Selection Sort
### Selection Sort

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>11</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

What do we do first?
Selection Sort

Find minimum element idx between start to end

What next?
Selection Sort

Swap the elements at start and minValIdx

What next?
Selection Sort

0 4 3 10 11 8

Decrease the interval.

What next?
Selection Sort

Find minimum element between start to end

What next?
Selection Sort

Start: 1
MinValIdx: 2

Swap the elements at start and minValIdx

What next?
Selection Sort

Decrease the interval.

What next?
Selection Sort

Find minimum element idx between start to end

What next?
## Selection Sort

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>10</td>
<td>11</td>
<td>8</td>
</tr>
</tbody>
</table>

- **start**: 2
- **minValIdx**: 2

Swap the elements at start and minValIdx

What next?
Selection Sort

0  1  2  3  4  5

0  3  4  10  11  8

Decrease the interval.

What next?
Selection Sort

Find minimum element idx between start to end

What next?
Selection Sort

Swap the elements at start and minValIdx

What next?
Selection Sort

Decrease the interval.

What next?
Selection Sort

Find minimum element idx between start to end

What next?
Selection Sort

0  3  4  8  10  11

Swap the elements at start and minValIdx

What next?
Selection Sort

Decrease the interval.

We’re done!
Selection sort

selectionSort(L):
    for startIdx in range(len(L)):
        minIdx = findMinimum(startIdx, L)
        swap(startIdx, minIdx, L)
Selection sort

findMinimum(startIdx, L):

    minIdx = startIdx

    for i in range(startIdx, len(L)):
        if L[i] < L[minIdx]:
            minIdx = i

    return minIdx
Selection sort

swap(i, j, L):

temp = L[i] # step 1
L[i] = L[j]    # step 2
L[j] = temp # step 3
Write a function that swaps two elements

NOTE:
The if statement at the bottom only executes if we run this file from the command line, e.g.

```python
>>> python3 swap.py
Before: [0, 1]
After: [1, 0]
```

This feature allows us to use the functions in this file in other programs using the syntax:

```python
import swap
swap.swap(0, 1, L)
```

```python
def swap(i, j, L):
    # Swaps the i-th and j-th elements of L
    # Params i, j (int): indexes into L
    # Param L (list): the list to change
    # Returns: None
    tmp = L[j]
    L[j] = L[i]
    L[i] = tmp
```

```python
if __name__ == '__main__':
    L = [0, 1]
    print("Before:", L)
    swap(0, 1, L)
    print("After:", L)
```
Write a function that checks if a list is sorted from least to greatest

```python
$ python3 isSorted.py
[0, 2, 4, 6] isSorted? True
[6, 2, 4, 0] isSorted? False
[10] isSorted? True

---

import random

def isSorted(L):
    """
    Returns True if the list L is sorted; False otherwise
    Param L (list): the list to test
    Return (bool)
    """
    for i in range(len(L)-1):
        if L[i] > L[i+1]:
            return False
    return True

if __name__ == '__main__':
    L = list(range(0,8,2))
    print(L, 'isSorted?', isSorted(L))
    random.shuffle(L)
    print(L, 'isSorted?', isSorted(L))
    L = [10]
    print(L, 'isSorted?', isSorted(L))
```
Sort a list in place using selection sort.

Use your existing implementation for swap and isSorted!

```python
# python3 selectionSort.py
Before: [10, 4, 3, 0, 11, 8]
swap 0 3
swap 1 2
swap 2 2
swap 3 5
swap 4 5
swap 5 5
After: [0, 3, 4, 8, 10, 11] isSorted? True
```