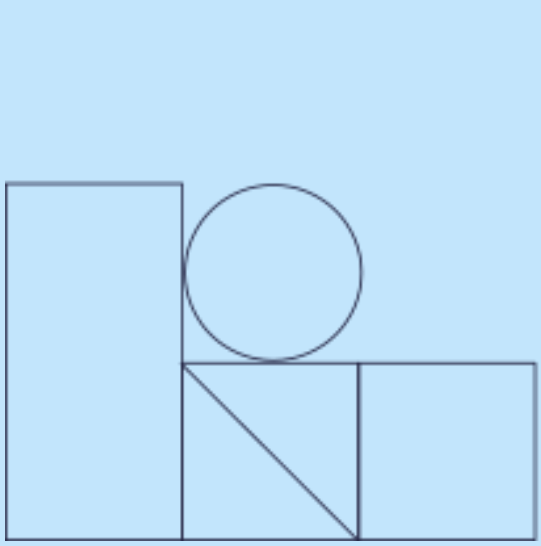
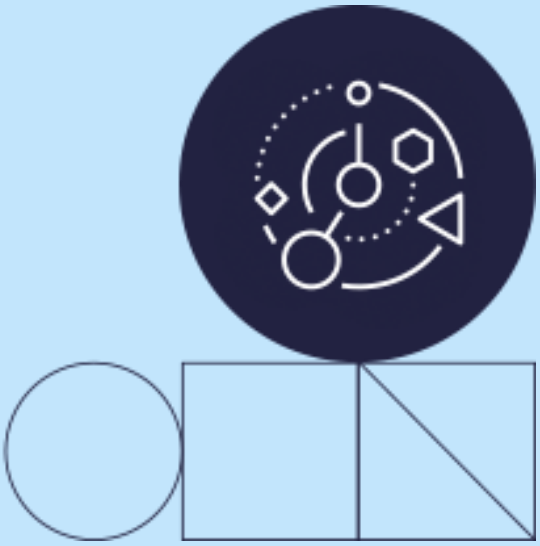
IBM SkillsBuild

IBM

Noções básicas e sintaxe de Python

Introdução à linguagem Python





Índice

Introdução

Python é freqüentemente a linguagem com a qual muitas pessoas começam no mundo da programação. É uma linguagem multiparadigma na qual você pode trabalhar com programação estruturada, programação orientada a objetos ou programação funcional e tem uma curva de aprendizado significativamente menor do que outras linguagens de programação de alto nível.

Python é uma linguagem de programação interpretada que se caracteriza por sua legibilidade e por promover software de qualidade. Se nos atermos à descrição fria, Python é uma linguagem multiparadigma (permite uma programação imperativa, orientada a objetos e, em menor medida, funcional), interpretada e dinamicamente tipada.

A Python foi criada em 1991 por Guido Van Rossum, que, até recentemente, continuava a liderar seu desenvolvimento sob o título de Benevolent Dictator For Life (BDFL).

O desenvolvimento da **Python** é coordenado sob o guarda-chuva da Python Software Foundation, que fornece recursos e uma estrutura organizacional. **Python** é Software Livre, sendo licenciado sob a Licença de Software Python.

Por que usar Python?

Com o número de linguagens de programação que existem hoje em dia, é fácil perguntar por que aprender **Python**? Que vantagens esta linguagem oferece? Abaixo está uma lista das principais vantagens que fizeram do **Python** uma linguagem amplamente utilizada.

Qualidade de Software

Python se concentra na legibilidade, consistência e qualidade do software. Isto permitiu que ela se destacasse de outras linguagens de programação. **Python** foi projetado para tornar os programas escritos nesta linguagem altamente legíveis e, portanto, mais fáceis de manter. A uniformidade do código **Python** torna um programa **Python** mais fácil de entender, mesmo por pessoas que não tenham escrito esse código. Tal é o impacto deste foco que outras linguagens de programação que surgiram recentemente também estão focando estes aspectos.

Aumentando a produtividade do programador

Os programas escritos em **Python** são tipicamente entre um terço e um quinto (às vezes até mais!) mais curtos que seus equivalentes escritos em C, C++ ou Java. Isto significa que os programas **Python** não só são mais rápidos de escrever, mas também geralmente mais robustos. Ter que escrever menos linhas de código significa que há menos código para “debugear” e menos para manter. Além disso, como **Python** é uma linguagem interpretada, os programas podem ser executados diretamente, sem os demorados processos de compilação e ligação necessários em outras línguas compiladas.

Portabilidade de código

A maioria dos softwares escritos em **Python** podem ser portados para os principais Sistemas Operacionais e plataformas com pouco ou nenhum esforço. A maioria dos softwares **Python** pode ser portada do Linux para o Windows, por exemplo, simplesmente copiando o código de uma máquina para outra.

Librería padrão

Python inclui uma grande coleção de bibliotecas, a biblioteca padrão, que cobre uma multiplicidade de funcionalidades: interfaces com o sistema operacional, manipulação de texto, conexão a bancos de dados, etc. É por isso que muitas vezes se diz que a **Python** vem com baterias incluídas (batteries included). Além da biblioteca padrão, há um enorme ecossistema de bibliotecas de terceiros disponíveis na web. Desde frameworks para robótica até bibliotecas de computação numérica que rivalizam (e em muitos casos superam) o Matlab, assim como framworks web, etc. Todos eles completamente livres e abertos ao usuário.

Gosto da programação

Este último é mais subjetivo, embora seja uma razão pela qual muitos programadores Python falam sobre ele. Devido à facilidade de programação e ao extenso ecossistema de bibliotecas disponíveis, muitos desenvolvedores dizem que a programação em **Python** se torna mais um prazer do que uma tarefa entediante. Alguns também falam da beleza do código **Python**, tanto por sua estrutura como por sua coerência. Além disso, alguns desenvolvedores que tiveram que voltar à programação em outras linguagens depois de terem usado **Python** por algum tempo expressaram frustração por terem que voltar a linguagens menos "bonitas".

Diferenças entre Python 2 e 3

Existem atualmente dois ramos principais de Python: 2.x, lançado no início de 2000, e 3.x, lançado pela primeira vez em 2008. À primeira vista, Python 3 pode parecer semelhante a Python 2, mas há numerosas diferenças profundas, incluindo o suporte extensivo Unicode em Python 3, renomeado módulos, mudanças nas importações, mudanças nas divisões, e muito mais. Antes de começarmos a aprender, há algumas coisas importantes a serem levadas em conta.

Para começar, devemos nos acostumar com a idéia de que Python 2.x e Python 3.x são linguagens completamente diferentes. Na realidade, eles são o mesmo Python, mas as diferenças são tantas que é difícil imaginá-los como a mesma linguagem. Um script compatível com Python 2.x pode não funcionar em Python 3.x e vice-versa. É melhor se acostumar à idéia de que eles são idiomas diferentes.

Antes de mais nada, por pura lógica entenderemos que Python 3 é mais moderno e atualizado que Python 2, e como um programador deve estar sempre trabalhando com as últimas versões das línguas, é aconselhável aprender Python 3. É mais otimizado, mais difundido e, é claro, é o futuro da língua.

Python 2 não é mais suportado a partir de (2020), o que significa que Python 2.7 foi a última versão de Python 2.x, nunca haverá uma Python 2.8, se vamos aprender Python, não faz sentido investir tempo em Python 2.

A única razão aceitável para aprender Python 2, é fazê-lo com a intenção de trabalhar na manutenção de aplicações já criadas em Python 2. O lógico é pensar que as empresas que uma vez apostaram em Python 2.x fariam uma migração, elas tiveram muito tempo. Mas a verdade é que, no mundo real, as empresas sempre têm códigos antigos que precisam manter e nem sempre são atualizados.

Agora, se uma empresa está pensando em fazer um novo desenvolvimento com Python 2.x, a verdade é que, como programadores responsáveis, devemos estar cientes de tudo o que está envolvido. As fraquezas do software, sem atualizações e sem suporte linguístico, podem se tornar um grande problema a longo prazo.

Mesmo assim, há maneiras de migrar e até mesmo manter ambas as versões no mesmo projeto:

Python 3 propõe implementar lançamentos mais rápidos, melhorias de desempenho, novas funcionalidades no uso de cordas, novos operadores de join e APIs internos mais consistentes e estáveis. Quais são as diferenças entre Python 2 e Python 3?

Suporte Unicode melhorado

Em Python 2, as cordas são armazenadas como ASCII por padrão e tivemos que adicionar um "u" se quiséssemos armazená-las como cordas Unicode. Em Python 3, isto não é mais necessário, pois as cordas são armazenadas como Unicode por padrão.

Isto é muito importante, pois o Unicode é muito mais versátil do que o ASCII. Cordas Unicode podem armazenar letras em diferentes idiomas, numerais romanos, símbolos, emojis, etc., dando-nos muito mais opções.

Melhoria da divisão inteira

Em Python 2, se você digitar um número sem nenhum dígito após o ponto decimal, seu cálculo é arredondado para o número inteiro mais próximo. Por exemplo, se tentarmos dividir 5 por 2 (5/2), o resultado será 2 devido aos arredondamentos.

Teríamos que escrevê-lo como 5.0 / 2.0 para obter a resposta exata de 2.5. Entretanto, em Python 3, a expressão 5/2 retornará o resultado esperado de 2,5

sem ter que se preocupar em adicionar esses zeros extras.

Sem dúvida, este pequeno exemplo simples demonstra como a sintaxe do Python 3 acaba sendo muito mais intuitiva, algo que dará muito mais conforto aos novatos que tentam aprender a linguagem de programação Python.

Diferenças sintáticas em certos comandos

Podemos ver esta diferença como trivial, e certamente é, pois não afeta a funcionalidade de Python, mas é uma diferença muito perceptível na sintaxe da declaração print que não deve ser ignorada.

Em Python 3, a declaração de print foi substituída por uma função de print(). Por exemplo, em Python 2 você escreve print "hola", mas em Python 3 você escreve print("hola"). Como iniciantes que procuram começar a aprender a linguagem de programação Python, esta diferença não deve nos preocupar, pois não vai nos afetar muito, mas ainda assim é um ponto a ser levado em conta.

Da mesma forma, a função raw_input() é renomeada para input().