Activity: Using Linters to Achieve Python Code Quality

Four Python libraries were used to assess code quality:

- PyLint
- PyCodeStyle
- PyDocStyle
- PyFlake

Ahead of this exercise, it is critical to highlight that all above libraries are static code analysers, therefore they are not able to dynamically evaluate code quality. Testing, such as unit and integration tests are vital to cover this aspect of code quality. Additionally, investigating for “code smells”, surface indicators of poor quality code is also important and must be managed.

It is clear from the outputs that PyLint is producing the most fixes to the code specified within the exercise across code quality areas, particularly code style and code efficiency. PyLint found possible improvements in below, amongst other items:

1. Variable naming styles for global and local variables
2. Unnecessary or missing whitespaces including newlines
3. Missing docstrings
4. Unreachable code (more code in line with return statement)
5. Variables called before assignment
6. Unnecessary imports
7. Uncalled variables

Pyflakes in comparison only found items with respect to waste in memory space via unused entities, such as:

1. Unnecessary imports, and
2. Uncalled variables

It is explicit in not detecting style issues within the code.

PyCodeStyle called out styling issues within the code, particularly:

1. Unnecessary or missing whitespaces including newlines
2. Long lines (over > 79 characters)

PyDocStyle is focused on finding issues with respect to docstrings, so missing docstrings or misspelled words within existing docstrings.