

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.

Pythonコード添削道場

“私的” Python 然コードスタイル

増田 泰

株式会社 アクセセンス・テクノロジー

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!

「Py然 (ぱいぜん)」とは

- Py然 = PyZen
- Zen of Python準拠: `>>> import this`
- 的確な構文と名前
- Py然 = Pythonic, PEP8-ism
- Pythonのコードに見える
- 他のPythonユーザにも理解しやすい

The Zen of Python, by Tim Peters

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 case

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 do, 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!

```
>>> 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.
```

```
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!
```

```
Namespaces are one honking great idea -- let's do more of those!
```

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.

Readability counts.

Sparse is better than dense.

Readability counts.

Although practicality beats purity.

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!

- Beautiful : キモイコードは嫌だ
- Explicit : 一目瞭然ならコードで語れる
- Simple : 簡単な方がメンテしやすい
- Complex : 汚いワンライナーより複合文
- Flat : ネストは思考を鈍らせる
- Sparse : スペースや空行は句読点と段落

The Zen of Python, by Tim Peters

PEP-8

- Guido と Barry によるコードスタイルガイド
 - 原文：<http://www.python.org/dev/peps/pep-0008/>
 - 和訳：<http://www.olderiver.org/python/pep-0008j.html>
 - 多くのPythonプロジェクトで採用
- PEPは「共通語」「書道」
 - なまっても片言でも通じるし死なないよ
 - でも相手によっちゃ取り合ってくれないよ
 - 通じないのはお互いによくない

The Zen of Python, by Tim Peters

PEP-8スタイルの基本

- インデントは4スペース
- 行の最大長は 79 文字
- トップレベルの関数／クラスは2行あける
- メソッド定義は1行あける
- コメントと docstring
- 命名規則

Namespaces are one honking great idea -- let's do more of those!

The Zen of Python, by Tim Peters

PEP-8推奨の命名規則

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.

- モジュール名：小文字、`_`なし `modulename`
- クラス／例外名：CapWords `MyClass`
- メソッド名：小文字+`_` `method_name`
- メソッドの変数：`self/cls` `def foo(self, ...)`
- 「定数」：大文字+`_` `MY_CONSTANT`

Special cases aren't special enough to break the rules.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be only one --obvious-- way to do it.

Although that way may not be obvious at first unless you're Dutch.

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!

The Zen of Python, by Tim Peters

コメントとdocstring

形式その1

```
# ファイルをコピーする
# @param dir1 コピー元ディレクトリ
# @param dir2 コピー先ディレクトリ
# @param files dir1内に存在するコピー対象ファイル名
def fcopy(dir1 , dir2 , files):
```

...

```
def fcopy(dir1 , dir2 , files):
```

```
    """ファイルをコピーする
```

引数:

dir1 -- コピー元ディレクトリ

dir2 -- コピー先ディレクトリ

files -- dir1内に存在するコピー対象ファイル名

```
    """
```

...

The Zen of Python, by Tim Peters

79文字の制限

- 長い行のできる理由
- 文字列リテラルが長過ぎる
- メソッド呼び出しがカスケードする
- 79字制限は決してきつくない
- 明示的な折り返しと論理行
- 構文の見直し

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.

Default arguments to functions should be ordered from most specific to least specific.

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.

There should be a better way to do it.

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!

The Zen of Python, by Tim Peters

長い行の折り返し

1. 引数が長過ぎる

```
someobject.somemethod(very_long_argument_foo, very_long_argument_bar)
```

```
someobject.somemethod(very_long_argument_foo, very_long_argument_bar)
```

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!

The Zen of Python, by Tim Peters

長い行の折り返し

2. メソッド呼び出しがカスケード

```
someobject.somemethod(arg1, arg2).another  
method(...).moremethod(...)
```

```
someobject.somemethod(arg1, arg2) \  
    .anothermethod(...)  
    .moremethod(...)
```

(ただしちょっと変態)

The Zen of Python, by Tim Peters

長い行の折り返し

3. 文字列が長過ぎる

```
message = "lorem ipsum, quia dolor sit,  
amet, consectetur, adipisci velit, sed qu  
ia non numquam eius modi ..."
```

```
message = ("lorem ipsum, quia dolor sit, amet,  
"consectetur, adipisci velit, sed "  
"quia non numquam eius modi ...")
```

Namespaces are one honking great idea -- let's do more of those!

The Zen of Python, by Tim Peters

長い行の折り返し

4. 内包表記とジェネレータ式

```
fn_list = [os.path.join( node[0],  
filename ) for node in os.walk( dir_path )  
for filename in node[2]]
```

```
fn_list = (  
os.path.join(path, filename)  
for path, sub, fn in os.walk(dir_path)  
for filename in fn)
```

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!

The Zen of Python, by Tim Peters

ロジックやインタフェースの設計を見直す

```
get_timestamp = lambda filename: os.stat(filename)[stat.ST_MTIME]
get_filelist = lambda dir_path: [os.path.join(node[0], filename)
for node in os.walk(dir_path) for filename in node[2]]
get_dir_info = lambda dir_path: dict(zip(get_filelist(dir_path),
[get_timestamp(filename) for filename in sorted(get_filelist
(dir_path))]))
```

```
def revised_get_dir_info(dir_name):
    filenames = (
        os.path.join(path, filename)
        for path, subdirs, files in os.walk(dir_name)
        for filename in files)
    return dict((fn, os.stat(fn)[stat.ST_MTIME])
                for fn in sorted(filenames))
```

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!

The Zen of Python, by Tim Peters

よりスマートな表現

文字列の乗算を使う

```
print "////////////////////"
```

```
print "/"*60 # 幅60でセパレータ出力
```

extend() や内包表記を活用する

```
for f in self.Aset - self.Bset:
```

```
    self.ABlist.append(f)
```

```
self.ABlist.extend(list(self.Aset-self.Bset))
```

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.
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!