



Ejercicio Guiado 2

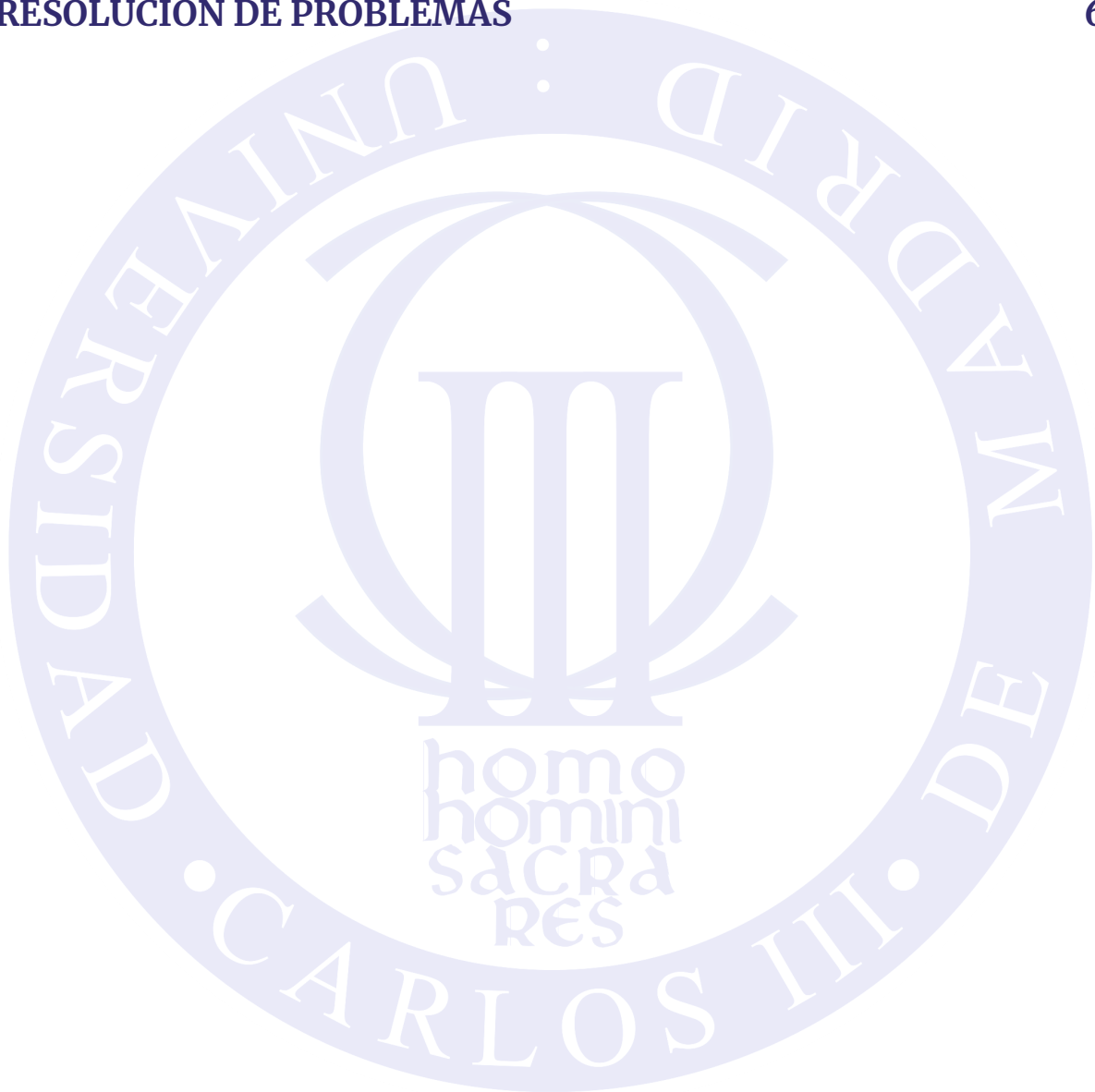
Aspectos Éticos y Legales de la Ingeniería del Software

Grupo 9

Ángela Serrano Casas
Arturo Soto Ruedas

07/02/2023

ENUNCIADO	3
WARNINGS Y CONVENCIONES EN EL CÓDIGO	5
RESOLUCIÓN DE PROBLEMAS	6



ENUNCIADO

Definir una normativa de codificación para el grupo y documentarla. Para ello se deben describir las reglas establecidas en un documento PDF. Se tendrá en cuenta la buena presentación de este documento.

A continuación se enumeran los cambios realizados en la normativa original brindada por Python. La primera parte será la regla original de la que se partía y la segunda parte (lo posterior a “ → ”) será la modificación de esta. Existen ciertas reglas que al inicio estaban comentadas, es decir no se aplicaban al código. Por ello, algunas de las normas se considerarán creadas mientras que otras serán solo modificadas.

1. `argument-naming-style=snake_case` → `argument-naming-style= camelCase`
En lugar de seguir un formato de tipo `snake_case` (palabras en minúsculas separadas por “_”) se decide modificar el estilo del nombre de los argumentos a `camelCase` (primer carácter y todas las palabras en minúscula menos el inicio de las siguientes palabras incluidas).

2. `class-attribute-naming-style= any` → `class-attribute-naming-style= camelCase`
Se decide que los atributos en las clases deben seguir el estilo `camelCase`.

3. `bad-names=foo, bar, baz, toto, tutu, tata` → `bad-names=foo, bar, baz, toto, tutu, tata, tete, titi, clase, funcion`.
Se ha modificado la lista de palabras prohibidas para el nombramiento de variables. Se ha añadido la palabra `funcion` y `clase` por razones obvias además de otras de prueba como “tete” y “titi”

4. `bad-names-rgxs=` → `bad-names-rgxs= [0-9]$`
Se ha restringido el nombramiento de las variables. Ninguna variable puede empezar por un número.

5. `class-const-naming-style= any` → `class-const-naming-style=UPPER_CASE`
El nombramiento de constantes en una clase pasa a ser en mayúsculas en su totalidad.

6. `class-const-rgx=` → `class-const-rgx= CT_[a-z0-9]{2,20}$`
Toda constante perteneciente a una clase debe iniciar por “CT_” seguida de cualquier carácter alfanumérico de longitud máxima 20

7. `class-naming-style=PascalCase` → `class-naming-style=UPPER_CASE`
El nombramiento de clases será en mayúsculas a partir de ahora.

8. `const-rgx=` → `const-rgx=CT_[a-z0-9]{2,20}$`
Creación de una nueva regla. El nombre de las constantes debe iniciar por el conjunto “CT_” seguido opcionalmente de un código alfanumérico con una longitud máxima de 20 caracteres.

9. `docstring-min-length=-1` → `docstring-min-length=5`
Aquellas funciones/módulos/clases que incluyan docstring deberán tener una longitud mínima de 5 caracteres. Esto implica que haya una breve explicación de las funcionalidades desarrolladas.

10. `function-naming-style=snake_case` → `function-naming-style=PascalCase`
Variación entre reglas. El nombre de las funciones seguirá un estilo de tipo PascalCase, es decir, cada inicio de palabra en mayúscula.

11. `good-names=i,j,k,ex,Run,_,var,total` → `good-names=i,j,k,ex,Run,_,var,total`
Se han incluido varios nombres de variables aceptados como correctos. Estos son nombres bastante comunes a la hora del desarrollo de código.

12. `include-naming-hint=no` → `include-naming-hint=yes`
Se incluirá una pequeña guía para conocer el correcto estilo de nombramiento de cada entidad definida en el código.

13. `reports=no` → `reports=yes`
Con el objetivo de recibir un reporte más completo acerca de los warnings devueltos por pylint, activamos los reports para que se desarrollen y visualicen mejor estos.

14. `ignore-long-lines=^\s*(#)?<?https?:/\S+>?$` → ELIMINADO
Hemos prescindido de esta regla puesto que no vemos necesario permitir líneas más largas de código.

15. `max-line-length=100` → `max-line-length=125`
Aumentamos el tamaño de la línea en un 25% de esta manera podemos generar comentarios más largos sin tener que cambiar de línea.

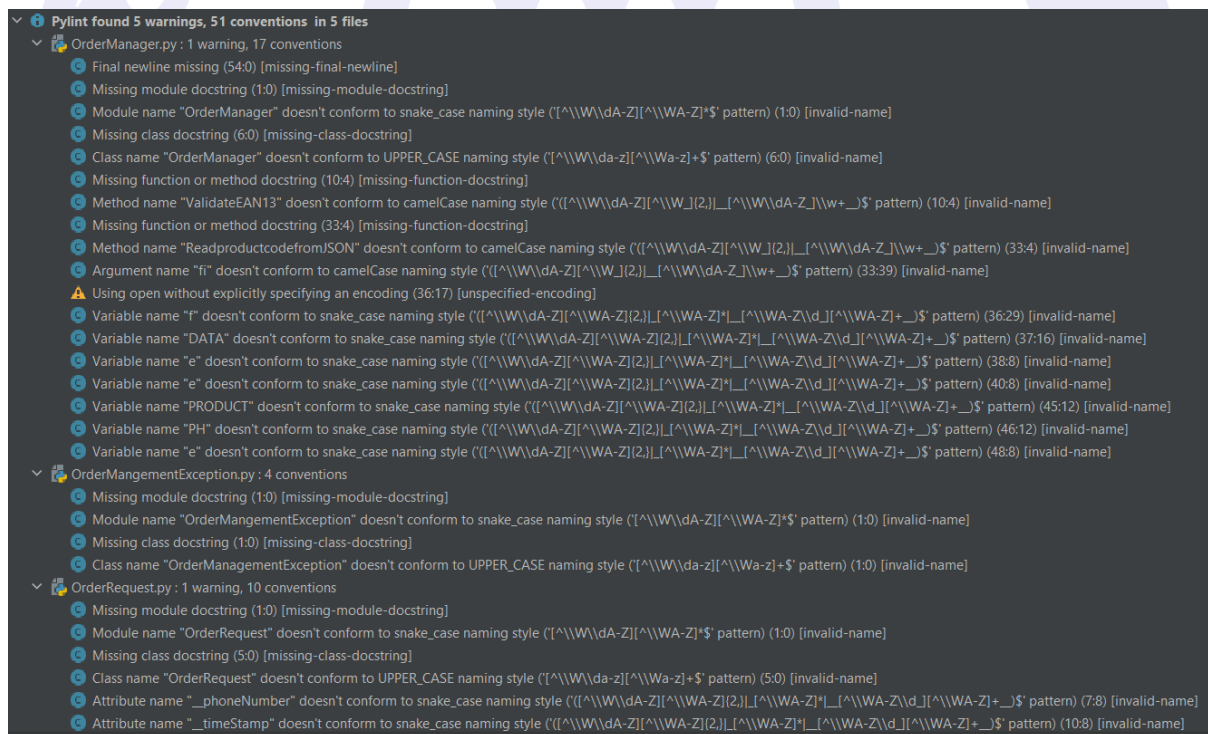
16. `max-returns=6` → `max-returns=4`
Reducimos el número de devoluciones que puede hacer una función a 4 puesto que el valor anterior nos parece excesivo.

17. check-protected-access-in-special-methods=no→check-protected-access-in-special-methods=yes

18. method-naming-style=snake_case → method-naming-style=camelCase
Los métodos de las clases cambian de estilo de snake_case a camelCase.

WARNINGS Y CONVENCIONES EN EL CÓDIGO

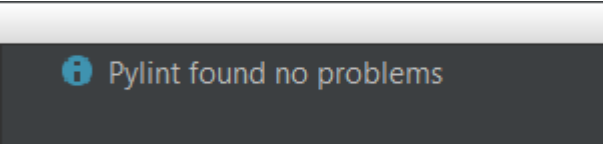
A continuación se adjunta una captura de pantalla del resultado de ejecutar pylint tras modificar los estándares. Mostrando así, los fallos, avisos y convenciones hallados en el código.



```
Pylint found 5 warnings, 51 conventions in 5 files
OrderManager.py : 1 warning, 17 conventions
  Final newline missing (54:0) [missing-final-newline]
  Missing module docstring (1:0) [missing-module-docstring]
  Module name "OrderManager" doesn't conform to snake_case naming style ('([a-zA-Z][a-zA-Z]*$' pattern) (1:0) [invalid-name]
  Missing class docstring (6:0) [missing-class-docstring]
  Class name "OrderManager" doesn't conform to UPPER_CASE naming style ('([A-Z][a-zA-Z]*$' pattern) (6:0) [invalid-name]
  Missing function or method docstring (10:4) [missing-function-docstring]
  Method name "ValidateEAN13" doesn't conform to camelCase naming style ('([a-zA-Z][a-zA-Z]*_[a-zA-Z][a-zA-Z]*$' pattern) (10:4) [invalid-name]
  Missing function or method docstring (33:4) [missing-function-docstring]
  Method name "ReadproductcodefromJSON" doesn't conform to camelCase naming style ('([a-zA-Z][a-zA-Z]*_[a-zA-Z][a-zA-Z]*$' pattern) (33:4) [invalid-name]
  Argument name "fi" doesn't conform to camelCase naming style ('([a-zA-Z][a-zA-Z]*_[a-zA-Z][a-zA-Z]*$' pattern) (33:39) [invalid-name]
  Using open without explicitly specifying an encoding (36:17) [unspecified-encoding]
  Variable name "f" doesn't conform to snake_case naming style ('([a-zA-Z][a-zA-Z]*_[a-zA-Z][a-zA-Z]*$' pattern) (36:29) [invalid-name]
  Variable name "DATA" doesn't conform to snake_case naming style ('([a-zA-Z][a-zA-Z]*_[a-zA-Z][a-zA-Z]*$' pattern) (37:16) [invalid-name]
  Variable name "e" doesn't conform to snake_case naming style ('([a-zA-Z][a-zA-Z]*_[a-zA-Z][a-zA-Z]*$' pattern) (38:8) [invalid-name]
  Variable name "e" doesn't conform to snake_case naming style ('([a-zA-Z][a-zA-Z]*_[a-zA-Z][a-zA-Z]*$' pattern) (40:8) [invalid-name]
  Variable name "PRODUCT" doesn't conform to snake_case naming style ('([a-zA-Z][a-zA-Z]*_[a-zA-Z][a-zA-Z]*$' pattern) (45:12) [invalid-name]
  Variable name "PH" doesn't conform to snake_case naming style ('([a-zA-Z][a-zA-Z]*_[a-zA-Z][a-zA-Z]*$' pattern) (46:12) [invalid-name]
  Variable name "e" doesn't conform to snake_case naming style ('([a-zA-Z][a-zA-Z]*_[a-zA-Z][a-zA-Z]*$' pattern) (48:8) [invalid-name]
OrderMangementException.py : 4 conventions
  Missing module docstring (1:0) [missing-module-docstring]
  Module name "OrderMangementException" doesn't conform to snake_case naming style ('([a-zA-Z][a-zA-Z]*$' pattern) (1:0) [invalid-name]
  Missing class docstring (1:0) [missing-class-docstring]
  Class name "OrderMangementException" doesn't conform to UPPER_CASE naming style ('([A-Z][a-zA-Z]*$' pattern) (1:0) [invalid-name]
OrderRequest.py : 1 warning, 10 conventions
  Missing module docstring (1:0) [missing-module-docstring]
  Module name "OrderRequest" doesn't conform to snake_case naming style ('([a-zA-Z][a-zA-Z]*$' pattern) (1:0) [invalid-name]
  Missing class docstring (5:0) [missing-class-docstring]
  Class name "OrderRequest" doesn't conform to UPPER_CASE naming style ('([A-Z][a-zA-Z]*$' pattern) (5:0) [invalid-name]
  Attribute name "__phoneNumber" doesn't conform to snake_case naming style ('([a-zA-Z][a-zA-Z]*_[a-zA-Z][a-zA-Z]*$' pattern) (7:8) [invalid-name]
  Attribute name "__timeStamp" doesn't conform to snake_case naming style ('([a-zA-Z][a-zA-Z]*_[a-zA-Z][a-zA-Z]*$' pattern) (10:8) [invalid-name]
```

RESOLUCIÓN DE PROBLEMAS

Tras modificar y adaptar todas las recomendaciones que mostraba pylint obtenemos el siguiente resultado:



Pylint found no problems

Finalmente, hemos resuelto en su totalidad las recomendaciones que nos hacía pylint. Para ello hemos tenido que prescindir de ciertas partes del código original del enunciado ya sea comentadolas o en algunos casos eliminando trozos de código directamente.

