Selection Sort Algorithm Analysis

```
def FindSmallest(items):
```
running time is linear on the number of elements in the list, i.e. Θ(n)

```
    """ Finds the smallest element in the list.

    pre: items is a list of integers."""
    s = 0
    i = 0
    while i < len(items)-1:
        if items[s] > items[i]:
            s = i
        i +=1
    return s

def SelectionSort(items):
    sorted = 0
    n = len(items)
    while sorted < n-2:
        m = FindSmallest(items[sorted:n])

        tmp = items[sorted]
        items[sorted]= items[m+sorted]
        items[m+sorted]=tmp
        sorted += 1
```

n-1 iterations

running time is Θ(n), but we realize that the list will be reducing by 1 with each iteration of the `while` loop, therefore let's do summation (see below)

1st iteration of the while loop: ***n*** iterations in the FindSmallest's while loop,
2nd iteration of the while loop: ***n-1*** iterations in the FindSmallest's while loop,
3rd iteration of the while loop: ***n-2*** iterations in the FindSmallest's while loop,
….
n-2nd iteration of the while loop: ***3*** iterations in the FindSmallest's while loop,
n-1st iteration of the while loop: ***2*** iterations in the FindSmallest's while loop.

Therefore, we have *n+(n-1)+(n-2)+...+3+2* = $(n-1)\frac{(n+2)}{2}$, which is **Θ(n²)**

Comment: *2, 3, 4, .. n* is a *finite arithmetic progression*, the sum of k elements of arithmetic progression, with a_1 being first element an a_k being last element is $k\frac{(a_1+a_k)}{2}$

The complexity of the algorithm is **Θ(n²)**

Timing the Selection Sort Algorithm:

It took 15.011640071868896 seconds to complete the Selection Sort of 10,000 values.
It took 1800.7854030132294 seconds to complete the Selection Sort of 100,000 values.

The second value is predictable if we recall that the complexity is quadratic, i.e. T(n) is Θ(n²). It means that if it took about 15 seconds for the algorithm to sort 10,000 elements, it will take about 15*100 = 1500 seconds (or 25 minutes) to sort 100,000 elements.
Explanation: $dn^2+en+f < T(n) < an^2+bn+c$, if we simplify it:
$dn^2 < T(n) < an^2$, then use n=10,000:
$d(10{,}000)^2 < 15 < a(10{,}000)^2$, then multiply all sides by 100:
$d(10{,}000)^2 * 100 < 15*100 < a(10{,}000)^2 * 100$, re-writing:
$d(100{,}000)^2 < 1500 < a(100{,}000)^2$ – running time estimate for 100,000 elements.
Moreover: Following these calculations, it will take about 150,000 seconds (or 2500 minutes, or about 42 hours) to sort 1,000,000 elements!!!

What can we possibly do to improve the performance of the Selection Sort?
- use built-in method **min** to finding the smallest element.

```
def SelectionSort2(items):
    sorted = 0
    n = len(items)
    while sorted < n-2:
        smallest = min(items[sorted:])
        m = items[sorted:].index(smallest)

        tmp = items[sorted]
        items[sorted]= items[m+sorted]
        items[m+sorted]=tmp
        sorted += 1
```

Result: dramatically improved running time!

It took 2.632875919342041 seconds to complete the Selection Sort of 10,000 values.
2.6 seconds in comparison with 15 seconds (previous time, before the change)!!!

It took 443.61888813972473 seconds to complete the Selection Sort of 100,000 values.
444 seconds in comparison with 1801 seconds!