# Ayudantía 02

## PEP8 y Estructuras de Datos

Felipe Garrido  Antonio Ossa

Departmento de Ciencia de la Computación
Pontificia Universidad Católica de Chile

IIC2233, 2015-1

# Tabla de contenidos

# ¿Qué es?

## Origen

**Guido van Rossum** es un científico de la computación, conocido por ser el autor del lenguaje de programación Python.

# ¿Qué es?

## Origen

**Guido van Rossum** es un científico de la computación, conocido por ser el autor del lenguaje de programación Python.

"Code is read much more often than it is written."

— Guido van Rossum

# ¿Qué es?

## Definición

- PEP 8 es la Guía de Estilo para Python, que busca consistencia en escritura de código para facilitar la posterior lectura.
- Es importante saber cuando usar esta guía, pues debe prevalecer la legibilidad del código y la consistencia con código ya escrito, como en el caso de proyectos grandes.
- La guía aplica tanto sobre código en sí, indentación, espacios en blanco, comentarios y nombramiento de variables

# Consejos

## Algunos de todos los de la guía...

- 4 espacios de indentación

# Consejos

## Algunos de todos los de la guía...

- 4 espacios de indentación
- 79 caracteres como máximo tamaño de linea

# Consejos

## Algunos de todos los de la guía...

- 4 espacios de indentación
- 79 caracteres como máximo tamaño de linea
- Usar espacios y lineas en blanco cuidadosamente

# Consejos

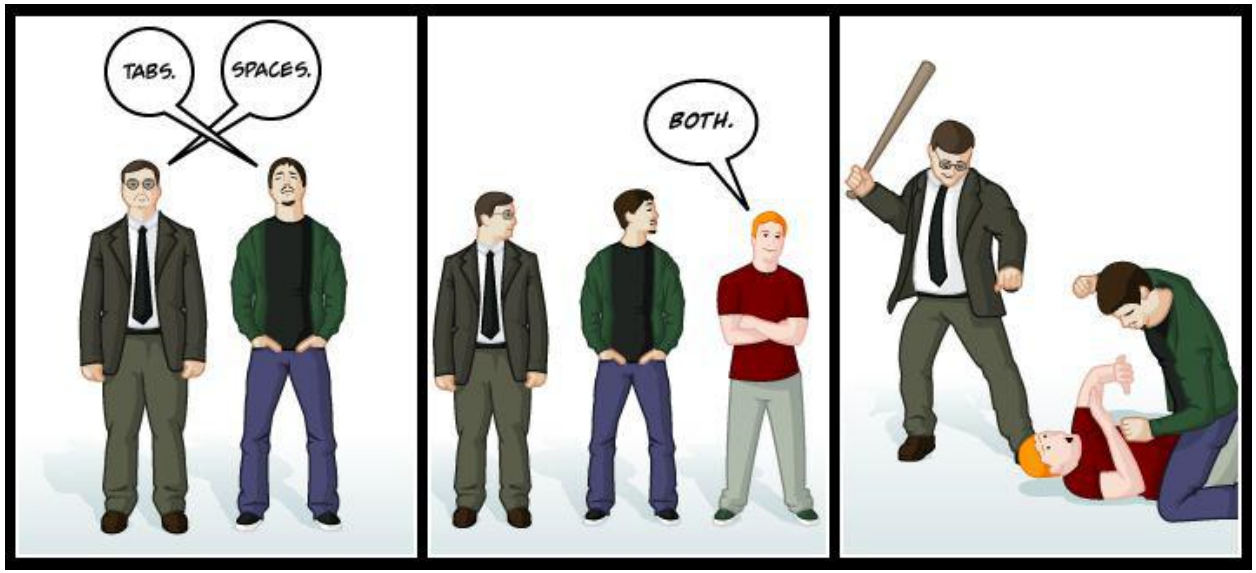
## Algunos de todos los de la guía...

- 4 espacios de indentación
- 79 caracteres como máximo tamaño de linea
- Usar espacios y lineas en blanco cuidadosamente
- Y...

# Consejos

## Algunos de todos los de la guía...

- 4 espacios de indentación
- 79 caracteres como máximo tamaño de linea
- Usar espacios y lineas en blanco cuidadosamente
- Y...




Nunca mezclar tabuladores y espacios

# Ejemplos

## Código "malo"

```
1   import os, sys
2   from collections import defaultdict
3
4   class Parking:
5       def __init__(self):
6               def no_car():
7                       return Car( plate ='UNKNOWN')
8               self.parking = defaultdict(no_car())
9       def park(self, car):
10              self.parking.append(car)
11
12  class Car:
13       def __init__(self, plate = 'NO PLATE'):
14              self.plate = plate
15
16  car= Car()
17  parking =     Parking()
18  parking.park (car)
```

## Cambios

Vamos resolviendo los problemas. Primero, la indentación debe estar dada por 4 espacios.

# Ejemplos

## Indentación arreglada

```
1  import os, sys
2  from collections import defaultdict
3
4  class Parking:
5      def __init__(self):
6          def no_car():
7              return Car( plate ='UNKNOWN')
8          self.parking = defaultdict(no_car())
9      def park(self, car):
10         self.parking.append(car)
11
12 class Car:
13     def __init__(self, plate = 'NO PLATE'):
14         self.plate = plate
15
16 car= Car()
17 parking =    Parking()
18 parking.park (car)
```

### Cambios

Los "import" no pueden estar en una misma línea.

# Ejemplos

## Imports corregidos

```
1  import os
2  import sys
3  from collections import defaultdict
4
5  class Parking:
6      def __init__(self):
7          def no_car():
8              return Car( plate ='UNKNOWN')
9          self.parking = defaultdict(no_car())
10     def park(self, car):
11         self.parking.append(car)
12
13 class Car:
14     def __init__(self, plate = 'NO PLATE'):
15         self.plate = plate
16
17 car= Car()
18 parking =    Parking()
19 parking.park (car)
```

### Cambios

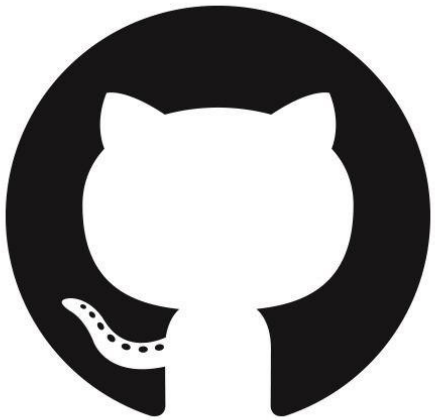
Ahora que nos encargamos de la indentación y de import, arreglamos los espacios y líneas en blanco.

# Ejemplos

## Listo, PEP8 :)

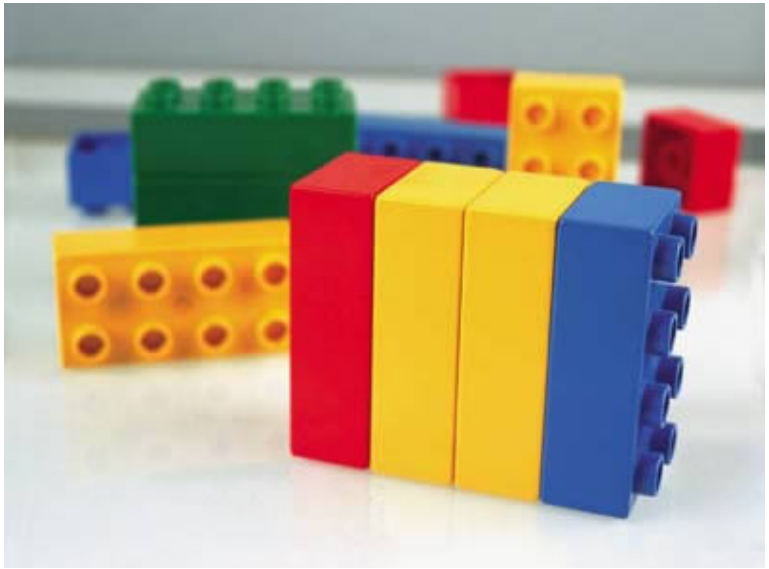
```
1   import os
2   import sys
3   from collections import defaultdict
4
5
6   class Parking:
7       def __init__(self):
8           def no_car():
9               return Car(plate='UNKNOWN')
10
11          self.parking = defaultdict(no_car())
12
13      def park(self, car):
14          self.parking.append(car)
15
16
17  class Car:
18      def __init__(self, plate='NO PLATE'):
19          self.plate = plate
20
21
22  car = Car()
23  parking = Parking()
24  parking.park(car)
```

# PEP8 en nuestra Wiki



**IIC2233-2015-1 > Syllabus > Wiki > PEP8**

# EDD



**Se utilizan para organizar la información y facilitar su manipulación.**

# Nodo

## Características

- Un nodo es la unidad básica de cualquier estructura de datos
- Contiene información
- Tiene referencia a otros nodos

# Nodo

## Código

```
1  class Nodo_Muy_Simple(object):
2
3      """Este es un nodo, la parte esencial de una EDD"""
4
5      def __init__(self, objeto=None):
6          self.objeto = objeto
7          self.siguiente = None
```

# Lista ligada

## Código

```
1   class ListaLigada(object):
2       def __init__(self):
3           self.__ultimo = self.__primero = None
4
5       def agregar(self, elemento):
6           if self.__primero is None:
7               self.__primero = Nodo(elemento)
8               self.__ultimo = self.__primero
9           else:
10              self.__ultimo.siguiente = Nodo(elemento)
11              self.__ultimo = self.__ultimo.siguiente
12
13      def obtener(self, indice):
14          n = 0
15          actual = self.__primero
16          while (n != indice):
17              if actual is None:
18                  raise IndexError('No es valido este indice')
19              else:
20                  actual = actual.siguiente
21              n += 1
22          return actual.objeto
```

# Árbol

## Características

- Un árbol es una EDD no lineal
- Los nodos de un árbol hacen referencia a uno o más nodos

"Un árbol está compuesto de áboles."

# Árbol

## Código

```
1   class Arbol(object):
2
3       """Este es un nodo modificado
4       para almacenar otros nodos de su mismo tipo"""
5
6       def __init__(self, elemento=None):
7           self.elemento = elemento
8           self.__hijos = Lista()
9
10      def agregar_hijo(self, elemento):
11          self.__hijos.agregar(Arbol(elemento))
12
13      def obtener_hijo(self, indice):
14          return self.__hijos.obtener(indice)
```

# Sumario

- PEP8 es una **Guía** de Estilo para Python, no son reglas.
- Los nodos son la estructura básica de las estructuras de datos más complejas.
- Usen los nodos para crear nuevas estructuras.

**BONUS:**

**El Zen de Python... import this**