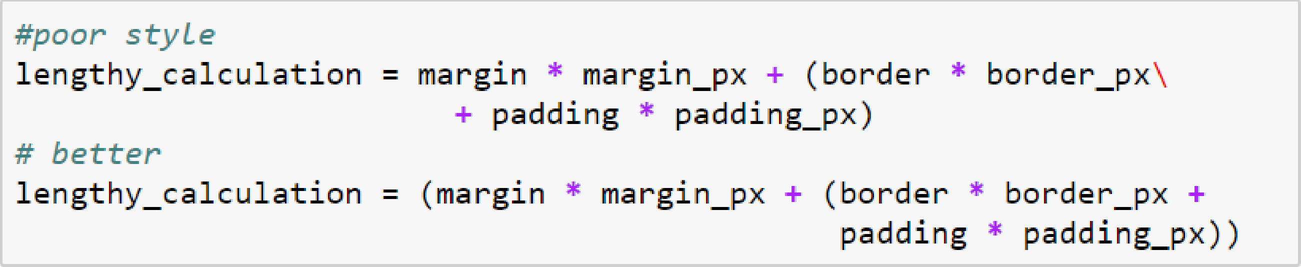
# Python Coding Style

- The officially recommended coding conventions for Python are provided by a document known as PEP8 (http://www.python.org/dev/peps/pep-0008/)
- Use *four spaces* per indentation level (and never tabs)
- In assignments, put spaces around the = sign
  - For example, a = 10, not a=10
- Separate operators from their operands with single spaces unless operations with different priorities are being combined
  - For example, write x = x + 5, but r2 = x**2 + y**2
- Don’t use spaces around the = in keyword arguments
  - For example, in function calls use foo(b=4.5) not foo(b = 4.5)
- Separate top-level function and class definitions by two blank lines; within a class, separate them by one blank line

# Python Coding Style

- Use a maximum of 80 characters per line, where you need to split a line of code over more than one line:
  - Favor implicit line continuation inside parentheses over the explicit use of the character \
  - In arithmetic expressions, break around binary operators so that the new line is *after* the operator
  - As far as possible, line up code so that expressions within parentheses line up, e.g.:



```
#poor style
lengthy_calculation = margin * margin_px + (border * border_px\
                        + padding * padding_px)
# better
lengthy_calculation = (margin * margin_px + (border * border_px +
                                             padding * padding_px))
```

- Avoid putting more than one statement on the same line separated by semicolons
  - For example, instead of a = 1; b = 2, write a, b = 1, 2
- Avoid wildcard imports (from foo import *)

# Python Coding Style

- Functions, modules and packages should have short, all-lowercase names
- Use underscores in function and module names if necessary, but avoid them in package names
- Class names should be in CamelCase, also known as CapWords
  - For example, `AminoAcid`, not `amino_acid`
- Define constants in all-capitals with underscores separating words
  - For example, `MAX_LINE_LENGTH`

# Python Coding Style

- Imports are always put at the top of the file, just after any module comments and docstrings, and before module globals and constants
- Imports should be grouped in the following order:
  - Standard library imports
  - Related third-party imports
  - Local application/library-specific imports
- You should put a blank line between each group of imports