# Last class this semester!

## Plan wk 13.2 (1 hour)

- Announcements
- Measuring execution time
  - Built-in sort vs. our sort
- Time Complexity
- Notion of Analysis of algorithms

- Grand finale:
  - how various sorts compare

## Announcements:

- Check announcement sent via Canvas (last night)
- **Keep checking announcements and check/ask in the discussion forum these days!**
- Off hs
  - This week TAs additional
  - Next week Monday consultation session

## How do we measure execution time of algorithms??

## Measuring execution time of /comparing time efficiency of algorithms?

- $\rightarrow$ Timing execution of algorithms
- Visualizing execution of algorithms
- …

## Investigating… Which one is faster??

**Python built-in sort ( mainly mergesort)**
**vs.**
**(our) selection sort**

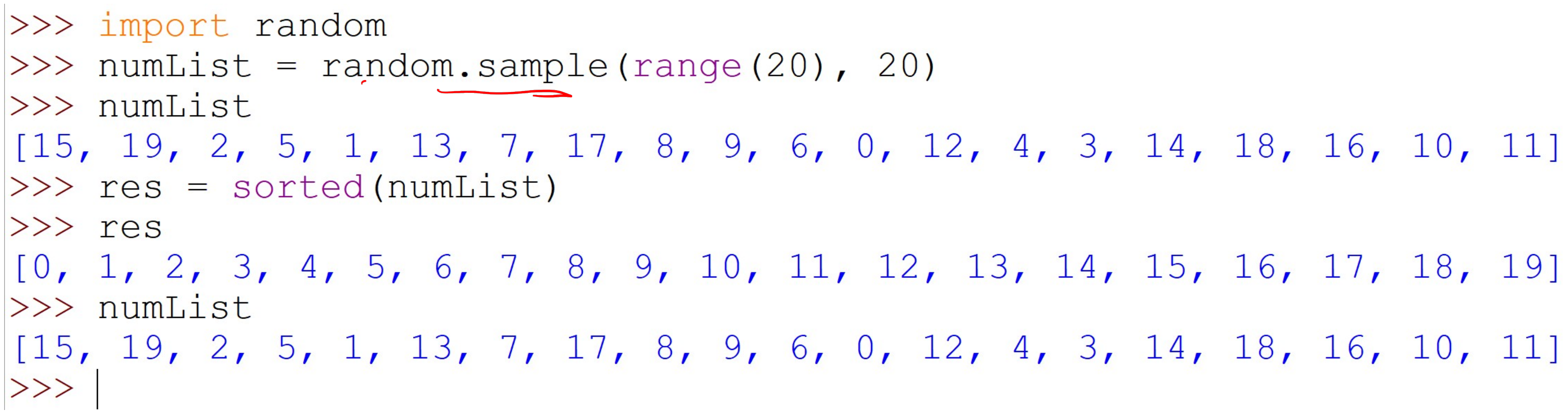
## Built-in sort in Python.

- “Tim’s sort”: mergesort combined with insertion-sort

```
result = sorted(original_list)
```

Function sorted (…) creates a new list and returns it. It does NOT modify the original list.

## Sorting a random list with built-in Python function sorted(…)



```
>>> import random
>>> numList = random.sample(range(20), 20)
>>> numList
[15, 19, 2, 5, 1, 13, 7, 17, 8, 9, 6, 0, 12, 4, 3, 14, 18, 16, 10, 11]
>>> res = sorted(numList)
>>> res
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
>>> numList
[15, 19, 2, 5, 1, 13, 7, 17, 8, 9, 6, 0, 12, 4, 3, 14, 18, 16, 10, 11]
>>> 
```

## Timing programs

```
import time

start = time.time()
#   … this is what we want to measure
end = time.time()
diff_in_secs = (end-start+1)

print(diff_in_secs)
```

## Which one is faster??

```
Which one is faster?? Built-in sorted(...)  or our selection sort?

Ok to continue? (y/n) --> y
10 ITEMS...

[4, 1, 6, 7, 2, 8, 5, 3, 0, 9]
[4, 1, 6, 7, 2, 8, 5, 3, 0, 9]

'sorted(..)' 1.0 seconds
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

our selection sort 1.0 seconds
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]


Ok to continue? (y/n) -->
Ok to continue? (y/n) --> y
100 ITEMS...
'sorted(..)' 1.0 seconds
our selection sort 1.00099778175354 seconds

Ok to continue? (y/n) --> y
1000 ITEMS...
'sorted(..)' 1.0 seconds
our selection sort 1.0388801097869873 seconds
```

## Empirically observed …

The time to sort a list (in seconds)…

| Number Items in the list | Python built-in sort (Merge sort + improvements) | Our selection sort |
|---|---|---|
| 10 | <=1 | <=1 |
| 100 | <=1 | 1.0009977… |
| 1,000 | <=1 | 1.0388801… |
| 10,000 | 1.00199389… | 3.89128184… |
| 30,000 | 1.0010461… | 22.4499542… |
| 50,000 | 1.0273454… | 67.788336… |
| 100,000 | 1.0447697… | 300.678108… |

## Best way to measure and compare algorithms?

- Timing algorithms
- Visualizing algorithms
- A theoretical measure: TIME COMPLEXITY