SOFTWARE DEVELOPMENT

Computer Science

Guided Exercise 2

Collective Code Ownership and Coding Standards

Course 2022/2023

Group 89

Sergio Barragán Blanco, Eduardo Alarcón Navarro

100472343 , 100472175

100472343@alumnos.uc3m.es, 100472175@alumnos.uc3m.es

Sergio Barragán 100472344
Eduardo Alarcón 100472175

TABLE OF CONTENTS:

Sergio Barragán 100472344
Eduardo Alarcón 100472175

Coding Standard for our team:

Files:

1. The import functions must be always on top of the file, each one in a separate line, following alphabetical order:

Example:

-Correct:

import copy
import random
from mario import Mario

-Incorrect:

import random
import math

2. At the beginning of a function, a comment must be included describing the function, of length at leat 2 lines:

```
def FunctionName():
"""
Comment on how the functions works, at least 2 lines of comments are
expected.
"""
```

3. A separate file should be implemented for each class, with the naming convention being UPPER_CASE, with at least 5 letters, following the regex `^.{5,}$`:

Example:

- Correct:
  - CLASS1
- Incorrect:
  - CLASS
  - class1
  - cls1

Names and Variables:

4. Variables of the name : `i,j,k,ex,Run,_,elem,ele,el` will always be accepted

   - Variables that won't be accepted include: bar, baz, toto, tutu, tata, con

5. If a variable has multiple words, the words will always be in CamelCase
   - Correct: finalList
   - Incorrect: final_List or FinalList or final_LIST

6. Constant variables and class constant variables will be defined at the beginning of each file, and the name will be following the regular expression `^[^-_]{5,}$ :`
    - Correct:
        - Shift
        - SHIFT
        - ShifT1
    - Incorrect:
        - SH_IFT
        - shift_length
        - shift-len

7. Functions must be named in PascalCase, following the same naming convention as the variables:
   Example:
    - Correct: def CreationPlayer():
    - Incorrect: def creationPlayer():

8. The maximum number of characters that a .py file must contain in a single line is 120 characters.
9. Global variables that are not used will not be allowed.

Standard for Methods ——————————————————————————————————

Creo que vamos a tener que cambiar lo de los tabs por espacios porque el pycharm hace lo que le sale de ahí mismo.

10. *Use Tab for indentation or as a substitute, 4 spaces, all the code must be indented depending on the hierarchy of the line.*
    Example:
    Correct:
    ```
    for elem in list1:
        do_sth()
    ```
    Incorrect:
    ```
    for elem in list1:
      do_sth()
    ```

Operations:

11. The module naming, when imported will be any
12. The keywords that will be used to flag the file will be check against the regular expression `^__(TODO|FIXME|HAZME|TOCAME|ARREGLAME|XXX|LOOK)__$`
13. The similarities between segments of the codes must be at least of 10 lines
14. Spelling mistakes will be ignored with the words `Sergio, Barragán, Eduardo, Alarcón, UC3M,NIA,` as there are Spanish names and terms that might not be in the dictionary

Sergio Barragán 100472344
Eduardo Alarcón 100472175

The following changes are not inside pylint but we want them on our coding standard:

15. When comparing values of the type boolean with True or False using ==, follow this standard:
    - Correct: if greeting:
    - Correct: if not done:
    - Incorrect: if greeting == True:
    - Incorrect: if done == False:
    - Never: if greeting is True:

16. Avoid unnecessary blank spaces: dict['key'] = list[index] | if x == 4: , instead, use them without the space: dict['key'] = list[index] | if x == 4:

17. Use +=, -=, *= and /= operators instead of normal operators when dealing with the same variable.
    Example:
    - Correct: elem += 10
    - Incorrect: elem = elem + 10

Comments:

18. Comments will be used # for one line comments and triple quotes (“””) for multiple line explanations, the one line comments explaining a code will be situated above said line.
    Example:
    - Correct:

```
#This is a comment of one line describing the for
for i in range(len(numbers)):

“””
This is a multi-lined comment
The comment must be here, between the triple quotes
“””
```

    - Incorrect

```
for i in range(len(numbers)):
# This is a comment describing
# the for

“””This is a one line comment”””
```

Other Standards:

19. For list, set, tuples and dictionaries, the parenthesis (or the {} in case of dictionaries) will be in the same line as the contents.
    Example:
        natural_numbers = [1, 2, 3, 4, ...]
        real_numbers = (-99999, -99998,.....……., 0, ...99998, 99999)

20. Between the elements of a list, after the comma(“,”) that separates the elements, a space will be placed (“ “).
    - Correct: list1 = [1, 2, 3, “element”, False]
    - Incorrect: list1 = [1,2,3 ,”element”,True]

To locate the changes we have made inside the *pylintrc* file, we have elaborated a `pylintre_changes.md` where each of the rules we have edited are, as well as the line on where they can be found inside the *pylintrc* configuration file.

Comments:

From the main.py file:

We have added this comment, to disable the import error, to have a clean *pylint* run.

```
# pylint: disable=import-error
# This one is added to avoid the warning of the import error, which is the import of UC3MLogistics
```

From the OrderRequest.py file:

We have added the comment to disable having to create a *“dosctring”* for the import of the json module.

```
# pylint: disable=missing-module-docstring
```

From the OrderManagerException.py file:

```
# pylint: disable=missing-function-docstring
# pylint: disable=unused-variable
# This one is added to avoid the warning of the unused variable, which is the name of the class
```

From the OrderManager.py file:

```
# pylint: disable=missing-module-docstring
# pylint: disable=unused-variable
```

Sergio Barragán 100472344
Eduardo Alarcón 100472175

```
# Used to remove the warning the missing “doctring” on the json module and
the warning of the unused variable, which is the name of the class
respectively
```

Recomendation:

- **Not in pylint, but if a module is generated by the user, the import will be done using the * instead of naming each class.**
  correct: from Mario import *
  Incorrect import *