

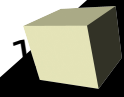


Quality in Python

« Bind two birds together: they have twice as many wings. However they are unable to fly. »
-- Sufi Proverb

Bader Ladjemi
<bader@tele2.fr>

AFPy





About me

- Bader Ladjemi
- Jabber: bader@amessage.de
- IRC: Bader on irc.freenode.net
- GSM: +33624334442
- <http://nic-nac-project.de/~bader/>
- AFPy(.org) active member





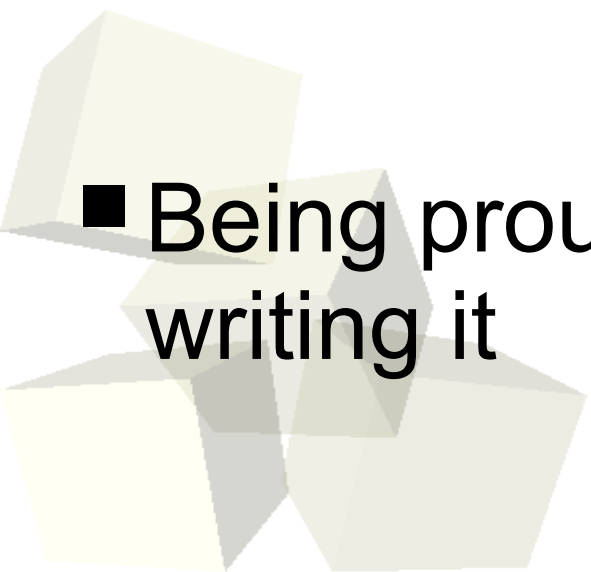
What does quality mean ?

- ISO 9000 defines quality as "degree to which a set of inherent characteristic fulfils requirements".



- "to fulfill the quality standards"

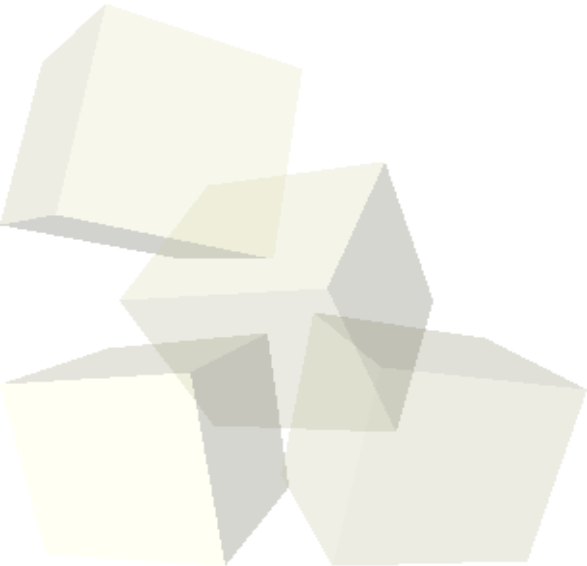
- Being proud of your code 6 months after writing it



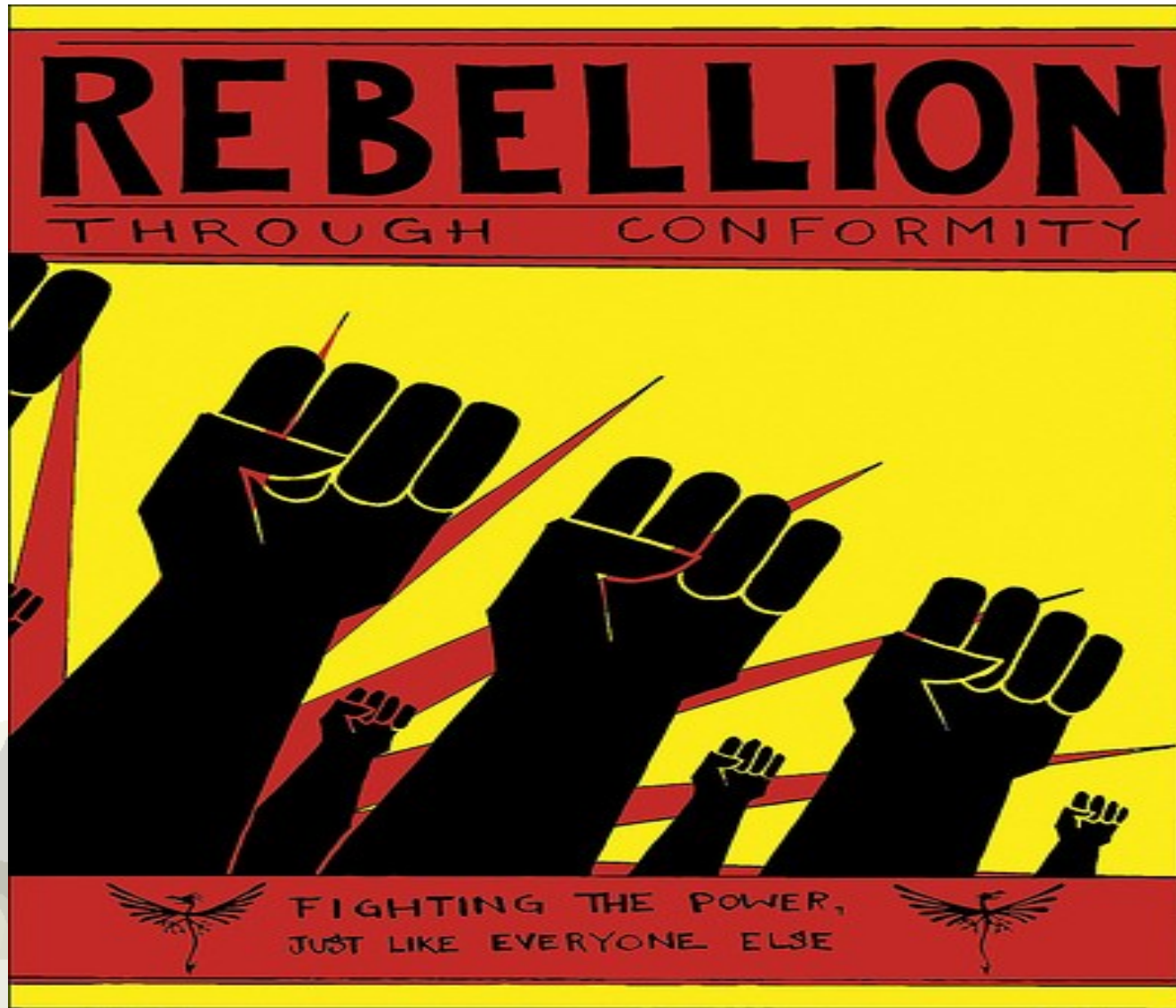


The Python way of life

- Conform to conventions
- Code tools
- Project requirements



Conventions Conformance





Python Enhancement Proposal

- Design issue

- Guideline

- New feature





PEP 20: “Zen of Python”

```
>>> import this
```

```
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.
```

```
[...]
```



PEP 8 -- Style Guide

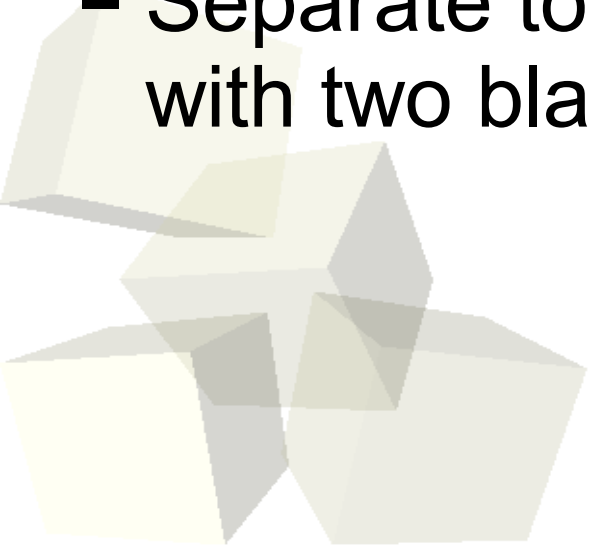
- Code is read much more than it is written
- PEP 20: “Readability counts”





PEP 8: Code lay-out

- Use 4 spaces per indentation level.
- Never mix tabs and spaces.
- Limit all lines to a maximum of 79 characters.
- Separate top-level function and class definitions with two blank lines.





(Tabs are Evil ?)

- yes, they are !

- ♦ <http://www.emacswiki.org/cgi-bin/wiki/TabsAreEvil>

- Emacs solution

- ♦ M x untabify

- ♦ in your .emacs

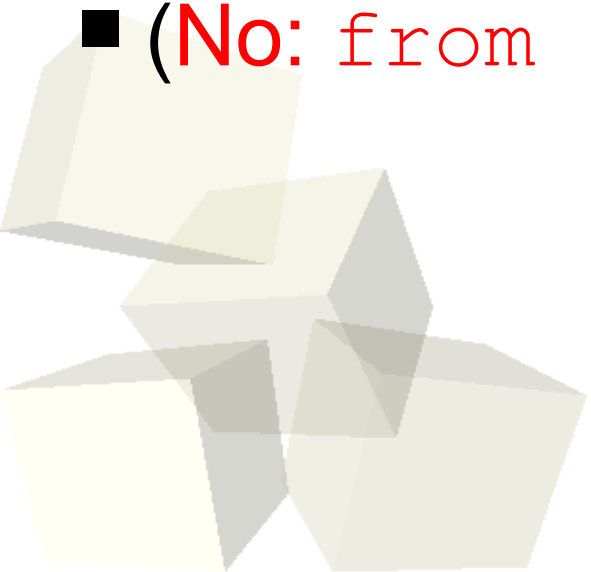
- `(setq-default indent-tabs-mode nil)`





PEP 8: Imports

- **No:** `import sys, os`
- Always put them at the top of the file
- (**No:** `from os import *`)





PEP 8: Whitespaces (1/8)

Not immediately inside parentheses, brackets or braces.

Yes:

```
spam(ham[1], {eggs: 2})
```

No:

```
spam( ham[ 1 ], { eggs: 2 } )
```





PEP 8: Whitespaces (2/8)

Not immediately before a comma, semicolon, or colon:

Yes:

```
if x == 4:  
    print x, y  
    x, y = y, x
```

No:

```
if x == 4 :  
    print x , y  
    x , y = y , x
```



PEP 8: Whitespaces (3/8)

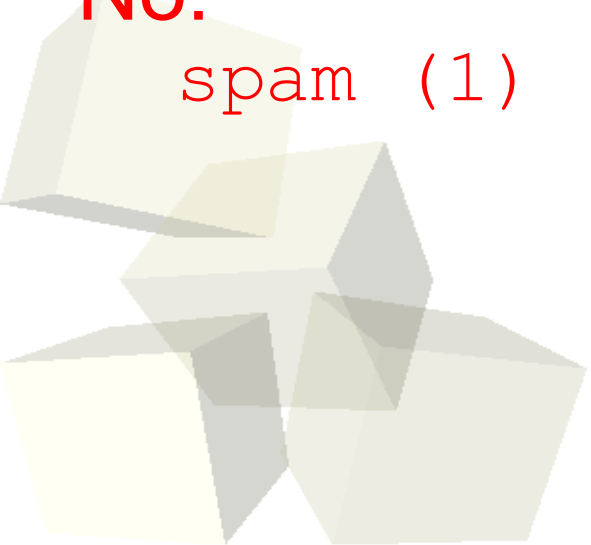
Not immediately before the open parenthesis that starts the argument list of a function call:

Yes:

```
spam(1)
```

No:

```
spam (1)
```





PEP 8: Whitespaces (4/8)

Immediately before the open parenthesis that starts an indexing or slicing:

Yes:

```
dict['key'] = list[index]
```

No:

```
dict ['key'] = list [index]
```





PEP 8: Whitespaces (5/8)

No more than one space around an assignment (or other) operator to align it with another.

Yes:

```
x = 1
y = 2
long_variable = 3
```

No:

```
x      = 1
y      = 2
long_variable = 3
```



PEP 8: Whitespaces (6/8)

Use spaces around arithmetic operators.

Yes:

```
submitted += 1  
x = x * 2 - 1  
c = (a + b) * (a - b)
```

No:

```
submitted +=1  
x = x*2 - 1  
c = (a+b) * (a-b)
```



PEP 8: Whitespaces (7/8)

Don't use spaces around the '=' sign when used to indicate a keyword argument or a default parameter value.

Yes:

```
def complex(real, imag=0.0):  
    return magic(r=real, i=imag)
```

No:

```
def complex(real, imag = 0.0):  
    return magic(r = real, i = imag)
```





PEP 8: Whitespaces (8/8)

Compound statements (multiple statements on the same line) are generally discouraged.

Yes:

```
if foo == 'blah':  
    do_one()  
    do_two()  
    do_three()
```

No:

```
if foo == 'blah': do_one(); do_two();  
do_three()
```



PEP 8: Comments

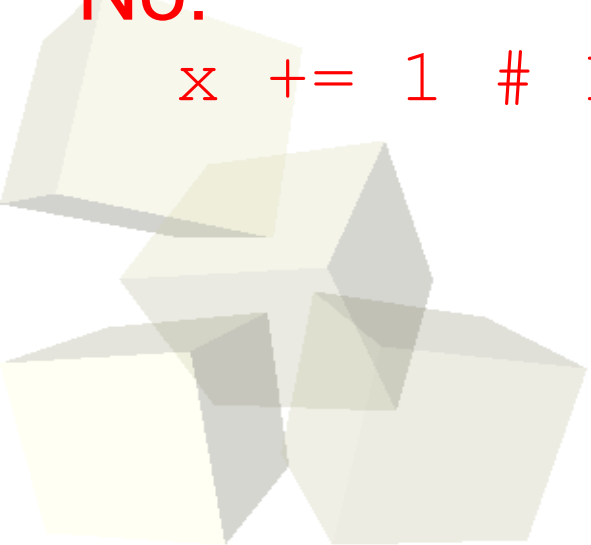
Comments should be complete sentences and in English please.

Yes:

```
x += 1 # Compensate for border
```

No:

```
x += 1 # Increment x
```



PEP 8: Naming Conventions (1/4)

- Avoid names: `l`, `o`, `I`
- Modules should have short, lowercase names, without underscores. `[a-z]{2,}[0-9]`
- Class names use the `CapWords` convention.
- Function names should be lowercase, with words separated by underscores `[a-z_][a-z0-9_]*`



PEP 8: Naming Conventions (2/4)

Arguments.

- Always use 'self' for the first argument to instance methods.
- Always use 'cls' for the first argument to class methods.



PEP 8: Naming Conventions (3/4)

Class members.

- Use one leading underscore only for non-public methods and instance variables.

- ♦ `def _private_method(self):`

- Append a single trailing underscore to avoid conflicts.

- ♦ `print_`

PEP 8: Naming Conventions (4/4)

Class attributes.

Expose attribute name, without complicated accessor/mutator methods.

Yes:

```
self.foo = 'baz'
```

No:

```
def get_foo(self):  
    return self._foo
```

```
def set_foo(self, foo):  
    self._foo = foo
```



PEP 8: Programming (1/3)

Boolean comparison.

Yes:

```
if greeting:
```

No:

```
if greeting == True:
```

Worse:

```
if greeting is True:
```



PEP 8: Programming (2/3)

Comparison.

None:

Yes:

```
a is None
```

No:

```
a == None
```

Type:

Yes:

```
if isinstance(obj, int):
```

No:

```
if type(obj) is type(1):
```





PEP 8: Programming(3/3)

Strings.

Concatenation:

Yes:

```
''.join(a, b)
```

No:

```
a+b
```

Is it a String ? (unicode, str, etc.)

Yes:

```
if isinstance(obj, basestring):
```

No:

```
if isinstance(obj, str):
```



PEP 257 – Docstrings

- Modules, class and functions should have docstrings

- Example:

```
def complex(real=0.0, imag=0.0):  
    """Form a complex number.
```

```
    Keyword arguments:
```

```
    real -- the real part (default 0.0)
```

```
    imag -- the imaginary part (default 0.0)
```

```
    """
```

- Not a function signature:

```
def function(a, b):  
    """function(a, b) -> list"""
```



Module Variables

- `__version__` (PEP 8)
 - ♦ `__version__ = "4.2"`
 - ♦ module version

- `__all__` (PEP 8)
 - ♦ `__all__ = ("foo_bar", "FooBar")`
 - ♦ list of public module objects that are designed for use via `from module import *`

- semi-officials: `__author__`,
`__copyright__`, `__license__`

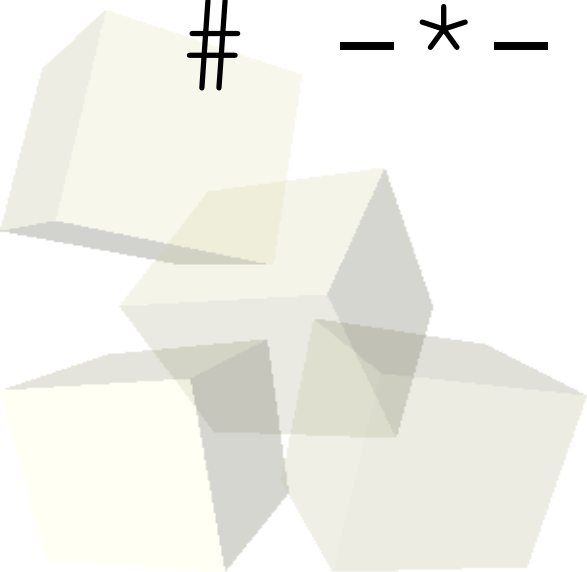


PEP 263 -- Encoding

- encoding matters
- you have to deal with encodings
- everything has an encoding
- ***especially*** if you're using **utf-8**

```
# ! /usr/bin/env python
```

```
# -*- coding: utf-8 -*-
```





nose, **doctest**, pyunit, unittest, DataTest,
gtktest, guitest, TestGears, testtoob, thrud,







- Syntax checker:
 - ◆ pyflakes, pylint, pychecker

- Doc generation:
 - ◆ pydoc, epydoc, happydoc, pudge

- Code Coverage
 - ◆ coverage.py, pycover

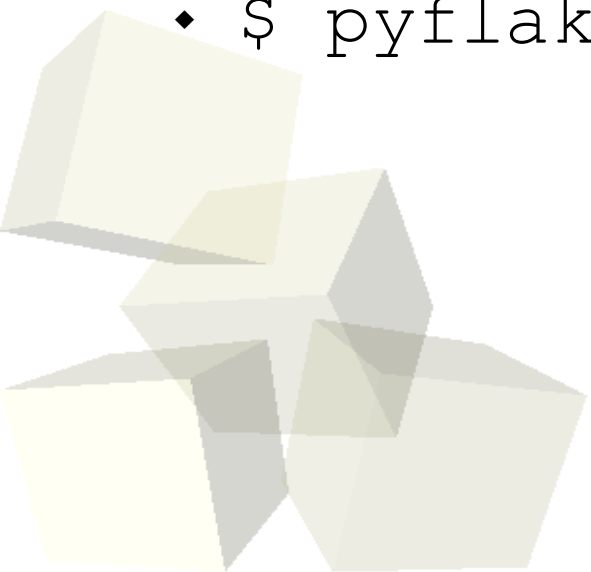
- Kwalitee
 - ◆ Cheesecake project (Google SoC 2006)



- Simple (relaxed?) syntax checker

- Output example:

 - ◆ `$ pyflakes /usr/lib/python2.4/* .py`





- checks for compliance with PEP 8

- much more pedantic

- highly configurable
 - ◆ default configuration
 - `$ pylint -generate-rcfile`
 - ◆ sample configuration:
 - <http://nic-nac-project.de/~bader/misc/pylintrc>



Code Coverage: coverage.py

- Measures code coverage during Python execution.
- Which lines are executable
- Which have been executed
- Usage:
 - ◆ \$ coverage -x module.py
 - ◆ \$ coverage -rm module.py



- README (reStructuredText)
- COPYRIGHT (GNU GPLv2)
- INSTALL
- CHANGELOG
- setup.py (setuptools > distutils)
- package/
- tests/
- doc/



Kwalitee (Coming Soon)

- CheeseCake, Google Summer of Code 2006
- *«What is a good module? That's hard to say. What is good code? That's also hard to say. "Quality" is not a well-defined term in computing ... and especially not Perl. One man's Thing of Beauty is another's man's Evil Hack. »*
- Set of indicators about a project
 - ◆ ex: has_readme
 - Checks if the distribution includes a README



« Tout ce qui peut être fait un autre jour, le peut être aujourd'hui. »

-- Montaigne





References

- PEP 1 – Python Enhancement Proposal
- PEP 8 – Style Guide
- PEP 20 – “Zen of Python”
- PEP 257 – Docstrings
- PEP 263 – Encoding
- Kwalitee <http://cpants.perl.org/kwalitee.html>





Questions

Don't forget to tell some joke *now*

