# **Week 7: Lab outline**

* A base class for sorting called *sort*
  + This class should not be instantiated since it doesn’t have any sorting functionality but is simply a skeleton/blueprint for children class (specific sorting algorithms)
  + Read here on how in Python there is not a keyword for specifying when a class is ‘abstract’ i.e. not to be instantiated. <https://stackoverflow.com/questions/7989042/preventing-a-class-from-direct-instantiation-in-python>
  + Can do it by raising ‘not implemented’ error for the methods of the base class
* We then have 3 different extensions of the base class *sort*
  + Merge sort
  + Bubble sort
  + Quick sort
* Each child class will overload the parent class sorting function with their own specific algorithmic implementation (.sort)
* Each child class will also have 2 properties - runtime, and maxspace - to be populated when the (.sort) method is run on an object of the class.
* The method - getruntime, and getmax space - will output the values held in the properties above.
* Each class may also have a function to report - best case, worst case, and average case, time complexity plots.

[Add a cheat sheet here for all the class related syntax that the students will be exposed to/ or will have to use in this lab session - we may try to be consistent by drawing some of these from the document shared on LMS (lecture 7.1)]

\*\* runtime to be populated by the number of iterations taken for the function to complete (number of iterations of the loops, currently working by initialising a counter that counts the number of times a function is being called)

\*\* maxspace to be populated by the number of variables created in the function call (global variable to be updated for recursive calls)