Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.

Seoul National University

Jan. 6 – 17, 2020

Python for Data Analytics

Lists

Lists

- Ordered collections of arbitrary objects (arrays of object references)
- Accessed by offset (items not sorted)
- Variable-length, heterogeneous, mutable, and arbitrarily nestable
- The most versatile and popular data type in Python

```python
prime = [2, 3, 5, 7, 11]
a = [2, 'three', 3.0, 5, 'seven', 11.0]
b = [1, 3, 3, 3, 2, 2]
c = [ 1, [8, 9], 12]
emptylist = []
```

Getting the List Size

- ## len(*list*)

  - Return the number of elements in the list

  - Actually, `len()` tells us the number of elements of any sequence or collection (e.g., string, list, tuple, dict, set, …)

```
>>> greet = 'Hello Spam'
>>> print(len(greet))
10
>>> x = [ 1, 2, 'spam', 99, 'ham' ]
>>> print(len(x))
5
```

Useful Functions on a List

- *list.*count(*x*)
  - Return the number of times x appears in the list

- max(*list*)
  - Return the maximum value in the list

- min(*list*)
  - Return the minimum value in the list

```
>>> numbers = [3, 1, 12, 14, 12, 6, 1, 12]
>>> print(numbers.count(12))
3
>>> print(min(numbers), max(numbers))
1 14
```

Building a List

- *list.*append(*x*)
  - Add an item to the existing list
  - The list stays in order and new elements are appended at the end of the list

```
>>> menu = list()
>>> print(menu)
[]
>>> menu.append('spam')
>>> menu.append('ham')
>>> menu.append('spam')
>>> print(menu)
['spam', 'ham', 'spam']
```

Concatenating/Replicating Lists

- *list1* **+** *list2* : create a new list by adding two existing lists together
- *list* ***** *n*: create a new list by replicating the original list *n* times

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print(c)
[1, 2, 3, 4, 5, 6]
>>> d = a*3
>>> print(d)
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Extending Lists

■ *list.*extend(*list2*)

- Extend the list by appending all the items from the other list
- Faster than a series of append()'s

```
>>> menu = ['spam', 'ham']
>>> menu.extend(['egg', 'sausage'])
>>> print(menu)
['spam', 'ham', 'egg', 'sausage']
>>> menu.extend(['spam']*3)
>>> print(menu)
['spam', 'ham', 'egg', 'sausage', 'spam', 'spam', 'spam']
```

Referencing a List Element

- Just like strings, use an index specified in square brackets

Emma	Olivia	Ava	Isabella	Sophia
0	1	2	3	4

```
>>> names = ['Emma', 'Olivia', 'Ava', 'Isabella', 'Sophia']
>>> print(names[0])
Emma
>>> print(names[2])
Ava
```

Finding an Element

- *list.*`index`*(x[, start[, end]])*
  - Return zero-based index in the list of the first item whose value is equal to *x*
  - Error if there is no such item
  - The optional *start* and *end* arguments are used to limit the search to a particular subsequence of the list

```
>>> names = ['Emma', 'Olivia', 'Ava', 'Emma', 'Isabella']
>>> print(names.index('Emma')
0
>>> print(names.index('Emma', 1, 4))
3
>>> print(names.index('Isabella', 3))
4
```

Slicing a List

- *list*[*start* : *end* : *step*]

  - *start*: the starting index of the list
    - If omitted, the beginning of the list if *step* > 0 or the end of the list if step < 0

  - *end*: the ending index of the list (up to but not including)
    - If omitted, the end of the list if *step* > 0 or the beginning of the list if step < 0

  - *step*: the number of elements to skip + 1 (1 if omitted)

  - *start* and *stop* can be a negative number, which means it counts from the end of the array

Emma	Olivia	Ava	Isabella	Sophia
0	1	2	3	4
-5	-4	-3	-2	-1

```
names = ['Emma', 'Olivia', 'Ava', 'Isabella', 'Sophia']
```

Slicing a List: Example

Emma	Olivia	Ava	Isabella	Sophia
0	1	2	3	4
-5	-4	-3	-2	-1

```
>>> names = ['Emma', 'Olivia', 'Ava', 'Isabella', 'Sophia']
>>> print(names[::2])
['Emma', 'Ava', 'Sophia']
>>> print(names[3:])
['Isabella', 'Sophia']
>>> print(names[-3:])
['Ava', 'Isabella', 'Sophia']
>> print(names[::-1])
['Sophia', 'Isabella', 'Ava', 'Olivia', 'Emma']
```

Lists are Mutable

- *list[i] = x*: change the *i*-th element of the list to *x*
- *list[i:j] = list2*: replace the elements from *i*-th to *j-th* with the new list

```
>>> names = ['Emma', 'Olivia', 'Ava', 'Isabella', 'Sophia']
>>> names[2] = 'Eve'
>>> print(names)
['Emma', 'Olivia', 'Eve', 'Isabella', 'Sophia']
>>> names[1:4] = [ 'Charlotte', 'Mia', 'Amelia']
>>> print(names)
['Emma', 'Charlotte', 'Mia', 'Amelia', 'Sophia']
>>> names[5:] = [ 'Ella', 'Avery' ]
>>> print(names)
['Emma', 'Charlotte', 'Mia', 'Amelia', 'Sophia', 'Ella', 'Avery']
```

Membership Operators

- **in** (**not in**) operator
  - Check if an item is in a list or not
  - Returns True or False

```
>>> menu = ['spam', 'ham']
>>> 'spam' in menu
True
>>> 'ham' not in menu
False
>>> 'egg' in menu
False
```

Reversing the List

■ *list*.reverse()

- Reverse the elements of the list in place
- cf. *list*[::-1] returns the new list with the elements in reversed order

```
>>> names = ['Emma', 'Olivia', 'Ava', 'Isabella', 'Sophia']
>>> names.reverse()
>>> print(names)
['Sophia', 'Isabella', 'Ava', 'Olivia', 'Emma']
>>> new_names = names[::-1]
>>> print(new_names)
['Emma', 'Olivia', 'Ava', 'Isabella', 'Sophia']
```

Inserting an Element

- *list*.insert(*i, x*)
  - Insert an item at a given position *i* (0 means the front of the list)

```
>>> menu = ['spam', 'ham']
>>> print(menu)
['spam', 'ham']
>>> menu.insert(1, 'egg')
>>> print(menu)
['spam', 'egg', 'ham']
>>> menu.insert(0, 'bacon')
>>> print(menu)
['bacon', 'spam', 'egg', 'ham']
```

Removing Elements

■ *list*.remove(*x*)

- Remove the first item from the list whose value is equal to *x*

■ del *list*[*i*] or del *list*[*i:j*]

- Deletes *i*-th element (or from *i*-th to *j*-th elements) from the list

```python
>>> menu = ['spam', 'ham', 'egg', 'sausage', 'bacon']
>>> menu.remove('ham')
>>> print(menu)
['spam', 'egg', 'sausage', 'bacon']
>>> del menu[1:3]
>>> print(menu)
['spam', 'bacon']
```

Popping an Element

■ *list*.pop([*i*])

- Remove the item at the given *i*-th position in the list, and return it
- If no index is specified, it removes and returns the last item in the list

```
>>> menu = ['spam', 'ham', 'egg', 'sausage', 'bacon']
>>> print(menu.pop(1))
ham
>>> print(menu.pop())
bacon
>>> print(menu.pop())
sausage
>>> print(menu.pop())
egg
```

Using Lists as Stacks

- Stack: Last-In, First-Out

```
>>> stack = list()
>>> stack.append(6)
>>> stack.append(7)
>>> stack.append(2)
>>> print(stack)
[6, 7, 2]
>>> print(stack.pop())
2
>>> print(stack.pop())
7
```

Implementing Lists

- A list has an array of references to other objects
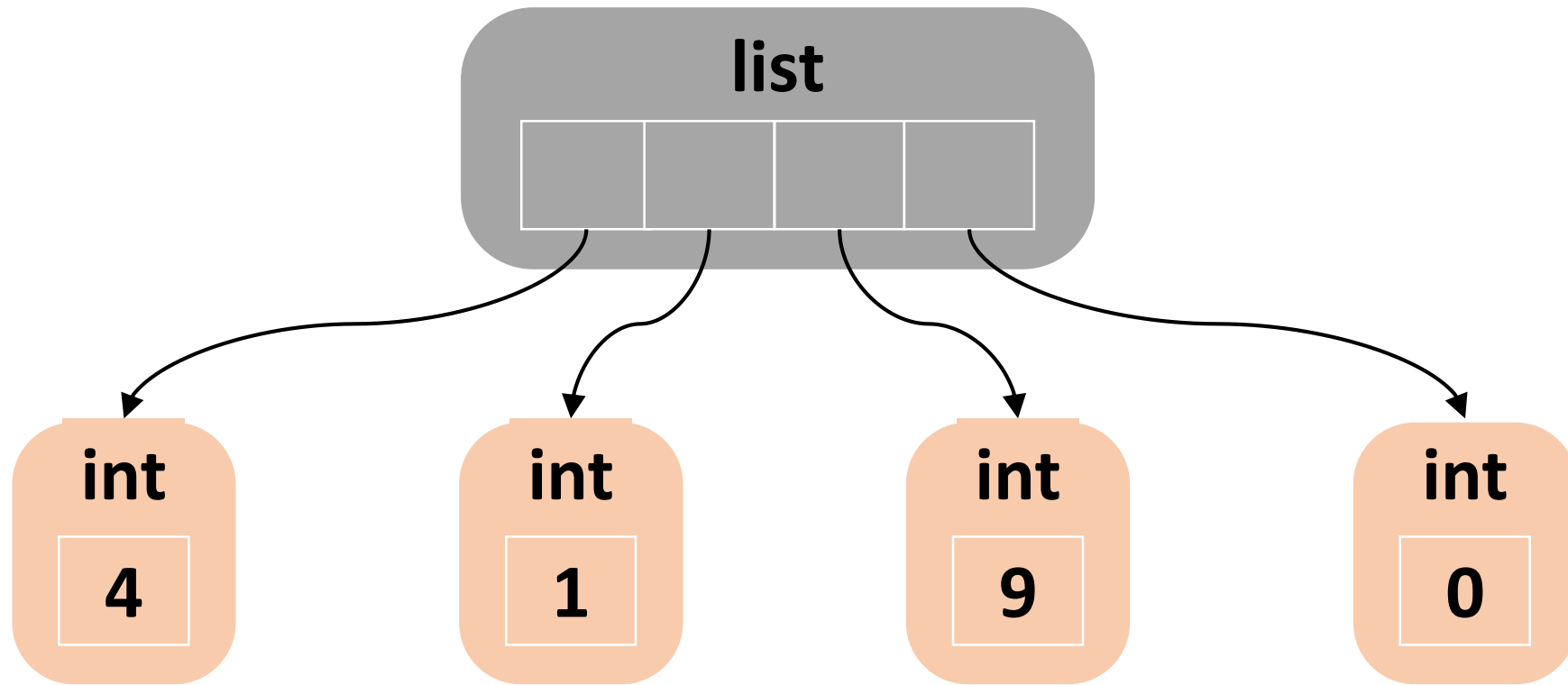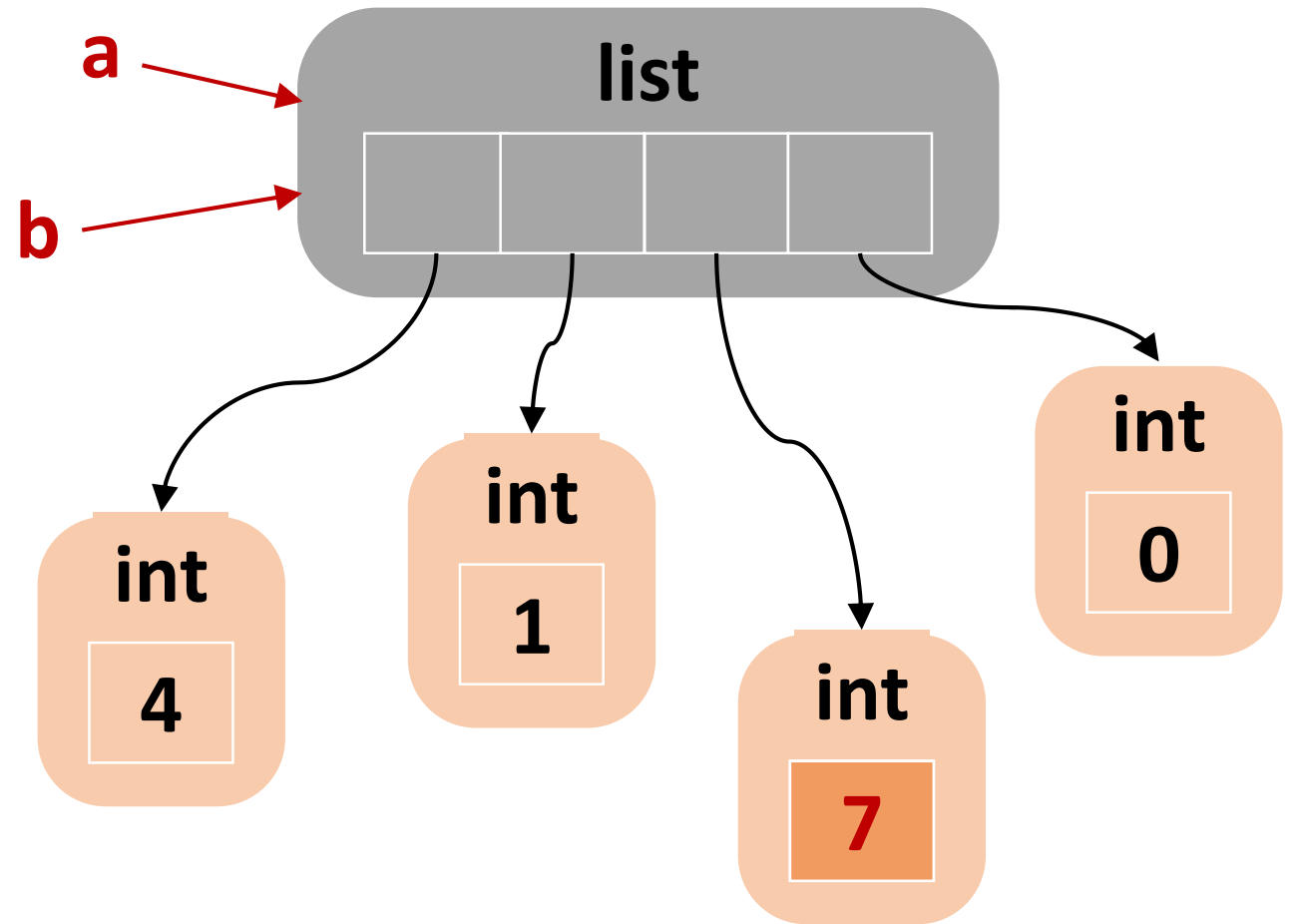
Assigning a List

```
>>> a = [4, 1, 9, 0]
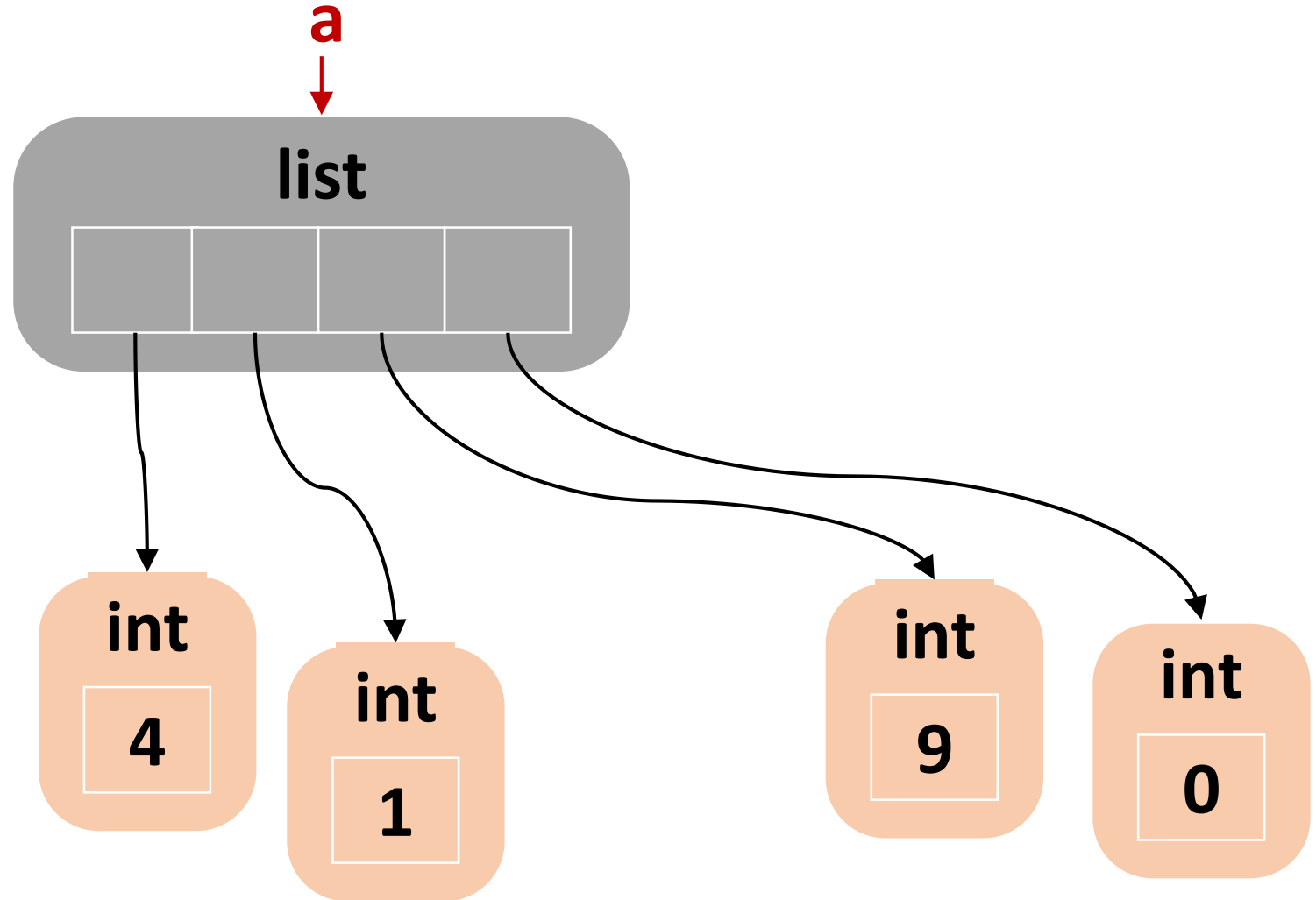```

# Assigning a List

```
>>> a = [4, 1, 9, 0]
>>> b = a
```

a → **list**

b →

int
4

int
1

int
9

int
0

Assigning a List

```
>>> a = [4, 1, 9, 0]
>>> b = a
>>> a[2] = 7
>>> print(a)
[4, 1, 7, 0]
>>> print(b)
[4, 1, 7, 0]
```

Copying a List

a
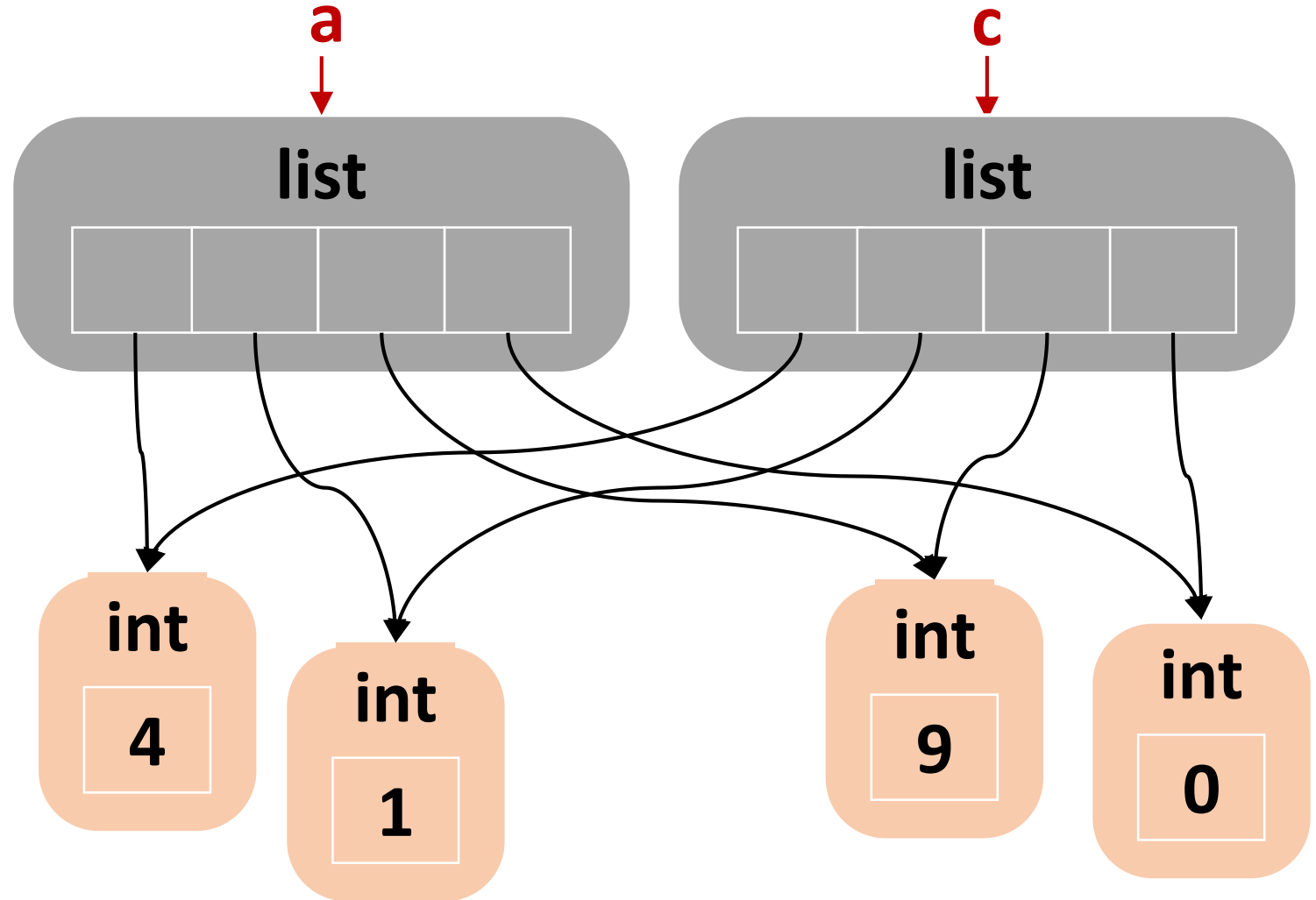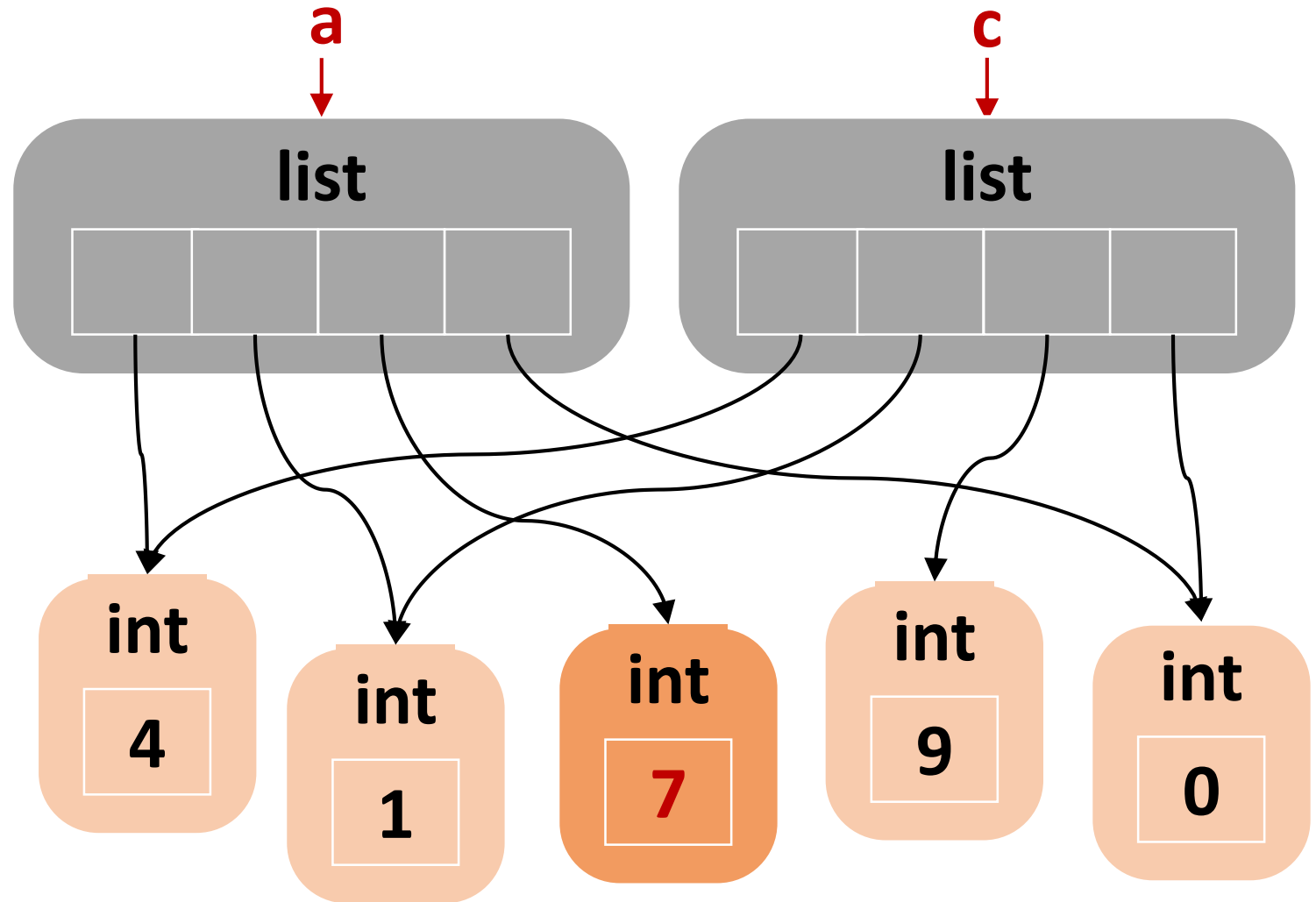
```
>>> a = [4, 1, 9, 0]
```

list

int
4

int
1

int
9

int
0

Copying a List

```
>>> a = [4, 1, 9, 0]
>>> c = a.copy()
```

Copying a List

```
>>> a = [4, 1, 9, 0]
>>> c = a.copy()
>>> a[2] = 7
>>> print(a)
[4, 1, 7, 0]
>>> print(c)
[4, 1, 9, 0]
```

Sorting Elements in a List (1)

- *list.*sort([reverse=True])
  - Sort the elements in the list
  - if reverse is True, the list elements are sorted in the reverse order
  - *list.*sort() changes the list in place, but don't return the list as a result

```
>>> names = ['Emma', 'Olivia', 'Ava', 'Isabella', 'Sophia']
>>> names.sort()
>>> print(names)
['Ava', 'Emma', 'Isabella', 'Olivia', 'Sophia']
>>> names.sort(reverse=True)
['Sophia', 'Olivia', 'Isabella', 'Emma', 'Ava']
```
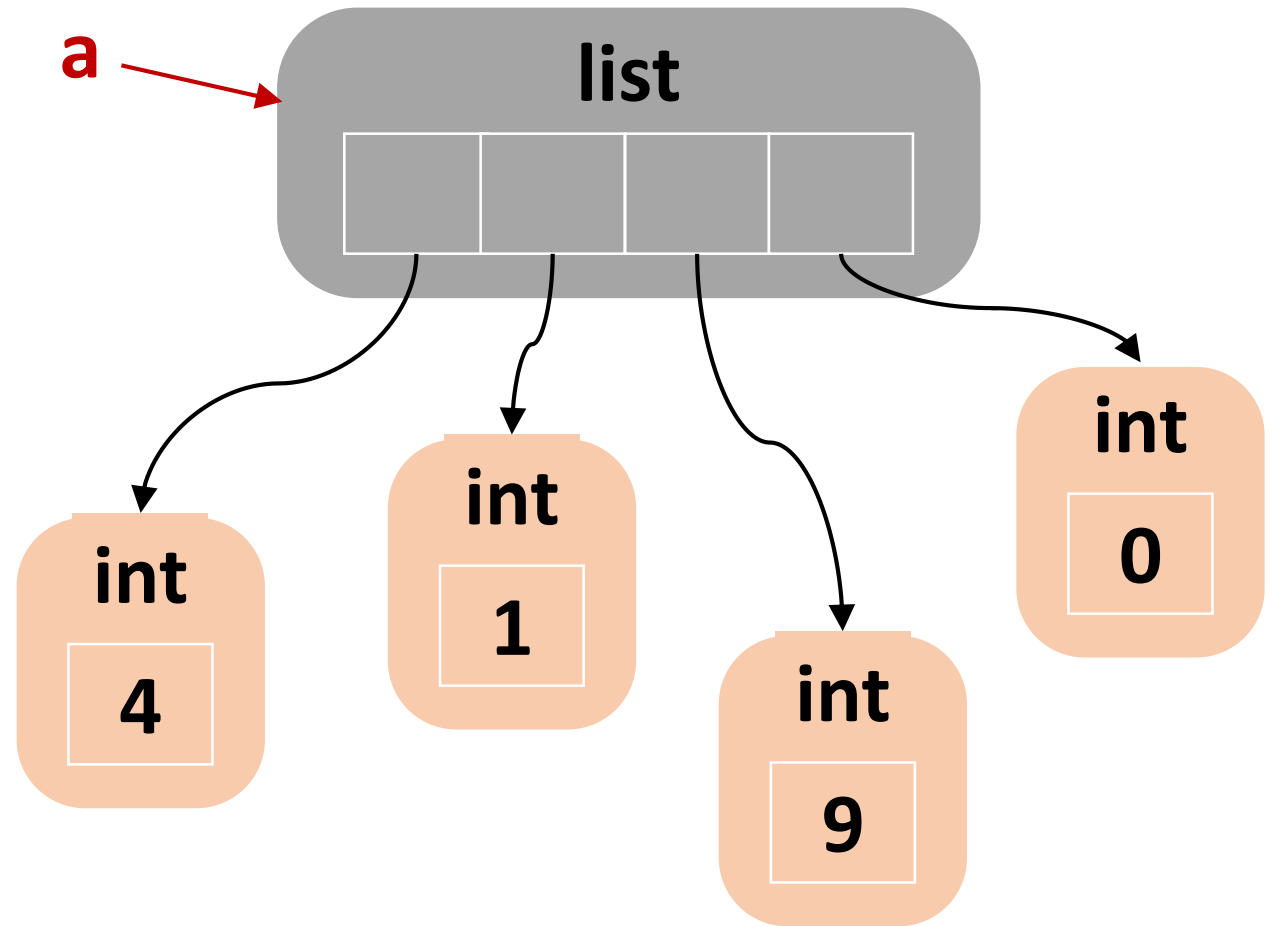
Sorting Elements in a List (2)

- **sorted(*list*[, reverse=True])**
  - Sort the elements in the list
  - if reverse is True, the list elements are sorted in the reverse order
  - sorted() returns a new list!

```
>>> names = ['Emma', 'Olivia', 'Ava', 'Isabella', 'Sophia']
>>> sorted_names = sorted(names)
>>> print(sorted_names)
['Ava', 'Emma', 'Isabella', 'Olivia', 'Sophia']
```
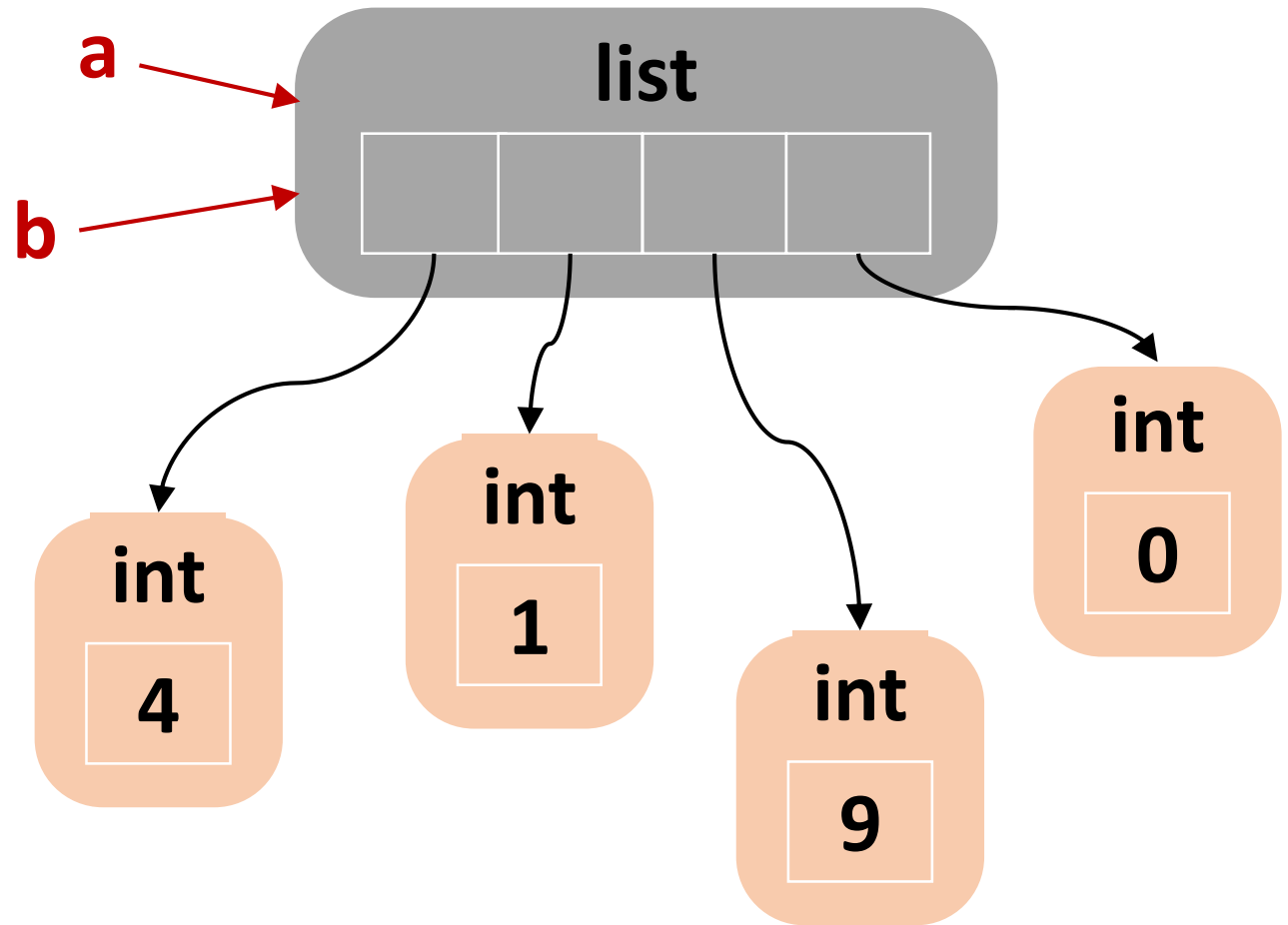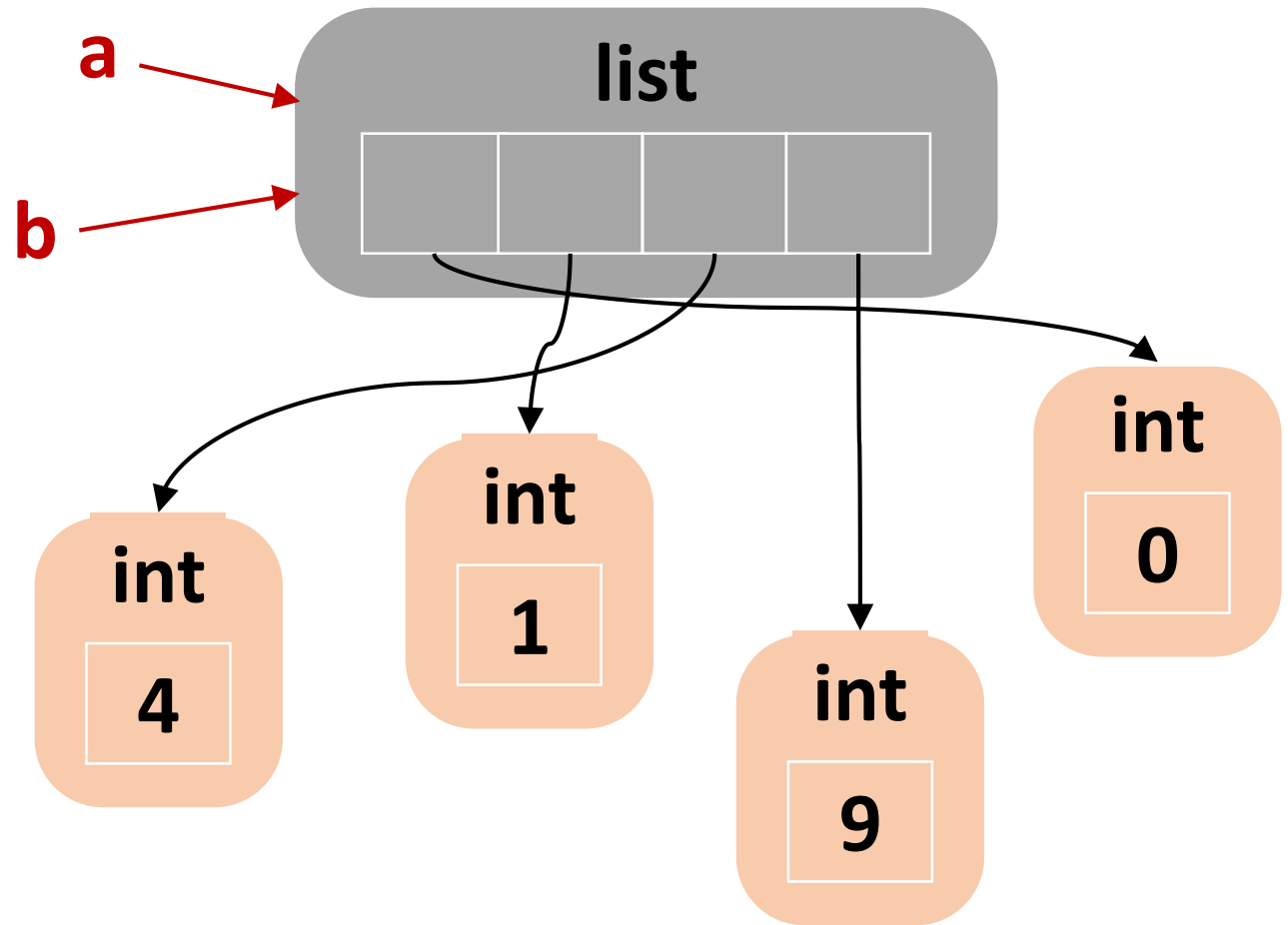
list.sort()

```
>>> a = [4, 1, 9, 0]
```

a → **list**

int 4

int 1

int 9

int 0

list.sort()

```
>>> a = [4, 1, 9, 0]
>>> b = a
```

a →

b →

list
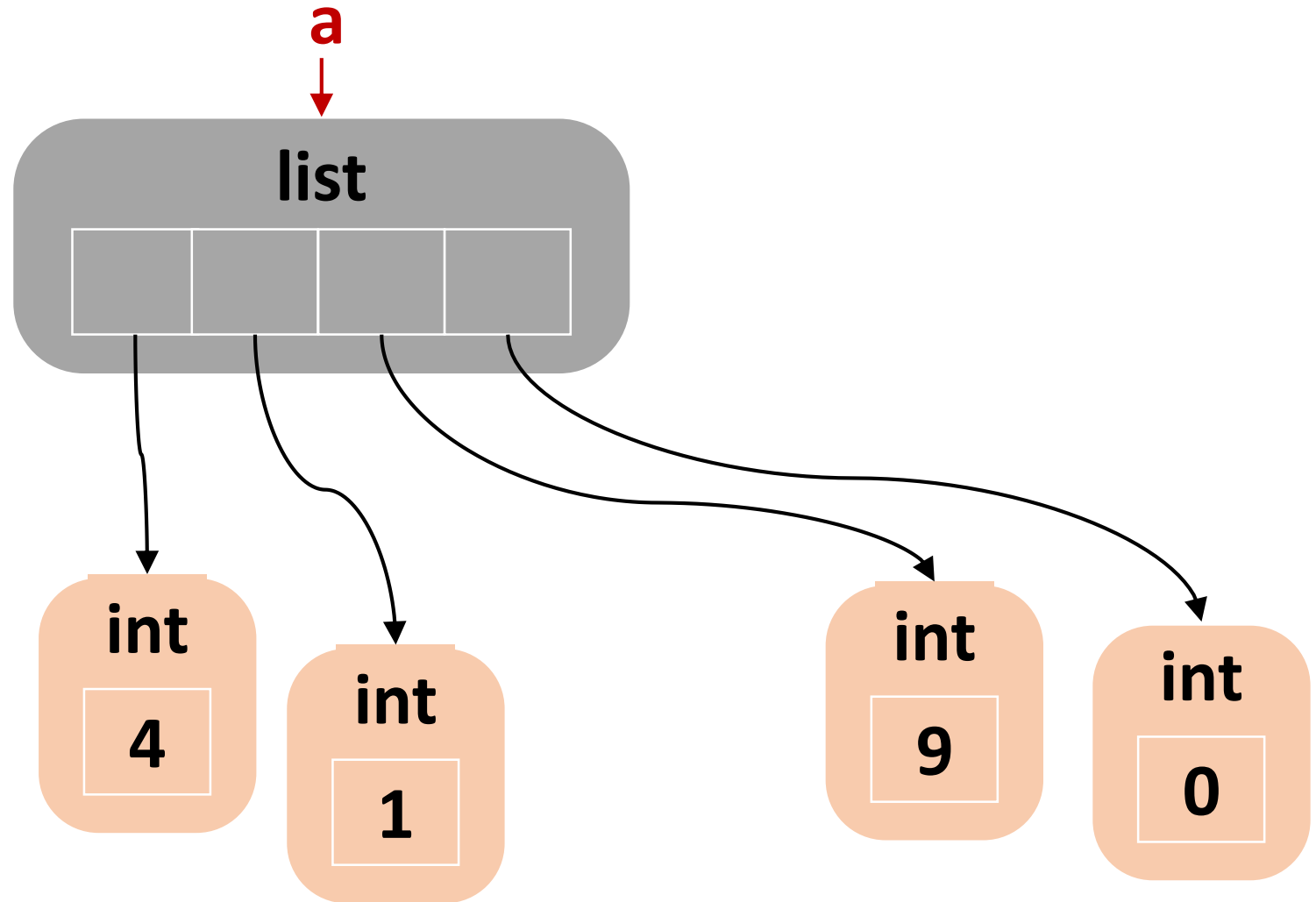
int
4

int
1

int
9

int
0

list.sort()

```
>>> a = [4, 1, 9, 0]
>>> b = a
>>> b.sort()
>>> print(b)
[0, 1, 4, 9]
>>> print(a)
[0, 1, 4, 9]
```

sorted(list)

a

```
>>> a = [4, 1, 9, 0]
```

list

int **4**

int **1**

int **9**

int **0**

sorted(list)

```
>>> a = [4, 1, 9, 0]
>>> c = sorted(a)
>>> print(c)
[0, 1, 4, 9]
>>> print(a)
[4, 1, 9, 0]
```