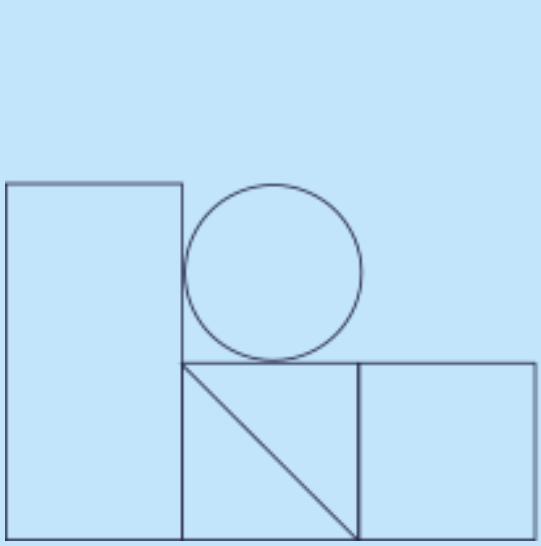
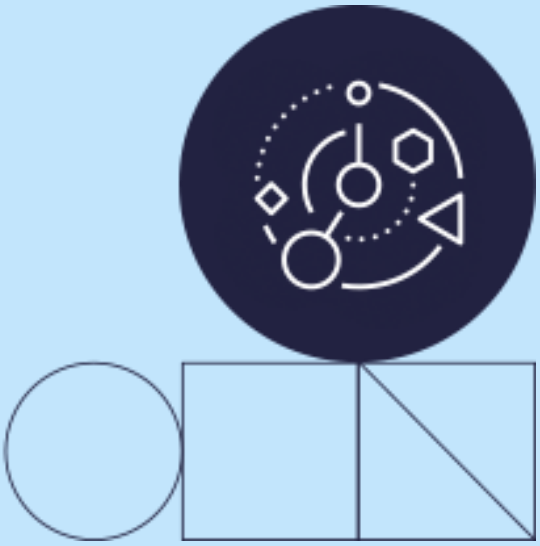
IBM SkillsBuild

IBM

Noções básicas e sintaxe de Python

Sintaxe Python





Índice

Introdução

Sintaxe **Python**, vendo como podemos começar a usar a linguagem, criando nossas primeiras variáveis e estruturas de controle.

O termo sintaxe se refere ao conjunto de regras que definem como o código deve ser escrito em uma determinada linguagem de programação. Ou seja, refere-se à forma como devemos escrever as instruções para que o computador, ou melhor, a linguagem de programação, nos entenda.

Na maioria das línguas há uma sintaxe comum, como o uso de = para atribuir dados a uma variável, ou o uso de {} para designar blocos de código, mas **Python** tem certas particularidades.

A sintaxe é programar o que a gramática é para as linguagens. Da mesma forma que a frase "eu estamos aqui" não é correta, o seguinte código em Python não seria correto, pois não respeita as regras da linguagem.

```
if ($variable){
    x=9;
}
```

Veremos isso em detalhes abaixo, mas **Python** não suporta o uso de $ nem precisa terminar linhas com ; como em outras línguas, nem precisa usar {} em estruturas de controle como if.

Por outro lado, da mesma forma que uma linguagem não é falada pelo simples conhecimento de todas as suas palavras, na programação não é suficiente conhecer a sintaxe de uma linguagem para programar corretamente nela. É verdade que, conhecendo a sintaxe, podemos começar a programar e fazer o que quisermos, mas o uso de uma linguagem de programação vai muito mais além da sintaxe.

Cordas, operadores aritméticos e o uso do if condicional

Para começar a perder o medo da sintaxe **Python**, vamos ver um exemplo onde vemos cordas, operadores aritméticos e o uso do if condicional.

O seguinte código simplesmente define três valores a, b e c, realiza algumas operações com eles e exibe o resultado na tela.

```
# Definimos una variable x con una cadena
x = "El valor de (a+b)*c es"

# Podemos realizar múltiples asignaciones
a, b, c = 4, 3, 2

# Realizamos unas operaciones con a,b,c
d = (a + b) * c

# Definimos una variable booleana
imprimir = True

# Si imprimir, print()
if imprimir:
    print(x, d)

# Salida: El valor de (a+b)*c es 14
```

Como você pode ver, a sintaxe do **Python** é muito semelhante à linguagem natural ou pseudo-código, o que torna a leitura relativamente fácil. Outra vantagem é que não precisamos de mais nada, o código acima pode ser executado como está. Se você está familiarizado com outras linguagens como **C** ou **Java**, você achará isto conveniente, já que não precisa criar a função habitual main().

Comentários

Os comentários são blocos de texto usados para comentar o código. Ou seja, fornecer a outros programadores ou a nossos futuros eus informações relevantes sobre o código que está sendo escrito. Para fins práticos, para **Python** é como se eles não existissem, já que não são códigos em si, apenas anotações.

Os comentários começam com # e qualquer coisa depois disso na mesma linha é considerada um comentário.

```
# Este é um comentário
```

Como em outras linguagens de programação, também podemos comentar várias linhas de código. Para isso, é necessário utilizar aspas triplas, sejam elas simples ''' ou duplas """. É necessário utilizá-los para abrir o bloco de comentários e fechá-lo.

```
'''
Este é um comentario
de várias linhas
de código'''
```

É interessante que, como programadores, temos o hábito de comentar nosso código o máximo possível. A maioria dos projetos de certo tamanho é feita em equipe, e deixar nosso código comentado facilita para o colega de equipe que vem atrás de nós. No caso de programarmos sozinhos, comentar o código também é uma excelente idéia, pois é provável que dentro de algum tempo (meses ou anos) teremos que rever ou escalar nosso código, e se o deixarmos bem comentado poderemos retomar o trabalho onde o deixamos muito mais facilmente.

Blocos de código e identificação

Na maioria das linguagens de programação, elementos contendo código (tais como procedimentos, funções, condicionadores, loops... etc.) fazem uso de parênteses, colchetes e suportes para incluir esse código. **Python** não é assim. Em **Python**, blocos de código são representados com identação, ou seja, colocando o código dentro de um dos elementos acima, vários espaços à direita, e embora haja algum debate sobre o uso de abas ou espaços, a regra geral é usar quatro espaços.

No seguinte código, temos um código condicional if. Logo em seguida, temos um print() ident com quatro espaços. Portanto, tudo o que tiver essa identação pertencerá ao bloco do if.

Em Python:

```
if True:
    print("True")
```

Em outras linguagens de programação, a sintaxe seria algo como:

```
if (True)
{print("True")};
```

Isto é muito importante, pois o código acima e o código seguinte não são o mesmo. Na verdade, o seguinte código daria um erro como o if não contém blocos de código, e isso é algo que você não pode fazer em **Python**.

```
if True:
print("True")
```

Por outro lado, ao contrário de outras linguagens de programação, não é necessário utilizar ; para terminar cada linha.

```
# Otros lenguajes como C
# requieren de ; al final de cada línea
x = 10;
```

Entretanto, em **Python** isto não é necessário, uma quebra de linha é suficient.

```
x = 5
y = 10
```

Mas você pode usar o ponto-e-vírgula ; para ter duas frases na mesma linha.

```
x = 5; y = 10
```

Múltiplas linhas

Em algumas situações pode ser o caso de desejarmos ter uma única instrução em várias linhas de código. Uma das principais razões para isso poderia ser que é muito longo e, de fato, a especificação **PEP8** recomenda que as linhas não devem exceder 79 caracteres.

Ao fazer uso de \ é possível dividir o código em várias linhas, o que em certos casos torna o código muito mais legível.

```
x = 1 + 2 + 3 + 4 +\
    5 + 6 + 7 + 8
```

Se, por outro lado, estamos dentro de um bloco cercado de parênteses (), bastaria saltar para a seguinte linha.

```
x = (1 + 2 + 3 + 4 +
     5 + 6 + 7 + 8)
```

O mesmo pode ser feito para chamadas a funções:

```
def funcion(a, b, c):
    return a+b+c

d = funcion(10,
23,
3)
```