**Swap**

Suppose we have a list

\[ L = [0, 4, -3, 10] \]

**swap(0,2,L)** should swap

the 0th and 2nd elements

of \( L \):

\[ L = [-3, 4, 0, 10] \]

An incorrect implementation:

```python
def swap(i, j, L):
    L[i] = L[j]
    L[j] = L[i]
```

Suppose \( L = [0, 1] \), what happens if we call \( \text{swap}(0, 1, L) \):

- **Correct implementation (use an extra variable)**

```python
+mp = L[i]
L[i] = L[j]
L[j] = +mp
```
Bubble sort

Idea: Compare pairs & swap if they are out of order

\[ 10 \ 4 \ 3 \ \phi \]

Iteration #1: \[ 10 \ 4 \ 3 \ \phi \]
\[ 4 \ 10 \ 3 \ \phi \]
\[ 4 \ 3 \ 10 \ \phi \]
\[ 4 \ \phi \ 3 \ 10 \]

Iteration #2: \[ 3 \ 4 \ \phi \ 10 \]
\[ 3 \ \phi \ 4 \ 10 \]

Iteration #3: \[ \phi \ 3 \ 4 \ 10 \]
\[ \phi \ 3 \ \phi \ 4 \ 10 \] done!

```python
def bubble_sort(L):
    for i in range(len(L)):
        for j in range(1, len(L)):
            if L[j-1] > L[j]:
                swap(j-1, j, L)
```

We can improve performance by not going through the whole list each iteration. The largest value will be at the end of the sublist.