Bubble sort:

The inner loop is to go through the list, select a pair of adjacent elements to compare and possibly swap if the second element is less than the first one. The outer loop will then increment to the next index as the starting point and begin the next iteration. So let’s say in the first outer loop, it will make sure all element starting with an even index have the correct position compared to the other element within the pair. The next outer loop will have all element starting with an odd index move to the correct relative position compared to the other element in the pair. That way, after all outer loop iteration, the list is sorted.

Insertion sort:

Insertion sort divides the input array into 2 parts: a sorted and a unsorted part. Variable i will be used to keep track of the sorted element in the array. And j will be the index of the element that needs to be sorted. Insertion sort assume the first element is sorted so j start at 1 and the while loop will keep swapping the jth element to all the elements before it until it finds the correct position for the jth element. If before it, no element is smaller than the jth element, j will reach 0 and the element will be stored at the first index, index 0 or j = 0, the while loop then breaks and it will go to another iteration with the outer for loop.

Run with pytest, all test passes, so our implementation of the two sorts are correct.

Run timeit script, as you can see from the pattern, as the size of the array increases, the time consumption of Bubble sort increases most significantly, our builtin python function is much faster than the other two sort function. Insertion sort hit second because it is a bit faster than Bubble sort.