The Zen of Python, by Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than \*right\* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

**Python Enhancement Proposals (PEP)**are suggestions for improvements to the language, made by experienced Python developers.  
**PEP 8** is a style guide on the subject of writing readable code. It contains a number of guidelines in reference to variable names, which are summarized here:  
- modules should have short, all-lowercase names;  
- class names should be in the CapWords style;  
- most variables and function names should be lowercase\_with\_underscores;  
- constants (variables that never change value) should be CAPS\_WITH\_UNDERSCORES;  
- names that would clash with Python keywords (such as 'class' or 'if') should have a trailing underscore.  
  
**PEP 8** also recommends using spaces around operators and after commas to increase readability.

Other **PEP 8** suggestions include the following:  
- lines shouldn't be longer than 80 characters;  
- 'from module import \*' should be avoided;  
- there should only be one statement per line.  
  
It also suggests that you use spaces, rather than tabs, to indent. However, to some extent, this is a matter of personal preference. If you use spaces, only use 4 per line. It's more important to choose one and stick to it.  
  
The most important advice in the PEP is to ignore it when it makes sense to do so. Don't bother with following PEP suggestions when it would cause your code to be less readable; inconsistent with the surrounding code; or not backwards compatible.  
However, by and large, following PEP 8 will greatly enhance the quality of your code.