

Welcome to 61A Lab!

We will begin at **5:10!**

Slides: **cs61a.bencuan.me**

Announcements

- Cats is due **this Thursday**
- HW 2 Recovery at **6pm**
 - You're welcome to join the zoom room and stay in the lab until 7!
- No HW this week! glhf on cats :)

The Plan

- List Comprehension review
- Mutation
- Lab hints
- Work time!

List Comprehensions



List Slicing

`lst[ start : end : step ]`

You can omit any of the above and use default values!

- Start: 0
- End: `len(lst)`
- Step: 1

Example: `lst[3:]` returns all but the first 3 elements of `lst`

The Syntax

[**op** for **el** in **lst** if **cond**]

```
result = []  
for el in lst:  
    if cond:  
        result += [op]  
return result
```

Example: get all even numbers in list

```
[ x for x in lst if x%2==0 ]
```

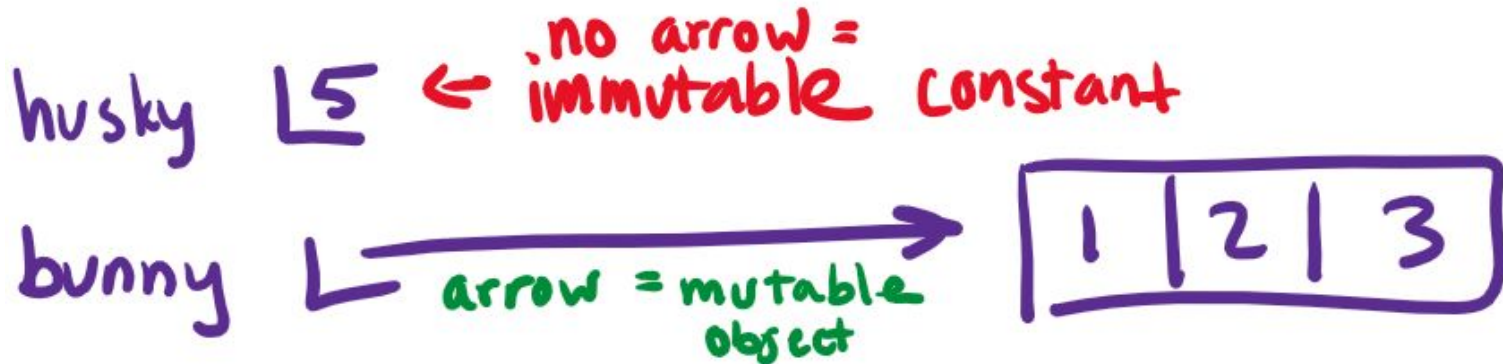
```
result = []  
for x in lst:  
    if x%2==0:  
        result += [x]  
return result
```

List Mutation

What is mutation?

mutating = changing

more specifically: a mutation is when you modify an object's contents



== vs is

a == b compares contents

“Do a and b hold the same values?”

a is b compares identity

“Are a and b arrows that point to the same object?”

list mutation functions (summary)

- `append(e1)`: Add `e1` to the end of the list. Return `None`.
- `extend(lst)`: Extend the list by concatenating it with `lst`. Return `None`.
- `insert(i, e1)`: Insert `e1` at index `i`. This does not replace any existing elements, but only adds the new element `e1`. Return `None`.
- `remove(e1)`: Remove the first occurrence of `e1` in list. Errors if `e1` is not in the list. Return `None` otherwise.
- `pop(i)`: Remove and return the element at index `i`.

append vs extend

Append puts one element onto the back of a list

`[1,2,3].append(5) => [1, 2, 3, 5]`

Extend appends lots of elements onto the back of a list

`[1,2,3].extend([5,6,7]) =? [1,2,3,5,6,7]`

pop vs remove

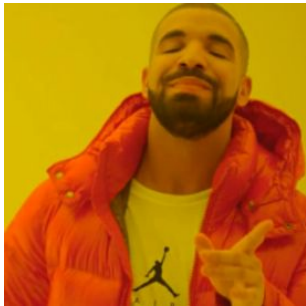
remove deletes a value and **returns None**.

pop deletes a value and **returns the value itself**.

a warning: mutation just works!



`lst = lst.append(2)`
DO NOT DO



`lst.append(2)`
GOOD!

list mutation functions (summary)

- `append(e1)`: Add `e1` to the end of the list. Return `None`.
- `extend(lst)`: Extend the list by concatenating it with `lst`. Return `None`.
- `insert(i, e1)`: Insert `e1` at index `i`. This does not replace any existing elements, but only adds the new element `e1`. Return `None`.
- `remove(e1)`: Remove the first occurrence of `e1` in list. Errors if `e1` is not in the list. Return `None` otherwise.
- `pop(i)`: Remove and return the element at index `i`.

Lab Hints

Lab Hints

- Remember the mutation warning! Use equals sparingly
- Write list comprehensions as for loops first
- Don't mutate in a for loop or bad things will happen. Use while instead!

Work Time!



go.cs61a.org/ben-queue