Universidad Carlos III

Curso Desarrollo del Software 2021

# Normativa de codificación

GRUPO:80 EQUIPO:08

Alumnos: **Diego Robes Lozano (100429029) / Gonzalo Llosá Cea (100429091)**

# Índice

# Introducción

Este documento pretende ser una guía de estilo para escribir código en python de una manera uniforme en un equipo de trabajo. Muchas de estas reglas podrán ser implementadas a través de Pylint pero las que no serán marcadas en este documento con una estrella (*).

# Comentarios

## Comentarios de documentación

La primera línea de cada archivo deberá comenzar por un comentario en bloque que indique los colaboradores del archivo y la siguiente línea la fecha de última modificación. El formato deberá ser el siguiente:

```
“”” Colaboradores: nombre1 apellido1, ..., nombreN apellidoN
Fecha: dia/mes/año.”””
```

Los comentarios de documentación tendrán un tamaño mínimo de 2 líneas, ya que la primera se utilizará para indicar los autores involucrados en el archivo y la segunda para indicar la fecha de última modificación.

Todos los métodos y todas las clases deben tener un comentario de al menos una línea que indique su propósito.

No hemos implementado esto último porque, para Pylint, comentar un método o una clase significa poner un bloque de comentario (el cual, como hemos ya decidido, ocupa dos líneas como mínimo). Nos parecía innecesario un bloque de comentario para cada clase y cada función.

Por defecto, Pylint exige comentarios para funciones y clases, así que hemos tenido que especificar que ignore estos errores en el apartado [MASTER] del pylintrc. .

Siempre se dejará un espacio inmediatamente después del inicio del comentario en bloque ("""..""") y no se dejará ninguno inmediatamente antes del final de este. El primer carácter debe estar escrito en mayúscula.

Ejemplo:

```
""" Comentario
      en
      bloque."""
```

* En Pylint hemos implementado la obligación de que los comentarios tengan mínimo dos líneas, pero no se controla qué se escribe en estos comentarios.

## Comentarios de línea

Todos los comentarios deberán dejar un espacio entre el símbolo “#” y el primer carácter del comentario. *

Ejemplo:

```
# Esta función…
```

El primer carácter de cualquier comentario estará en mayúsculas y deberán acabar en un punto.*

Los comentarios nunca compartirán línea con alguna parte funcional del código.*

No:

```
for(i in array): # Un for que hace…
```

Si:

```
# Un for que hace…
for(i in array):
```

El comentario siempre se referirá a la/s línea/s por debajo de él, por lo que se deberá dejar una línea en blanco encima de cada comentario.*

Ejemplo:

```
i = 0

# Un for que...
for(i in array):
```

# Convenciones de nombrado

## Nombrado de módulos

Los módulos se deben nombrar haciendo uso del estilo camelCase.

Ejemplo:

```
tamañoEdificio = n
```

## Nombrado de argumentos

Los argumentos se deben nombrar haciendo uso del estilo camelCase.

## Nombrado de atributos

Los atributos se deben nombrar haciendo uso del estilo camelCase. Por este motivo no ha sido posible mantener los atributos privados como __nombre, ya que no cumplen la sintaxis de camelCase. Es por ello que en vez de doble barra baja estas se han sustituido por el prefijo “priv”, mantenido sus respectivos setters y getters.

## Nombrado de clases

Las clases se deben nombrar haciendo uso del estilo PascalCase.

Ejemplo:

```
class EdificioTactico()...
```

Las variables de clase deberán llevar la palabra clave self delante de ellas.

Ejemplo:

```
self.nombre
```

### Nombrado de variables

Las variables se deben nombrar haciendo uso del estilo camelCase.

### Nombrado de constantes

Las constantes se deben nombrar haciendo uso del estilo UPPER_CASE. Además estas constantes deberán estar separadas en un fichero constantes.py.

Ejemplo:

```
PI = 3.14159
```

### Nombrado de funciones

Las funciones se deben nombrar haciendo uso del estilo camelCase.

### Nombrado de métodos

Los métodos se deben nombrar haciendo uso del estilo camelCase.

## Distribución del código

### Número máximo de bloques anidados

Como máximo se podrán anidar 4 bloques de código con el fin de que este sea más legible.

### Indentación

Las indentaciones se realizarán con una sola tabulación.

Ejemplos *incorrectos*:

```
for (i in array):
          a = b;
for (i in array):
        a = b;
```

Ejemplo *correcto*:

```
for (i in array):
     a = b;
```

Cuando una línea requiera ser separada en dos, esta se deberá indentar con una longitud de 4.

Si:

```
unaFuncion(atributo1, atributo2, atributo3,
     atributo4, atributo5, atributo6)
```

No:

```
unaFuncion(atributo1, atributo2, atributo3,
atributo4, atributo5, atributo6)
```

### Caracteres por línea

El número máximo de caracteres por línea permitidos será de 150.

### Número máximo de argumentos que puede recibir una función / método

El número de argumentos máximo que puede recibir una función o método es de 7.

### Número máximo de buleanos en una sentencia if

El número máximo de buleanos en una sentencia if es 6.

Cada fichero contendrá como máximo una clase y esta se deberá llamar de la misma forma que la clase que contiene.*
Las variables podrán ser públicas.
No se pueden declarar variables con el nombre de "kk" o "caca"

## Espacios en blanco en expresiones y sentencias

Las declaraciones de variables deberán dejar un espacio entre el nombre de la variables, el símbolo igual y el valor de la variables.*

Ejemplo:

```
numManzanas = 3
```

No puede haber un espacio entre el nombre del método al que se llama y el paréntesis de la lista de argumentos del mismo. *

No:

```
sumar (1, 2)
```

Si:

```
sumar(1, 2)
```

No puede haber un espacio inmediatamente antes del paréntesis de indexado.*

No:

```
lista [i]
```

Si:

```
lista[i]
```

Los argumentos de las clases y de las funciones tienen que ir separados por un espacio además de por una coma de la siguiente manera:*

Si:

```
class Clase(arg1, arg2, arg3):
```

No:

```
class Clase(arg1,arg2,arg3):
class Clase(arg1 , arg2 , arg3):
```

Los operadores lógico-matemáticos tienen que ir siempre separados por espacios a izquierda y a derecha.*

Ejemplos:

```
a = b + c
a = b % c
if ((a and b) or c)
```