PerguntasFrequentes / SobrePython

Conteúdo

O que é Python?

Python é uma linguagem de altíssimo nível (*VHLL - Very High Level Language*), de sintaxe moderna, orientada a objetos, interpretada via *bytecode*, com tipagem forte (não há conversões automáticas) e dinâmica (não há declaração de variáveis e elas podem conter diferentes objetos), modular, multiplataforma, de fácil aprendizado e de implementação livre. Python foi criada por GuidoVanRossum.

Como se pronuncia Python?

Deve-se dizer "Páifon",lembrando que "th" pronuncia-se colocando a língua entre os dentes e emitindo um som de "f". "Páiton" também é uma pronuncia aceitável.

De onde surgiu esse nome?

Ao contrário do que normalmente se pensa, a origem do nome da linguagem não é a espécie de serpente "Pitón" e sim o grupo inglês de humoristas "Monty Python". O uso da serpente como símbolo da linguagem se difundiu depois da publicação do ProgrammingPython da editora O'Reilly.

Em que tipo de aplicações eu usaria Python?

Python é uma linguagem de uso geral que pode ser empregada em vários tipos de problemas. A biblioteca padrão inclui módulos para processamento de texto e expressões regulares, protocolos de rede (HTTP, FTP, SMTP, POP, XML-RPC, IMAP), acesso aos serviços do sistema operacional, criptografia, interface gráfica etc. Além da biblioteca padrão, existe uma grande variade de extensões adicionais para todo tipo de aplicação.

Python é tipicamente usado em aplicações web e como linguagem de *scripting* para administração de sistemas. A facilidade de integração com C faz de Python uma linguagem embutida atrativa em aplicações de maior porte. A possibilidade de uso de componentes COM faz de Python uma alternativa mais agradável (e barata) ao Visual Basic. Finalmente, com o uso de ferramentas como o freeze ou Py2Exe é possível distribuir aplicações Python *stand-alone*, sem que o usuário tenha que instalar o interpretador Python separadamente.

Praticamente tudo o que se faria com qualquer linguagem de programação, seja ela interpretada ou compilada pode-se se fazer com python: protótipos de sistemas, automatizar tarefas repetitivas como manipulação de texto, cópia de arquivos e outros. Pode-se também criar programas que funcionam no modo texto, tanto interativos como servidores (ou daemons). Pode-se fazer programas em modo gráfico usando a interface nativa do seu sistema, ou então utilizando Tk, GTk, Qt, wxWidgets e tantas outras.

Em que tipo de aplicações eu não usaria Python?

Em teoria pode-se fazer qualquer coisa com a linguagem (é Turing completa ;), mas na prática, devido à recursos de CPU, implementação e uso de memória isso nem sempre é possível. Aplicações que exigem manipulações de baixo-nível são mais complicadas de se fazer (por exemplo, troca de contexto em um Sistema Operacional), rotinas relacionadas a controladores de dispositivos que exigem respostas muito rápidas são pouco adequados também de se fazer em Python, para isso é melhor utilizar uma linguagem compilada como C/C++ ou então reescrever partes críticas como módulos em C de um programa principal em Python.

O Python também não é muito bom quando se trata de threads. Se sua aplicação precisa de ter muitos deles ou precisa de certas funções como prioridades entre eles, provavelmente você não poderá contar com o Python em seu estado atual. Deve ser lembrado, porém, que existe suporte a threads e em muitos casos ele é suficiente (por exemplo, para delegar a um thread a GUI e a outro o trabalho duro).

Qual a licença usada e quais as restrições de uso?

Python é distribuída sob uma licença própria (compatível com a GPL), que impõe poucas restrições. É permitida a distribuição, comercial ou não, tanto da linguagem quanto de aplicações desenvolvidas nela, em formato binário ou código fonte, bastando cumprir a exigência de manter o aviso de *Copyright* da PSF (*Python Software Foundation*). Veja a licença em mais detalhes aqui: Licença do Python 2.x ou Licença do Python 3.x.

Como achar uma boa lista de sites de hospedagem gratuitos que suportam Python (e ZOPE) ?

Você pode encontrar uma lista em http://www.oinko.net/freepython/

Por que aprender Python?

Porque Python é uma linguagem simples e elegante. Porque Python é fácil de aprender. Porque Python pode ser usado para resolver uma grande variedade de problemas. Porque Python incentiva você a escrever seus programas da maneira correta, sem que isso se torne um empecilho à produtividade.

Python tem uma curva de aprendizado bastante interessante, permitindo que novos programadores, mesmo os que nunca tenham programado antes, sejam imediatamente produtivos escrevendo scripts procedurais. O programador pode executar o interpretador como um *shell*, vendo imediatamente o resultado da saída de cada comando e explorando os recursos da linguagem interativamente.

Para construir aplicações mais complexas, Python possibilita a fácil migração para a programação orientada a objetos. Um programa pode evoluir naturalmente para esse paradigma à medida que se torna mais complexo. A facilidade inicial do Python não barateia a linguagem, como é comum em linguagens que têm por objetivo expresso serem fáceis de aprender. Python é simples de aprender porque é uma linguagem bem planejada.

Ocultando o Código

Muitas pessoas querem saber como transformar o código python em binário, ocultando-o de leitores indesejáveis. Apesar de **não** existir um jeito perfeito de fazer isso em **nenhuma linguagem**, devido aos *disassemblers*, a página OcultandoCodigoPython fornece algumas informações.

Divulgando Python

Todos que trabalham com Python já devem estar acostumados com as mesmas perguntas sempre, começando com "Python? O que é isto?", acompanhadas de um nariz meio torto e uma testa franzida. Nossa intenção é criar uma lista de Perguntas Freqüentes, com o objetivo de mostrar que Python é uma linguagem que pode ser utilizada no ensino de lógica e programação em cursos de graduação. Apesar deste objetivo, acreditamos que as respostas podem ser úteis a todos os interessados em conhecer um pouco mais sobre Python. Esta seção de perguntas e respostas foi compilada peloMarcoAndréLopesMendes a partir de respostas de diversos participantes da lista de discussão python-brasil no Google Groups.

Por que eu deveria usar Python e não <insira aqui a sua linguagem favorita>?

Python e Perl são linguagens com propósitos bastante parecidos entretanto Python promove a facilidade de leitura em contraste ao modo "somente de escrita" que muitos programadores adotam em Perl. Um outro lema oposto ao Perl é que existe somente um jeito de se fazer uma coisa, em vez de se utilizar diversos dielatos que Perl permite.

Python e Java são linguagens bastante diferentes, o que torna a comparação direta difícil. Enquanto Python sugere um desenvolvimento rápido, do tipo "editar-executar" (Python compila automaticamente quando executamos o programa), Java exige que o programador declare tipos, visibilidade de funções, separe cada classe (pública) em arquivos diferentes, e o desenvolvimento é do tipo "editar-compilar-executar" (ainda que o arquivo gerado tenha que ser interpretado...). Outra vantagem do Python são suas estruturas nativas de dados, tais como listas e dicionários. Java usa classes de sua biblioteca padrão para as mesmas funcionalidades, com algumas complicações (leia-se *casts*) adicionais.

Python e C. C é de médio nível e assim como o *assembly*, expõe conceitos estruturais da arquitetura da máquina e complica a implementação de conceitos modernos com Orientação a Objetos.

Python e Pascal. Pascal é "linguagem de brinquedo" (não estou falando de Object Pascal), nos anos 80 era uma ótima linguagem para iniciar a programar mas hoje este papel pode ser cumprido com vantagens por Python.

Python e PHP. PHP é uma linguagem de programação especialmente direcionada para a programação de websites dinâmicos. Até existe o projeto PHP-GTK, ou seja, tornar o PHP também uma linguagem de programação com suporte à objetos gráficos, porém é complicado, trabalhoso e problemático. O Python é uma linguagem sem objetivo principal: pode ser tanto utilizada em programas modo texto, quanto suporte a objetos gráficos e páginas dinâmicas. Aprendendo Python, você terá a capacidade de montar programas para diversas plataformas e objetivos sem nem mesmo prescisar trocar de linguagem.

Python e Visual Basic/Delphi. Em termos de Win32, o Python não perde em nada para VB/Delphi pois oferece o acesso completo ao MFC e outras bibliotecas gráficas mais produtivas. Em contrapartida, oferece uma linguagem Orientada a Objetos DE VERDADE enquanto que essas outras duas apenas implementam parte dos conceitos da OOP. Outras vantagens importantes são o custo/benefício e o fato de ser multi-plataforma. Em termos de .Net, oIronPython oferece bom suporte, com a vantagem de suportar também o Mono. (MarinhoBrandão)

Uma linguagem interpretada não é lenta?

Sim e não. Uma linguagem interpretada em geral é mais lenta que uma linguagem compilada. Se a linguagem estará sendo utilizada por estudantes em iniciação, então os programadores deveriam ter uma preocupação maior com a corretude e a facilidade de programação do que com o desempenho do código, desenvolvendo programas simples, mais fáceis de entender, onde a velocidade não é a maior das considerações.

Além do mais, é possível fazer código otimizado muito mais rápido em Python do que em C, por exemplo. Além disso, atualmente estão se desenvolvendo tecnologias de compilação JIT (*just in time*) que reduzem drasticamente a diferença entre aplicações desenvolvidas em linguagens compiladas e linguagens interpretadas.

Para ver uma prova disto, basta uma visita rápida a: BenchmarkAdHoc

Se o programa a ser implementado for algum tipo de programa mastigador de grandes volumes de dados ou um sistema de tempo real para controle de hardware, talvez a lentidão seja perceptível. A maioria dos problemas não requer tanto processamento assim, computadores pessoais têm poder de CPU suficiente para rodar satisfatoriamente a maioria dos programas interpretados. Não estamos mais nos anos 80, com Pascal e máquinas de 8 bits para justificar o argumento de lentidão. Ainda mais com o advento do *bytecoding* e compiladores JIT, o fato de ser interpretado há muito tempo deixou de ser um

penalizador. É claro que é difícil chegar ao tempo de execução de uma aplicação feita em C, mas por outro lado, uma perda de velocidade significa um aumento no "tempo do programador", no fato dele escrever mais rápido e menos propenso ao erro.

Alguns dados sobre a produtividade do programador podem ser vistos neste estudo (PDF em inglês) realizado por Lutz Prechelt.

Programas escritos em Python podem ser mais lentos para executar em algumas situações, mas, em geral, levam menos tempo para implementá-los. Se o problema é realmente desempenho, deve-se implementar as partes críticas em linguagens compiladas ou até mesmo em *assembly*. A questão é quanto do código total possui estes requisitos.

Que empresas usam Python?

Quanto à utilização comercial de Python: Muitas empresas de grande porte vêm adotando Python em suas linhas de desenvolvimento. A mais importante delas é o Google. Pode-se citar ainda outras: Industrial Light and Magic, NASA, Thawte, Inktomi, IBM. No Brasil: Serpro, Haxent, Async, Conectiva, Embratel, GPr, Hiperlógica. Sem falar nas que utilizam Python indiretamente, através do Zope e do Plone.

Atualmente, a linguagem ocupa a sexta posição no ranking de linguagens utilizadas em projetos de Software Livre hospedados no SourceForge.

Uma pergunta importante a ser feita é que empresas não usam e por quê?

No site oficial do Python, pode-se encontrar uma lista de empresas que usam Python. Outra fonte é o Pythonology.

Que escolas/faculdades/universidades usam Python?

São ainda poucas, mas com resultados excelentes.

O IME e a POLI na USP, na UFSC, nos departamentos de Física e Matemática. Na Unicamp em cursos de extensão e na PUC-Campinas em cursos de graduação. Na Universidade Federal Rural de Pernambuco no curso de Licenciatura em Computação. O NAPI (Nucleo de Apoio a Projetos de Informática), da UCPel (Universidade Católica de Pelotas), utiliza Python em grande parte dos projetos de pesquisa realizados.

Na Suíça na Fachhochschule de Zurique em cursos de Pós-Graduação.

O caso mais famoso talvez seja o da Yorktown High School, Arlington, Virginia. O Prof. Jeffrey Elkner, desta escola, é um dos autores do livro *How to Think Like a Computer Scientist - Learning with Python*. No MIT (Massachusetts Institute of Technology), em Boston, Python é usado para introduzir programação a iniciantes na aula "6.00: Introduction to Computer Science and Programming".

A partir deste ano, estamos introduzindo Python na disciplina de Programação I do curso de Bacharelado em Sistemas de Informação do Instituto Superior Tupy em Joinville, Santa Catarina, substituindo o C++.

Para maiores informações sobre este tema, acesse a lista EDU-SIG.

Python é uma linguagem séria ou de brinquedo?

As duas coisas, dependendo do ponto de vista.

Python não é uma linguagem de brinquedo se isso significar que é impossível fazer aplicações de uso real com ela, como editores ou desktops. Aliás, muito pelo contrário. Diversas empresas utilizam Python de maneira 'séria' para ligar seus muitos componentes - veja o Pythonology novamente. Python é utilizada como linguagem de *scripting* em diversos programas sérios, como jogos, renderizadores (desses que fazem filmes de sucesso). Se Python não fosse séria - nesse sentido - a NASA e a OTAN não a usariam.

Por outro lado, Python é uma linguagem de brinquedo se isso significa que é possível se divertir com ela. Python segue as definições de "brinquedo" do projetista de jogos Greg Cotikyan, mas isso depende muito mais do estado de espírito de quem está o.

Python é uma linguagem séria, ótima para quem está iniciando e, melhor ainda, poderosa para quem já é programador. Algo difícil de se obter em uma linguagem, diga-se de passagem. É possível se desenvolver qualquer tipo de aplicação em Python. É uma linguagem que dá liberdade ao programador e faz com que ele tenha que se preocupar em resolver os seus problemas e não se preocupar em deixar o compilador "feliz".

Por outro lado, uma linguagem cujo nome vem de um seriado de comédia não deve ficar feliz em ser considerada uma linguagem "séria". Definitivamente ela é de brinquedo. Programa em Python é resgatar o prazer e a diversão do ato de criar e interagir.

Dá pra escrever programas sérios com Python?

Se você é um programador sério, sim.

O que existe de "programa sério" que utiliza Python não é brincadeira (perdão pelo trocadilho), o Google pode ser citado como o principal exemplo mais uma vez.

Dá para escrever programas "profissionais" com Python, sendo o Zope talvez o exemplo mais notório. Além dele podemos citar: skencil, txt2tags, ERP5, Plone, Schooltool, Chandler, Plucker, ...

Dá pra escrever programas gráficos com Python?

Sim, e as opções são tantas que podem até atrapalhar. Pode-se utilizar quase a totalidade dos widgetsets disponíveis para Linux em Python e em Windows é possível utilizar a MFC e as APIs de mais alto nível desta plataforma.

Pode-se utilizar: wxPython, PyGTk, PyQt, PythonCard, Tkinter, PyGlade, PMW, Kiwi, ncurses, OpenGL. Tem também: PIL, VPython.

Para uma comparação mais extensa entre as diversas opções para desenvolvimento de aplicações GUI, dêem uma olhada na ComparacaoDeGUIs.

Uma boa referência é o Python Eggs.

O Blender é um excelente exemplo de programa gráfico que utiliza python no desenvolvimento de seu código. Permite inclusive a facilidade de se inserir plug-ins através desta linguagem.

Ser fácil não é um problema?

Sim e não.

Sim, isto pode acontecer. Mas nada que não seja controlável. A facilidade pode gerar um péssimo hábito: o de sentar em um computador e começar a programar sem planejar ou sem de fato resolver o problema desejado antes. Mas acredito que todas as linguagens, mesmo as compiladas, hoje em dia, não estão livres desse problema.

Se você ensina um estudante a utilizar um IDE, ele terá um botão para "compilar e executar" automaticamente o programa. É praticamente a mesma coisa. O professor portanto deve trabalhar isso com os alunos, de tal maneira que esse hábito não se desenvolva.

Ser fácil pode também ser um problema para masoquistas que desejam usar Python ou não-masoquistas que insistem em não usar e ainda assim competir em tempo e qualidade com quem usa.

Não. Ser fácil quer dizer que você não vai perder tempo "traduzindo mentalmente" o que pensa do problema em sua solução em Python, ou seja, você não tem muito "atrito" com a linguagem. Isto é muito importante, principalmente para cursos introdutórios.

Os alunos não ficarão com os conceitos fracos ou mal acostumados?

Sim e não.

Sim. Os alunos tendem a ficar com alguns conceitos fracos. Isso, no entanto, não é culpa da linguagem. Os alunos que estudarem de fato vão aprender e reter os conceitos - desde que o professor não tente ensinar história da computação, programação estruturada, programação lógica e funcional, inteligência artificial e construção de compiladores em seu curso de 4 meses. Se seguir um currículo mínimo: introdução a computação e programação estruturada (*if-then-else*, laços, e conceitos básicos), os alunos que praticarem vão absorver tudo o que é ensinado.

Se os alunos só aprenderem Python, as chances são maiores. Universidade não tem esse nome à toa, não existe um único conceito que se for aprendido torna todos os demais obsoletos. Python é uma excelente ferramenta, ponto final.

Não. Python suporta os paradigmas estrutural, modular, funcional e Orientado a Objetos. São vários conceitos a serem explorados.

Os conceitos serão ensinados. Os algoritmos podem ser explicados normalmente em Python e, mesmo que Python já disponibilize algumas estruturas de dados como listas e dicionários, é possível fazer com que o aluno reimplemente-as. Tudo depende do foco que o professor quer dar ao seu curso.

Conceitos como alocação de memória, *heap*, pilha, listas ligadas, arvores binárias podem ser explicados numa etapa mais avançada dos cursos. No início é mais interessante explica para o aluno coisas como: implementar uma calculadora pós-fixada utilizando pilha, ordenar uma lista, fazer uma pesquisa binária nessa lista, percorrer uma árvore binária, utilizando as estruturas de dados nativas de Python.

Dá pra ensinar X usando Python? Onde X pode ser "classes", "herança", "ponteiros", "threads", "estruturas de dados", "ordenação", "recursão", "sockets", ...

Sim, com exceção talvez de ponteiros, o que tem um lado positivo. O conceito de *aliases* de Python e o método de chamada (por referência) esconde bem os ponteiros. Por outro lado, é possível simular o conceito de ponteiros usando dicionários, se for realmente necessário ensinar.

De maneira geral, as bibliotecas Python procuram não modificar muito o *modus operandi* da API C subjacente, e isso é *muito* bom. Por exemplo, quem aprende BSD/*Sockets* em Python vai saber lidar com *Sockets* em C, basta conhecer as idiossincrasias do C, como *casts* e estruturas.

Para ser justo, é possível programar sem ponteiros também em C++ (grande parte da má fama de C++ vem do seu mau uso), mas infelizmente isso não isentará o aluno de ter de aprender ponteiros, pois em alguns casos seu uso é mandatório.

Certas coisas não devem ser ensinadas utilizando Python, a não ser como linguagem de prototipagem. Por exemplo, ponteiros estão muito mais relacionados com programação de sistemas em baixo nível, e Python tem como foco um nível acima. No entanto, para o estudo de algoritmos, é difícil que exista coisa melhor.

Algumas estruturas de dados já consagradas são nativas em Python, mas nada impede que elas sejam reimplementadas. Além disso, existem outras estruturas de dados que não fazem parte da linguagem.

E o mercado de trabalho pros alunos?

Está faltando gente qualificada em Python no mercado. E quem programa em Python pode entender Zope e Plone e assim ganhar mais espaço no mercado.

No que se refere a conhecimentos, quanto mais melhor! Python não vai evitar que os alunos tenham que aprender outras linguagens ou ferramentas. Estar preparado para o mercado de trabalho é ser um curioso insaciável, ter prazer em resolver os problemas da profissão (porque trabalhar é resolver problemas) e agir com humildade e respeito com seus colegas. Seguindo estas regras, alguém tem que se esforçar muito para fracassar.

O mercado de trabalho funciona com uma regra simples: de demanda e oferta. As duas necessitam crescer juntas ou ocorre um desequilíbrio.

Atualmente a demanda para profissionais Python vem crescendo de forma tímida por um único motivo: falta de oferta.

Não deveríamos esperar até haver um mercado maior, para então começar a ensinar utilizando Python?

Depende do perfil da universidade. Existem universidades que preferem **treinar** o aluno para o mercado de trabalho atual. Essa é uma das características das universidades que possuem cursos de 2 anos.

Por outro lado existe um outro tipo de universidade, que prefere **preparar** melhor o aluno e torná-lo apto a se adaptar ao mercado de forma muito mais eficiente. A não muitos anos atrás, existiam universidades

que treinavam seus alunos para desenvolver software em Clipper e não em Pascal (por exemplo). A explicação para isso é a de que o mercado pedia Clipper. Quando o Delphi da Borland surgiu, quem se adaptou melhor? O aluno que aprendeu Clipper solicitado pelo mercado ou o aluno que aprendeu os fundamentos da programação em Pascal?

As universidades deveriam estar a vários passos adiante do mercado, pois a função delas é a de **preparar** os alunos para o futuro e não **treiná-los** para o mercado.

Que livros e outras bibliografias existem sobre Python?

Além da documentação oficial, existem vários livros:

- How to Think Like a Computer Scientist - Learning with Python
- Learning Python
- Python Programming on Win32
- Python in a Nutshell
- Core Python
- Python Cookbook
- Python How to Program
- Thinking in Python
- Python - Guia de Consulta Rápida, da Novatec (Em português)
- Python - Curso Completo

Documentacao existente na Internet:

Em Português:

- PensarCCPy - Tradução em andamento do *"How to Think Like a Computer Scientist - Learning with Python"*, para o português.

Uma boa relação está disponível aqui em DocumentacaoPython. Além disto, o OsvaldoSantana está escrevendo um livro sobre Python.

Qual o melhor livro para aprender a programar em Python?

1. Python Tutorial escrito pelo criador da linguagem, GuidoVanRossum e a tradução Tutorial de Python.
2. Livro Learning Python e a tradução Aprendendo Python.
3. Livro Dive into Python e a tradução Mergulhando no Python.
4. Livro How To Think Like a Computer Scientist - Learning with Python - primeira edição e a tradução em andamento Aprenda Computação com Python. Existem ainda a segunda edição em inglês e um novo livro feito a partir dele chamado How to Think Like a (Python) Programmer.

Mais informações na seção Aprenda Mais.

Como faço para executar uma aplicação gráfica (GUI) sem que apareça a janela do prompt de comandos do Windows?

Existem duas formas de resolver este problema:

- modifique a extensão do arquivo principal da sua aplicação de *.py* para *.pyw*
- execute a aplicação com o interpretador *pythonw.exe* e não com o *python.exe*.

Onde posso tirar minhas dúvidas?

- Documentação que vem junto com a linguagem
- O verdadeiro oráculo moderno: http://www.google.com
- python-list@python.org (inglês)
- python-brasil no Google Groups (português)
- http://www.pythonbrasil.com.br (português)

Este FAQ está disponível no site python.org.br na seção FAQ.