Python Class Style Guide

One big advantage of Python is its readability. When you write your code, always think of it as literature or art, something to be admired by a reader.

If you don't understand a style listed below, we have probably not yet learned that part of Python.

1. You may break the style guide in certain circumstances – but you must have a good reason for everything you do that does not follow the style guide. Document that reason!

2. Do what is expected. Don't give your reader a reason to think about anything but the intention of the code. No distractions; nothing unnecessary.

3. Be tasteful about documentation:

   - Your code should not include documentation about language features, or about anything obvious.
   - Do provide complete documentation about your modules, functions, and classes in doc strings.
   - Use the `#` when you want to document details for the reader of the code.
   - There should be no external documentation, i.e., no `README` files.

4. Indentations are 4 spaces. Don't create deep nests of indents. Instead, improve your architecture. Maybe make more functions.

5. No duplicate code (after you have learned to make functions).

6. All functionality should be in functions (after you have learned to make functions).

7. Labels are meaningful, long if necessary, and self-documenting. The case of your labels means a lot to the reader:

   - `variable_labels` are lower case with an underline to separate words.
   - `CONSTANT_LABELS` are all upper case.
   - `FunctionLabels` have every word capitalized, including the first. They start with verbs or are verbs.
   - `ClassName` labels have every word capitalized, including the first. They are nouns.
   - `module_labels.py` are lower case with an underline to separate words.

8. A comma has a space after it, and no spaces before it.

9. Usually the assignment operator has one space on each side:

   ```
   bacon_strips = 3
   ```

   But:

   - In function definitions, there are no spaces around the = in defaulted arguments:

     ```
     def RateIt(name, rating=100):
       pass
     ```

   - In function calls, when providing keyword arguments, there are no spaces around the =:

     ```
     RateIt("Alice's", rating=99)
     ```

10. Parentheses have no spaces around them. Also:

    - No extra parentheses, unless they help readability.
    - Conditionals don't have parentheses:

      ```
      if chocolate_candy == "dark":
          pass
      ```