Python para Informáticos

Explorando a Informação

Version 2.7.2

Autor: Charles Severance
Tradução: @victorjabur

Histórico de Publicação:

Maio 2015: Checagem editorial obrigado a Sue Blumenberg.

Outubro 2013: Revisão principal dos Capítulos 13 e 14 para mudar para JSON e usar OAuth. Novo capítulo adicionado na Visualização.

Setembro 2013: Livro publicado na Amazon CreateSpace

Janeiro 2010: Livro publicado usando a máquina da Universidade de Michigan Espresso Book.

Dezembro 2009: Revisão principal dos capítulos 2-10 de *Think Python: How to Think Like a Computer Scientist* e escrita dos capítulos 1 e 11-15 para produzir *Python for Informatics: Exploring Information*

Junho 2008: Revisão prncipal, título alterado para *Think Python: How to Think Like a Computer Scientist*.

Agosto 2007: Revisão principal, título alterado para *How to Think Like a (Python) Programmer*.

Abril 2002: Primeira edição de *How to Think Like a Computer Scientist*.

O código fonte LaTeX para a *Think Python: How to Think Like a Computer Scientist* versão deste livro está disponível em `http://www.thinkpython.com`.

Prefácio

Python para Informáticos: Adaptação de um livro aberto

É muito comum que acadêmicos, em sua profissão, necessitem publicar continuamente materiais ou artigos quando querem criar algo do zero. Este livro é um experimento em não partir da estaca zero, mas sim "remixar" o livro entitulado *Think Python: How to Think Like a Computer Scientist* escrito por Allen B. Downey, Jeff Elkner, e outros.

Em dezembro de 2009, quando estava me preparando para ministrar a disciplina **SI502 - Programação para Redes** na Universidade de Michigan para o quinto semestre e decidi que era hora de escrever um livro de Python focado em explorar dados ao invés de entender algoritmos e abstrações. Minha meta em SI502 é ensinar pessoas a terem habilidades na manipulação de dados para a vida usando Python. Alguns dos meus estudantes estavam planejando tornarem-se profissionais em programação de computadores. Ao invés disso, eles escolheram ser bibliotecários, gerentes, advogados, biólogos, economistas, etc., e preferiram utilizar habilmente a tecnologia nas áreas de suas escolhas.

Eu nunca consegui encontrar o livro perfeito sobre Python que fosse orientado a dados para utilizar no meu curso, então eu comecei a escrever o meu próprio. Com muita sorte, em uma reunião eventual três semanas antes de eu começar a escrever o meu novo livro do zero, em um descanso no feriado, Dr. Atul Prakash me mostrou o *Think Python* livro que ele tinha usado para ministrar seu Curso de Python naquele semestre. Era um texto muito bem escrito sobre Ciência da Computação com foco em explicações diretas e simples de se aprender.

Toda a estrutura do livro foi alterada, visando a resolução de problemas de análise de dados de um modo tão simples e rápido quanto possível, acrescido de uma série de exemplos executáveis e exercícios sobre análise de dados desde o início.

Os capítulos 2–10 são similares ao livro *Think Python* mas precisaram de muitas alterações. Exemplos com numeração e exercícios foram substituídos por exercícios orientados a dados. Tópicos foram apresentados na ordem necessária para construir soluções sofisticadas em análise de dados. Alguns tópicos tais como `try` e `except` foram movidos mais para o final e apresentados como parte do capítulo de condicionais. Funções foram necessárias para simplificar a complexidade na manipulação dos programas introduzidos anteriormente nas primeiras

lições em abstração. Quase todas as funções definidas pelo usuário foram removidas dos exemplos do código e exercícios, com exceção do Capítulo 4. A palavra "recursão"[1] não aparece no livro inteiro.

Nos capítulos 1 e 11–16, todo o material é novo, focado em exemplos reais de uso e exemplos simples de Python para análise de dados incluindo expressões regulares para busca e transformação, automação de tarefas no seu computador, recuperação de dados na internet, extração de dados de páginas web, utilização de web services, transformação de dados em XML para JSON, e a criação e utilização de bancos de dados utilizando SQL (Linguagem estruturada de consulta em bancos de dados).

O último objetivo de todas estas alterações é a mudança de foco, de Ciência da Computação para uma Informática que inclui somente tópicos que podem ser utilizados em uma turma de primeira viagem (iniciantes) que podem ser úteis mesmo se a escolha deles não for seguir uma carreira profissional em programação de computadores.

Estudantes que acharem este livro interessante e quiserem se aprofundar devem olhar o livro de Allen B. Downey's *Think Python*. Porque há muita sinergia entre os dois livros, estudantes irão rapidamente desenvolver habilidades na área com a técnica de programação e o pensamento em algoritmos, que são cobertos em *Think Python*. Os dois livros possuem um estilo de escrita similar, é possível mover-se para o livro *Think Python* com o mínimo de esforço.

Com os direitos autorais de *Think Python*, Allen me deu permissão para trocar a licença do livro em relação ao livro no qual este material é baseado de GNU Licença Livre de Documentação para a mais recente Creative Commons Attribution — Licença de compartilhamento sem ciência do autor. Esta baseia-se na documentação aberta de licenças mudando da GFDL para a CC-BY-SA (i.e., Wikipedia). Usando a licença CC-BY-SA, os mantenedores deste livro recomendam fortemente a tradição "copyleft" que incentiva os novos autores a reutilizarem este material da forma como considerarem adequada.

Eu sinto que este livro serve de exemplo sobre como materiais abertos (gratuitos) são importantes para o futuro da educação, e quero agradecer ao Allen B. Downey e à editora da Universidade de Cambridge por sua decisão de tornar este livro disponível sob uma licença aberta de direitos autorais. Eu espero que eles fiquem satisfeitos com os resultados dos meus esforços e eu desejo que você leitor esteja satisfeito com *nosso* esforço coletivo.

Eu quero fazer um agradecimento ao Allen B. Downey e Lauren Cowles por sua ajuda, paciência, e instrução em lidar com este trabalho e resolver os problemas de direitos autorais que cercam este livro.

Charles Severance
www.dr-chuck.com

[1] Com exceção, naturalmente, desta linha.

Ann Arbor, MI, USA
9 de Setembro de 2013

Charles Severance é um Professor Associado à Escola de Informação da Universidade de Michigan.

Tradução:
@victorjabur

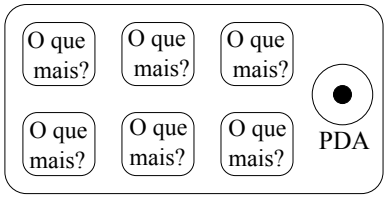
Sumário

Capítulo 1

Por que você deve aprender a escrever programas ?

Escrever programas (ou programação) é uma atividade muito criativa e recompensadora. Você pode escrever programas por muitas razões, que vão desde resolver um difícil problema de análise de dados a se divertir ajudando alguém a resolver um problema. Este livro assume que *qualquer pessoa* precisa saber como programar, e uma vez que você sabe como programar, você irá imaginar o que você quer fazer com suas novas habilidades.

Nós estamos cercados no nosso dia a dia por computadores, desde notebooks até celulares. Nós podemos achar que estes computadores são nossos "assistentes pessoais" que podem cuidar de muitas coisas a nosso favor. O hardware desses computadores no nosso dia a dia é essencialmente construído para nos responder a uma pergunta, "O que você quer que eu faça agora ?"



Programadores adicionam um sistema operacional e um conjunto de aplicações ao hardware e nós terminamos com um Assistente Pessoal Digital que é muito útil e capaz de nos ajudar a fazer diversas coisas.

Nossos computadores são rápidos, tem vasta quantidade de memória e pode ser muito útil para nós, somente se conhecermos a linguagem falada para explicar para um computador o que nós gostaríamos de fazer "em seguida". Se nós conhecemos esta linguagem, nós podemos pedir ao computador para fazer tarefas repetitivas a nosso favor. Curiosamente, as coisas que os computadores podem fazer melhor são frequentemente aquelas coisas que humanos acham chatas e entediantes.

Por exemplo, olhe para os três primeiros parágrafos deste capítulo e me diga qual é a palavra mais usada e quantas vezes. Contá-las é muito doloroso porque não é o tipo de problema que mentes humanas foram feitas para resolver. Para um computador o oposto é verdade, ler e entender o texto de um pedaço de papel é difícil, mas contar palavras dizendo a você quantas vezes ela aparece é muito fácil:

```
python palavras.py
Digite o nome do arquivo: palavras.txt
para 16
```

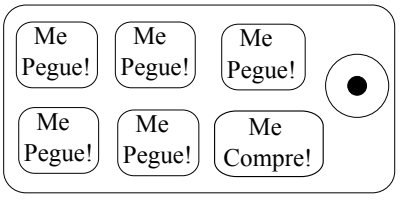
Nosso "assistente de análise pessoal de informações" rapidamente conta para nós que a palavra "para" foi utilizada dezesseis vezes nos primeiros três parágrafos deste capítulo.

Este fato de que os computadores são bons em coisas que humanos não são é a razão pela qual você precisa tornar-se qualificado em falar a "linguagem do computador". Uma vez que você aprende esta nova linguagem, pode delegar tarefas mundanas para o seu parceiro (o computador), ganhando mais tempo para fazer coisas que você foi especialmente adaptado para fazer. Você agrega criatividade, intuição e originalidade para o seu parceiro.

1.1 Criatividade e motivação

Embora este livro não se destine a programadores profissionais, programação profissional pode ser um trabalho muito gratificante, tanto financeiramente quanto pessoalmente. Construir programas úteis, elegantes, inteligentes para que outros utilizem é uma atividade criativa. Seu computador ou assistente pessoal digital (PDA) geralmente contém muitos programas diferentes feitos por diversos grupos de programadores, todos competindo por sua atenção e seu interesse. Eles tentam dar o seu melhor para atender suas necessidades e dar a você uma boa experiência de usabilidade no processo. Em algumas situações, quando você executa um trecho de software, os programadores são diretamente recompensados por sua escolha.

Se nós pensarmos em programas como resultado criativo de grupos de programadores, então talvez a figura a seguir seja uma versão mais sensata de nosso PDA:

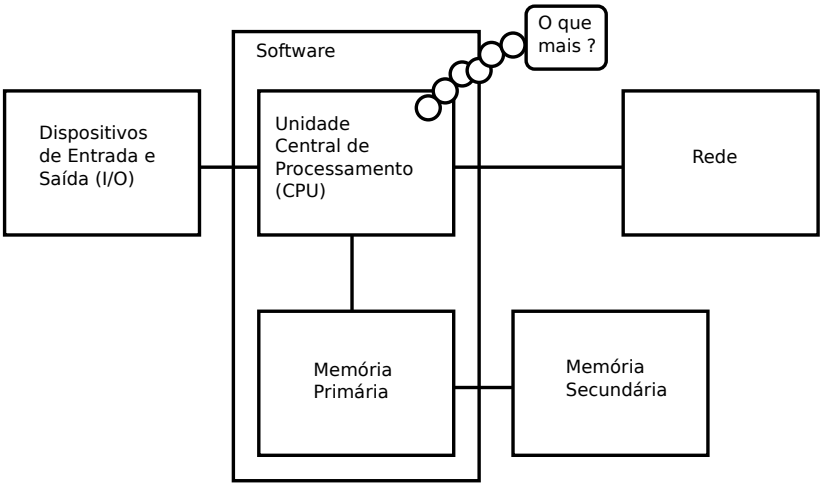


Por enquanto, nossa motivação primária não é ganhar dinheiro ou agradar usuários finais, para nós sermos mais produtivos na manipulação de dados e informações que nós encontraremos em nossas vidas. Quando você começar, você será tanto

o programador quanto o usuário final de seus programas. Conforme você ganhar habilidades como programador e melhorar a criatividade em seus próprios programas, mais você pode pensar em programar para os outros.

1.2 Arquitetura física do Computador - Hardware

Antes de começar a estudar a linguagem, nós falamos em dar instruções aos computadores para desenvolver software, nós precisamos aprender um pouco mais sobre como os computadores são construídos. Se você desmontar seu computador ou celular e olhar por dentro, você encontrará as seguintes partes:



As definições resumidas destas partes são:

- A **Unidade Central de Processamento** (ou CPU) é a parte do computador que é feita para sempre te perguntar: "O que mais ?" Se seu computador possui uma frequência de 3.0 Gigahertz, significa que a CPU irá te perguntar "O que mais ?" três bilhões de vezes por segundo. Você irá aprender como conversar tão rápido com a CPU.

- A **Memória Principal** é utilizada para armazenar informação que a CPU precisa com muita pressa. A memória principal é aproximadamente tão rápida quanto a CPU. Mas a informação armazenada na memória principal se perde quando o computador é desligado (volátil).

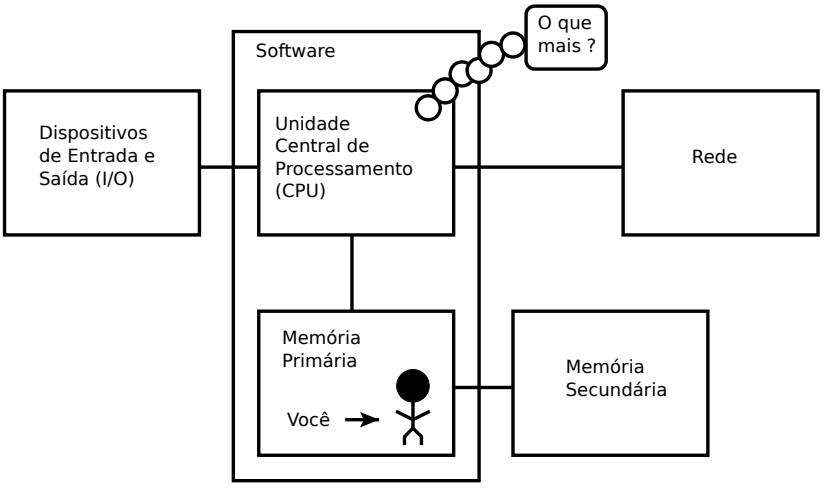
- A **Memória Secundária** é também utilizada para armazenar informação, mas ela é muito mais lenta que a memória principal. A vantegem da memória secundária é que ela pode armazenar informação que não se perde quando o computador é desligado. Exemplos de memória secundária são discos rígidos (HD), pen drives, cartões de memória (sd card) (tipicamente) encontradas no formato de USB e portáteis.

- Os **Dispositivos de Entrada e Saídas** são simplesmente nosso monitor (tela), teclado, mouse, microfone, caixa de som, touchpad, etc. Eles são todas as formas com as quais interagimos com o computador.

- Atualmente, a maioria dos computadores tem uma **Conexão de Rede** para buscar informação em uma rede. Nós podemos pensar a rede como um lugar muito lento para armazenar e buscar dados que podem não estar "disponíveis". Em essência, a rede é mais lenta e às vezes parece uma forma não confiável de **Memória Secundária**.

É melhor deixar a maior parte dos detalhes de como estes componentes funcionam para os construtores dos computadores, isso nos ajuda a ter alguma terminologia que podemos utilizar para conversar sobre as diferentes partes nos programas que vamos escrever.

Como um programador, seu trabalho é usar e orquestrar cada um destes recursos para resolver um problema que você precisa resolver e analisar os dados que você obtém da solução. Como um programador você irá "conversar" com a CPU e contar a ela o que fazer em um próximo passo. Algumas vezes você irá dizer à CPU para usar a memória principal, a memória secundária, a rede ou os dispositivos de entrada e saída.



Índice Remissivo