PEP 8

Introduction :-

This document gives coding conventions for the Python code comprising the standard libary in the main Python distribution. Please see the companion informational PEP describing style guidelines for the C code in the C implementation of Python.
This document and PEP 257 (Docstring conventions) were adapted from Guido's ~~orignal~~ original Python style guide essay, with some additions from Barry's style guide.
This style guide evolves over time as additional conventions are identified and past conventions are rendered obsolete by changes in the language itself.
Many projects have their own coding style guidelines. In the event of any conflicts, such project-specific guides take precedence for that project.

A Foolish Consistency is the Hobgoblin of little Minds :-

One of Guido's key insights is that code is read much more often than it is written. The guidelines provided here are ~~intended~~ intended to improve the read-ability of code and make it consistent across the wide spectrum of Python code. As PEP 20 says, "Readability counts".
A style guide is about consistency with the style guide is important. Consistency within a Project is more important. Consistency within one module or function is the most important.

However, Know when to be inconsistent - sometimes Style guide recommendation just aren't applicable. when in doubt, use your best judgement. Look at other example and decide what looks best. And don't hesitate to ask!

In Particular: do not break backward compatibility just to comply with this PEP!

Some other good reasons to ignore a particular guideline:

1) When applying the guideline would make the code less readable, even for someone who is used to reading code that follows this PEP.

2) To be consistent with surrounding code that also break it (maybe for historic reasons) -- although this is also an opportunity to clean up someone else's mess (in true XP style).

3) Because the code in question predates the introduction of the guideline and there is no other reason to be modifying that code.

4) When the code needs to remains compatible with older version of Python that don't support the feature recommended by the style guide.

CODE Lay-out

Indentation :-

Use 4 spaces per indentation level.

Continuation lines should align wrapped elements either vertically using Python's implict line joining inside Parentheses, brackets and braces or using a hanging indent. When using a hanging indent the following should be considered; there should be no arguments on the first line and further indentation should be used to clearly distinguish itself as a continuation line:

Correct:

Aligned with opening delimiter.

```
foo = long_function_name(var_one, var_two, var_three, var_four)
```

Add 4 spaces (an extra level of indentation) to distinguish argument from the rest.

```
def long_function_name(
        var_one, var_two, var_three,
        var_four):
Print(var_one)
```

Hanging indents should add a level.

```
foo = long_function_name(
    var_one, var_two
    var_three, var_four)
```

wrong:

Arguments on first line forbidden when not using vertical alignment.

```
foo = long_function_name(var_one, var_two, var_three, var_four)
```

further indentation required as indentation is not distinguishable.

```
def long_function_name( var_one, var_two, var_three,
                        var_four):
    Print (var_one)
```

The 4-space rule is optional for continuation lines.

optional:

Hanging indents *may* be indented to other 4 spaces.

```
foo = long_function_name(
    var_one, var_two,
    var_three, var_four)
```

when the conditional part of an if-statement is long enough to require that it be written across multiple line, it's worth noting that the combination of a two character keyword (ie if) Plus a single space, plus an opening parenthesis creates a natural 4-space indent for the subsequent lines of the multiline conditional

This can produce a visual conflict with the indented suite of code nested inside the if statement, which would also naturally be indented to 4 spaces. This PEP takes no explicit position on how (or whether) to further visually distinguish such conditional lines from the nested suite inside the if-statement. Acceptable options in this situation include, but are not limited to:

No extra indentation.

```
if (this_is_one_thing and
    that_is_another_thing):
    do_something()
```