ORAL FEEDBACK FRAMEWORK

Comprehension

To what extent does the student understand the code they have written?

- Review the investigate slides in their workbook.
- Ask questions about lines of code they have written to solve problems, getting them to explain how and why their algorithms work.

Maintainability

To what extent and how consistently has the student used best practices in creating readable code?

- The use of comments, subroutines, sensible identifier names and whitespace.
- Using code structures that are easy to understand.
- The use of the most appropriate iteration: counter or condition (from objective 6).

Scalability

To what extent could subroutines be used in other programs later and how well would the program perform if the data set it uses is increased significantly?

- Using subroutines and iterations instead of repeating blocks of code.
- Using self-contained subroutines with local variables and functions that return values.
- Using arrays and lists instead of multiple variables (from objective 8).

Robustness

To what extent can the program easily crash?

- Using validation (from objective 7).
- Using exception handling (from objective 9).

Approach

To what extent is the code the best algorithm for solving the problem?

- Creating time efficient algorithms (minimising the CPU cycles).
- Creating space efficient algorithms (minimising the use of memory) including using global variables only when it makes sense to do so.
- Alternative algorithms may also be considered even though they do not gain any significant advantage to appreciate the different approaches programmers might take and why.