Sorting: Bubble Sort and Selection Sort

1. Bubble Sort

> **The rule.** Walk from left to right through the list. At each position, compare the *two neighbours*. If the left one is bigger, **swap** them. One full walk across the list is called a **Interation**.

Starting list:

5	1	4	2	8

a. Complete **Interation 1**. The first row is done for you.

Compare	Swap?	List after this comparison				
5 and 1	yes	1	5	4	2	8
5 and 4	yes / no					
5 and 2	yes / no					
5 and 8	yes / no					

b. Now write the list at the end of each remaining Interation.

End of Interation 2:					
End of Interation 3:					
End of Interation 4:					

c. After Interation 3 the list was already sorted, but the algorithm still ran Interation 4. For a list of n elements, how many Interationes does this version do?

Number of Interations: _____

2. Selection Sort

> **The rule.** Look at the *unsorted* part of the list (everything from position i to the end). Find the **smallest** value in it, and **swap** it into position i. Then move on to position $i + 1$.

Starting list:

7	3	9	1	5

a. Complete the table. Shaded boxes mean "already sorted — do not touch". The first row is done for you.

Step	Unsorted part	Smallest value in it	List after the swap
1	7 3 9 1 5	1	1 | 3 | 9 | 7 | 5
2	3 9 7 5	_____	1 | | | |
3	_____	_____	1 | | | |
4	_____	_____	1 | | | |

b. In Step 2 the smallest value was *already* in the right place. What does the algorithm do in that case?

c. Fill in the two blanks so this code matches the rule in the grey box above.

```
def selection_sort(lst):
    for i in range(len(lst)):
        smallest = i
        for j in range(_____________, len(lst)):
            if lst[j] < lst[smallest]:
                smallest = _____________
        lst[i], lst[smallest] = lst[smallest], lst[i]
    return lst
```

Hint: why does the inner loop *not* start at 0?