Python Style Conventions for BU PyCon

Adapted from: **https://www.python.org/dev/peps/pep-0008/**

Indentation

1. When indenting, use **only spaces, not tabs**. Some text editors allow soft tabs, which are ok. Soft tabs are essentially using the “tab” button to indent 4 or 5 spaces.
2. Each level of indentation should be **4 spaces**.
3. Indentations from multiple lines of code should line up to matching parenthesis, e.g.:

```
x = function(arg1, arg2, arg3,
             arg4, arg5)
```

4. Rule 3 has an exception of an extra level of indentation if the code is followed by an indented level, for example in a loop or if statement. E.g.,:

```
if (a > b and color == ‘red’ or
    height == 3.0 and c != 0):
  do something()
```

Line Length

1. Limit all lines of code to a maximum of 80 characters.
2. Limit all lines of comments/docstrings to 72 characters.
3. Wrapping of multiple lines should be done by wrapping long expressions in parentheses.

4. Backslashes may be used for line continuation when you think parenthetical wrapping is adding too many parentheses in a given statement, as in *with* or *assert* statements.

Blank Lines

1. Top-level functions and class definitions should be separated with two blank lines.
2. Class methods and groups of related functions should be separated with one blank line.

Importing

1. Modules should be imported one at a time, i.e., on separate lines.
2. You can import multiple functions from the same module on one line.
3. Imports are declared at the top of the file, before globals and constants.
4. **Do not wildcard import any modules**, e.g.:

```
from numpy import *
```

Strings

1. Use single quotation marks for strings. Consistency is nice!

Comments

1. Write your comments in English. Not everyone is a polyglot.
2. Comments should be **complete sentences**, with the first word in a sentence capitalized (unless the first word is an identifier).
3. Short comments can have the period omitted from the end of the comment.

4. Block comments should have a single # followed by a single space. If the block comment applies to indented code that follows it, the comments themselves should also be indented the same amount for clarity.
5. Inline comments should be at least 2 spaces away from the end of the statement.
6. For docstring conventions, go here: https://www.python.org/dev/peps/pep-0257/

Naming Conventions

There are not many naming conventions easy to adopt, so here are some guidelines to loosely help us:

1. Use **descriptive** names. This may seem obvious, but I see sloppy code all the time. When using a *for* loop, as an example, “for i in j:” is terrible. A better way might be “for job in jobList:” or “for num in countArray:”. Same goes for variables.
2. The cases we use to name variables can vary — I prefer the mixed case convention when it’s a compound-word-variable. Some examples include dataList, numArray, objArr (a little vaguer but saves space when obvious), and dummyVar.
3. When your name might clash with a Python keyword, use a single trailing underscore. E.g., list_
4. If you want to define a function with the “weak internal use” indicator, use a single leading underscore. This keeps it from being wildcard imported, but since we won’t be doing that, I wouldn’t worry too much.
5. **Never invent new functions or variables with the double leading and double trailing underscores** (e.g., __init__). Only use them for the specific documented uses, like using __init__ for class instance initialization.

6. Avoid names with one character, especially ones that look similar to other characters.
7. Modules should have short, lower-case names. Underscores are ok to help with readability.
8. Class names should use the CapWord convention. Examples include MyClass(), LargeDict(), and SimulatorClass().
9. Exception names should follow the CapWord convention as well (since they should be classes), but they should also end in “Error”.
10. Functions should be all lower case with underscores to help readability, or it can use mixedCase if that is already the prevailing convention for backwards compatibility.
11. Define constants with all caps, with underscores separating words. Channel your inner senior citizen for these!

Final Thoughts

Just remember — code should always be as readable as possible! If something isn’t listed here or in the full linked style guide, use your judgment as to what is the most readable way. For example, I like to line up my equal signs if they are close together, but not if they are far apart. Do what you think is easiest to read and understand.

For further recommendations than these basic guidelines, see https://www.python.org/dev/peps/pep-0008/ and the links therein.