Final project: Due Saturday, 11:55pm.

Final exam: Monday, April 29, 5:30pm.

Location: FJ-B

- **In pairs**, use the index cards to devise a sorting algorithm: an algorithm that takes a list of mixed-up numbers and puts them in sorted order, from lowest to highest.
- **In groups of 4**, discuss what each pair came up with. Are your ideas similar or different?
- As a group of 4, write down your algorithm on paper.
  - Your algorithm can re-arrange the elements of the mixed-up list, or copy them into a new list.
- In pairs, write your algorithm in Python.

Insertion Sort

For every item in L (except first), from left to right:

- find new position for item between 0 and item’s old position
- slide all items from 0 to newpos-1 to the right one slot
- put item into its new position

[0]	[1]	[2]	[3]	[4]	
8	7	5	6	3	Item = 7
7	8	5	6	3	Item = 5
5	7	8	6	3	Item = 6
5	6	7	8	3	Item = 3
3	5	6	7	8	

Insertion Sort

```
for oldpos in range(0, len(L)):
  item = L[oldpos]
  newpos = 0
  while L[newpos] > L[oldpos]
    newpos = newpos + 1
  slide_right(L, newpos, oldpos – 1)
  L[newpos] = item
```

Selection Sort

```
run loop from pos=0 to pos=len(L)-2
    find smallest item in L[pos : len(L)], let that
            position be called small_position
    swap L[pos], L[small_position]
```

[0]	[1]	[2]	[3]	[4]	
8	7	5	6	3	Smallest = 3
3	7	5	6	8	Smallest = 5
3	5	7	6	8	Smallest = 6
3	5	6	7	8	Smallest = 7
3	5	6	7	8	

Selection Sort

```
for pos in range(0, len(L) – 1):
    smallest_pos = pos
    for test_pos in range(pos, len(L)):
        if L[test_pos] < L[smallest_pos]:
            smallest_pos = test_pos
    swap L[pos], L[smallest_pos]
```

Bubble Sort

```
Loop:
    loop over positions in L from 0 to len(L) – 2:
        if L[pos] > L[pos + 1], then swap them
    do outer loop again if any swaps were made
```

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

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

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

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