## Chapter 5.2: Searching and Sorting - Insertion Sort

#### Refresh

In the last lesson we looked at a Bubble Sort and you might want to refresh your memory by watching <u>this video</u>. We stated that the Bubble Sort is not an efficient algorithm can you remember why?

### **Objectives**

- You should be able to explain how an Insertion Sort works.
- You should be able to write a simple program in Python that performs an Insertion Sort on a simple set of items.

#### Introduction

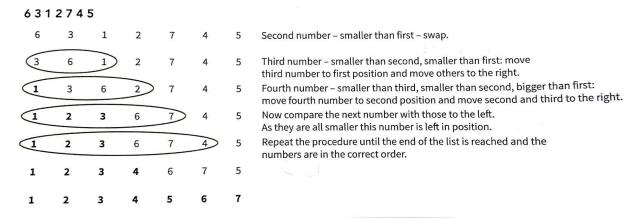
An Insertion Sort works by examining each item in turn and inserting it into the right position.

The steps of the Insertion Sort are:

- 1. IF there is only one item STOP
- 2. Look at the second item and compare it with the first
- 3. IF it is greater than the first then leave it in place ELSE IF it is less than swap the two numbers
- 4. Check the third number. If it is smaller than the second one then compare it with the first one. IF it is smaller than the first, place it in this position by moving the first two number along.
- 5. REPEAT this procedure with all the numbers by comparing them with the numbers to the left until a smaller number is found. IF a smaller number is found then place the number in the position to the right of it by moving the other numbers along.

## WORKED EXAMPLE

Using insertion sort, put this list in ascending order.



Watch this video to see how the Insertion Sort works.

This <u>animation</u> goes through the algorithm step by step.

#### **Pseudo Code Examples**

Below is one way to program the algorithm in Python. It is just the **pseudo code** and not the actual Python code.

## **Chapter 5.2: Searching and Sorting - Insertion Sort**

```
1
     Procedure InsertionSort (List, First, Last)
2
         For CurrentPointer <- First + 1 To Last
             CurrentValue <- List[CurrentPointer]
3
             Pointer <- CurrentPointer - 1
 4
 5
             While List[Pointer] > CurrentValue AND Pointer > 0
                 List[Pointer+1] <- List[Pointer]
6
7
                 Pointer <- Pointer - 1
8
             EndWhile
9
             List[Pointer+1] <- CurrentValue
10
         EndFor
11
    EndProcedure
```

Task 1 Produce a table showing the steps needed to sort the following numbers into ascending order using the insertion sort method. 20, 15, 3, 13, 9, 2, 6.

| Position 1 | Position 2 | Position 3 | Position 4 | Position 5 | Position 6 | Position 7 |
|------------|------------|------------|------------|------------|------------|------------|
| 20         | 15         | 3          | 13         | 9          | 2          | 6          |
| 15         | 20         | 3          | 13         | 9          | 2          | 6          |
| 3          | 15         | 20         | 13         | 9          | 2          | 6          |
| 3          | 13         | 15         | 20         | 9          | 2          | 6          |
| 3          | 9          | 13         | 15         | 20         | 2          | 6          |
| 2          | 3          | 9          | 13         | 15         | 20         | 6          |
| 2          | 3          | 6          | 9          | 13         | 15         | 20         |

A teacher has sorted the test results for a particular class in an array. Write the code that would sort the list into ascending order using an insertion sort

Link to code or screen shot of annotated code

Insertion sort has a list of sorted and unsorted numbers if there is more than one number is a list. In the code you can see a lot about the position of elements because elements are moved into the sorted list and sorted until eventually everything is sorted.

# Chapter 5.2: Searching and Sorting - Insertion Sort