Selection Sort

Sorting a list

- Why is sorting important?
  - Binary Search
  - Finding median
  - Checking for duplicates
  - Building a frequency table of values

Strategy 1: SELECTION SORT

SWAPPING

```
graph TD
New_pile --> Find_Minimum
Find_Minimum --> Swap_to_first_min_element_first_position
Swap_to_first_min_element_first_position
 --> Swap_second_min_element_second_position--> continue
```

```
def selectionSort(L):
  n = len(L)
  if n<1:
    return(L)
  for i in range(n):
    # Assume L[:i] is sorted
    mpos = i
    # mpos = position of minimum L[i:]
    for j in range(i+1,n):
      if L[j]<L[mpos]:
        mpos = j

  (L[i],L[mpos]) = (L[mpos],L[i])
```

```
    # Now L[:i+1] is sorted
    return(L)
```

Outer loop iterates: `n times`

Inner loop iterates: `n-i` times

- Efficiency : `T(n) = n + n-1 + —— + 1`
- Total : `T(n)` = `n(n+1)`
- `O(n^2)` : In all cases we go n square