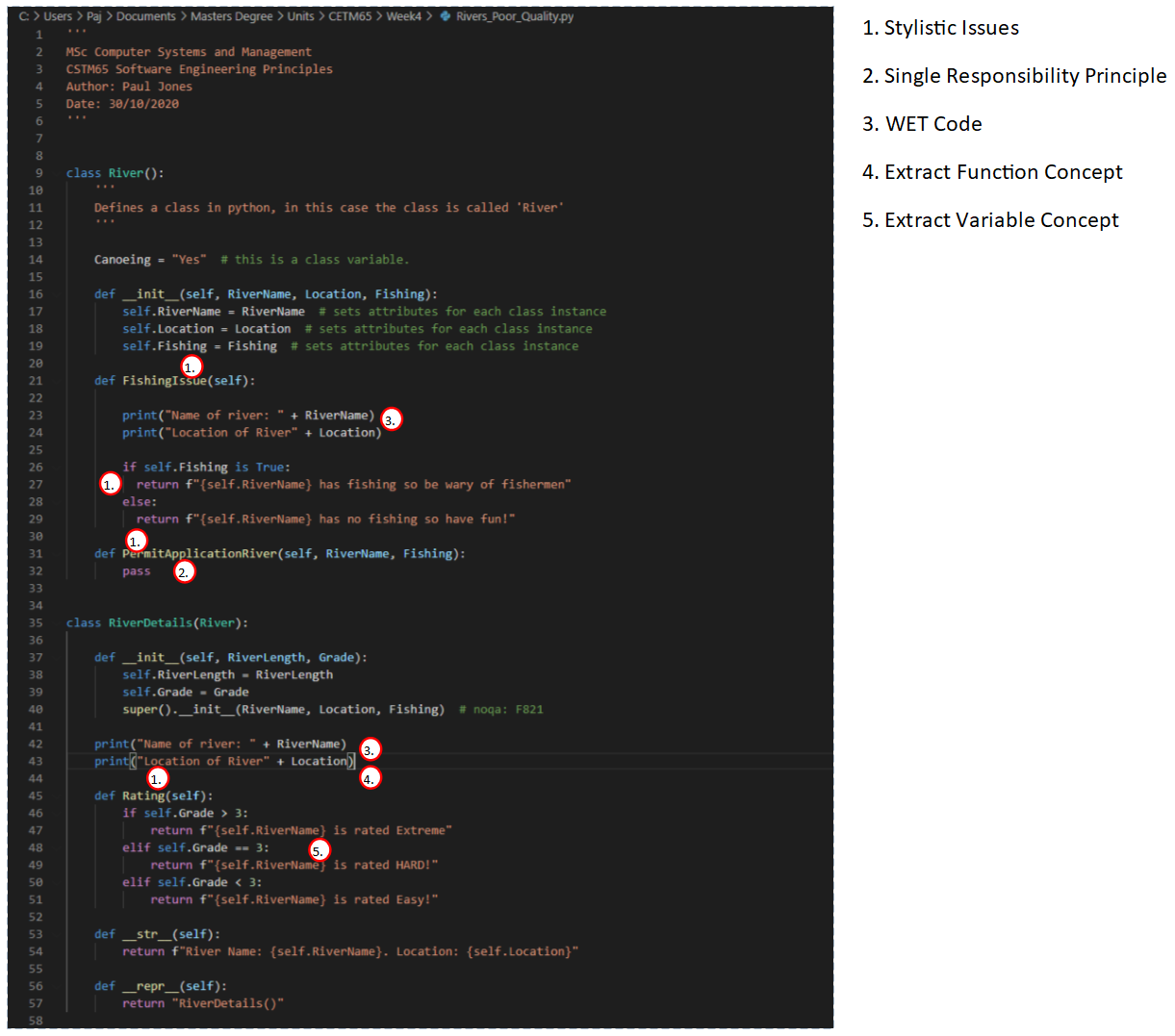
**Improving Code Quality**

There are many different aspects of a written code base that can make it ‘unclean’ and require it to be re-factored. Although unclean code can, and does, run, there may be ways to improve the code to make it easier to read. Both for yourself and for other programmers who may look at and utilise your code at a later date.

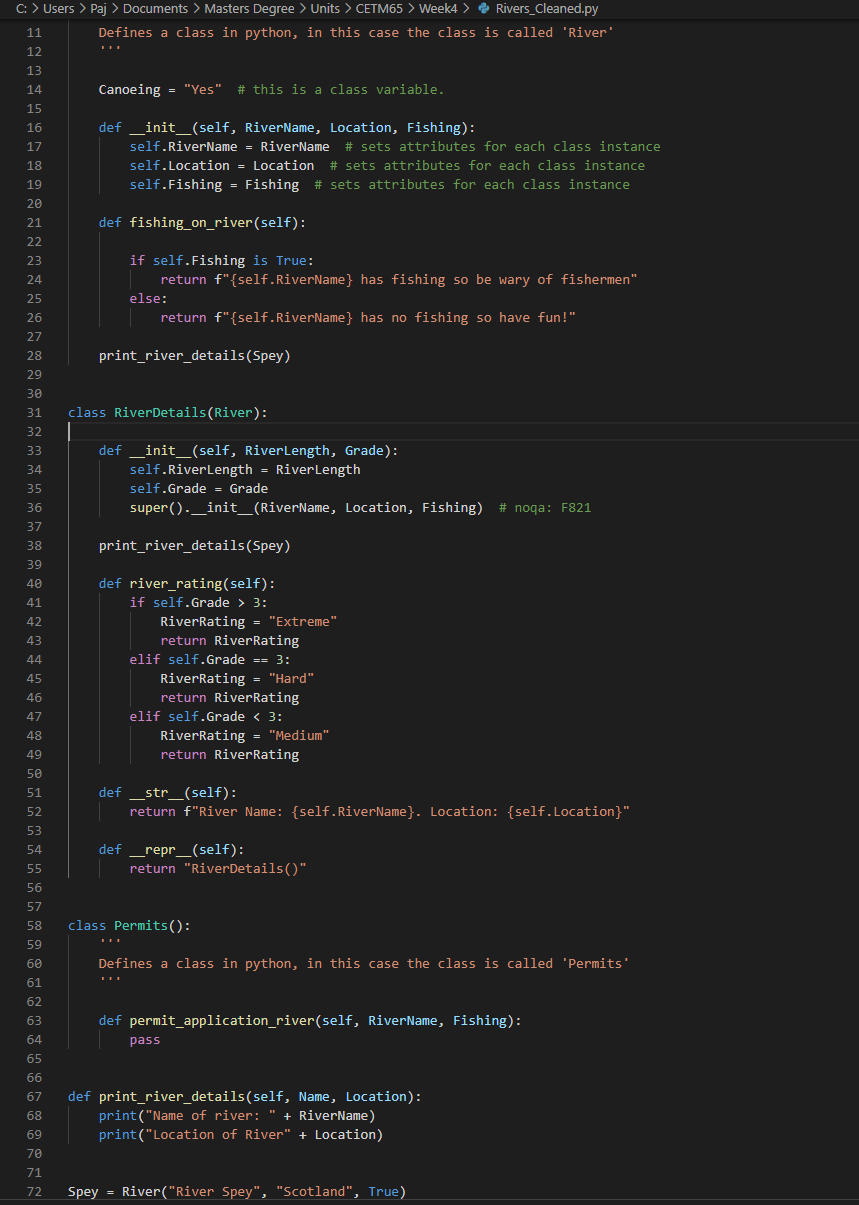
The following screenshot shows a piece of code that, whilst workable and runnable with zero errors, is not as clean as it should be. It does not conform to PEP8 standards and has examples of WET code (code that is needlessly repeated) as well as code that breaks the Single Responsibility Principle and code that requires Extract Function and Extract Variable concepts to be applied.

We re-factor code to make it more easily readable by a human. This does not always mean making the code shorter. You will see with this example that shorter code is not always cleaner code.



As you can see, the above code has many areas of poor quality. Stylistic issues include naming conventions not being correct for functions (these should be snake case) and/or function names being inadequate in terms of description. Incorrect indentation for the selection statement on line 26 to 29 is also a stylistic error. In terms of Single Responsibility Principle. The River class includes a function for permit application, which means the class in now responsible for both rivers and permits and this needs correcting. There is wet code throughout the two classes and this wet code should also be subject to the extract function concept. Finally, there should be an extracted variable for the rating selection statement.

The below screenshot shows the corrected code after it has been refactored, sanitised and made clean:



As you can see. The stylistic errors have been corrected. There is proper indentation of 4 spaces for the selection statements, functions are named with snake case and include more detail about the purpose of the function.

The permit function has been removed from the river class and placed into a new ‘Permits()’ class so as to enforce the single responsibility principle. The wet code has been stripped out using an extract function concept and placed into its own function that can be called multiple times from different functions and/or classes. Finally, the selection statement has had the results stored into a variable and that variable then returned.

There is still work that could be done to further clean the code, refactoring is an iterative process and one that should be constantly used when programming solutions.