

CSCI 145 · Introduction to Computing · Fall 2026

More Lists

Lecture 9 — sorting, aliasing, map/filter/reduce, list comprehensions

Outline

- Sorting and reversing

- Two names, one list

- Map, filter, and reduce

- List comprehensions

Part 1 of 4

Sorting and reversing

sorted vs. .sort()

CODE

```
nums = [4, 2, 3]
```

```
sorted(nums)
```

```
nums.sort()
```

```
nums.reverse()
```

RETURNS

[2, 3, 4]

None

None

NUMS AFTER

4 2 3

4 2 3

2 3 4

4 3 2

- `sorted(nums)` returns a **new** sorted list
`nums` doesn't change
- `.sort()` and `.reverse()` change the list in place
They return `None`

Sorting documentation: docs.python.org/3/howto/sorting.html

A common mistake with `.sort()`

`.sort()` changes the list

```
>>> nums = [4, 2, 3]
>>> nums.sort()
>>> nums
[2, 3, 4]
```

```
>>> nums = [4, 2, 3]
>>> nums = nums.sort()
>>> print(nums)
None
```

- `.sort()` returns `None`
so `nums = nums.sort()`
sets `nums` to `None`
- Same mistake as
`evens = evens.append(num)`
in Lecture 8
- Want a new sorted list?
Use `sorted(nums)`

Part 2 of 4

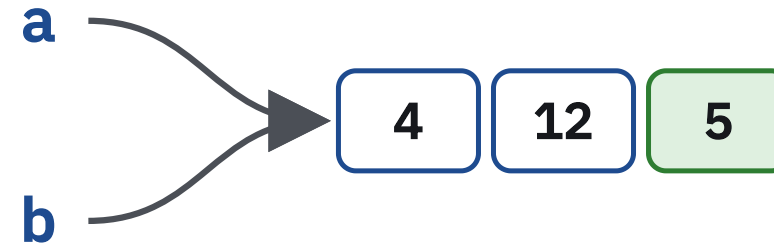
Two names, one list

A variable names a list

```
a = [4, 12, 3]
b = a
b[2] = 5
print(a)
```

OUTPUT

```
[4, 12, 5]
```

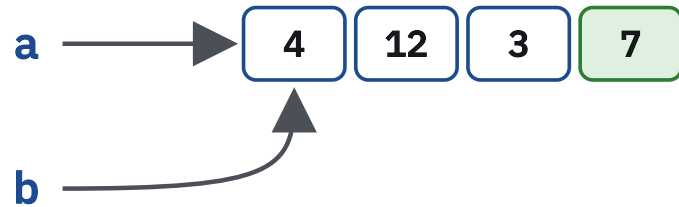


- `b = a` doesn't copy the list
`a` and `b` are two names for one list: **aliases**
- A change made through `b` also shows up through `a`

Changing the list vs. naming a new list

CHANGES THE LIST

```
a = [4, 12, 3]
b = a
b.append(7)
print(a)
```



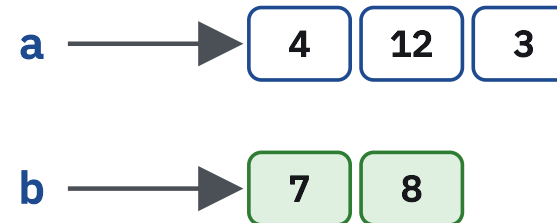
OUTPUT

```
[4, 12, 3, 7]
```

`b.append(7)` changes the list
Every name for it sees the change

NAMES A NEW LIST

```
a = [4, 12, 3]
b = a
b = [7, 8]
print(a)
```



OUTPUT

```
[4, 12, 3]
```

`b = [7, 8]` makes `b` name a new list
`a` still names the old one

What does this print?

```
nums = [1, 2, 3]
for num in nums:
    num = num * 2
print(nums)
```

Go to **slido.com** and enter the code: 3674841

Or [join directly here](#)



What does this print?

```
nums = [1, 2, 3]
for num in nums:
    num = num * 2
print(nums)
```

It prints [1, 2, 3]

num now names a new value
The list doesn't change

```
nums = [1, 2, 3]
for i in range(len(nums)):
    nums[i] = nums[i] * 2
print(nums)
```

To change the list, assign to nums[i]

Loop over the indices

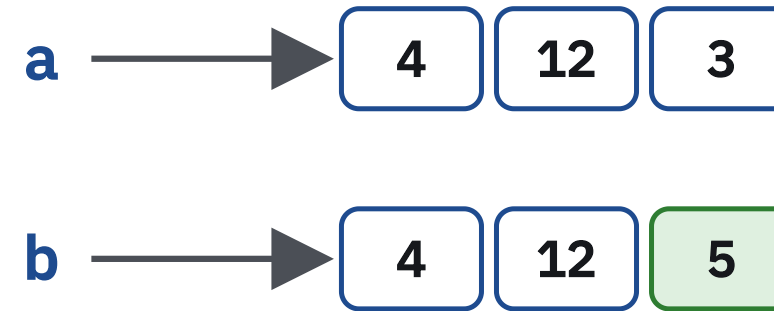
Now it prints [2, 4, 6]

Making a copy

```
a = [4, 12, 3]
b = a[:]
b[2] = 5
print(a)
print(b)
```

OUTPUT

```
[4, 12, 3]
[4, 12, 5]
```



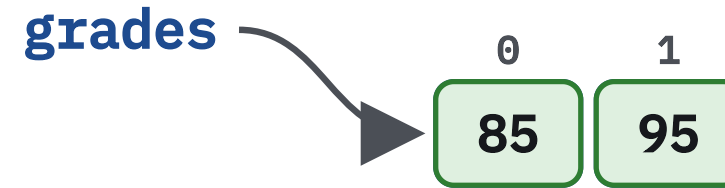
- A slice is a new list, so `a[:]` makes a **copy**
A new list with the same items
- `a.copy()` and `list(a)` make a copy too

A function can change a list you pass in

```
def add_bonus(scores):  
    for i in range(len(scores)):  
        scores[i] += 5  
  
grades = [80, 90]  
add_bonus(grades)  
print(grades)
```

OUTPUT

```
[85, 95]
```



- Calling `add_bonus(grades)` makes `scores` an alias for `grades`
- So `scores[i] += 5` changes `grades`

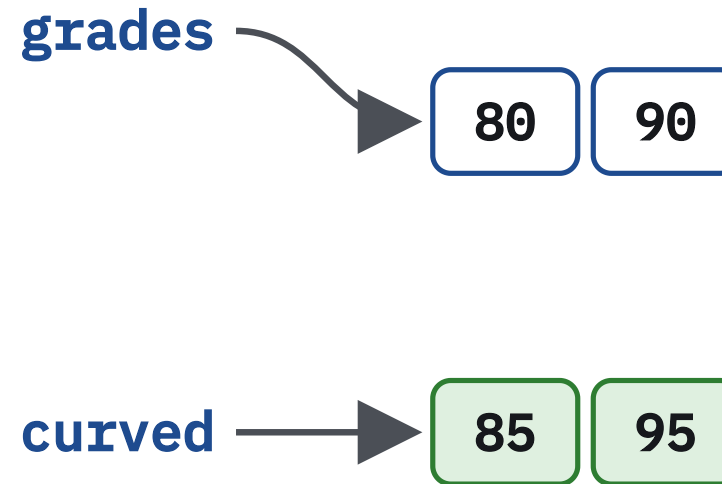
A function can return a new list instead

```
def with_bonus(scores):  
    result = []  
    for score in scores:  
        result.append(score + 5)  
    return result
```

```
grades = [80, 90]  
curved = with_bonus(grades)  
print(grades, curved)
```

OUTPUT

```
[80, 90] [85, 95]
```



- `result` is a new list, separate from `grades`
- Unless changing the list is the point, build and return a new one

Part 3 of 4

Map, filter, and reduce

Three common list patterns

You already wrote one of each

Map

Change every item

Lecture 8: squares

1 2 3



1 4 9

Filter

Keep some of the items

Lecture 8: keep_evens

3 4 5



4

Reduce

Combine into one value

Lecture 8: highest

88 92 75 100



100

Map and **filter** build a new list

Reduce combines the items into one value

Reduce: find the bug in `list_sum`

`list_sum([5, 10])` should return 15

```
def list_sum(nums):  
    total = 0  
    for num in range(len(nums)):  
        total += num  
    return total
```

Returns 1

`num` is each **index**: 0, then 1
So the total is 0 + 1

```
def list_sum(nums):  
    total = 0  
    for num in nums:  
        total += num  
    return total
```

✓ Fix: loop over the items

Now `num` is 5, then 10
It returns 15

Map, filter, or reduce?

FROM A LIST OF GRADES
Find the highest

Reduce

FROM A LIST OF NAMES
Make each one uppercase

Map

FROM A LIST OF EMAIL ADDRESSES
Keep the ones ending in `@middlebury.edu`

Filter

FROM A LIST OF WORDS
Count the ones that start with `"a"`

Filter, then reduce

Part 4 of 4

List comprehensions

A list comprehension: a loop in one line

A **list comprehension** builds a new list

Map: square every number in `nums`

LOOP

```
squares = []  
for num in nums:  
    squares.append(num * num)
```

LIST COMPREHENSION

```
squares = [num * num for num in nums]
```

GENERAL FORM

```
[expression for variable in sequence]
```

for each item in the list

expression for the new item

Adding a condition: filter

Filter: keep the even numbers in `nums`

LOOP

```
evens = []  
for num in nums:  
    if num % 2 == 0:  
        evens.append(num)
```

LIST COMPREHENSION

```
evens = [num for num in nums if num % 2 == 0]
```

GENERAL FORM

```
[expression for variable in sequence if condition]
```

for each item in the list

if keeps only the items that pass

expression for the new item

What does this print?

```
nums = [1, 5, 2, 8]
result = [num * 10 for num in nums if num > 2]
print(result)
```

Go to **slido.com** and enter the code:

3674841

Or [join directly here](#)



What does this print?

```
nums = [1, 5, 2, 8]
result = [num * 10 for num in nums if num > 2]
print(result)
```

It prints [50, 80]

The `if` keeps 5 and 8

Then `num * 10` maps each one

```
nums = [1, 5, 2, 8]
result = []
for num in nums:
    if num > 2:
        result.append(num * 10)
print(result)
```

The same list, built with a loop

Filter with `if`

Map with the expression you append

More list comprehensions

Try writing each one before it appears

- First letters of `words`

```
[word[0] for word in words]
```

[Map

- Words in `words` shorter than 4

```
[word for word in words if len(word) < 4]
```

[Filter

- Squares of positive numbers in `nums`

```
[num * num for num in nums if num > 0]
```

[Map

[Filter

- Factors of `n`

```
[i for i in range(1, n + 1) if n % i == 0]
```

[Filter

A comprehension can map and filter, but not reduce

For one value, use a loop or `sum`, `min`, `max`

Activity

1. Download [day09_lists.py](#) and save it in your `cs145/class` folder.
2. Launch Thonny. From the File menu, open `cs145/class/day09_lists.py`.
3. Write `lengths`, `long_words`, and `product` with loops.
4. Which pattern is each one: map, filter, or reduce?
5. Write `lengths_comp` as a one-line list comprehension.
6. If you have time, complete `long_words_comp` and `double_in_place`.

To do

- Review this slide deck and the corresponding notes: [Notes 8](#)
- [Homework 3](#) is due Thursday 10/15
- No lab this Friday: fall break

Next time

Lecture 10 — recursion