Python Coding Style Guild

jslee

Reference by Google python style guild

http://google-styleguide.googlecode.com/svn/trunk/pyguide.html

PEP 8 -- Style Guide for Python Code

http://www.python.org/dev/peps/pep-0008/

Who am I？

- 不太會寫js的jslee
- 目前正在創立YouHack.com
- 曾經任職過的公司有台達電子雲端技術中心，D-Link、中華電信、Yam、Yahoo!以及飛行網(Kuro P2P download)

19 Language Rule

17 Style Rule

Start

pychecker

在Git Sever 上加一個hook 去tigger pychecker.
commit 時若沒辦法過pychecker 就沒辦法 commit 成功.
已確保進git server的code都是pass pychecker

+ pep8 (check)
+ pyflakes (static check)

Git Hook Script:
https://gist.github.com/jsleetw/faa9113528c6b5db2a98

imports

use:
import x
from x import y
from x import y as z

do not use:
from x import *

import 不要用*號

Package

use:

```
from sound.effects import echo
```

do not use:

```
import sound.effects.echo
```

不要點點點到天邊

Exceptions

```
rasie MyException('Error message')

try:
    raise Error
except Error as error:
    pass

raise MyException, 'Error message'

except:
```

不要懶惰開個string就丟，或是except:全部都吃走

Global variables

全域變數能別用就別用

Nested/Local/Inner Classes and Functions

Nested/Local/Inner Classes and Functions 不錯

List Comprehensions

do not use:

```
result = [(x, y) for x in range(10) for y in range(5) if x * y > 10]

return ((x, y, z)
        for x in xrange(5)
        for y in xrange(5)
        if x != y
        for z in xrange(5)
        if y != z)
```

不要搞一行文字天書

Generators

Generator (Yields:) 有需要就拿來用

Lambda Functions

Lambda一行可，超過60-80字請用nested

Conditional Expressions

```
x = 1 if cond else 2.
```

簡單的條件表達式一行可

Default Argument Values

```
def foo(a, b=None):
        if b is None:
            b = []
```

```
def foo(a, b=[]):
    ......

  foo(1)
  foo(1, b=2)

  foo(1, 2)
```

Propeties

建立一個Class介面要簡潔輕巧, 內部要考慮周詳

Code example on document.

True/False evaluations

Never use == or != to compare singletons like None.
Use is or is not.

```
if not x and x is not None:
```

When handling integers use == or !=:

```
if not users:
    print 'no users'
if foo == 0:
    self.handle_zero()
if i % 10 == 0:
    self.handle_multiple_of_ten()
```

```
if len(users) == 0:
    print 'no users'
if foo is not None and not foo:
    self.handle_zero()
if not i % 10:
    self.handle_multiple_of_ten()
```

Use the "implicit" false if at all possible.

Deprecated Language Features

使用 Python 2.7.3 有支援的語法

不要用 Python 3.3.0. 才支援的語法

Lexical Scoping

```
def get_adder(summand1):
"""Returns a function that adds numbers to a given number."""
    def adder(summand2):
      return summand1 + summand2
    return adder

i = 4
def foo(x):
    def bar():
        print i,
    # ...
    # A bunch of code here
    # ...
    for i in x:  # Ah, i *is* local to Foo, so this is what Bar sees
        print i,
    bar()
```

Function and Method Decorators

Decorators 是"top level code"

```
class C(object):
    @my_decorator
    def method(self):
        # method body ...
```

is equivalent to:

```
class C(object):
    def method(self):
        # method body ...
    method = my_decorator(method)
```

Decorators明智使用但不濫用

Threading

不要依賴內建type的原子性

ex.

__hash__

__eq__

use threading.Condition instead of using lower-level locks.

Power Features

access to bytecode
on-the-fly compilation
dynamic inheritance
object reparenting
import hacks
reflection
modification of system internals
etc.

python很多威能，但不要使用奇技巧淫

Semicolons

不要使用分號

Line length

最長80字一行，但是通常不夠用

Parentheses

```
if foo:
    bar()
while x:
    x = bar()
if x and y:
    bar()
if not x:
    bar()
return foo
for (x, y) in dict.items(): ...

if (x):
    bar()
if not(x):
    bar()
return (foo)
```

Indentation

使用4 space

no (tabs / mix tab / space / 2 space)

Blank Line

開頭空兩行，其餘空一行
其他不要亂空行

White space

使用pychecker

Shebang Line

需要直接執行的.py 才用#!/usr/bin/python

Comments

1.註解多多益善，累積人品

2.code reviewer可要求coder補註解

Classes

```
class SampleClass(object):
    pass

class OuterClass(object):
    class InnerClass(object):
        pass

 class ChildClass(ParentClass):
     """Explicitly inherits from another class already."""
```

```
class SampleClass:
        pass

class OuterClass:
    class InnerClass:
        pass
```

繼承使用object別偷懶

Strings 1

```
x = a + b
x = '%s %s! % (imperative, expletive)
x = 'name: %s; score: %d' % (name, n)
```

```
x = '%s %s' % (a, b) # use + in this case
x = imperative + ',' + expletive + '!'
x = name: ' + name + ': score:' + str(n)
```

優先使用%

Strings 2

```
items = ['<table>']
for last_name, first_name in employee_list:
    items.append('<tr><td>%s, %s</td></tr>' % (last_name, first_name)
items.append('<table>')
 employee_table = ''.join(items)
```

```
employee_table = '<table>'
for last_name, first_name in employee_list:
    employee_table += '<tr><td>%s, %s</td></tr>' % (last_name,
first_name)
employee_table += '</table>'
```

在loop裡不要用+ +=處理string

Strings 3

```
print("This is much nicer.\n"
      "Do it  this way.\n")

print '''This is pretty ugly.
Don't do this
'''
```

使用’’’別太醜，寧願不用，對齊就有美感

Files and Sockets 1

resource使用完就關掉

```
with open("hello.txt") as hello_file:
    for line in hello_file:
        print line
```

使用with 處理file

Files and Sockets 2

```
import contextlib

with contextlib.closing(urlopen("http://www.python.org/")) as
front_page:
    for line in front_page:
        print line
```

非file使用.closing

TODO Comments

#TODO(kl@gmail.com): Use a "*" here for string repetition.
#TODO(Zeke) Change this use relations.

使用TODO寫TODO的時候要屬名，是自己也可以是別人

Imports formatting

```
import os
import sys

import os, sys

import foo
from foo import bar
from foo.bar import baz
from foo.bar import Quux
from Foob import ar
```

明確指出從哪裡import

Statements

```
if foo: bar(foo)

if foo: bar(foo)
else: baz(foo)

try:
bar(foo)
expect ValueError: baz(foo)
```

一行只能用在很簡單的if上

Access Control

分清楚Public Private

Naming

Type	Public	Internal
Packages	lower_with_under	
Modules	lower_with_under	_lower_with_under
Classes	CapWords	_CapWords
Exceptions	CapWords	
Functions	lower_with_under()	_lower_with_under()
Global/Class Constants	CAPS_WITH_UNDER	_CAPS_WITH_UNDER
Global/Class Variables	lower_with_under	_lower_with_under
Instance Variables	lower_with_under	_lower_with_under (protected) or __lower_with_under (private)
Method Names	lower_with_under()	_lower_with_under() (protected) or __lower_with_under() (private)
Function/Method Parameters	lower_with_under	
Local Variables	lower_with_under	

Main

```
def main():
    .....

if __name__ == '__main__':
    main()
```

Parting Words

最後重點是為了要一致，好溝通

End

Q&A