Idiomatic Python

Mustafa Khattab
https://github.com/mkhattab
April 15, 2014

Why is it important?

- Readability
- Consistency
- In some cases, performance
- Most importantly, people will like you

Ingredients

- Read PEP8
- Command line linting tool (e.g. pylint, pyflakes, pep8, etc.)
- A decent editor (like emacs, not vim :)

import this

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!

Whitespace

- 4 spaces per indentation. Please don’t use 2 or 8 or whatever.
- Don’t use tabs, period.
- One (1) blank line between after functions/methods.
- Two (2) blank lines between classes.
- Try to limit line width to 79 characters.

Imports

- Wild card inputs should be avoided
  (e.g. from os import *)
- Each module import should be in a separate line.
- Unless, importing names from the same module
  (using from <mod> import foo, bar)
- Imports should be grouped in the following order:
  standard library, third party, local application specific.

Naming conventions

- Variable, method, function naming styles:
  - lowercase
  - lower_case_with_underscores
  - UPPERCASE (globals, constants)
  - UPPERCASE_WITH_UNDERSCORES
- Classes
  - FooBar
- Special Conventions
  - _class_attribute -- indicates an internal method or variable
  - __class_attribute -- does name mangling to prevent inadvertent access
  - __foo__ -- “magic” methods add special powers

Docstrings

- See PEP 257

- You *should* use docstrings on public methods and interfaces.

Naming conventions

References

- PEP 8
- Code Like a Pythonista: Idiomatic Python
- Read source code: Flask, Fabric, etc.