NORMATIVA DE CÓDIGO

Universidad Carlos III

Grado Ingeniería Informática 2022-23

Desarrollo de Software (Grupo 81)

Práctica realizada por:

- **Jaime Vaquero Rabahieh** (NIA: 100472248, Email: 100472248@alumnos.uc3m.es)
- **Alejandro Díaz Cuéllar** (NIA: 100472173, Email: 100472173@alumnos.uc3m.es)

Aclaración: esta guía o normativa de código solo incluye reglas que han sido modificadas en pylintrc, es decir, aunque existan reglas que pylint permita su modificación, como el nombrado de archivos, hemos decidido no cambiarlas.

TABLA DE CONTENIDO

1) Formato y estándar para archivos

Regla 1 - Número máximo de caracteres por línea (max-line-length)

- **Valor por defecto:** 100.
- **Nuevo valor:** 79.
- **Descripción:** Como en el estándar PEP-8, no puede haber más de 79 caracteres por línea.

```
# Regla 1
# Maximum number of characters on a single line.
max-line-length=79
```

Regla 2 - Número máximo de líneas por archivo (max-module-lines)

- **Valor por defecto**: 1000.
- **Nuevo valor:** 2500.
- **Descripción**: No pueden haber más de 2500 líneas por archivo.

```
#Regla 2
# Maximum number of lines in a module.
max-module-lines=2500
```

2) Estándar para variables

Regla 3 - Nombre para las constantes, usando rgx (const-naming-style y const-rgx)

- **Valor por defecto:** UPPER_CASE
- **Nuevo valor:** ^[A-Z].*$
- **Descripción:** Los nombres de constantes deberán empezar por una letra mayúscula, seguida de cualquier otra combinación de letras, números o signos
- **Ejemplos**: Constante, Const23 o CONSTANTE son constantes válidas mientras que constante, const23 o 12CONST no lo son

```
# Regla 3
# Regular expression matching correct constant names. Overrides const-naming-
# style. If left empty, constant names will be checked with the set naming
# style.
const-rgx= ^[A-Z].*$
```

Regla 4 - Nombre para las variables, usando rgx (variable-naming-style y variable-rgx):

- **Valor por defecto:** snake_case
- **Nuevo valor:** ^[a-z].*$
- **Descripción:** Los nombres de variables deben empezar por una letra minúscula, seguida de una combinación de letras, números y signos.
- **Ejemplos:** var1, var_ej, variable o varA son nombramientos válidos mientras que 4var, -ej o Var no lo son.

```
#Regla 4
# Regular expression matching correct variable names. Overrides variable-
# naming-style. If left empty, variable names will be checked with the set
# naming style.
variable-rgx= ^[a-z][a-z,0-9]*[_[a-z][a-z,0-9]*]*$
```

3) Estándar para clases y métodos

Regla 5 - Nombre para las clases (class-naming-style)

- **Valor por defecto:** PascalCase
- **Nuevo valor:** camelCase
- **Descripción:** Los nombres de clases deberán utilizar el formato camelCase
- **Ejemplos**: objetoPersona(), orderManager() son ejemplos válidos, mientras que ObjetoPersona() o ORDER_MANAGER() no lo serían

```
# Regla 5
# Naming style matching correct class names.
class-naming-style=camelCase
```

Regla 6 -Nombre de las constantes de clases (class-const-naming-style y class-consts-rgx):

- **Valor por defecto:** UPPER_CASE
- **Nuevo valor:** ^[A-Z]*$
- **Descripción:** Los nombres de constantes de clases deben empezar por una letra consonante. El resto puede ser números, letras y símbolos.
- **Ejemplos:** Const, CONST, Cos1 o Cos_Ej son nombramientos válidos mientras que 1Ej, const o +Ej no lo son.

```
# Regla 6
# Regular expression matching correct class constant names. Overrides class-
# const-naming-style. If left empty, class constant names will be checked with
# the set naming style.
class-const-rgx= [^[0-9][A-Z]]$
```

Regla 7 - Nombre para los atributos de clases (class-atribute-naming-style)

- **Valor por defecto:** any
- **Nuevo valor:** snake_case
- **Descripción:** Los nombres de atributos de clases deberán estar escritos en snake_case
- **Ejemplos**: Ejemplos de atributos válidos son ean_13, codigo, o codigo_ean_13, mientras que ejemplos de atributos no válidos serían EAN13, Codigo, o codigoEAN13

```
# Regla 7
# Naming style matching correct class attribute names.
class-attribute-naming-style=snake_case
```

Regla 8 -Número de atributos por clase(max-attributes)

- **Valor por defecto:** 7
- **Nuevo valor:** 10
- **Descripción:** Como máximo, deben haber 10 atributos por clase.

```
# Regla 8
# Maximum number of attributes for a class (see R0902).
max-attributes=10
```

Regla 9 - Nombre para los métodos de clases (method-naming-style)

- **Valor por defecto:** snake_case
- **Nuevo valor:** camelCase
- **Descripción:** Los nombres de métodos de clases deberán estar escritos en camelCase
- **Ejemplos**: validateEan13() o testJson() son ejemplos válidos, mientras que ejemplos como VALIDATE() O Test_JSON() no lo serían

```
# Regla 9
# Naming style matching correct method names.
method-naming-style=camelCase
```

Regla 10 - Número de padres por clase(max-parents)

- **Valor por defecto:** 7
- **Nuevo valor:** 2
- **Descripción:** Cada clase puede tener 2 padres como mucho.

```
# Regla 10
# Maximum number of parents for a class (see R0901).
max-parents=2
```

4) Estándar para funciones y argumentos

Regla 11 - Nombre para las funciones (function-naming-style)

- **Valor por defecto:** snake_case
- **Nuevo valor:** camelCase
- **Descripción:** Los nombres de las funciones deberán estar escritos en formato camelCase
- **Ejemplos**: Nombres de funciones válidos son sumaNumeros() o multiplicaValores() mientras que sucesion_fibonacci() o SumaMultiplos()

```
# Regla 11
# Naming style matching correct function names.
function-naming-style=camelCase
```

Regla 12 - Nombre para argumentos (arguments-rgx)

- **Valor por defecto:** snake_case
- **Nuevo valor:** ^[a-z].*$
- **Descripción:** Los nombres de argumentos deben empezar por una letra minúscula, seguida de una combinación de letras, números y signos.
- **Ejemplos:** arg1, argA, ejemplo o arg_usado son nombramientos válidos mientras que Arg, 1Ej y _num no lo son.

```
# Regla 12
# Regular expression matching correct argument names. Overrides argument-
# naming-style. If left empty, argument names will be checked with the set
# naming style.
argument-rgx= ^[a-z][a-z,0-9]*[_[a-z][a-z,0-9]*]*$
```

Regla 13 - Número máximo de argumentos en funciones y métodos (max-args)

- **Valor por defecto:** 5
- **Nuevo valor:** 10
- **Descripción:** El número máximo de argumentos que puede haber en una función o método será de 10

```
# Regla 13
# Maximum number of arguments for function / method.
max-args=10
```

Regla 14 - Número máximo de returns por función (max-return)

- **Valor por defecto:** 6
- **Nuevo valor:** 10
- **Descripción:** Como máximo, pueden haber 10 returns por función.

```
#Regla 14
# Maximum number of return / yield for function / method body.
max-returns=10
```