PYLINT CODING STANDARD

1. Variable naming

We decided to use a variable naming convention similar to snake_case that is ruled by the following regular expression:

([a-zA-Z_][a-z0-9_]*){1, 60}

Variable names can start with Uppercase or lowercase but cannot have uppercase letters anywhere else. This first character may be followed by lowercase letters or numbers. Character _ is also accepted, but not as the first character. The maximum length for a variable name is 60, and the minimum is 1.

2. Function naming

The convention we established to determine the name of our functions is camelCase.

3. Variables accepted

We learned that it is possible to make some exceptions to the naming rules and add variable names that we want to be accepted for sure. We added some accepted labels like “x” and “y”, additionally to some that were already accepted like “j” and “k”.

4. Method naming: -> camel case

The convention we used to label the methods in python is camelCase.

5. Unused imports

We fixed pylint to raise an alarm to tell us if there is any unused imports in __init__ files in some point of the code, in order to make our code more clear and understandable.

6. Arguments for functions

We fixed the maximum number of arguments for functions and methods to 7. We learned that 5 is the number of arguments commonly allowed but thought that some more arguments would be helpful for coding functions in several cases.

7. Constant naming

The convention we decided to use to name the constant is snake_case, as we consider this naming style more clean and easy to write compared to UPPER_CASE.

8. Characters per line

We set the maximum number of characters in a line to 150 characters.

9. Lines per module

The maximum number of lines that we set to write in a module is 2000 lines, so that we can write huge pieces of code in a single module.

10. Single line class

We permit to write a class in a single line as we considered that this can be handful when coding simple classes.

11. Single line if statement

Our pylint allows the body of an if to be on the same line as the test in the case that there is no “else”.

12. Note tags

We learned that pylint offers the possibility of adding personal note tags to highlight remarkable pieces of code by writing those words in our code, so we added the tag “IMPORTANT” to quickly find some line that we find particularly relevant.

13. Bad variable names

Finally, we discovered that it is also possible to fix any string you want not to be a variable name. In our code, we established that the labels “pepe” and “juan” cannot be the name of a variable.