por muito tempo (tipo, para sempre).
- B. Se a minhoca cair do anzol, nunca saberemos ou a substituiremos.
- C. O que acontece se as minhocas acabarem?

D. Especificamos o que fazer com o peixe depois de puxá-lo?

E. O que aconteceu com a vara de pescar?

F. _____

Você descobrirá as respostas no final do capítulo. Eu comprei este livro técnico caro para aprender a programar e você está me falando de receitas? Isso não parece nada promissor ou, bem, técnico.



Assim como há várias receitas para o mesmo prato, com os algoritmos você verá que há muitas maneiras de resolver o mesmo problema. Algumas mais saborosas que as outras.

Na verdade, uma receita é uma maneira absolutamente perfeita de descrever um conjunto de instruções para um computador. Você pode até encontrar aquele termo vagamente usado aqui e ali em livros de programação avançada. Você pode até encontrar livros sobre técnicas comuns de desenvolvimento de software que são chamados de livros de receitas (cookbooks). Dito isso, se quiser que sejamos técnicos, vamos lá — um cientista da computação ou um desenvolvedor de software sério normalmente chamaria uma receita de algoritmo. O que é um algoritmo? Bem, nada mais do que uma receita — é uma sequência de instruções que resolvem algum problema. Vemos com frequência que os algoritmos são escritos primeiro na forma de código informal, chamada de pseudocódigo.

Quando falamos de receita, pseudocódigo ou algoritmo não podemos esquecer que o principal objetivo é desenvolver uma descrição de alto nível de como resolver um problema antes de entrar nos detalhes de escrita do código que o computador consegue entender e executar.

Neste livro, você nos verá usar todos esses termos como sinônimos — e, ah, em sua próxima entrevista de trabalho talvez você queira usar o termo *algoritmo* ou até *pseudocódigo* para garantir um bônus maior de contratação no salário inicial (porém, ainda não há nada de errado com a palavra *receita*).

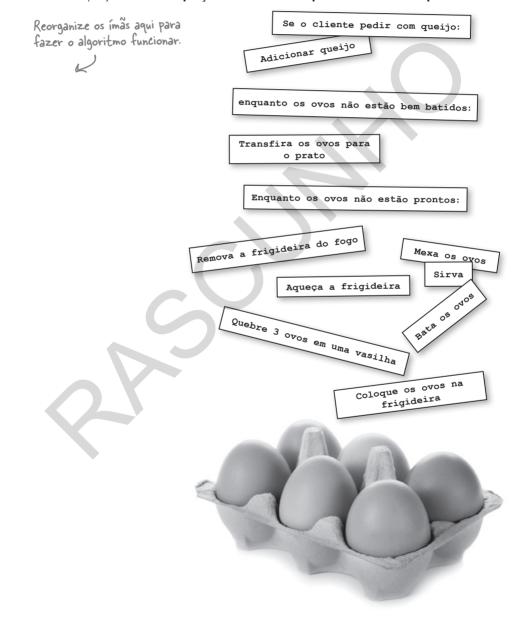
Já falaremos mais sobre pseudocódigo.

Como veremos, isso pode tornar a tarefa de programar mais direta e menos propensa ao erro.



Ímãs de Código

Vamos praticar um pouco as receitas os algoritmos. Colocamos o algoritmo da Lanchonete Use a Cabeça! para fazer um omelete de três ovos na geladeira para lembrar dele, mas alguém bagunçou tudo. Você consegue colocar os ímãs em ordem novamente para que o algoritmo funcione? Note que a Lanchonete Use a Cabeça! faz dois tipos de omeletes: normal e com queijo. **Não se esqueça de conferir a resposta no final do capítulo.**



Como funciona a programação

Então você quer que o computador faça uma tarefa para você, e sabe que precisará dividi-la em várias instruções para que o computador possa entender, mas como *realmente dizer* ao computador para fazer algo? É aí que entra a linguagem de programação — com ela é possível descrever sua tarefa em termos que *você e o computador* entendam. Mas antes de entrarmos de cabeça nas linguagens de programação, vamos ver os passos que você precisa seguir para escrever código:



Crie seu algoritmo

É aqui que você pega o problema ou tarefa a ser resolvido e o transforma em uma receita de alto nível, pseudocódigo ou algoritmo que descreve os passos que o computador deverá realizar para alcançar o resultado desejado.



Arremesse a linha no lago.

Observe a boia até que ela seja puxada para dentro da água.

Puxe a linha e o peixe.

Se acabou de pescar, volte para casa; se não, volte ao passo 1.

Este é o passo em que mapeamos nossa solução antes de fazer a tradução para a linguagem de programação.



Em seguida, pegue a receita e a traduza para um conjunto de instruções específico escrito em uma linguagem de programação. Este é a etapa de *programação*, e chamamos o resultado de *programa* ou só de "seu código" (ou, mais formalmente, *código fonte*).



gar minhoca

Arremessar a isca A boia afundou?

Colocar minhoca no anzol

Este é o
passo em que
"programamos",
onde você
transforma
o algoritmo
em código (ou
código fonte)
que estará
pronto para ser
executado no
próximo passo.

Execute o programa

Por fim, pegue o código fonte e passe-o ao computador, que começará a executar as instruções. Dependendo da linguagem usada, este processo pode ser chamado de interpretar, rodar, avaliar ou executar o código.

Também usamos esses termos como sinônimos com frequência Quando o código fonte fica pronto, você pode executá-lo. Se tudo correr bem e ele tiver sido bem programado, o computador lhe dará o resultado desejado.

Estamos mesmo falando a mesma língua?

Pense em uma linguagem de programação como uma linguagem com propósito especial criada explicitamente para transmitir tarefas a um computador. As linguagens de programação lhe dão um modo de descrever as receitas de maneira clara e precisa o suficiente para que um computador possa entendê-las.

Para aprender uma linguagem de programação há duas coisas que você precisa saber: o que você pode dizer com a linguagem e o que elas significam? Um cientista da computação chamaria isso de **sintaxe** ou **semântica** da linguagem. Por ora, apenas guarde esses termos na caixola; vamos deixá-lo a par de tudo no decorrer do livro.

Agora, conforme podemos ver, assim como as línguas faladas, há *muitas* linguagens de programação e, como você já deve ter percebido, neste livro, usaremos a linguagem de programação Python. Vamos dar uma olhada melhor nas linguagens e em Python...



esperamos que você leia ou escreva código. Minha nossa, você ainda tem o livro inteiro pela frente — por enquanto, estamos apenas nos familiarizando com o código, como ele é e como funciona. O importante neste capítulo é só internalizar tudo.

Você verá que as técnicas aprendidas neste livro podem ser aplicadas a quase qualquer linguagem de programação que você poderá encontrar no futuro.

CORRESPONDENTES

À esquerda, temos algumas declarações escritas em português, e à direita, as declarações escritas em linguagem de programação. Ligue cada uma das declarações em inglês à sua tradução correspondente em código. Fizemos a primeira para você. Lembre-se de conferir a solução no final do capítulo antes de seguir em frente.

Escreva "Olá" na tela.

Se a temperatura for maior do que 22 graus, então escreva "Vestir shorts" na tela. Uma lista de compras com pão, leite e ovos.

Servir cinco bebidas.

Perguntar ao usuário: "Qual é o seu nome?"

O mundo das linguagens de programação

Se você está lendo este livro já deve ter ouvido falar de várias linguagens de programação. Apenas passando por uma seção de livros de programação em uma livraria podemos encontrar Java, C, C++, LISP, Scheme, Objective-C, Perl, PHP, Swift, Clojure, Haskell, COBOL, Ruby, Fortran, Smalltalk, BASIC, Algol, JavaScript e, é claro, Python, para citar algumas. Você também pode estar se perguntando de onde vieram todos esses nomes. A verdade é que os nomes das linguagens de programação são como nomes de bandas de rock: significam algo para as pessoas que criaram a linguagem. Veja o Java, por exemplo: o nome vem do café (o nome preferido, Oak, já havia sido usado). Haskell recebeu o nome em homenagem a um matemático, e o nome C foi escolhido porque C foi a sucessora de A e B na Bell Labs. Mas por que há tantas linguagens e sobre o qual a finalidade delas? Vejamos o que alguns amigos nos dizem sobre as linguagens que usam:

Sou do time Objective-C. Desenvolvo apps de iPhone o dia todo e adoro como ela é parecida com C, só que mais dinâmica e orientada a objetos. Também estou aprendendo uma nova linguagem Apple chamada Swift.

Java me faz pensar no nível dos objetos, não em código de baixo nível, e relaciona para mim muitas coisas de baixo nível, como gestão de memória e threading

Eu vivo no mundo do WordPress, que é escrito em PHP, então PHP é a minha linguagem de referência. Algumas pessoas a chamam de linguagem de script, mas ela faz tudo o que eu preciso.



pensando computacionalmente

Eu uso principalmente **C**. Escrevo partes de sistemas operacionais que precisam ser supereficientes. No meu trabalho, todo ciclo de CPU e toda endereçamento de memória é importante.



Você pode dizer que sou acadêmico, mas eu adoro linguagens no estilo Scheme e LISP. Para mim, tudo se trata de funções de ordem elevada e abstração. Fico feliz em ver linguagens funcionais como Clojure sendo realmente usadas na indústria.



Sou desenvolvedora web e **JavaScript** é minha linguagem principal. É a linguagem de fato de todos os navegadores e também está sendo usada para escrever serviços web de backend.



Sou administrador de sistemas e uso muito Perl para escrever vários scripts de sistema. Ela é concisa, mas muito expressiva também. Só um pouco de código já faz muita coisa.



Nós amamos Python. É uma linguagem conhecida por ser muito legível e limpa, com um ótimo suporte de bibliotecas que o permite escrever código para todos os tipos de domínio; há também uma comunidade enorme de pessoas envolvidas com ela.

Python é conhecida como uma das melhores linguagens para iniciantes, ela evolui com você à medida que suas habilidades amadurecem. É também uma linguagem real, utilizada pelo Google, Disney e NASA para desenvolver sistemas sérios.



Escolhas, escolhas...

Como você pode ver, há muitas linguagens e opiniões por aí, e nós mal falamos das linguagens modernas. Você também pode ver que há muitas terminologias que vêm com essas linguagens e, à medida que você progride, esses termos farão mais sentido. Por enquanto, saiba que há várias linguagens por aí, e mais são criadas todos os dias.

Logo, neste livro, quais delas deveríamos usar? A questão é: antes de tudo, queremos aprender a pensar *computacionalmente* — independente da linguagem que você se deparar no futuro, estará preparado para aprendê-la. Mas, dito isso, precisamos começar com *alguma linguagem* e, como você já sabe, usaremos Python. Por quê? Nossos amigos acima falaram tudo: ela é considerada uma das melhores linguagens para iniciantes por ser muito legível e consistente. É também uma linguagem poderosa, pois não importa o que você quiser fazer com ela (agora ou depois de ler este livro), conseguirá encontrar suporte em termos de extensões de código (chamados de *módulos* ou *bibliotecas*) e uma comunidade solidária de desenvolvedores disposta a ajudá-lo. Por fim, alguns desenvolvedores dirão até que Python é *mais divertida* que outras linguagens. Então, o que poderia sair de errado?

Aponte o seu lápis

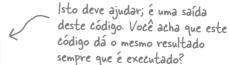
Veja como é fácil escrever Python

Você ainda não sabe Python, mas podemos apostar que você até tem uma boa ideia de como o código Python funciona. Dê uma olhada em cada linha do código abaixo e veja se consegue adivinhar o que ela faz. Escreva as respostas abaixo. Se não souber, as respostas estão na próxima página. Fizemos a primeira para você.

<pre>customers = ['Jimmy', 'Kim', 'John', 'Stacie']</pre>	Faz uma lista de clientes.
<pre>winner = random.choice(customers)</pre>	
flavor = 'vanilla'	
<pre>print('Congratulations ' + winner + ' you have won an ice cream sundae!')</pre>	
<pre>prompt = 'Would you like a cherry on top?'</pre>	
<pre>wants cherry = input(prompt)</pre>	
order = flavor + ' sundae '	
<pre>if (wants_cherry == 'yes'): order = order + ' with a cherry on top'</pre>	
mint/long L. Landan L. L. San L. L. Laire	
<pre>print('One ' + order + ' for ' + winner +</pre>	

Python Cutpot

Congressionies State you have win as the cream suntae?
Whold you like a cherry on top? you
One wantile mundes with a cherry on top for State coming
right up ...



Aponte o seu lápis Solução

Veja como é fácil escrever Python

Você ainda não sabe Python, mas podemos apostar que você até tem uma boa ideia de como o código Python funciona. Dê uma olhada em cada linha do código abaixo e veja se consegue adivinhar o que ela faz. Escreva as respostas abaixo. Se não souber, as respostas estão na próxima página. Fizemos a primeira para você.

customers = ['Jimmy', 'Kim', 'John', 'Stacie']

winner = random.choice(customers)

flavor = 'vanilla'

print('Congratulations ' + winner +
 ' you have won an ice cream sundae!')

prompt = 'Would you like a cherry on top?'

wants _ cherry = input(prompt)

order = flavor + ' sundae

if (wants_cherry == 'yes'):
 order = order + ' with a cherry on top'

Python Output

Congratulations Stanle you have won an ice cream sundeef. Would you like a cherry on topf you

One vanilla sundae with a cherry on top for Stanie omning right up... Faz uma lista de clientes.

Escolhe um dos clientes aleatoriamente.

Configura o nome ou variável chamada flavor como o texto 'vanilla'.

Exibe uma mensagem de parabéns na tela que inclui o nome do cliente vencedor. Por exemplo, se Kim vencer, o código exibe "Congratulations Kim you have won an ice cream sundae!"

Define o nome ou variável chamada prompt como o texto "Would you like a cherry on top?"

Pede ao usuário para digitar alguma coisa e atribui-la para wants_cherry. Note que quando o usuário é solicitado pela input, o prompt é exibido primeiro (como visto na saída de Python).

Define order como o texto 'vanilla' seguido por 'sundae'.

Se o usuário respondeu sim para 'Would you like a cherry on top?', então adiciona o texto "with a cherry on top" ao pedido.

Exibe que o pedido do vencedor já está a caminho.

P.S. Se você não conseguir se segurar e precisar digitar esse código, adicione import random no topo de cada arquivo antes de executá-lo. Veremos o que isso faz mais tarde, mas note que rodar o código a esta altura não é necessário e nem tão útil. Dito isso, sabemos que alguém vai querer tentar. Sim, estamos falando de você!

Como escrever e rodar código com Python

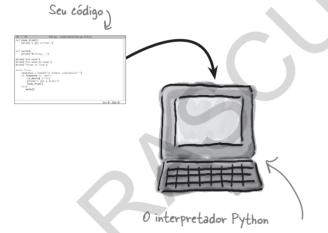
Agora que falamos sobre código e até o vimos, é hora de começar a pensar sobre como você na prática escreveria e executaria o código de verdade. Como mencionado, dependendo da linguagem e do ambiente, há muitos modelos diferentes de como fazer isso. Vamos ter uma noção de como você escreverá e executará o código Python:

Escrevendo seu código

Primeiro, digite seu código em um editor e salve. Você pode usar qualquer editor de texto, como o Bloco de Notas do Windows ou o TextEdit do Mac, para escrever o código Python. Dito isso, a maioria dos desenvolvedores usa editores especializados chamados IDEs (ou Ambientes de Desenvolvimento Integrado) para escrever o código. Por quê? Os IDEs são como processadores de texto — fornecem vários recursos legais como autocompletar de palavras-chaves comuns de Python, destaques de sintaxe da linguagem (ou erros), bem como, funcionalidades integradas de teste. O Python também tem oportunamente uma IDE chamada IDLE, que veremos em breve.



TEste é o editor IDLE de Python.



Como o código é interpretado

Descrevemos Python como a linguagem que você e o computador entendem. E, como aprendemos, um interpretador faz o trabalho de ler o código e executá-lo. Para isso, ele realmente traduz seu código nos bastidores em um código de máquina de nível mais baixo que pode ser executado diretamente pelo hardware do seu computador. Você não precisa se preocupar com como ele faz isso; só saiba que o interpretador fará o trabalho de executar cada declaração do seu código Python.

Rodando o código

Rodar o código é tão fácil quanto passá-lo pelo interpretador Python, um programa que se encarrega de tudo que é necessário para executar o código escrito. Falaremos sobre os detalhes daqui a pouco, mas você pode acessar o interpretador por meio do IDLE ou diretamente via prompt de comando do seu computador.

