Algorithm: Selection Sort

Selection Sort

Key Aspects:

- In-place comparison sort.
- The algorithm divides the list into two portions: sorted and unsorted.
- Selects the minimum element and moves it to the front
- Inefficient for large lists.

The list is sorted in ascending order

Algorithm:

- Start from the first element in the list.
- Find the minimum element in the list.
- Swap it with the first element.
- The sorted portion now has one element.
- Move to the next index and repeat the previous steps, adding the minimum element found in the unsorted portion as the last element of the sorted portion of the list.

Time Complexity:

- Worst-Case Time Complexity: Quadratic O(n^2)
- Average-Case Time Complexity: Quadratic O(n^2)
- Best-Case Time Complexity: Quadratic O(n^2)

Selection Sort = Select Minimum

Selection Sort

Code:

```
def selection_sort(lst):
    # Traverse the list.
    # i represents the index where the unsorted portion of the list starts.
    for i in range(len(lst)):
        # The min_index variable describes the index of the
        # smallest element in the remaining unsorted array.
        min_index = i
        # For each element in the unsorted portion of the list,
        # check if the element is smaller than the current min
        # and assign that index as the new min_index.
        for curr_index in range(i+1, len(lst)):
            if lst[min_index] > lst[curr_index]:
                min_index = curr_index
        # Swap the min element with the first element of the
        # unsorted portion of the list.
        lst[i], lst[min_index] = lst[min_index], lst[i]
```

Example:

```
>>> selection_sort([5, 1, 3])
```

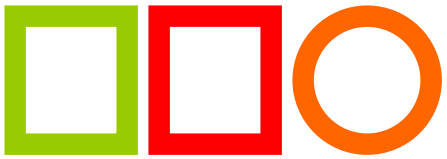
```
==========> Outer Loop iteration #1

List: [5, 1, 3]
Sorted portion: []
Unsorted portion: [5, 1, 3]
The unsorted portion starts at index: 0

--> Inner Loop iteration
Current element: 1
Min element so far: 5
Is the current element smaller than the min element? Yes
1 is now the new min element. It is located at index: 1

--> Inner Loop iteration
Current element: 3
Min element so far: 1
Is the current element smaller than the min element? No
No need to change the min element

-> Out of inner loop
Previous list: [5, 1, 3]

Swapping the first element in the unsorted portion: 5
With the min element found: 1
New list: [1, 5, 3]
```

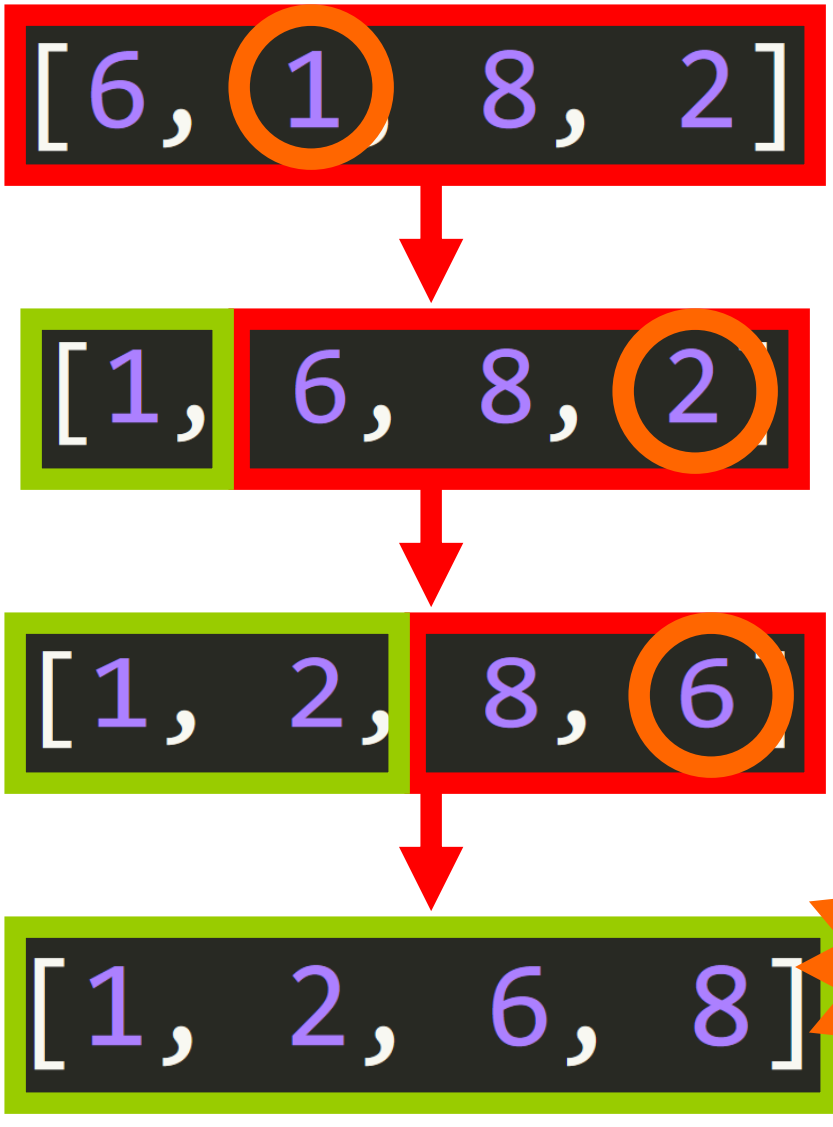
```
==========> Outer Loop iteration #2

List: [1, 5, 3]
Sorted portion: [1]
Unsorted portion: [5, 3]
The unsorted portion starts at index: 1

--> Inner Loop iteration
Current element: 3
Min element so far: 5
Is the current element smaller than the min element? Yes
3 is now the new min element. It is located at index: 2

-> Out of inner loop
Previous list: [1, 5, 3]

Swapping the first element in the unsorted portion: 5
With the min element found: 3
New list: [1, 3, 5]

==========> Outer Loop iteration #3

The list is now sorted!
[1, 3, 5]
```

Selection Sort

Example:

```
>>> selection_sort([6, 1, 8, 2])
```





Selection Sort

Example:

```
>>> selection_sort([6, 1, 8, 2])
```

```
==========> Starting Selection Sort <==========


=========> Outer Loop iteration #1

List: [6, 1, 8, 2]
Sorted portion: []
Unsorted portion: [6, 1, 8, 2]
The unsorted portion starts at index: 0

--> Inner Loop iteration
Current element: 1
Min element so far: 6
Is the current element smaller than the min element? Yes
1 is now the new min element. It is located at index: 1

--> Inner Loop iteration
Current element: 8
Min element so far: 1
Is the current element smaller than the min element? No
No need to change the min element

--> Inner Loop iteration
Current element: 2
Min element so far: 1
Is the current element smaller than the min element? No
No need to change the min element

-> Out of inner loop
Previous list: [6, 1, 8, 2]

Swapping the first element in the unsorted portion: 6
With the min element found: 1
New list: [1, 6, 8, 2]
```

Selection Sort

Example:

```
=========> Outer Loop iteration #2

List: [1, 6, 8, 2]
Sorted portion: [1]
Unsorted portion: [6, 8, 2]
The unsorted portion starts at index: 1

--> Inner Loop iteration
Current element: 8
Min element so far: 6
Is the current element smaller than the min element? No
No need to change the min element

--> Inner Loop iteration
Current element: 2
Min element so far: 6
Is the current element smaller than the min element? Yes
2 is now the new min element. It is located at index: 3

-> Out of inner loop
Previous list: [1, 6, 8, 2]

Swapping the first element in the unsorted portion: 6
With the min element found: 2
New list: [1, 2, 8, 6]

=========> Outer Loop iteration #3

List: [1, 2, 8, 6]
Sorted portion: [1, 2]
Unsorted portion: [8, 6]
The unsorted portion starts at index: 2

--> Inner Loop iteration
Current element: 6
Min element so far: 8
Is the current element smaller than the min element? Yes
6 is now the new min element. It is located at index: 3

-> Out of inner loop
Previous list: [1, 2, 8, 6]

Swapping the first element in the unsorted portion: 8
With the min element found: 6
New list: [1, 2, 6, 8]

=========> Outer Loop iteration #4

The list is now sorted!
[1, 2, 6, 8]
```