PYTHON PARA AS HUMANIDADES DIGITAIS

Witton Becerra Mayorga
Adriana López
Alex L. Rojas

Becerra Mayorga, Witton, autor
Python para as humanidades digitais / Witton Becerra Mayorga, Adriana López, Alex L. Rojas. -- Primeira edição. -- Bogotá: Ecoe Ediciones, 2024.
250 páginas. -- (Computadores e tecnologia da informação. Programação e desenvolvimento de software)

Inclui curricula vitae dos autores -- Inclui bloco de códigos e índices alfabéticos -- Inclui referências bibliográficas.

ISBN 978-958-503-891-2 (digital)

1. Python (Linguagem de Programação de Computadores) 2. Humanidades - Pesquisa - Processamento de Dados 3. Programação (Computadores Eletrônicos) 4. Desenvolvimento de software I. López, Adriana, autora II. Rojas, Alex L., autor

CDD: 005.133 ed.23 CO-BoBN- a1131254

Área: *Computação e tecnologia da informação*
Subárea: *desenvolvimento de software*

ECOE
EDICIONES

info@ecoeediciones.com
www.ecoeediciones.com
Carrera 19 # 63 C 32
Teléfono: (+57) 321 226 46 09
Bogotá, Colombia

Primera edición: Bogotá, 2024

e-ISBN: 978-958-503-891-2

Diretora Editorial: Ana María Rueda G.
Coordenadora Editorial: Alejandra Rondón Forero
Editora de aquisições: Alejandra Cely R.
Diagramação: Denise Rodríguez Ríos
Capa: Sindy Nicol Pulido C.
Tradução: Deepl SE, Valeria Rondón Rincón

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DAS TABELAS

Sistema de Información en Línea

No final do livro, encontrará o código para acessar o **Sistema de Información en Línea** - SIL - onde poderá encontrar todos os blocos de código incluídos no livro.

ecoeediciones.com

PREFÁCIO

As Humanidades Digitais são um campo interdisciplinar que combina o estudo tradicional das Ciências Humanas com a programação, a fim de abordar os fenômenos e objetos usados nessas disciplinas de uma maneira diferente, graças à programação, que permite que a pesquisa seja abordada de uma perspectiva diferente.

Embora existam muitas possibilidades subjacentes ao espectro desse conceito, as que queremos expressar aqui consistem basicamente no acúmulo, na manipulação, na análise e no estudo de dados de literatura, linguística, história, direito, jornalismo etc. A base principal dessas disciplinas são, por exemplo, objetos físicos, como livros, que, quando digitalizados, permitem que sejam abordados de uma maneira diferente, graças à coleta de dados por meio de programação que permite a criação de novos protocolos de pesquisa para estudar as humanidades de uma maneira mais ampla, com resultados mais completos e que geram novidades em relação aos objetos de estudo tradicionais.

Como pano de fundo, há vários projetos que geraram essa nova maneira de ver o estudo das ciências humanas, por exemplo: "Michael Hart criou o Projeto Gutenberg em 1971, que se tornou um dos primeiros projetos de digitalização de livros. O objetivo do projeto era tornar os livros de domínio público disponíveis gratuitamente on-line. Até 2019, o Projeto Gutenberg havia digitalizado mais de 60.000 livros." (Warwick, 2012, p. 13). No mesmo sentido, na década seguinte, foi criada a Text Encoding Initiative (TEI), que: "é um consórcio internacional que desenvolve e mantém um padrão para a codificação de textos eletrônicos. O TEI foi criado em 1987 para fornecer uma metodologia para a criação de textos eletrônicos e tem sido fundamental para a

criação de projetos de digitalização de textos. O TEI é amplamente usado em projetos de humanidades digitais que envolvem digitalização e análise de textos." (Warwick, 2012, p. 20). Até o momento, o que está em questão é a digitalização de textos.

Como se pode observar, a iniciativa desses projetos baseia-se na digitalização de conteúdo físico, especialmente livros. Nesse sentido, em princípio, foi proposta a digitalização de vários textos do patrimônio mundial ou, como uma abordagem móvel, foi considerada a acumulação e o resgate desse material por meio de sua digitalização. Assim, o Projeto Perseus: "é uma iniciativa que começou em 1987 com o objetivo de criar uma biblioteca digital de textos e recursos em áreas como literatura, história e filosofia. O projeto também inclui ferramentas de análise e visualização de dados. A biblioteca digital do Projeto Perseus se tornou uma das mais usadas pelos pesquisadores de humanidades digitais." (Schreibman *et al*., 2008, p. 31). Da mesma forma, os autores apontam os estudos da Biblioteca de Alexandria como um pilar para esse olhar sobre essa nova forma de acumular textos: "Em 1991, a UNESCO lançou uma iniciativa para recriar a Biblioteca de Alexandria em formato digital. O projeto envolve a digitalização e a análise de textos antigos e a criação de uma biblioteca digital que pode ser acessada de qualquer lugar do mundo. Os estudos da Biblioteca de Alexandria foram fundamentais para a criação de uma rede global de bibliotecas digitais." (Schreibman *et al*., 2008, p. 33).

Como se pode ver, essas referências importantes propõem a criação de coleções bibliográficas digitais. A questão é, então, o que aconteceu após a confirmação de que era possível criar grandes quantidades de textos digitais. Essa é a origem da perspectiva atual das Humanidades Digitais, que consiste em estudar esses textos por meio de programas de computador.

Com a abundância de dados digitalizados, surgiram posteriormente metodologias para estudar os textos. Portanto, hoje, a perspectiva das Humanidades Digitais é estudar essas grandes quantidades de texto por meio de programas de computador com base em modelos de estudo mais amplos e abrangentes, nas palavras de Jockers:

> As Humanidades Digitais não se referem apenas à aplicação de tecnologias digitais, mas também ao desenvolvimento de novas teorias e métodos para a pesquisa em humanidades, como análise de redes sociais, mineração de dados, visualização e aprendizado de máquina. Essas abordagens permitem que os pesquisadores analisem conjuntos de dados grandes e complexos, descubram padrões e relações que não são imediatamente aplicáveis e gerem novas percepções sobre a cultura e a sociedade humanas" (2013, p. 34).

Em conclusão, o que está sendo explorado em profundidade nas Humanidades Digitais atualmente é metodologias como análise de redes sociais, mineração de dados, visualização de dados usando gráficos que mostram textos e seus relacionamentos e o agora famoso *machine learning* que produziu uma revolução como o ChatGPT.

Este livro foi escrito para qualquer humanista que esteja interessado em aprender a programar em Python. Em nossa experiência, esse interesse geralmente vem com um problema real que o indivíduo deseja resolver e ouviu dizer que pode ser resolvido com Python. Por esse motivo, recomendamos que, ao longo do material, o senhor pratique não apenas com os exemplos do livro, mas com material relacionado ao problema do humanista.

Na primeira parte do livro, concentramo-nos em aprender as estruturas básicas de armazenamento de dados em Python e sua manipulação básica. Também extraímos dados de arquivos de texto, arquivos PDF e páginas da Internet com a ajuda de bibliotecas como BeatifulSoup e pypdf. Os arquivos de texto e pdf podem estar disponíveis em nosso disco rígido ou na Internet, portanto, é necessário ter uma conexão com a Internet para trabalhar com o material proposto. Para extrair e manipular as informações desses arquivos, é essencial conhecer e manipular as estruturas básicas do Python, bem como outras estruturas disponíveis em bibliotecas criadas especificamente para manipular determinados tipos de informações. Por esse motivo, é importante ter acesso a uma distribuição Python que tenha essas bibliotecas instaladas por padrão. Conforme explicado no primeiro capítulo, usaremos a distribuição Anaconda, de modo que não precisaremos nos preocupar com a instalação de bibliotecas extras ou incompatibilidades.

Diferentemente de outros livros introdutórios de Python, em que o uso de conceitos matemáticos é o foco central para a apresentação de todo o material, neste livro o foco é trabalhar com texto. Há vários formatos para armazenar texto, mas não consideramos todos eles neste livro. Nosso foco é o texto simples e os formatos csv, HTML e JSON. Entretanto, se for necessário interagir com qualquer outro formato, acreditamos que, com o material que aprendemos neste livro, temos as ferramentas para lê-lo e manipulá-lo.

Como humanistas digitais, sabemos que o texto é uma fonte extremamente rica de informações. Por exemplo, desde que começamos a ler este livro, centenas de mensagens de texto, e-mails, comentários em diferentes plataformas etc. foram criados. Além disso, de todo o texto que está disponível na Internet e em bancos de dados. Quando tivermos acesso ao texto, devemos organizá-lo de forma estruturada e garantir que ele esteja o mais livre de erros possível.

Na segunda parte do livro, consideramos ferramentas mais especializadas para lidar com textos e conjuntos de dados. No Capítulo 5, começamos a nos aprofundar no uso de expressões regulares para generalizar padrões de caracteres. Em seguida, voltamos ao estudo de conjuntos de dados e aprendemos sobre outras ferramentas para manipulá-los.

Nos dois últimos capítulos do livro, consideramos duas ferramentas comumente usadas por humanistas digitais: gráficos e processamento de linguagem natural. No

Capítulo 6, somos apresentados ao processamento de linguagem natural, usando a biblioteca spaCy. Já no Capítulo 7, aprendemos a representar graficamente as redes e a obter estatísticas básicas a partir delas. Nosso tratamento desses dois tópicos é de natureza introdutória, pois um tratamento aprofundado requer conhecimento de material mais avançado.

Todos os exemplos deste livro podem ser reproduzidos, pois todas as fontes de informação estão disponíveis gratuitamente. Portanto, convidamos nossos leitores a repetir e modificar nossos exemplos. Por fim, este livro foi digitado usando Quarto[I], e todas as figuras foram criadas pelos autores usando LaTeX, Tik z, mermaid e Python.

I. https://quarto.org/docs/books

1

INTRODUÇÃO

Neste capítulo, apresentamos o material preliminar para aprender a programar em Python. Especificamente, propomos algumas das opções disponíveis para executar programas escritos em Python e uma breve descrição dos termos comumente usados na programação nessa linguagem. Se já tiver o Python instalado no computador ou acesso a uma ferramenta na Internet, poderá pular a primeira seção.

Quando temos um problema que podemos resolver de forma programática, é essencial poder executar o código que escrevemos. Por esse motivo, é importante que tenhamos acesso ao Python ou a uma interface que nos permita escrever, testar e executar nosso código. Em nossa opinião, ao aprender a programar, é importante receber feedback sobre o código que estamos escrevendo para nos motivar a continuar. Além disso, detectar um erro de digitação ou de sintaxe em uma linha de código é muito mais fácil do que quando se tem várias linhas de código. Por esse motivo, em vez de usar Python diretamente, aconselhamos o interessado neste texto a usar um ambiente de desenvolvimento integrado, conhecido como IDE (*Integrated Development Environment*), em vez da instalação básica da linguagem. Como existem várias dessas ferramentas que servem como ambiente de trabalho, na seção a seguir, apresentamos algumas das opções disponíveis para que possa fazer uma escolha.

1.1. Instalação

A instalação do Python depende do sistema operacional e da versão desse sistema. Além disso, a instalação básica não contém todas as ferramentas de que podemos precisar em nossos projetos. Portanto, é necessário instalar códigos extras escritos por outros programadores, como veremos no Capítulo 3. Por conseguinte, instalar o Python com todas as ferramentas de que precisamos em nossos projetos geralmente é uma tarefa desafiadora. Há pelo menos duas soluções para esse problema, que apresentamos a seguir para trabalhar com o material deste texto.

Por um lado, podemos evitar instalar o Python em nosso computador e trabalhar *on-line*. Há várias páginas na Internet que nos permitem escrever e executar o código que escrevemos sem a necessidade de instalar nada em nosso computador. Os aplicativos que recomendamos têm uma interface semelhante entre si e a ideia original vem do projeto Ipython (Perez & Granger, 2007), que se concentra no Python. Mas ela foi generalizada para várias linguagens de programação com o projeto Jupyter[1]. *Os notebooks* Jupyter são uma das formas mais populares de trabalhar em Python e compartilhar código com outras pessoas. No momento em que este artigo foi escrito, o IDE com maior flexibilidade no projeto Jupyter era o JupyterLab[2]. Uma captura de tela dessa interface é mostrada na Figura 1.1. Como pode ser visto nessa figura, o *notebook* contém células nas quais o código pode ser digitado diretamente e, se não houver erros, executado.

Com o JupyterLab, já podemos escrever nossa primeira linha de código; no entanto, esse aplicativo destina-se apenas a ser testado e, em seguida, baixado e usado localmente em nosso computador.

Outra opção disponível na Internet é o Google Colaboratory. Como o próprio nome sugere, é um aplicativo desenvolvido pelo Google com um ambiente *semelhante ao notebook* Jupyter que é executado na nuvem; portanto, não é necessário instalar nada em nosso computador de trabalho, mas é preciso ter uma conta do Google.

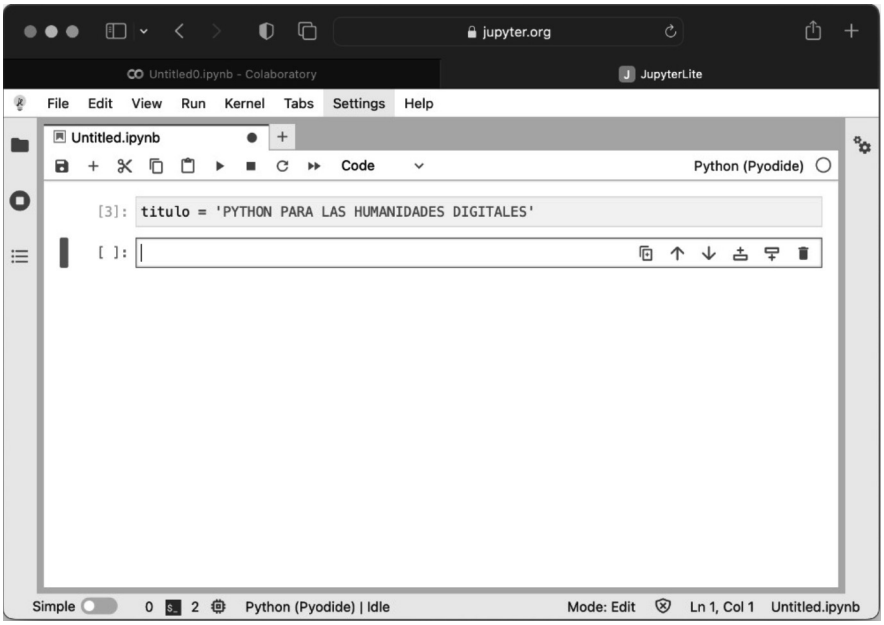
Na Figura 1.2, podemos ver que a interface do Google Colaboratory é semelhante à interface do JupyterLab. Para acessar a interface, precisamos inserir as informações da nossa conta do Google, e nossos programas serão armazenados no Google Drive em uma pasta chamada Colab *notebooks*. Um problema que pode surgir ao usar o Colab em conjunto com o material deste livro é a disponibilidade dos dados

1 https://jupyter.org/

2 https://jupyter.org/try-jupyter/lab/

que usamos no Colab. Portanto, em alguns casos, será necessário fazer o upload dos dados manualmente para ter acesso a eles.

Figura 1.1. Interface do Jupyter Lab



A última opção que consideraremos é instalar o Python localmente em nossa estação de trabalho. Isso pode ser uma tarefa complicada, portanto, nossa recomendação é usar um ambiente que instale e assuma o controle dessa instalação. O ambiente mais popular, e nossa recomendação, é o Anaconda. Esse ambiente pode ser baixado em https://www.anaconda.com/download. Uma vez na página, ele detectará o sistema operacional em que se está trabalhando e, ao digitar Download, iniciará o download. Para instalar o Anaconda, siga as instruções do instalador.

Em resumo, o Anaconda elimina a necessidade de aprender a instalar várias das ferramentas de programação mais populares. Em particular, a Tabela 1.1 lista as bibliotecas que usaremos neste livro e que são encontradas nessa distribuição. Uma biblioteca é uma coleção de código reutilizável que nos permite executar tarefas específicas.

Figura 1.2. A interface do Google Colab

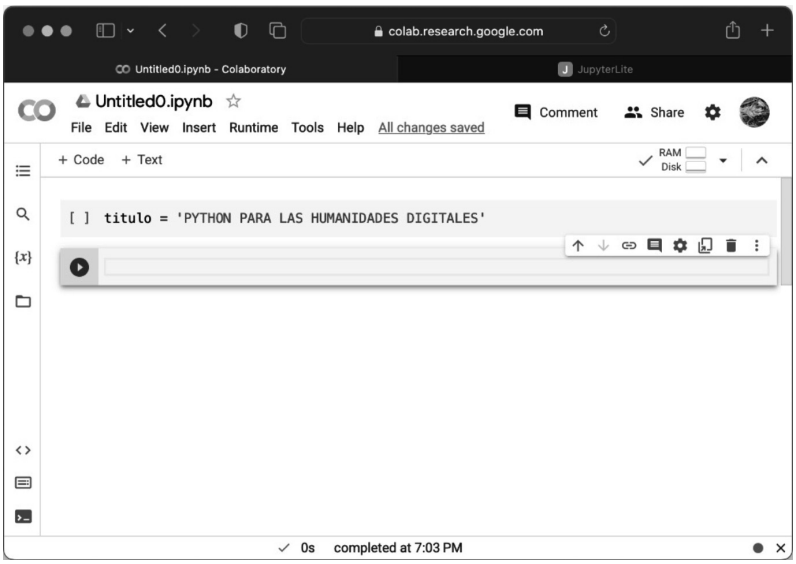


Tabela 1.1. Bibliotecas usadas neste livro e disponíveis na distribuição do Anaconda

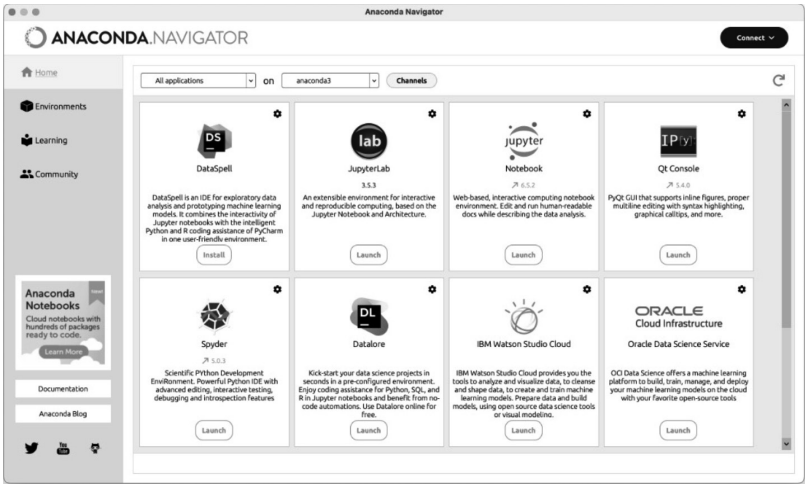
Livraria	Breve descrição
BeatifulSoup	Cria uma estrutura que permite o acesso ao texto e outras informações disponíveis em arquivos formatados em html
matplotlib	É a biblioteca mais popular para criar e manipular gráficos
pypdf	Leitura e manipulação de arquivos PDF
pandas	Leitura, manipulação e gravação de conjuntos de dados

O Python divide seu código em livrarias, pacotes e módulos. As bibliotecas são uma coleção de pacotes, os pacotes são uma coleção de módulos e os módulos são uma coleção de código que define variáveis, funções e classes. A ideia por trás dessa organização é dividir as tarefas em subtarefas, de modo que o código possa ser revisado e modificado com mais facilidade. Na prática, porém, há alguns pacotes que são tão complexos e têm vários aplicativos que são chamados de livrarias. Por exemplo, o pacote pandas, que chamamos de livrarias na Tabela 1.1. Por esse motivo, é comum na prática trocar esses dois termos.

Observe que, embora esse ambiente já esteja instalado, ainda precisamos selecionar como interagir com o Python. Quando o navegador Anaconda abrir, o que pode levar alguns segundos, você verá uma tela como a da Figura 1.3. Nesse navegador, podemos selecionar o IDE que queremos usar para escrever nosso código. As três opções mais populares são JupyterLab, Spyder e PyCharm. Ao executar o JupyterLab, uma janela é aberta em um navegador da Web e, como mencionado acima, podemos executar o código linha por linha, ou melhor, célula por célula.

Por outro lado, o Spyder e o PyCharm são normalmente usados para executar um programa inteiro, e não linha por linha de código. Nossa recomendação é que começar com o JupyterLab e, quando tiver adquirido experiência, passe a usar o Spyder ou o PyCharm, se desejar.

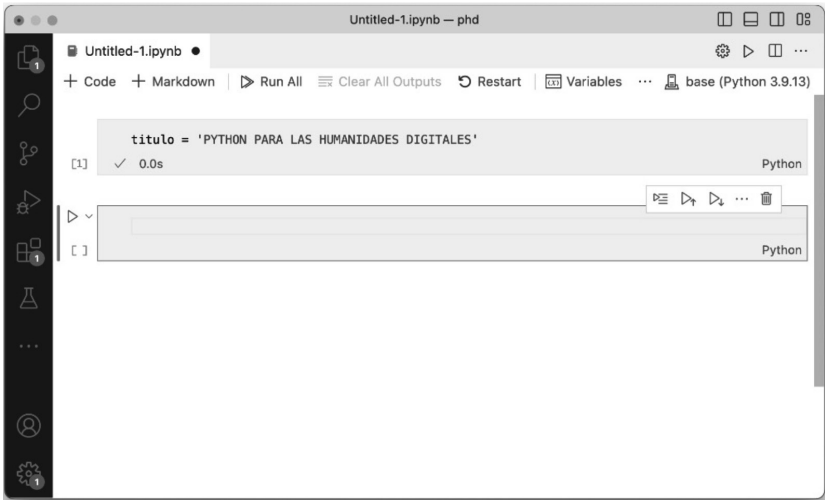
Figura 1.3. Navegador Anaconda



Para concluir esta seção, há uma outra IDE que não está disponível com Anaconda, mas que também é popular entre os programadores: VS Code 2.

Como podemos ver na Figura 1.4, o VS Code 2 tem uma interface semelhante à dos *notebooks* Jupyter; mas, ao contrário do JupyterLab, o VS Code 2 não é executado em um navegador, mas é um aplicativo autônomo que oferece outras ferramentas.

Figura 1.4. Interface do VS Code 2

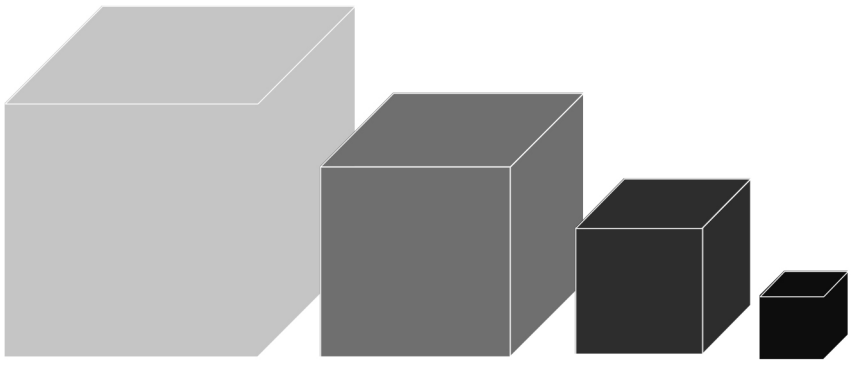


1.2. Conceitos básicos

Python é uma linguagem de programação multiparadigma, que, em particular, oferece suporte à programação orientada a objetos. Para isso, o Python permite definir *classes* de objetos. Com cada classe, um tipo específico de objeto é modelado, especificando os dados que cada objeto contém e o comportamento desses dados. Em outras palavras, uma classe funciona como um modelo ao qual podemos atribuir valores específicos, gerando assim uma instância da classe que chamamos de objeto. Um exemplo clássico desses conceitos que nos permite entendê-los melhor é um complexo residencial no qual a planta das casas corresponde à classe e as casas construídas correspondem aos objetos. A seguir, apresentamos outro exemplo.

Exemplo 1.1 (Cubos). Suponhamos que tenhamos uma *classe* chamada cubo. Essa classe cria uma imagem em nosso cérebro de um cubo e é possível decidir, por exemplo, a cor e o tamanho desse cubo. Portanto, esse conceito de classe é abstrato. Entretanto, ao representar graficamente quatro objetos semelhantes a cubos aos quais foram atribuídos cores e tamanhos, como na Figura 1.5, esses *objetos* deixam de ser abstratos porque podemos visualizá-los de acordo com seus atributos, ou seja, as cores e os tamanhos atribuídos.

Figura 1.5. Quatro instâncias da classe cube



Neste livro, não abordaremos a criação de classes nem nos aprofundaremos no conceito de programação orientada a objetos, mas mencionaremos o conceito de *objeto* várias vezes, por isso é importante ter uma ideia básica do que ele significa. Por exemplo, nossa interação com o Python geralmente requer a criação de dados ou a importação de dados disponíveis na Internet ou em qualquer outro lugar a que tenhamos acesso. Esses dados devem ser armazenados em algum lugar da memória do computador e devem ter um nome pelo qual possamos acessá-los diretamente. O nome pelo qual obtemos acesso aos dados é conhecido como *variável*. Por outro lado, os dados que armazenamos têm características diferentes. Por exemplo, podemos armazenar o texto de um livro ou uma imagem, que são objetos diferentes, portanto, o formato de armazenamento também é diferente.

Em outras palavras, as características de cada um dos objetos afetam a forma como os dados são armazenados na memória.

A maior parte do código que apresentamos neste livro está organizada em "Blocos de código" aos quais fazemos referência no texto e, além disso, uma lista de todos eles pode ser encontrada no índice de blocos de código no final do livro. Nesses blocos, geralmente não incluímos comentários devido a restrições de espaço. Por outro lado, todo o código usa uma fonte monoespaçada; dessa forma, o texto do livro é diferenciado do código. Além disso, sempre que falamos sobre caracteres e texto no Python, usamos aspas simples, ' ', com fonte monoespaçada. Por fim, também usamos fonte moo espaçada para os nomes de linguagens de programação, nomes de arquivos, bibliotecas, atributos, *funções e métodos*. No caso de funções e métodos, sempre incluímos parênteses para diferenciá-los.

1.3. Acesso aos dados do site

Neste livro, trabalharemos com dados disponíveis gratuitamente na Internet. Portanto, todos os códigos apresentados e exercícios propostos podem ser feitos sem problemas de acesso. Em geral, esses dados não precisam ser armazenados no disco rígido do seu computador. Entretanto, é muito provável que, em nosso trabalho com nossos próprios dados, tenhamos de armazenar as informações localmente. Nossa recomendação é que cada projeto tenha sua própria pasta e talvez essa pasta deva ter subpastas como Dados, Código e Figures. Seguindo essa proposta, para este livro, vamos supor que a pasta de trabalho seja phd, que todo o nosso código seja armazenado na pasta Código, que os dados, sejam documentos de texto ou imagens, sejam armazenados em Dados e que as figuras que geramos sejam armazenadas em Figuras. Na Figura 1.6, apresentamos um esquema da organização das pastas. Para mostrar a estrutura, incluímos alguns dos arquivos com os quais trabalharemos em capítulos futuros, e não é necessário, no momento, que eles estejam em nosso diretório de trabalho.

1.4. Erros comuns

Um dos recursos do Python é que, quando ocorre um erro de execução no código, uma longa descrição do erro é impressa na tela.