PEP 8

Estilo en python

PEP 8 - Estilo en python

Qué es?

Es una guía de estilo

Cuál es su objetivo?

Mejorar la legibilidad del código y hacerlo consistente

Alguien la usa realmente?

Sí, prácticamente todo proyecto opensource basa en esta guía.

Quién la creó?

Guido van Rossum y Barry Warsaw

PEP 8 - Estilo en python

Basado en el Zen de Python (PEP20)

```
>>>import this
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.
Errors should never pass silently.
Unless explicitly silenced.
```

PEP 8 - Estilo en python

Indentation

4 espacios o un tab (mejor 4 espacios).

No mezclar.

En las funciones debe ser claro cuales son los argumentos.

PEP 8 - Estilo en python

Sí

```
# Aligned with opening delimiter
foo = long_function_name(var_one, var_two,
                         var_three, var_four)
# More indentation included to distinguish this from the rest.
def long_function_name(
            var_one, var_two, var_three,
            var_four):
      print(var_one)
```

No

```
# Arguments on first line forbidden when not using vertical alignment
foo = long_function_name(var_one, var_two,
      var_three, var_four)
# Further indentation required as indentation is not distinguishable
def long_function_name(
      var_one, var_two, var_three,
      var_four):
      print(var_one)
```

PEP 8 - Estilo en python

Limita las líneas a un máximo de 79 caracteres.

```
class Rectangle(Blob):
    def __init__(self, width, height,
                 color='black', emphasis=None, highlight=0):
        if (width == 0 and height == 0 and
            color == 'red' and emphasis == 'strong' or
            highlight > 100):
            raise ValueError("sorry, you lose")
        if width == 0 and height == 0 and (color == 'red' or
                                           emphasis is None):
            raise ValueError("I don't think so")
        Blob.__init__(self, width, height,
                      color, emphasis, highlight)
```

PEP 8 - Estilo en python

Líneas en blanco

Separar las funciones y clases raíz (o top-level) con dos líneas vacías.

Los métodos dentro de una clase con una línea blanca.

Líneas vacías extra pueden ser usadas para separar grupos de funciones relacionadas.

Usar líneas vacías dentro de las funciones para indicar secciones lógicas separadas.

PEP 8 - Estilo en python

Imports

Sí	No
`import os` `import sys`	`import os, sys`
`from subprocess import Popen, PIPE`	

Cuándo?

Justo después del comentario del módulo, antes de las constantes y variables globales.

Ordenando imports

Agrupado por global a específico:

1 - Librería estándar

2 - Módulos de terceros

3 - Imports locales

PEP 8 - Estilo en python

Espacios en las expresiones

Paréntesis, corchetes o llave:.

```
Sí:  cocinar(ham[1], {eggs: 2})
No:  cocinar( ham[ 1 ], { eggs: 2 } )
```

Coma, punto y coma o dos puntos:

```
Sí: if x == 4: print x, y; x, y = y, x
No: if x == 4 : print x , y ; x , y = y , x
```

Paréntesis en llamadas a funciones:

```
Sí: sumar(1, 2)
No: sumar ( 1,1)
```

Listas y diccionarios:

```
Sí: dict['key'] = list[index]
No: dict ['key'] = list [index]
```

Asignaciones:

```
Sí: cantidad = 10
No: cantidad=10
    espacios      = 10
    mas_espacios  = 10
```

PEP 8 - Estilo en python

Comentarios

Los comentarios que contradicen el código son peores que no tener comentarios. Hacer una prioridad mantener los comentarios al día!

Los comentarios deben ser oraciones completas. Si un comentario es una frase u oración, su primera palabra debe ser mayúscula, a menos que sea un identificador que comienza con una letra minúscula.

Si un comentario es corto, el punto al final puede ser omitido. Los comentarios de bloque generalmente consisten de uno o más párrafos construidos con frases completas, y cada frase debe terminar con un punto.

Programadores de Python de países de habla no inglesa: por favor escriba sus comentarios en Inglés, a menos que esté 120% seguro de que el código no será leído por personas que no hablan su idioma.

PEP 8 - Estilo en python

Estilos de nombrado

- b (single lowercase letter)
- B (single uppercase letter)
- lowercase
- lower_case_with_underscores
- UPPERCASE
- UPPER_CASE_WITH_UNDERSCORES
- CapitalizedWords (or CapWords, or CamelCase)
- mixedCase
- Capitalized_Words_With_Underscores (ugly!)

Nota: Cuando se usa abreviaturas en CamelCase, usar en mayúscula todas las letras de la abreviatura. HTTPServerError es mejor que HttpServerError.

PEP 8 - Estilo en python

Estilos de nombrado

Nombres que empiezan con "_" son para decir que es de uso interno:

def _single_leading_underscore:

Nombres que terminan con "_" se utilizan para evitar conflictos de nombre:

Python keyword, e.g: `Tkinter.Toplevel(master, class_='ClassName')`

Nombres que terminan y empiezan con doble "_":

`def __double_leading_and_trailing_underscore__:`

"magic" objects or attributes that live in user-controlled namespaces.

Ejemplo: `'__init__'`, `'__iter__'`, `'__le__'`, `'__len__'`, `'__lt__'`

PEP 8 - Estilo en python

Estilos de nombrado

Nunca usar el caracter '`l`' (L minúscula), '`0`' (o mayúscula), o '`I`'

(i mayúscula) como nombre de variables simple.

Los nombres de módulo deberían tener todas sus letras en minúscula, los "_" se pueden utilizar **sólo** si mejora la lectura, el uso de "_" en nombre de módulos no es aconsejado.

Los nombres de clases utilizan **CapWords**.

```
class EstiloEnPython():
```

Clases de uso interno pueden tener "_" al comienzo `_PrivateClass`

```
class _EstiloEnPythonPrivado():
```

Las excepciones son derivadas en la clase **Exception** por lo tanto se nombran como las clases pere se le agrega el sufijo "**Error**" (sólo si es un error)

```
class InvalidIDError(Exception):
class DoesNotExist(Exception):
```

PEP 8 - Estilo en python

Estilos de nombrado

Los nombres de funciones siempre deben ser en minúsculas con palabras separadas por "_" para mejorar, siempre que sea necesario para mejorar la legibilidad.

```
Sí:  def mi_funcion():
No:  def MiFuncion():
```

Siempre usar **self** para nombrar el primer argumento de los métodos de instancia.

```
def metodo_instancia(self, otro_argumento):
```

Siempre usar **cls** para nombrar el primer argumento de los métodos de clase.

```
@classmethod
def metodo_clase(cls, otro_argumento):
```

Si el nombre de un argumento colisiona con una palabra reservada generalmente es mejor agregar un "_" al final del nombre que usar una abreviación o un cambio.

```
def metodo(class_, clss):
```

PEP 8 - Estilo en python

Recomendaciones

Comparaciones con singletons como **None** siempre deben hacerse utilizando **is** o **is not**

Ej: preguntar por **`if x:`** cuando querías decir **`if x is not None:`**

Usar excepciones basadas en clases.

```
class ErrorDB(Exception):
"""Base class for errors in the database."""
```

PEP 8 - Estilo en python

Recomendaciones

Sí:

```
try:
    value = collection[key]
except KeyError:
    return key_not_found(key)
else:
    return handle_value(value)
```

No:

```
try:
    # Muy amplia!
    return handle_value(collection[key])
except KeyError:
    # También captura KeyError de handle_value()
    return key_not_found(key)
```

PEP 8 - Estilo en python

Recomendaciones

Comparaciones

```
Sí:  if foo.startswith('bar'):
No:  if foo[:3] == 'bar':
```

Saber si un objeto es de cierto tipo:

```
Sí: if isinstance(obj, int):
No: if type(obj) is type(1):
```

Preguntar si una lista esta vacia:

```
Sí: if not seq:
    if seq:
No: if len(seq)
    if not len(seq)
```

PEP 8 - Estilo en python

Validando el código:

Link de descarga PEP8 https://pypi.python.org/pypi/pep8

Se puede usar sin instalar, sólo necesitan el archivo pep8.py

```
$ python pep8.py /usr/lib/python2.6/linecache.py
/usr/lib/python2.6/linecache.py:13:1: E302 expected 2 blank lines, found 1
/usr/lib/python2.6/linecache.py:23:11: E261 at least two spaces before inline comment
```

```
$ python pep8.py ./directorio
$ python pep8.py -qq --statistics archivo.py
$ python pep8.py -qq --statistics ./directorio
```

La opción "--statistics" hace un resumen al final del análisis contando la cantidad de ocurrencias de errores, se suele usar en conjunto con la opción "-qq", omite nombrar los archivos con error para sólo ver las estadísticas.

PEP 8 - Estilo en python

Preguntas?