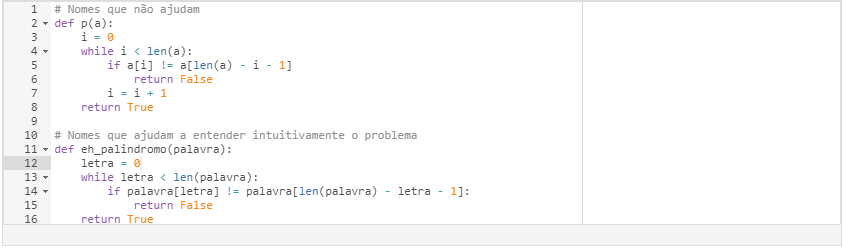
Assim como nas linguagens naturais, além de formas **corretas** de expressar um programa, existem também jeitos **bons**de se expressar (a frase "é, por exemplo, possível, em geral, expressar-se, sempre que necessário, desta forma" é um exemplo de frase **correta** mas não **boa** em português, pois sua leitura é difícil). Nosso curso tem lidado até agora com a escrita correta do código, ou seja, o código deve poder ser lido e executado sem erros pelo computador e produzir o resultado que queremos. Mas é importante saber também como programar de maneira a **comunicar** mais facilmente os objetivos do programa e como ele funciona.

Existem algumas normas gerais de estilo que servem para qualquer linguagem de programação (da mesma forma que nas linguagens naturais). No entanto, assim como no sudeste do Brasil se usa artigo antes de nomes próprios ("O Alberto gosta de maçã") e no nordeste não ("Alberto gosta de maçã"), existem também normas de estilo que dependem da tradição da comunidade. No caso da computação, as comunidades em geral correspondem aos usuários de uma linguagem de programação específica, e espera-se que um programa seja escrito de maneira **idiomática** em cada linguagem. O que se busca com isso é produzir código que seja mais facilmente legível para você e para outros programadores que eventualmente podem vir a consultá-lo, tornando-o mais simples de entender, corrigir, expandir e manter.

No caso da linguagem python, esse estilo é às vezes chamado de **pythônico**, e a comunidade produziu a [PEP8](https://www.python.org/dev/peps/pep-0008/), um guia de "dicas de estilo" para programadores Python. Compilamos aqui algumas dessas dicas. Diferentemente do resto do curso, elas são menos rígidas, já que seu programa funciona mesmo sem elas, mas ainda são interessantes de se levar em consideração.

Nomes São Importantes!

A melhor forma de tornar seu código legível para você mesmo e para outros é pensar bem nos nomes de variáveis e funções. Até os anos 1970, o uso de nomes longos tinha impacto negativo sobre o desempenho e consumo de memória de um programa, o que gerou um estilo excessivamente conciso na época. Por várias razões, esse estilo hoje é considerado nocivo (além de desnecessário), mas alguns programadores ainda são influenciados por ele:



## Indentação: Quatro Espaços

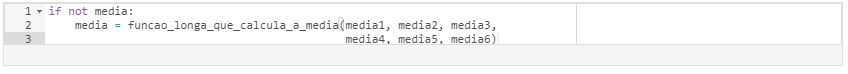
Para indentar, prefira o uso de espaços em vez de tabs. Utilize tabs somente para manter a consistência com programas antigos (note que em Python misturar espaços e tabs é proibido, e causa erros).

## Não usar linhas muito grandes

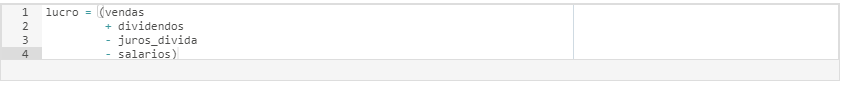
A PEP8 recomenda utilizar linhas de até 79 caracteres. Linhas maiores podem não ser legíveis em todos os dispositivos. Isso significa que, em alguns casos, precisamos quebrar linhas.

### **Como quebrar linhas?**

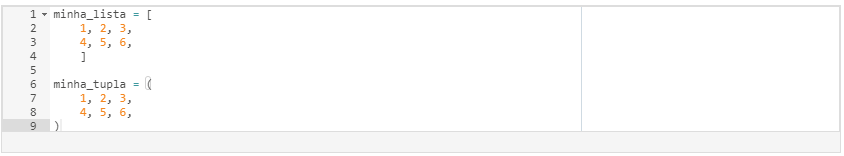
Alinhe a nova linha com o nível da parte que foi quebrada:



Para operações matemáticas, divida as linhas antes dos operadores matemáticos, não depois (em algumas linguagens, a convenção é invertida, mas há boas razões para o estilo pythônico):

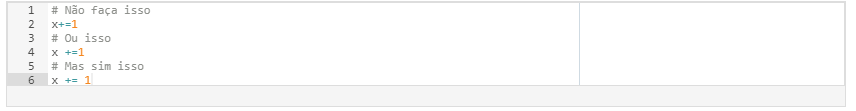


Para fechar parênteses ou chaves, a nova linha pode estar alinhada tanto com a linha anterior como com a primeira linha:

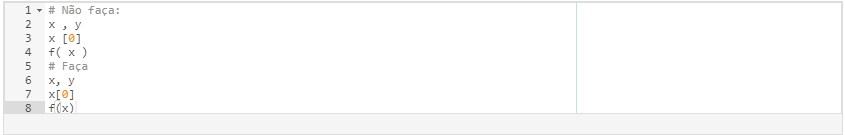


## Espaços entre operadores

Para aumentar a legibilidade, sempre coloque espaços entre operadores matemáticos de atribuição (=, +=, -=) , comparação (>, <, >=, <=, in, not in, is, is not, entre outros) ou booleanos (and, not, or):



Não use espaços depois de parênteses, chaves, ou antes de vírgulas e dois-pontos:



Em caso de dúvida, siga o que você faria se estivesse escrevendo em linguagem natural.

## Comentários

Comentários são encorajados, mas só se não forem redundantes (ou pior, contraditórios) com o código! Dê preferência a comentários em linhas separadas (não no final da linha) e os escreva como sentenças normais, começando com maiúscula e com um espaço depois do símbolo "#". Comentários no final da linha devem ser precedidos de dois espaços:

