Capítulo 1 Introdução

Este livro lhe dará uma visão geral da gramática da linguagem Python.

Parece que a linha entre a linguagem de programação propriamente dita e a biblioteca padrão está se tornando mais tênue atualmente. Embora não seja nosso objetivo examinar a biblioteca padrão do Python (que está muito além do escopo deste livro), abordaremos alguns conceitos importantes dos módulos da biblioteca padrão que são considerados mais ou menos parte da linguagem.

Essa "referência" começa com a parte mais chata do Python. ÿ Se você planeja ler o livro do começo ao fim, digamos, em vez de usá-lo apenas como uma referência rápida, pode pular os primeiros capítulos em sua primeira leitura e voltar a eles mais tarde, quando necessário.

O termo "programa" significa coisas diferentes em diferentes contextos e em diferentes linguagens de programação. Começamos com explicações um tanto formais sobre o que é um programa Python e como os programas Python são executados, por exemplo, no interpretador Python.

Os programas Python são logicamente organizados em pacotes e módulos, que veremos a seguir. Um *módulo* corresponde a um arquivo em um sistema de arquivos físico, e os módulos, incluindo os módulos de pacote, são as unidades básicas de reutilização e compartilhamento de código em Python.

Em seguida, examinamos brevemente alguns dos elementos lexicais do código-fonte do Python. Existem algumas pequenas variações em diferentes linguagens de programação, mas suas composições lexicais são bastante semelhantes, e o Python não é diferente. Esta parte pode ser pulada em sua "leitura".

De um modo geral, um programa consiste em *código e dados.* Código refere-se a instruções. Dados em Python são representados por *objetos.* O objeto em Python é um componente fundamental. Tudo o que tratamos em

um programa Python são objetos. Começamos a parte principal do livro apresentando vários conceitos importantes relacionados aos objetos.

Embora o Python use um sistema de tipo dinâmico, os tipos ainda desempenham os papéis fundamentais na linguagem de programação Python. Primeiro passamos por alguns dos tipos internos básicos em Python. O Python inclui alguns tipos integrados, e esta referência aborda apenas alguns deles. Conforme indicado, isso geralmente é verdade em todos os tópicos discutidos nesta referência. A completude não é o objetivo desta *mini referência.*

O Python também inclui alguns tipos compostos integrados , como lista e dicionário, que são componentes importantes de qualquer programa não trivial. Examinaremos brevemente esses tipos no próximo capítulo.
Tipos avançados, por exemplo, funções e classes, em particular, são explicados em detalhes posteriormente neste livro.

Como acontece com qualquer linguagem de programação imperativa, o Python possui expressões e declarações. Uma expressão prescreve como calcular um valor usando outras expressões e valores. O Python suporta a maioria dos operadores e expressões comuns encontrados em outras linguagens imperativas. Se você tiver experiência em outra linguagem de programação procedural, poderá folhear a maior parte desta parte.

Uma declaração é uma instrução. As declarações controlam o fluxo de um programa para atingir o objetivo desejado. Em Python, existem dois tipos de declarações, declarações simples e declarações compostas. Instruções compostas ou complexas podem "incluir" outras instruções simples ou compostas.

Instruções simples incluem instrução expression, instrução de atribuição, instrução assert , instrução pass , instrução del , instrução return , instrução raise , instrução break , instrução continue , instrução global e instrução não local . As instruções compostas do Python incluem instrução if , instrução while , instrução for , instrução try , instrução with , instrução match , instrução function def , definição de classe e outras instruções relacionadas à corrotina.

Uma das mudanças mais significativas no Python nos últimos 30 anos foi a adição de correspondência de padrão estrutural à linguagem, a partir do Python 3.10 (2021). A correspondência de padrões foi popularizada pela primeira vez por linguagens de programação funcionais como Haskell, e agora está se tornando cada vez mais amplamente disponível em muitas linguagens de programação diferentes, como C#, ferrugem, swift, scala e (sim) até Java. ÿ

À medida que adotamos mais esse recurso, como comunidade, a correspondência de padrões provavelmente mudará a forma como programamos em Python nos próximos anos. Embora atualmente seja suportado apenas no contexto da declaração de correspondência , é concebível, e de fato esperado, que a correspondência de padrões esteja disponível de forma mais ampla em toda a linguagem em um futuro próximo, considerando sua simplicidade, elegância e poder.

Uma definição de função é uma instrução composta. Dedicamos um capítulo próprio à definição da função. Este capítulo também inclui outros tópicos relacionados a funções, como expressões lambda e decoradores.

Outra instrução composta, a definição de classe, é explicada a seguir.
Outros conceitos relacionados à classe, como enums e decoradores de classe, também são descritos neste capítulo. Também fornecemos uma introdução informal aos fundamentos dos estilos de programação orientada a objetos em Python, incluindo herança múltipla.

Tipos especiais de funções, geradores e co-rotinas, são discutidos separadamente na última parte do livro, no capítulo Cor-rotinas. Tocamos brevemente no estilo de programação assíncrona (de alto nível) em Python usando as novas palavras-chave async e await .

Como afirmado, não cobrimos exaustivamente todos os tópicos em Python nesta "mini referência", incluindo a programação multi-thread e multiprocesso (de baixo nível) , etc. programadores Python de nível avançado devem estar familiarizados.