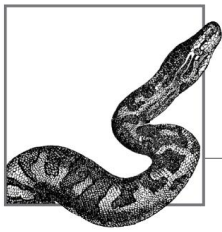
Introdução ao Python



1

Introdução ao Python

Python, uma linguagem de programação de uso geral, já existe há algum tempo: Guido van Rossum, o criador do Python, começou a desenvolvê-lo em 1990. Essa linguagem estável e madura é de alto nível, dinâmica, orientada a objetos e cruzada. plataforma - todas características muito atraentes. O Python é executado em todas as principais plataformas de hardware e sistemas operacionais, portanto, não restringe suas escolhas.

Python oferece alta produtividade para todas as fases do ciclo de vida do software: análise, design, prototipagem, codificação, teste, depuração, ajuste, documentação, implantação e, é claro, manutenção. A popularidade do Python tem tido um crescimento constante e incansável ao longo dos anos. Hoje, a familiaridade com o Python é uma vantagem para todo programador, já que o Python se infiltrou em todos os nichos e tem papéis úteis a desempenhar como parte de qualquer solução de software.

Python fornece uma mistura única de elegância, simplicidade, praticidade e poder absoluto.
Você rapidamente se tornará produtivo com o Python, graças à sua consistência e regularidade, sua rica biblioteca padrão e muitos pacotes e ferramentas de terceiros que estão prontamente disponíveis para ele. Python é fácil de aprender, por isso é bastante adequado se você é novo em programação, mas também é poderoso o suficiente para o especialista mais sofisticado.

A Linguagem Python

A linguagem Python, embora não seja minimalista, é sobressalente, por boas razões pragmáticas.
Uma vez que uma linguagem oferece uma boa maneira de expressar uma ideia de design, adicionar outras maneiras tem, na melhor das hipóteses, benefícios modestos, enquanto o custo, em termos de complexidade da linguagem, cresce mais do que linearmente com o número de recursos. Uma linguagem complicada é mais difícil de aprender e dominar (e de implementar de forma eficiente e sem bugs) do que uma mais simples. Complicações e peculiaridades em uma linguagem prejudicam a produtividade no desenvolvimento de software

desenvolvimento e manutenção, particularmente em grandes projetos, onde muitos desenvolvedores cooperam e muitas vezes mantêm o código originalmente escrito por outros.

Python é simples, mas não simplista. Ele adere à ideia de que, se uma língua se comporta de uma certa maneira em alguns contextos, ela deveria funcionar de maneira ideal em todos os contextos. Python também segue o princípio de que uma linguagem não deve ter atalhos “convenientes”, casos especiais, exceções ad hoc, distinções excessivamente sutis ou otimizações ocultas misteriosas e complicadas. Uma boa linguagem, como qualquer outro artefato bem projetado, deve equilibrar tais princípios gerais com bom gosto, bom senso e um alto grau de praticidade.

Python é uma linguagem de programação de uso geral: as características do Python são úteis em praticamente qualquer área de desenvolvimento de software. Não há área em que o Python não possa fazer parte de uma ótima solução. “Parte” é uma palavra importante aqui; embora muitos desenvolvedores achem que o Python preenche todas as suas necessidades, ele não precisa ficar sozinho: os programas Python podem cooperar com uma variedade de outros componentes de software, tornando-o uma linguagem ideal para unir componentes escritos em outras linguagens. Um dos objetivos do design da linguagem era que ela deveria “funcionar bem com os outros”.

Python é uma linguagem de nível muito alto (VHLL). Isso significa que o Python usa um nível mais alto de abstração, conceitualmente mais distante da máquina subjacente, do que as linguagens compiladas clássicas, como C, C++ e Fortran, que são tradicionalmente chamadas de “linguagens de alto nível”. Python é mais simples, mais rápido de processar (tanto para cérebros humanos quanto para ferramentas programáticas) e mais regular do que linguagens clássicas de alto nível. Isso permite alta produtividade do programador e torna o Python uma ferramenta de desenvolvimento atraente. Bons compiladores para linguagens compiladas clássicas podem gerar código de máquina binário que é executado mais rápido que o código Python. No entanto, na maioria dos casos, o desempenho dos aplicativos codificados em Python é suficiente. Quando não estiver, aplique as técnicas de otimização abordadas em “Otimização” na página 477 para melhorar o desempenho do seu programa enquanto mantém o benefício da alta produtividade.

Linguagens um pouco mais novas, como Java e C#, são de nível ligeiramente mais alto do que as clássicas, como C e Fortran, e compartilham algumas características de linguagens clássicas (como a necessidade de usar declarações), bem como alguns VHLLs, como Python (como o uso de bytecode portátil como destino de compilação em implementações típicas e coleta de lixo para liberar os programadores da necessidade de gerenciar memória).
Se você achar que é mais produtivo com Java ou C# do que com C ou Fortran, experimente Python (possivelmente nas implementações Jython ou IronPython, abordadas em “ Implementações Python” na página 5) para se tornar ainda mais produtivo.

Em termos de nível de linguagem, o Python é comparável a outros VHLLs poderosos, como JavaScript, Ruby e Perl. As vantagens da simplicidade e regularidade, no entanto, permanecem do lado do Python.

Python é uma linguagem de programação orientada a objetos, mas permite que você desenvolva código usando estilos orientados a objetos e procedurais, e também um toque de programação funcional, misturando e combinando como seu aplicativo requer. Os recursos orientados a objetos do Python são conceitualmente semelhantes aos do C++, mas mais simples de usar.

A biblioteca padrão do Python e os módulos de extensão

Há mais na programação Python do que apenas a linguagem Python: a biblioteca padrão e outros módulos de extensão são quase tão importantes para o uso efetivo do Python quanto a própria linguagem. A biblioteca padrão do Python fornece muitos módulos Python bem projetados, sólidos e 100% puros para reutilização conveniente. Ele inclui módulos para tarefas como representação de dados, processamento de texto, interação com o sistema operacional e sistema de arquivos e programação da web. Como esses módulos são escritos em Python, eles funcionam em todas as plataformas suportadas por Python.

Os módulos de extensão, da biblioteca padrão ou de outro lugar, permitem que o código Python acesse a funcionalidade fornecida pelo sistema operacional subjacente ou outros componentes de software, como interfaces gráficas de usuário (GUIs), bancos de dados e redes.
As extensões também oferecem velocidade máxima em tarefas computacionais intensivas, como análise de XML e cálculos de matriz numérica. Os módulos de extensão que não são codificados em Python, no entanto, não necessariamente desfrutam da mesma portabilidade automática entre plataformas que o código Python puro.

Você pode escrever módulos de extensão para propósitos especiais em linguagens de nível inferior para obter desempenho máximo para peças pequenas e com uso intensivo de computação cujo protótipo originalmente foi feito em Python. Você também pode usar ferramentas como Cython e CFFI para agrupar bibliotecas C/C++ existentes em módulos de extensão Python, conforme abordado em “Estendendo o Python sem a API C do Python” na página 677. Por último, mas não menos importante, você pode incorporar o Python em aplicativos codificados em outras linguagens, expondo a funcionalidade existente do aplicativo a scripts Python por meio de módulos de extensão Python específicos do aplicativo.

Este livro documenta muitos módulos, tanto da biblioteca padrão quanto de outras fontes, em áreas como programação de rede do lado do cliente e do servidor, bancos de dados, manipulação de arquivos de texto e binários e interação com o sistema operacional.

Implementações Python

Python tem atualmente quatro implementações de qualidade de produção (CPython, Jython, IronPython, PyPy) e uma nova implementação de alto desempenho em desenvolvimento inicial, Pyston, que não abordaremos mais. Também mencionamos várias outras implementações mais experimentais na seção seguinte.

Este livro aborda principalmente o CPython, a implementação mais amplamente usada, à qual geralmente nos referimos apenas como “Python” para simplificar. No entanto, a distinção entre uma linguagem e suas implementações é importante.

CPython

Classic Python (também conhecido como CPython, muitas vezes chamado apenas de Python) é a implementação de qualidade de produção mais atualizada, sólida e completa do Python. Pode ser considerada a “implementação de referência” da linguagem. CPython é um compilador, interpretador e conjunto de módulos de extensão integrados e opcionais, todos codificados no padrão C.

O CPython pode ser usado em qualquer plataforma em que o compilador C esteja em conformidade com o padrão ISO/IEC 9899:1990 (ou seja, todas as plataformas modernas e populares). Em “Instalação” na página 17, explicamos como baixar e instalar o CPython. Todo este livro, exceto algumas seções explicitamente marcadas de outra forma, se aplica ao CPython. CPython suporta tanto v3 (versão 3.5 ou superior) quanto v2 (versão 2.7).

Jython

Jython é uma implementação Python para qualquer Java Virtual Machine (JVM) compatível com Java 7 ou superior. Essas JVMs estão disponíveis para todas as plataformas populares e modernas. No momento em que este livro foi escrito, o Jython oferece suporte à v2, mas ainda não à v3. Com Jython, você pode usar todas as bibliotecas e estruturas Java. Para uso ideal do Jython, você precisa de alguma familiaridade com as classes Java fundamentais. Você não precisa codificar em Java, mas a documentação e os exemplos de bibliotecas Java são formulados em termos Java, portanto, você precisa ter um conhecimento prévio de Java para lê-los e entendê-los. Você também precisa usar ferramentas de suporte Java para tarefas como manipulação de arquivos .jar e assinatura de applets. Este livro trata de Python, não de Java. Para usar Jython, você deve complementar este livro com Jython Essentials, de Noel Rappin e Samuele Pedroni (O'Reilly), Java in a Nutshell, de David Flanagan (O'Reilly) e/ou e Denitive Guide to Jython (Open Source Versão) por Josh Juneau, Jim Baker, et al. (Apress, disponível no O'Reilly's Safari e no Jython.org), bem como alguns dos muitos outros recursos Java disponíveis.

IronPython

IronPython é uma implementação Python (até o momento, apenas v2, ainda não v3) para o Common Language Runtime (CLR) projetado pela Microsoft, mais comumente conhecido como .NET, que agora é de código aberto e portado para Linux e macOS. Com o IronPython, você pode usar todas as bibliotecas e estruturas CLR. Além da própria implementação da Microsoft, uma implementação de plataforma cruzada do CLR conhecida como Mono funciona com outros sistemas operacionais não pertencentes à Microsoft, bem como com o Windows. Para o uso ideal do IronPython, você precisa de alguma familiaridade com as bibliotecas CLR fundamentais. Você não precisa codificar em C#, mas a documentação e os exemplos de bibliotecas CLR existentes geralmente são expressos em termos de C#, portanto, você precisa de uma familiaridade com C# para lê-los e entendê-los. Você também precisa usar ferramentas de suporte CLR para tarefas como fazer assemblies CLR. Este livro trata do Python, não do CLR. Para uso do IronPython, você deve complementar este livro com a própria documentação on-line do IronPython e, se necessário, alguns dos muitos outros recursos disponíveis sobre .NET, CLR, C#, Mono e assim por diante.

PyPy

PyPy é uma implementação rápida e flexível do Python, codificada em um subconjunto do próprio Python, capaz de direcionar várias linguagens de nível inferior e máquinas virtuais usando técnicas avançadas, como inferência de tipos. A maior força do PyPy é sua capacidade de gerar código de máquina nativo “just in time” enquanto executa seu programa Python. O PyPy implementa a v2 e, até o momento, o suporte ao Python 3.5 está sendo lançado em

alfa. O PyPy tem vantagens substanciais no gerenciamento de velocidade e memória e está sendo usado em nível de produção.

Escolhendo entre CPython, Jython, IronPython e PyPy Se sua plataforma,

como a maioria, é capaz de executar todos os CPython, Jython, IronPython e PyPy, como você escolhe entre eles? Em primeiro lugar, não escolha prematuramente: baixe e instale todos eles. Eles coexistem sem problemas e são todos gratuitos.
Tê-los todos em sua máquina de desenvolvimento custa apenas algum tempo de download e um pouco de espaço extra em disco, e permite compará-los diretamente.

A principal diferença entre as implementações é o ambiente em que são executadas e as bibliotecas e estruturas que podem usar. Se você precisa de um ambiente JVM, Jython é sua melhor escolha. Se você precisar de um ambiente CLR (também conhecido como ".NET"), aproveite o IronPython. Se você precisa de uma versão personalizada do Python ou precisa de alto desempenho para programas de execução longa, considere o PyPy.

Se você estiver trabalhando principalmente em um ambiente tradicional, o CPython é uma excelente opção. Se você não tem uma forte preferência por um ou outro, comece com a implementação de referência padrão do CPython, que é mais amplamente suportada por add-ons e extensões de terceiros, e oferece a altamente preferível v3 (em toda a glória 3.5).1

Em outras palavras, quando você estiver experimentando, aprendendo e testando coisas, use CPython, o mais amplamente suportado e maduro. Para desenvolver e implantar, sua melhor escolha depende dos módulos de extensão que deseja usar e como deseja distribuir seus programas. Os aplicativos CPython geralmente são mais rápidos do que Jython ou IronPython, especialmente se você usar módulos de extensão como NumPy (abordado em "Processamento de Array" na página 444); no entanto, o PyPy geralmente pode ser ainda mais rápido, graças à compilação just-in-time para o código da máquina. Pode valer a pena comparar seu código CPython com o PyPy.

CPython é mais maduro: já existe há mais tempo, enquanto Jython, IronPython e PyPy são mais novos e menos comprovados no campo. O desenvolvimento das versões CPython tende a avançar mais do que Jython, IronPython e PyPy: no momento da escrita, por exemplo, o nível de linguagem suportado é, para v2, 2.7 para todos eles; para v3, 3.5 e 3.6 agora estão disponíveis em CPython e PyPy tem 3.5 em alfa.

No entanto, o Jython pode usar qualquer classe Java como um módulo de extensão, seja a classe proveniente de uma biblioteca Java padrão, de uma biblioteca de terceiros ou de uma biblioteca desenvolvida por você. Da mesma forma, o IronPython pode usar qualquer classe CLR, seja das bibliotecas CLR padrão ou codificadas em C#, Visual Basic .NET ou outras linguagens compatíveis com CLR. O PyPy é compatível com a maioria das bibliotecas CPython padrão e

1 Ao longo deste livro, mencionaremos alguns destaques do Python 3.6, recém-lançado enquanto escrevemos - embora o 3.6 ainda não esteja maduro o suficiente para você usar em lançamentos de produção, saber o que está por vir quando o 3.6 amadurecer não pode deixar de ser útil para suas escolhas técnicas.

com extensões codificadas em C usando Cython, abordado em “Cython” na página 678, e CFFI, mencionado em “Estendendo o Python sem a API C do Python” na página 677.

Um aplicativo codificado em Jython é um aplicativo Java 100% puro, com todas as vantagens e problemas de implantação do Java, e é executado em qualquer máquina de destino com uma JVM adequada. As oportunidades de empacotamento também são idênticas às do Java. Da mesma forma, um aplicativo codificado em IronPython é totalmente compatível com as especificações do .NET.

Jython, IronPython, PyPy e CPython são implementações boas e fiéis do Python, razoavelmente próximas umas das outras em termos de usabilidade e desempenho. Como cada uma das plataformas JVM e CLR carrega muita bagagem, mas também fornece grandes quantidades de bibliotecas, estruturas e ferramentas úteis, qualquer uma das implementações pode desfrutar de vantagens práticas decisivas em um cenário de implantação específico. É sábio familiarizar-se com os pontos fortes e fracos de cada um e, então, escolher o ideal para cada tarefa de desenvolvimento.

Outros desenvolvimentos, implementações e distribuições O Python tornou-

se popular o suficiente para que haja vários outros grupos e indivíduos que se interessaram por seu desenvolvimento e forneceram recursos e implementações fora do foco da equipe principal de desenvolvimento.

Os programadores novos no Python costumam achar difícil de instalar. Atualmente, quase todos os sistemas baseados em Unix incluem uma implementação v2, e versões recentes do Ubuntu incluem v3 como o “sistema Python”. Se você leva a sério o desenvolvimento de software em Python, a primeira coisa que deve fazer é deixar o Python instalado no sistema em paz! Independentemente de qualquer outra coisa, o Python é cada vez mais usado pelo próprio sistema operacional, portanto, ajustar a instalação do Python pode causar problemas.

Portanto, considere a instalação de uma ou mais implementações do Python que você pode usar livremente para sua conveniência de desenvolvimento, sabendo que nada do que você faz afeta o sistema operacional. Também recomendamos o uso de ambientes virtuais (consulte “Ambientes Python” na página 186) para isolar projetos uns dos outros, permitindo que eles façam uso do que de outra forma poderiam ser dependências conflitantes (por exemplo, se dois de seus projetos exigirem versões diferentes do mesmo módulo).

A popularidade do Python levou à criação de muitas comunidades ativas, e o ecossistema da linguagem é muito ativo. As seções a seguir descrevem alguns dos desenvolvimentos recentes mais interessantes, mas nossa falha em incluir um projeto aqui reflete limitações de espaço e tempo, em vez de implicar qualquer desaprovação de nossa parte.

Pyjion

Pyjion é um projeto de código aberto da Microsoft cujo objetivo principal é adicionar uma API ao CPython para gerenciar compiladores just-in-time (JIT). Os objetivos secundários incluem a criação de um compilador JIT para o ambiente CLR principal da Microsoft, que embora seja o coração do ambiente .NET agora é de código aberto e uma estrutura para o desenvolvimento de compiladores JIT. Portanto, não é uma implementação do Python, mas o mencionamos porque oferece o potencial de traduzir o bytecode do CPython em um código altamente eficiente para muitos ambientes diferentes. A integração do Pyjion no CPython será habilitada pelo PEP 523, que foi implementado em Python
3.6.

IPython

O IPython aprimora o CPython padrão para torná-lo mais poderoso e conveniente para uso interativo. O IPython estende as capacidades do interpretador ao permitir sintaxe de chamada de função abreviada e funcionalidade extensível conhecida como mágicas introduzidas pelo caractere de porcentagem (%) . Ele também fornece shell escapes, permitindo que uma variável Python receba o resultado de um comando shell. Você pode usar um ponto de interrogação para consultar a documentação de um objeto (dois pontos de interrogação para documentação estendida); todos os recursos padrão do Python também estão disponíveis.

O IPython fez progressos particulares no mundo científico e focado em dados e lentamente se transformou (através do desenvolvimento do IPython Notebook, agora refatorado e renomeado como Jupyter Notebook e mostrado na figura a seguir) em um ambiente de programação interativo que, entre trechos de código, também permite que você incorpore comentários no estilo de programação letrada (incluindo notação matemática) e mostre a saída do código em execução, com gráficos avançados produzidos por subsistemas como matplotlib e bokeh. Um exemplo de gráfico matplotlib é mostrado na metade inferior da figura. Financiado de forma tranquilizadora nos últimos anos, o Jupyter/IPython é uma das histórias de sucesso mais proeminentes do Python.

Moving Window Average

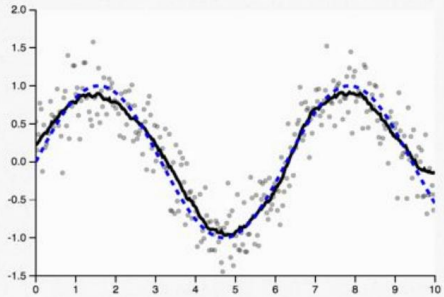
```
In [6]: np.random.seed(0)
        t = np.linspace(0, 10, 300)
        x = np.sin(t)
        dx = np.random.normal(0, 0.3, 300)

        kernel = np.ones(25) / 25.
        x_smooth = np.convolve(x + dx, kernel, mode='same')

        fig, ax = plt.subplots()
        ax.plot(t, x + dx, linestyle='', marker='o',
                color='black', markersize=3, alpha=0.3)
        ax.plot(t, x_smooth, '-k', lw=3)
        ax.plot(t, x, '--k', lw=3, color='blue')

        display_d3(fig)
```

Out[6]:



MicroPython: seu filho já é programador?

A tendência contínua de miniaturização colocou o Python bem dentro do alcance do hobby. Computadores de placa única como o Raspberry Pi e o Beagle Board permitem executar o Python em um ambiente Linux completo. Abaixo desse nível, existe uma interessante classe de dispositivos conhecidos como microcontroladores, chips programáveis com hardware configurável. Isso amplia o escopo de projetos amadores e profissionais - por exemplo, facilitando a detecção analógica e digital, permitindo aplicações como medições de luz e temperatura com pouco hardware adicional.

Exemplos típicos no momento em que escrevo foram o Adafruit Feather e o WiPy, mas novos dispositivos aparecem (e desaparecem) o tempo todo. Graças ao projeto MicroPython , esses dispositivos, alguns dos quais vêm com funcionalidades avançadas como WiFi, agora podem ser programados em Python.

MicroPython é uma implementação do Python 3 que pode produzir bytecode ou código de máquina executável (embora muitos usuários desconhecem o último fato). Ele implementa a sintaxe da v3, mas não possui a maior parte da biblioteca padrão. Módulos de controle de hardware especiais permitem a configuração e acionamento de várias peças de hardware configurável integrado, e o acesso à biblioteca de soquetes do Python permite que os dispositivos sejam clientes para serviços de Internet. Dispositivos externos e eventos de timer podem acionar o c

Um dispositivo normalmente oferece acesso de intérprete por meio de uma porta serial USB ou pelo protocolo Telnet (ainda não temos conhecimento de nenhuma implementação de ssh , portanto, tome cuidado com o firewall desses dispositivos corretamente: eles não devem ser acessados diretamente pela Internet sem as devidas precauções!). Você pode programar a sequência de inicialização do dispositivo em Python criando um arquivo boot.py na memória do dispositivo, e esse arquivo pode executar código Python arbitrário de qualquer complexidade.

Graças ao MicroPython, a linguagem Python pode ser um player completo na Internet das Coisas. No momento em que escrevo, o dispositivo micro:bit da BBC compatível com Python está sendo distribuído para todos os alunos do 7º ano no Reino Unido (cerca de um milhão).
Tem certeza de que seus filhos ainda não conhecem um pouco de Python?

Anaconda e Miniconda

Uma das distribuições Python de maior sucesso nos últimos anos é o Anaconda da Continuum Analytics. Este pacote de código aberto vem com um grande número de módulos pré-configurados e testados, além da biblioteca padrão. Em muitos casos, você pode descobrir que ele contém todas as dependências necessárias para o seu trabalho.
Em sistemas baseados em Unix, ele é instalado de maneira muito simples em um único diretório: para ativá-lo, basta adicionar o diretório bin do Anaconda na frente do seu shell PATH.

Anaconda é uma distribuição Python baseada em uma tecnologia de empacotamento chamada conda. Uma implementação irmã, Miniconda, dá acesso às mesmas bibliotecas, mas não vem com elas pré-carregadas, tornando-o mais adequado para a criação de ambientes sob medida. O Conda não usa os ambientes virtuais padrão, mas contém recursos equivalentes para permitir a separação das dependências para vários projetos.

Nuitka: Converta seu Python para C++

Nuitka é um compilador alternativo que converte seu código Python para C++, com suporte para v2 e v3. O Nuitka permite uma otimização conveniente do seu código, com velocidades atuais duas vezes mais rápidas que o CPython. Os marcos futuros incluem melhor inferência de tipos e ainda mais velocidade. A página inicial da Nuitka tem mais informações.

Grumpy: um transpiler Python-to-Go

Grumpy é um compilador experimental que converte seu código Python para Go, com suporte apenas para v2; o código Go então compila para um executável nativo, permitindo a importação de módulos codificados em Go (embora não de extensões Python normais) e alta escalabilidade. O Grumpy é otimizado para atender a grandes cargas de trabalho paralelas com muitos encadeamentos leves, podendo um dia substituir o CPython como o tempo de execução usado para atender aos milhões de solicitações do YouTube por segundo, conforme explicado no blog do Google Open Source . O Grumpy é mantido no projeto GitHub Grumpy.

Transcriptação: Converta seu Python em JavaScript

Muitas tentativas foram feitas para determinar como o Python poderia se tornar uma linguagem baseada em navegador, mas o domínio do JavaScript tem sido tenaz. O sistema Transcrypt é um pacote Python instalável por pip para converter código Python em executável pelo navegador

JavaScript. Você tem acesso total ao DOM do navegador, permitindo que seu código manipule dinamicamente o conteúdo da janela e use bibliotecas JavaScript.

Embora crie código minificado, o Transcrypt fornece mapas de origem completos que permitem depurar com referência à fonte Python em vez do Javascript gerado. Você pode escrever manipuladores de eventos do navegador em Python, misturando-o livremente com HTML e JavaScript. O Python pode nunca se tornar uma linguagem de navegador de primeira classe, mas o Transcrypt significa que você pode não precisar mais se preocupar com isso.

Outro projeto (mais maduro, mas muito ativo) para permitir que você crie scripts para suas páginas da Web com Python é o Brython, e ainda existem outros. Para ajudá-lo a escolher entre eles, você pode encontrar informações e links úteis neste tópico do Stack Overflow.

Questões de licenciamento e preço

O CPython é coberto pela Python Software Foundation License Versão 2, que é compatível com GNU Public License (GPL), mas permite que você use Python para qualquer desenvolvimento de software proprietário, gratuito ou de código aberto, semelhante ao BSD/Apache/ licenças MIT. As licenças de Jython, IronPython e PyPy são igualmente liberais. Qualquer coisa que você baixar dos principais sites Python, Jython, IronPython e PyPy não custará um centavo. Essas licenças não restringem quais licenças e condições de preço você pode usar para o software que desenvolver usando as ferramentas, bibliotecas e documentação que cobrem.

No entanto, nem tudo relacionado ao Python está livre de custos de licenciamento ou aborrecimentos. Muitas fontes, ferramentas e módulos de extensão do Python de terceiros que você pode baixar gratuitamente têm licenças liberais, semelhantes às do próprio Python. Outros são cobertos pela GPL ou Lesser GPL (LGPL), restringindo as condições de licenciamento que você pode colocar em trabalhos derivados. Alguns módulos e ferramentas desenvolvidos comercialmente podem exigir que você pague uma taxa, incondicionalmente ou se você os usar para fins lucrativos.

Nada substitui o exame cuidadoso das condições de licenciamento e preços.
Antes de investir tempo e energia em qualquer ferramenta ou componente de software, verifique se você pode viver com sua licença. Muitas vezes, especialmente em um ambiente corporativo, tais questões legais podem envolver a consulta de advogados. Os módulos e ferramentas abordados neste livro, a menos que digamos explicitamente o contrário, podem ser considerados, no momento da redação deste livro, para download gratuito, código-fonte aberto e cobertos por uma licença liberal semelhante à do Python. No entanto, não reivindicamos nenhum conhecimento jurídico e as licenças podem mudar com o tempo, portanto, é sempre prudente verificar duas vezes.

Desenvolvimento Python e Versões

O Python é desenvolvido, mantido e lançado por uma equipe de desenvolvedores principais liderada por Guido van Rossum, inventor, arquiteto e Benevolent Dictator For Life (BDFL) do Python. Este título significa que Guido tem a palavra final sobre o que se torna parte da linguagem Python e da biblioteca padrão. A propriedade intelectual do Python pertence à Python Software Foundation (PSF), uma corporação sem fins lucrativos dedicada à promoção do Python, descrita em “Python Software Foundation” na página 16 .

atualmente Mercurial, mas fazendo a transição para o git, conforme documentado no Python Developer's Guide), e a maioria dos committers do Python são membros ou Fellows do PSF.

As mudanças propostas para Python são detalhadas em documentos públicos chamados Python Enhancement Proposals (PEPs), debatidas (às vezes um voto consultivo é feito) por desenvolvedores Python e pela comunidade Python mais ampla e, finalmente, aprovadas ou rejeitadas por Guido (ou seus delegados). que levam em consideração os debates e votações, mas não estão vinculados a eles. Centenas de pessoas contribuem para o desenvolvimento do Python por meio de PEPs, discussões, relatórios de bugs e patches para fontes, bibliotecas e documentos do Python.

A equipe principal do Python lança versões secundárias do Python (3.x para valores crescentes de x), também conhecidas como “lançamentos de recursos”, atualmente em um ritmo de aproximadamente uma vez a cada 18 meses. O Python 2.7, lançado pela primeira vez em julho de 2010, foi o último recurso lançado da série 2.x. As versões atuais e futuras da v2 são e serão apenas versões pontuais de correção de bugs, sem adicionar nenhum recurso. No momento em que este livro foi escrito, a versão atual de correção de bug 2.7.13 está disponível. Nunca haverá uma versão 2.8: recomendamos migrar para v3.

Cada versão secundária v3 (em oposição à versão de ponto de correção de bug) adiciona recursos que tornam o Python mais poderoso e mais simples de usar, mas também cuida para manter a compatibilidade com versões anteriores. O Python 3.0, que foi autorizado a quebrar a compatibilidade com versões anteriores para remover recursos “legados” redundantes e simplificar a linguagem, foi lançado pela primeira vez em dezembro de 2008. O Python 3.5 (a versão que abordamos como “v3”) foi lançado pela primeira vez em setembro de 2015; 3.6 foi lançado em dezembro de 2016, quando estávamos terminando de escrever este livro.

Cada versão secundária 3.x é disponibilizada primeiro em versões alfa, marcadas como 3.x a0, 3.x a1 e assim por diante. Depois dos alfas, vem pelo menos um lançamento beta, 3.x b1, e depois dos betas, pelo menos um candidato a lançamento, 3.x rc1. No momento em que a versão final do 3.x (3.x.0) sai, ele é bastante sólido, confiável e testado em todas as principais plataformas. Qualquer programador Python pode ajudar a garantir isso baixando alfas, betas e candidatos a lançamento, testando-os e preenchendo relatórios de erros para quaisquer problemas que surjam.

Assim que um lançamento secundário é lançado, parte da atenção da equipe principal muda para o próximo lançamento secundário. No entanto, uma versão secundária normalmente obtém versões pontuais sucessivas (ou seja, 3.x.1, 3.x.2 e assim por diante) que não adicionam nenhuma funcionalidade, mas podem corrigir erros, portar o Python para novas plataformas, aprimorar a documentação e adicionar otimizações e ferramentas.

Este livro se concentra no Python 2.7 e 3.5 (e em todos os seus lançamentos pontuais), os lançamentos mais estáveis e difundidos no momento em que este livro foi escrito (com notas laterais sobre os destaques do lançamento mais recente, 3.6). Python 2.7 é o que chamamos de v2, enquanto 3.5 (com seus sucessores) é o que chamamos de v3. Documentamos quais partes da linguagem e bibliotecas existem apenas na v3 e não podem ser usadas na v2, bem como as da v3 que podem ser usadas na v2, por exemplo, com a sintaxe de importação de __future__ (consulte Importações de __future__ na página 707) . Sempre que dizemos que um recurso está na v3, queremos dizer que está no Python 3.5 (e presumivelmente em todas as versões futuras, incluindo a 3.6, mas não necessariamente na, digamos, 3.4); dizer que um recurso está na v2 significa que ele está no Python 2.7.x (atualmente 2.7.13), mas não necessariamente no 2.6 ou anterior. Em alguns casos menores, especificamos uma liberação pontual, como 3.5.3, para alterações de idioma em que a liberação pontual é importante.

Este livro não aborda versões mais antigas do Python, como 2.5 ou 2.6; essas versões têm mais de cinco anos e não devem ser usadas para nenhum desenvolvimento.
No entanto, você pode ter que se preocupar com essas versões legadas se elas estiverem incorporadas em algum aplicativo do qual você precisa fazer o script. Felizmente, a compatibilidade com versões anteriores do Python é bastante boa nas versões principais: v2 é capaz de processar adequadamente quase todos os programas Python válidos que foram escritos para Python 1.5.2 ou posterior. Como mencionado, no entanto, o Python 3 foi um lançamento importante que quebrou a compatibilidade com versões anteriores, embora seja possível escrever um código que funcione tanto na v2 quanto na v3 se for tomado cuidado.
Consulte o Capítulo 26 para saber mais sobre como migrar o código legado para a v3. Você pode encontrar código e documentação online para todas as versões antigas do Python.

Recursos do Python

O recurso Python mais rico é a web. O melhor ponto de partida é a página inicial do Python, cheia de links interessantes para explorar. O site Jython é obrigatório se você estiver interessado em Jython. IronPython tem um site .net. PyPy está documentado no site PyPy.

Documentação

Python, Jython, IronPython e PyPy vêm com boa documentação. Os manuais estão disponíveis em vários formatos, adequados para visualização, pesquisa e impressão.
Você pode ler os manuais do CPython online (geralmente nos referimos a eles como “os documentos online”): tanto a v2 quanto a v3 estão disponíveis. Você também pode encontrar vários formatos para download para v2 e v3. A página de documentação do Python possui links para uma grande variedade de documentos. A página de documentação do Jython possui links para documentos específicos do Jython, bem como documentos gerais do Python; também existem páginas de documentação para IronPython e para PyPy. Você também pode encontrar documentos de perguntas frequentes (FAQ) para Python, Jython, IronPython e PyPy online.

Documentação do Python para não programadores

A maior parte da documentação do Python (incluindo este livro) pressupõe algum conhecimento de desenvolvimento de software. No entanto, o Python é bastante adequado para programadores iniciantes, portanto, há exceções a essa regra. Bons textos introdutórios on-line para não programadores incluem:

- “Tutorial para Python para não programadores” de Josh Cogliati, disponível online em duas versões: para v2 e para v3
- “Learning to Program” de Alan Gauld, disponível online (abrange tanto a v2 quanto a v3)
- “Think Python, 2ª edição”, de Allen Downey, disponível online (abrange principalmente a v3, com notas para uso da v2)

Um excelente recurso para aprender Python (para não programadores e também para programadores menos experientes) é o wiki do Guia para Iniciantes, que inclui uma riqueza de

links e conselhos. É curado pela comunidade, portanto, provavelmente permanecerá atualizado à medida que os livros, cursos, ferramentas disponíveis e assim por diante continuarem evoluindo e melhorando.

Módulos de extensão e fontes Python Um

bom ponto de partida para extensões e fontes Python é o Python Package Index (ainda conhecido pelos veteranos como “The Cheese Shop” e geralmente referido agora como PyPI), que no momento em que este livro foi escrito oferece mais de 95.000 pacotes,2 cada um com descrições e indicações.

A distribuição padrão do código-fonte Python contém excelente código-fonte Python na biblioteca padrão e nos diretórios Demos e Tools, bem como fonte C para os vários módulos de extensão integrados. Mesmo que você não tenha interesse em compilar o Python a partir do código-fonte, sugerimos que você baixe e descompacte a distribuição do código-fonte do Python para fins de estudo; ou, se preferir, leia-o online.

Muitos módulos e ferramentas do Python abordados neste livro também possuem sites dedicados. Incluímos referências a esses sites nos capítulos apropriados deste livro.

livros

Embora a web seja uma rica fonte de informação, os livros ainda têm seu lugar (se não concordássemos com isso, não teríamos escrito este livro e você não o estaria lendo). Livros sobre Python são numerosos. Aqui estão alguns que recomendamos, embora alguns cubram versões mais antigas em vez das atuais:

- Se você está apenas aprendendo a programar, Python for Dummies, de Stef e Aahz Maruch (For Dummies), embora datado (2006), é uma ótima introdução à programação e ao Python elementar (v2) em particular .
- Se você conhece um pouco de programação, mas está apenas começando a aprender Python, Use a Cabeça Python, 2ª edição, de Paul Barry (O'Reilly), pode ser útil para você. Como todos os livros da série Use a Cabeça , ele usa gráficos e humor para ensinar seu

assunto. • Dive Into Python e Dive Into Python 3, de Mark Pilgrim (APress), ensinam pelo exemplo, de forma rápida e completa, bastante adequado para pessoas que já são programadores especialistas em outras linguagens. Você também pode baixar gratuitamente o livro, em qualquer um dos vários formatos, para v2 ou v3. •

Beginning Python: From Novice to Professional, de Magnus Lie Hetland (APress), ensina tanto por meio de explicações completas quanto pelo desenvolvimento completo de 10 programas completos em várias áreas de aplicação.

- Python Essential Reference, de David Beazley (New Riders), é uma referência completa importância para a linguagem Python e suas bibliotecas padrão.

2 Quando você ler isto, provavelmente haverá mais de 100.000 pacotes.

- Para obter um resumo conciso e um lembrete dos fundamentos do Python, confira o Python Pocket Reference, de Mark Lutz (O'Reilly).

Comunidade

Um dos maiores pontos fortes do Python é sua comunidade robusta, amigável e receptiva. Os programadores e colaboradores do Python encontram o f2f3 em conferências, "hackathons" (geralmente conhecidos como sprints na comunidade Python) e grupos de usuários locais; discutir ativamente interesses compartilhados; e ajudar uns aos outros em listas de discussão e mídias sociais

Python Software Foundation

Além de deter os direitos de propriedade intelectual da linguagem de programação Python, grande parte da missão da Python Software Foundation (PSF) é "apoiar e facilitar o crescimento de uma comunidade diversificada e internacional de programadores Python". O PSF patrocina grupos de usuários, conferências e sprints e fornece subsídios para desenvolvimento, divulgação e educação, entre outras atividades.
O PSF tem dezenas de Fellows (indicados por suas contribuições ao Python e incluindo toda a equipe principal do Python, bem como todos os autores deste livro); centenas de membros que contribuem com tempo, trabalho e dinheiro; e dezenas de patrocinadores corporativos. Sob o novo modelo de associação aberta, qualquer pessoa que use e ofereça suporte ao Python pode se tornar membro do PSF. Confira a página de adesão para obter informações sobre como se tornar um membro do PSF. Se estiver interessado em contribuir com o próprio Python, consulte o Python Developer's Guide.

Conferências Python

Existem muitas conferências Python em todo o mundo. As conferências gerais sobre Python incluem conferências internacionais e regionais, como PyCon e EuroPython, e outras mais localizadas, como PyOhio e Pycon Italia. Conferências tópicas incluem SciPy, PyData e DjangoCon. As conferências geralmente são seguidas por sprints de codificação, onde os colaboradores do Python se reúnem para vários dias de codificação focados em projetos de código aberto específicos e em muita camaradagem. Você pode encontrar uma lista de conferências na página Community Workshops . Vídeos de palestras de conferências anteriores podem ser encontrados no site pyvideo.

Grupos de usuários e

encontros A comunidade Python tem grupos de usuários locais em todos os continentes, exceto na Antártica, conforme listado no wiki de Grupos de usuários locais. Além disso, existem encontros Python listados para mais de 70 países. Women in Python tem um grupo internacional de PyLadies, bem como capítulos locais de PyLadies.

3 Cara a cara (ou seja, pessoalmente).

Grupos de Interesse Especial

As discussões sobre alguns assuntos especializados relacionados ao Python ocorrem nas listas de discussão dos Grupos de Interesse Especial do Python (SIGs). A página SIGS possui uma lista e indicações para informações gerais e específicas sobre eles. Mais de uma dúzia de SIGs estão ativos no momento em que este livro foi escrito. Aqui estão alguns exemplos:

• cplusplus-sig: Ligações entre C++ e Python • mobile-sig: Python em

dispositivos móveis • edu-sig: Python na educação

Mídia social

Para um feed RSS de blogs relacionados a Python, consulte planetpython ou siga-os no Twitter @planetpython. No Twitter, você também pode seguir @ThePSF. O FreeNode no IRC hospeda vários canais relacionados ao Python: o principal é o #python. No Facebook, você pode ingressar no grupo público Python Software Developers ; no Google+, a comunidade não oficial do Python; no LinkedIn, verifique sua página Python para saber mais. Perguntas e respostas técnicas sobre Python também podem ser encontradas e acompanhadas em stackoverflow.com sob uma variedade de tags, incluindo python, python-2.7, python-3.x e outros. Python é um tópico com curadoria ativa no Stack Overflow, onde muitas respostas úteis com discussões esclarecedoras podem ser encontradas.

Listas de discussão e grupos de notícias

A página de listas da comunidade tem links para listas de discussão e grupos de notícias relacionados ao Python. O grupo de notícias da Usenet para discussões sobre Python é comp.lang.python. O grupo de notícias também está disponível como uma lista de discussão; para assinar, preencha o formulário. Anúncios oficiais relacionados a Python são postados em python anunciar, ou sua lista de discussão equivalente.
Para pedir ajuda individual com problemas específicos, escreva para help@python.org. Para obter ajuda para aprender ou ensinar Python, escreva para tutor@python.org ou entre na lista. Para problemas ou discussões sobre portabilidade entre v2 e v3, junte-se à lista de portabilidade de python .

Para um resumo semanal útil de notícias e artigos relacionados a Python, você pode se inscrever no Python Weekly.

Em 2015, a Python Software Foundation criou uma lista de discussão voltada para a comunidade para dar suporte ao seu novo modelo de associação aberta, na qual você pode se inscrever.

Instalação

Você pode instalar o Python nas versões clássica (CPython), JVM (Jython), .NET (IronPython) e PyPy, na maioria das plataformas. Com um sistema de desenvolvimento adequado (C para CPython, Java para Jython, .NET para IronPython; PyPy, codificado no próprio Python, só precisa do CPython instalado primeiro), você pode instalar as versões Python das respectivas distribuições de código-fonte. Em plataformas populares, você também tem a alternativa (altamente recomendada!) de instalar a partir de distribuições binárias pré-compiladas.

Instalando o Python se ele vier pré-instalado Se sua plataforma vier com uma versão pré-instalada do Python, é melhor instalar outra versão independente e atualizada. Ao fazer isso, você não deve remover ou sobrescrever a versão original de sua plataforma – em vez disso, instale a outra versão “lado a lado” com a primeira. Dessa forma, você garante que não vai atrapalhar nenhum outro software que faça parte da sua plataforma: esse software pode depender da versão exata do Python que veio com a própria plataforma.

A instalação do CPython a partir de uma distribuição binária é mais rápida, economiza trabalho substancial em algumas plataformas e é a única possibilidade se você não tiver um compilador C adequado. Instalar a partir do código-fonte oferece mais controle e flexibilidade e é obrigatório se você não conseguir encontrar uma distribuição binária pré-compilada adequada para sua plataforma. Mesmo se você instalar a partir de binários, é melhor baixar também a distribuição de origem, pois inclui exemplos e demos que podem estar faltando em binários pré-construídos.

Instalando o Python a partir de binários

Se sua plataforma for popular e atual, você encontrará versões binárias empacotadas e pré-construídas do Python prontas para instalação. Os pacotes binários são normalmente auto-instalados, diretamente como programas executáveis ou por meio de ferramentas de sistema apropriadas, como o RedHat Package Manager (RPM) em algumas versões do Linux e o Microsoft Installer (MSI) no Windows. Depois de baixar um pacote, instale-o executando o programa e escolhendo os parâmetros de instalação, como o diretório onde o Python será instalado. No Windows, selecione a opção "Adicionar Python 3.5 ao PATH" para que o instalador adicione o local de instalação ao PATH para usar o Python facilmente em um prompt de comando (consulte " O programa python" na página 23).

Para fazer o download dos binários “oficiais” do Python, visite o site do Python, clique em Downloads e, na página seguinte, clique no botão chamado Python 2.7 ou Python 3.x para fazer o download do binário adequado à plataforma do navegador.

Muitos terceiros fornecem instaladores binários gratuitos do Python para outras plataformas. Existem instaladores para distribuições Linux, seja sua distribuição baseada em RPM (RedHat, Fedora, Mandriva, SUSE, etc.) ou Debian e Ubuntu. A página Other Downloads fornece links para distribuições binárias para plataformas agora um tanto exóticas, como OS/2, RISC OS, IBM AS/400 e assim por diante.

O Anaconda é uma distribuição binária que inclui v2 ou v3, mais o gerenciador de pacotes conda e centenas de extensões de terceiros, principalmente para ciências, matemática, engenharia e análise de dados. Está disponível para Linux, Windows e macOS. O mesmo pacote, mas sem todas essas extensões (você pode instalar seletivamente subconjuntos delas com conda), também está disponível e é conhecido como Miniconda.

Mac OS

O macOS da Apple vem com v2 (somente modo de texto), até o momento. No entanto, recomendamos que você instale a versão mais recente e os aprimoramentos seguindo as instruções e links no site do Python em Downloads; devido aos ciclos de lançamento da Apple, a versão do Python incluída na sua versão do macOS pode estar desatualizada ou incompleta. A versão mais recente do Python é instalada além, e não no lugar, da versão fornecida pela Apple; A Apple usa sua própria versão do Python e extensões proprietárias para implementar alguns dos softwares que distribui como parte do macOS, portanto, não é aconselhável arriscar perturbar essa versão de qualquer maneira.

O popular gerenciador de pacotes de código aberto macOS de terceiros, Homebrew, oferece, entre muitos outros pacotes, excelentes versões do Python, v2 e v3. Cuidado, porém, que geralmente apenas uma versão de cada está disponível.

Instalando o Python a partir do código-fonte

Para instalar o CPython a partir do código-fonte, você precisa de uma plataforma com um compilador C compatível com ISO e ferramentas como make. No Windows, a maneira normal de criar o Python é com o Visual Studio (idealmente VS2008 e VS2010 para v2 ou VS 2015 para v3).

Para baixar o código-fonte do Python, visite o site do Python, clique em Downloads e escolha v2 ou v3. Para mais opções, passe o mouse sobre Downloads e use o menu exibido.

A extensão de arquivo .tgz sob o link denominado “tarball de origem Gzipada” é equivalente a .tar.gz (ou seja, um arquivo tar de arquivos compactados pelo poderoso e popular compressor gzip). Como alternativa, use o link “XZ compactado fonte tarball” para obter uma versão com uma extensão de .tar.xz em vez de .tgz, compactado com o compressor xz ainda mais poderoso, se você tiver ferramentas que possam lidar com a compactação XZ.

Microsoft Windows

No Windows, instalar o Python a partir do código-fonte pode ser uma tarefa árdua, a menos que você já esteja familiarizado com o Visual Studio e esteja acostumado a trabalhar em janelas orientadas a texto conhecidas como prompt de comando.

Se as instruções a seguir causarem problemas, continue com “Instalando o Python a partir de binários” na página 18. É melhor fazer uma instalação separada dos binários de qualquer maneira, mesmo se você também instalar a partir do código-fonte. Se você notar algo estranho ao usar a versão que instalou da fonte, verifique novamente a instalação dos binários.
Se a estranheza desaparecer, deve ser devido a alguma peculiaridade em sua instalação da fonte, então você sabe que deve verificar novamente o último.

Nas seções a seguir, para maior clareza, assumimos que você criou uma nova pasta chamada %USERPROFILE%\py, (por exemplo, c:\users\tim\py). Abra esta pasta digitando, por exemplo, **%USERPROFILE%** na barra de endereço de uma janela do Windows Explorer e continue a partir daí. Faça o download do arquivo tgz de origem — por exemplo, Python-3.5.2.tgz (ou outra versão do Python de sua escolha) para essa pasta. Claro, nomeie e coloque a pasta como melhor lhe convier: nossa escolha de nome é apenas para fins expositivos.

Descompactando e descompactando o código-fonte

Python Você pode descompactar e descompactar um arquivo .tgz ou .tar.xz, por exemplo, com o programa gratuito 7-Zip. Baixe e instale o 7-Zip da página de download e execute-o no arquivo tgz das fontes (por exemplo, c:\users\tim\py\Python-3.5.2.tgz) que você baixou. (O arquivo tgz pode ter baixado para %USERPROFILE%\downloads, caso em que você precisa movê-lo para sua pasta \py antes de descompactá-lo com 7-Zip.) Agora você tem uma pasta %USERPROFILE% \py\Python-3.5 .2 (ou outra versão baixada originalmente), a raiz de uma árvore que contém toda a distribuição padrão do Python em forma de fonte.

Construindo o código-fonte

Python Abra o arquivo de texto %USERPROFILE%\py\Python-3.5.2\PCBuild\readme.txt (ou qualquer versão do Python que você baixou) com o editor de texto de sua escolha e siga as instruções detalhadas encontradas lá .

Plataformas semelhantes ao Unix

Em plataformas do tipo Unix, a instalação do Python a partir do código-fonte geralmente é simples. Nas seções a seguir, para maior clareza, presumimos que você criou um novo diretório chamado ~/Py e baixou o arquivo tgz do código-fonte, por exemplo, Python-3.5.2.tgz (ou outra versão do Python de sua escolha) para esse diretório. Claro, nomeie e coloque o diretório da maneira que melhor lhe convier: nossa escolha de nome é apenas para fins expositivos.

Descompactando e descompactando o código-fonte

Python Você pode descompactar e descompactar um arquivo .tgz ou .tar.xz com a popular versão GNU do tar. Basta digitar o seguinte em um prompt de shell:

```
$ cd ~/Py
$ tar xzf Python-3.5.2.tgz
```

Agora você tem um diretório ~/Py/Python-3.5.2, a raiz de uma árvore que contém toda a distribuição padrão do Python em forma de fonte.

Configurando, construindo e

testando Notas detalhadas estão em ~/Py/Python-3.5.2/README sob o título “Instruções de construção,” e recomendamos estudar essas notas. No caso mais simples, no entanto, tudo o que você precisa é fornecer os seguintes comandos em um prompt de shell:

```
$ cd ~/Py/Python-3.5.2 $ ./
configure
    [configurar grava muita informação - recortado aqui] $ make
[make
    leva um bom tempo e emite muita informação]
```

Se você executar **make** sem primeiro executar ./**configure, make** executa implicitamente ./**config ure.** Quando **o make** terminar, verifique se o Python que você acabou de criar funciona conforme o esperado:

```
$ fazer teste
    [demora bastante, emite muita informação]
```

Normalmente, **make test** confirma que sua compilação está funcionando, mas também informa que alguns testes foram ignorados porque módulos opcionais estavam faltando.

Alguns dos módulos são específicos da plataforma (por exemplo, alguns podem funcionar apenas em máquinas que executam o antigo sistema operacional Irix da SGI), então não se preocupe com eles. No entanto, outros módulos são ignorados porque dependem de outros pacotes de código aberto que não estão instalados em sua máquina. Por exemplo, o módulo _tkinter—necessário para rodar o pacote Tkinter GUI e o ambiente de desenvolvimento integrado IDLE, que vem com Python—só pode ser construído se ./**configure** puder encontrar uma instalação de Tcl/Tk 8.0 ou posterior em sua máquina. Consulte ~/Py/Python-3.5.2/README para obter mais detalhes e advertências específicas sobre diferentes plataformas Unix e semelhantes a Unix.

Construir a partir do código-fonte permite que você ajuste sua configuração de várias maneiras. Por exemplo, você pode criar Python de uma maneira especial que ajuda a rastrear vazamentos de memória ao desenvolver extensões Python codificadas em C, abordadas em “Construindo e instalando extensões Python codificadas em C” na página 643. ./**configure --help** é uma boa fonte de informações sobre as opções de configuração que você pode usar.

Instalando após a compilação

Por padrão, ./**configure** prepara o Python para instalação em /usr/local/bin e /usr/local/lib. Você pode alterar essas configurações executando ./**configure** com a opção **--prefix** antes de executar **make.** Por exemplo, se você deseja uma instalação privada do Python no subdiretório py35 do seu diretório inicial, execute:

```
$ cd ~/Py/Python-3.5.2 $ ./
configure --prefix=~/py35
```

e continue com **make** como na seção anterior. Assim que terminar de compilar e testar o Python, para executar a instalação real de todos os arquivos, execute:

```
$ fazer instalar
```

O usuário executando **make install** deve ter permissões de gravação nos diretórios de destino. Dependendo de sua escolha de diretórios de destino e permissões nesses diretórios, você pode precisar **su** para root, bin ou algum outro usuário ao executar **make install.** Um idioma comum para essa finalidade é **sudo make install:** se **o sudo** solicitar uma senha, digite a senha do usuário atual, não a do root. Uma abordagem alternativa e recomendada é instalar em um ambiente virtual, conforme abordado em “Ambientes Python” na página 186.

Instalando o Jython

Para baixar o Jython, visite a página inicial do Jython e siga o link Download. A versão mais recente, no momento da redação deste artigo, é o Jython 2.7.0, que oferece suporte ao Python 2.7 e vem como dois JARs executáveis: um apenas para executar o Jython e outro

para incorporá-lo em aplicativos Java. Execute a instalação seguindo as instruções de instalação.

Instalando o IronPython

Para instalar o IronPython, você precisa ter uma implementação atual do Common Language Runtime (CLR, AKA .NET) instalada em sua máquina. Tanto o Mono quanto o Microsoft .NET Framework funcionam bem com o IronPython. Para baixar o IronPython, visite a página inicial do IronPython e clique no ícone em forma de seta chamado Download IronPython 2.7; isso fornece um instalador executável - execute-o para instalar o IronPython.

Instalando o PyPy

Para instalar o PyPy, baixe a versão escolhida das fontes do PyPy (a versão mais recente, no momento em que este livro foi escrito, é o PyPy 5.6.0, mas não deixe que a numeração o confunda - ele suporta totalmente o Python 2.7, que chamamos de v2 em este livro!), então siga as instruções se quiser compilar o PyPy a partir dos fontes. Como alternativa, baixe um instalador binário para sua plataforma (se compatível), descompacte-o de seu formato .zip ou .tar.bz2 e execute o arquivo executável resultante.