rtgoSni

Sorting

Real life examples of sorting/searching…

- Investigating flights– we may get a long list… we can choose sorting by
  - Price
  - Departure time
  - Arrival time
  - Duration

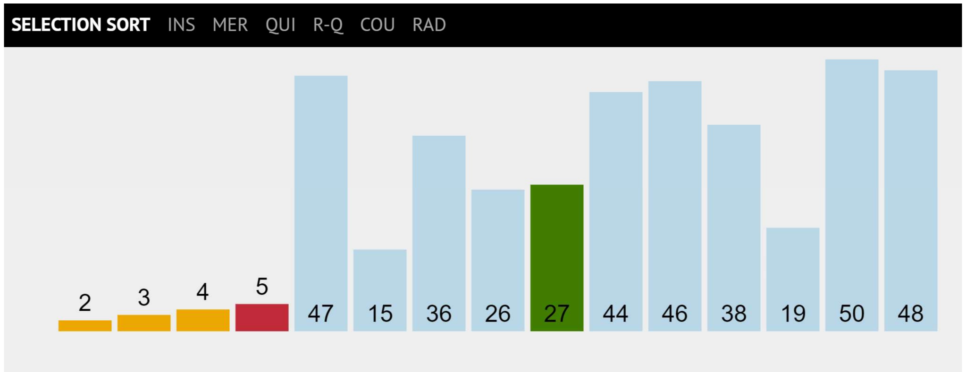
- Finding a book in a library (physical or online)

There are many sorting methods!!!

Selection sort

- Let’s first visualize the algorithm!!

Selection sort



https://visualgo.net/bn/sorting

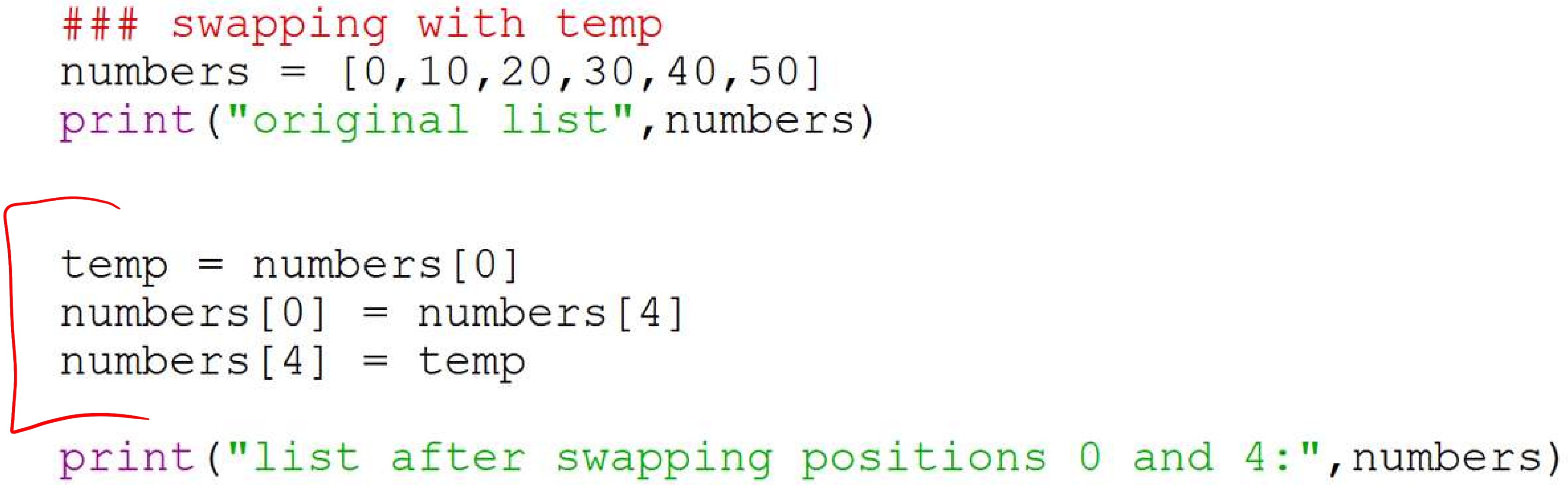
Before looking into the selection sort code, let’s look into/recap …

- Swapping two elements in a list

- Visiting elements in sublists

Swapping based Sort algorithms:

First: about swapping



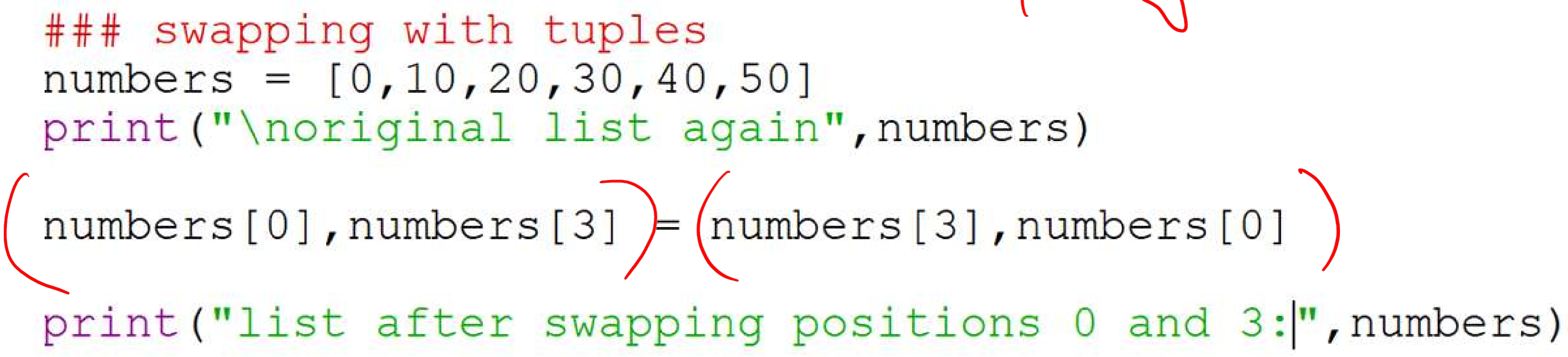
```
### swapping with temp
numbers = [0,10,20,30,40,50]
print("original list",numbers)

temp = numbers[0]
numbers[0] = numbers[4]
numbers[4] = temp

print("list after swapping positions 0 and 4:",numbers)
```

In Python, swapping can be done with tuples also

Pythony!

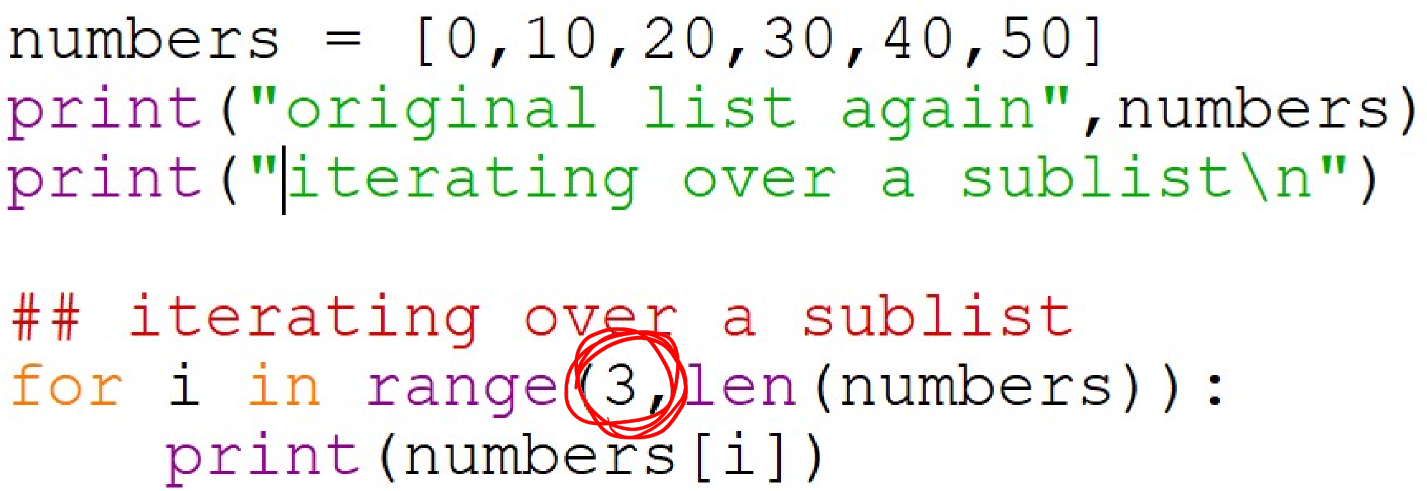


```
### swapping with tuples
numbers = [0,10,20,30,40,50]
print("\noriginal list again",numbers)

numbers[0],numbers[3] = numbers[3],numbers[0]

print("list after swapping positions 0 and 3:",numbers)
```

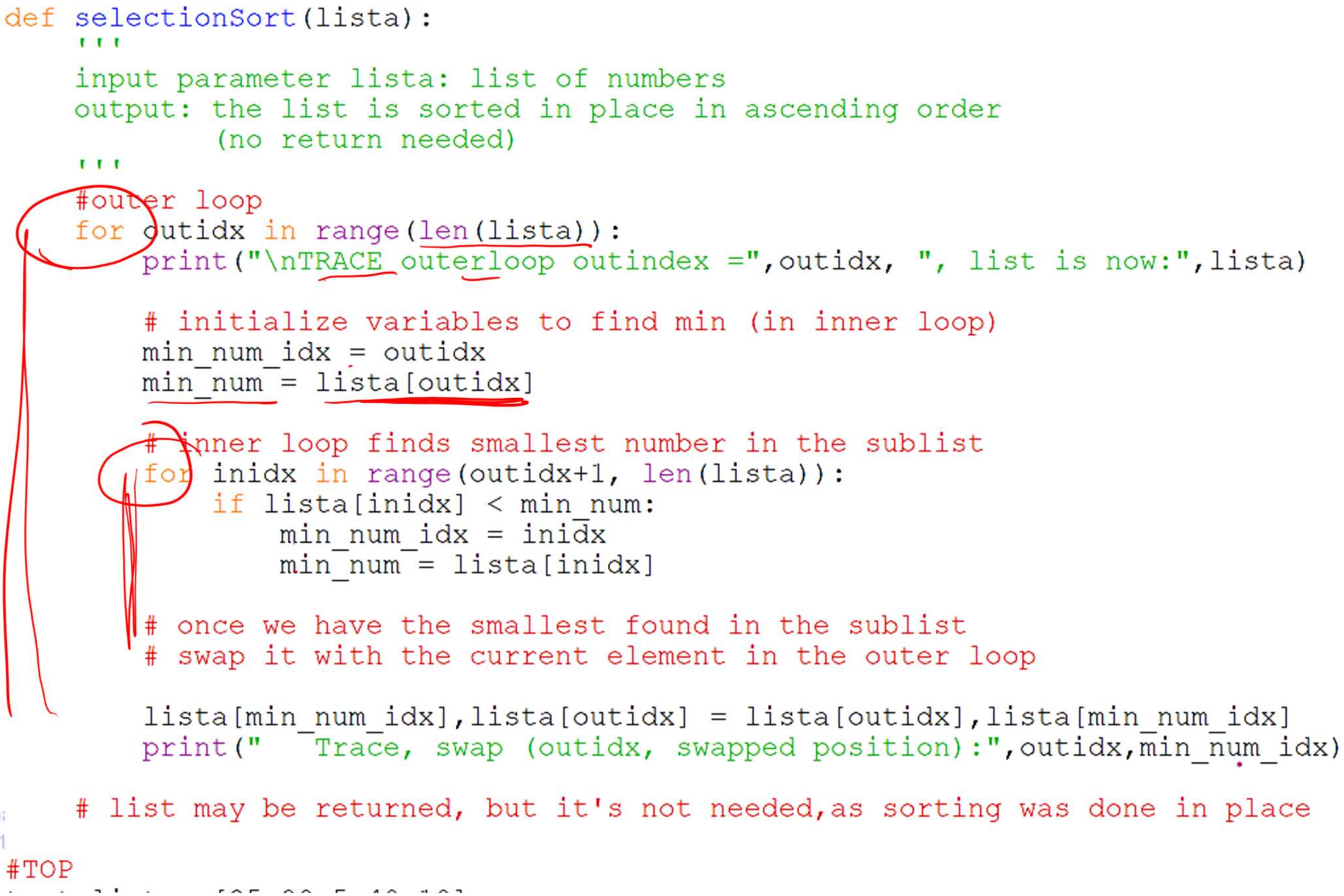
Let’s now recall about: iterating over a sublist



```
numbers = [0,10,20,30,40,50]
print("original list again",numbers)
print("iterating over a sublist\n")

## iterating over a sublist
for i in range(3,len(numbers)):
    print(numbers[i])
```

- Home → class notes and materials → Week 13
  - Sel_sort_PLAN.py



```
def selectionSort(lista):
    '''
    input parameter lista: list of numbers
    output: the list is sorted in place in ascending order
            (no return needed)
    '''
    #outer loop
    for outidx in range(len(lista)):
        print("\nTRACE outerloop outindex =",outidx, ", list is now:",lista)

        # initialize variables to find min (in inner loop)
        min_num_idx = outidx
        min_num = lista[outidx]

        # inner loop finds smallest number in the sublist
        for inidx in range(outidx+1, len(lista)):
            if lista[inidx] < min_num:
                min_num_idx = inidx
                min_num = lista[inidx]

        # once we have the smallest found in the sublist
        # swap it with the current element in the outer loop

        lista[min_num_idx],lista[outidx] = lista[outidx],lista[min_num_idx]
        print("   Trace, swap (outidx, swapped position):",outidx,min_num_idx)

    # list may be returned, but it's not needed,as sorting was done in place

#TOP
```

```
def selectionSort(lista):
    '''
    input parameter lista: list of numbers
    output: the list is sorted in place in ascending order
            (no return needed)
    '''
    #outer loop
    for outidx in range(len(lista)):
        print("\nTRACE outerloop outindex =",outidx, ", list is now:",lista)

        # initialize variables to find min (in inner loop)
        min_num_idx = outidx
        min_num = lista[outidx]

        # inner loop finds smallest number in the sublist
        for inidx in range(outidx+1, len(lista)):
            if lista[inidx] < min_num:
                min_num_idx = inidx
                min_num = lista[inidx]

        # once we have the smallest found in the sublist
        # swap it with the current element in the outer loop

        lista[min_num_idx],lista[outidx] = lista[outidx],lista[min_num_idx]
        print("   Trace, swap (outidx, swapped position):",outidx,min_num_idx)

    # list may be returned, but it's not needed,as sorting was done in place

#TOP
test_list = [25,89,5,40,10]
print("\noriginal list",test_list)
selectionSort(test list)
```

Run (by hand, executing with tracing) selection sort in border cases

- What happens if the original list is already sorted?
- What happens if the original list is sorted in reverse order?

Merge Sort

- Merge sort general IDEA:

- It is intrinsically a recursive algorithm.
  - (of course it can be implemented iteratively also!)

- This algorithm illustrates at the same time another example of sorting, and of recursion.

Merge sort – Thinking recursively

Recursive part:

Partition the list in half

Sort each half

Combine (merge keeping order) the sorted halves.

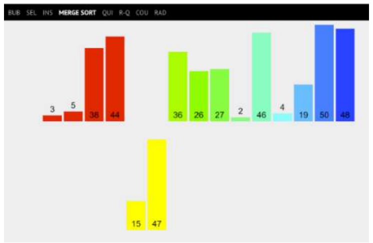
Base case:

Merge sort “unfolding the recursion”

Partition the list in half, and going down the recursion each half is again partitioned in half, and again and again… until reaching the base case.

As the recursion comes up, in each level, combine the already sorted halves (from the level below), and return that to the upper level

https://visualgo.net/bn/sorting



Mergesort: pseudocode

```
def mergeSort(alist):
    if (alist has 2 or more elements):
        sortedLeft = mergeSort(left half of alist)
        sortedRight = mergeSort(right half of alist)
        result = merge(sortedLeft , sortedRight)
    else:
        result = alist    # 1 element only or empty,
                          # the list is already sorted
    return result
```

Upcoming:

- Mergesort
- How does it compare with selection sort?
- Time complexity

To-do's!

- Keep working on project!
- Rubric TBA
- Work on remaining mock up questions or other practice!
- Latest topics: run different cases, normal and border cases
- Readings: useful! But especially latest topics, you will not be tested more than level in class
- Official survey: 23% as of this morning
- End of sem survey, after project brief reflection: TBP

- Consult if in doubt (extra off hs)!