Sun*

Sun*

Convention for Intern & Fresher

Naming conventions

Module Names:

- **Short, lowercase names, without underscores**

  Example: myfile.py

Class Names:

- **CapWords convention.**

  Example: MyClass

Exception Names:

- **If a module defines a single exception raised for all sorts conditions, it is generally called “Error”.**

  Otherwise use CapWords convention (i.e. MyError.)

Naming Conventions

Method Names and Instance Variables

- **The “Style Guide for Python Code” recommends using lowercase with words separated by underscores (examples: my_variable). But since most of our code uses mixedCase, I recommend using this style (example: myVariable)**
- **Use one leading underscore only for internal method and instance variables (i.e protected) Example: _myProtectedVar**
- **Use two leading underscores to denote class-private names Example: __myProtectedVar**
- **Don’t use leading or trailing underscores for public attributes unless they conflict with reserved words, in which case, a single trailing underscore is preferrable (example: class_)**

Word of Caution

Don’t REINVENT THE WHEEL!

If the style causes the code less readable, don’t use it! (You might have an overall, bigger problem.)

Be consistent with older code

Older versions that were used before the inception of the document shouldn’t have to follow this. (But, why are you using old Python?)

Organizing Imports

- They should be always put at the top of the file. just after any module comments and docstrings, and before module globals and constants
- Imports should me on separate lines

  Wrong: import sys, os

  Right: import sys

  import os

  The following is OK, though:

  from types import StringType, ListType

Organizing Imports

- **Imports should be grouped in the following order with a blank line between each group of imports:**

  + standard library imports

  + related major package imports

  + application specific imports

Indentation

Indentation should always be four (4) spaces long.

For long lines (>79 characters):

- **Functional arguments can always be represented by aligning the first letter of subsequent lines with the first character of the original line**
- **Function calls can use single indents for subsequent lines.**
- **Function definitions should use double indents for subsequent lines**

```
if (tag == "span".encode().decode("iso-8859-1") and
        attrs[0][0] == "class".encode().decode("iso-8859-1") and
        attrs[0][1] == "ul".encode().decode("iso-8859-1")):
    self._record_name = True
```

Tabs or Spaces

- **Python 2.x code should always use spaces**
- **Python 3.x code can use spaces or tabs.**

  + To that end, make sure that you are consistent with your tabs and spaces!
- **General recommendation is to use spaces**

Line Length

- **Maximum of 72 characters (never exceed 79 characters)**
- **You can break a long line using “\” .**

Docstring and Type hint

-

Extension VSCODE

Indentation should always be four (4) spaces long.

For long lines (>79 characters):

- **Functional arguments can always be represented by aligning the first letter of subsequent lines with the first character of the original line**
- **Function calls can use single indents for subsequent lines.**
- **Function definitions should use double indents for subsequent lines**