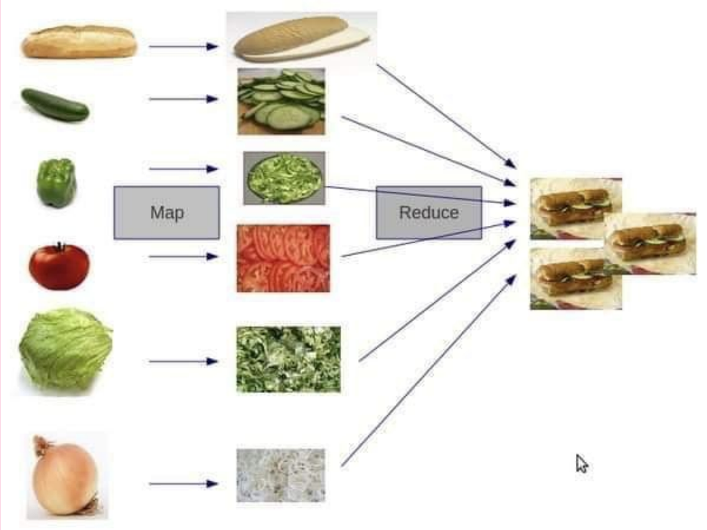


Hi! Welcome to 61A Discussion :)

We will begin at **8:10**!

Attendance: **go.cs61a.org/ben-disc**

Slides: **cs61a.bencuan.me**

Announcements

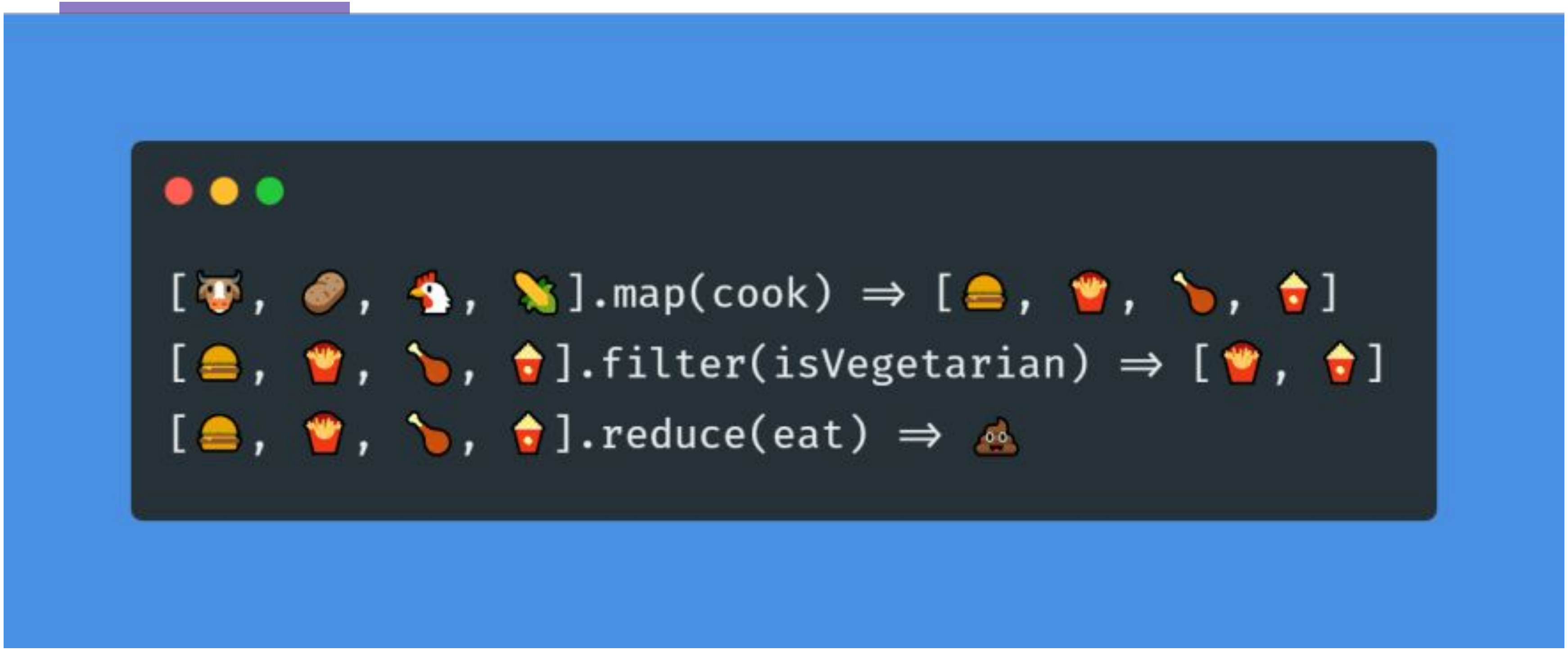
- Cats due today!!
    - Will end a bit early for Q’s/debugging hopefully

Agenda

- Attendance
- Sequences (map, filter, reduce)
- Mutability
- OOP

Map, Filter, Reduce

Emoji version



```
[🐮, 🥔, 🐔, 🌽].map(cook) ⇒ [🍔, 🍟, 🍗, 🍿]
[🍔, 🍟, 🍗, 🍿].filter(isVegetarian) ⇒ [🍟, 🍿]
[🍔, 🍟, 🍗, 🍿].reduce(eat) ⇒ 💩
```

credit: https://twitter.com/steveluscher

More formal version

- **Map(f, lst):** turn every x in lst into f(x)
  - f returns same type it gets in
- **Filter(f, lst):** get x only if f(x) == True
  - f always returns a boolean
- **Reduce(f, lst):** use f(a,b) to repeatedly combine all x
  - f takes in 2 numbers, returns 1 number
  - Reduce returns a **value**, not a list!

Examples

```
>> a = [1, 2, 3]
>> map(lambda x: x*x, a)
[1, 4, 9]
```

Examples

```
>> a = [1, 2, 3]
>> filter(lambda x: x % 2 == 0, a)
[2]
```

Examples

```
>> a = [1, 2, 3]
>> reduce(lambda x,y: x+y, a)
6
```

Q1: Make your own mapfilterreduce!

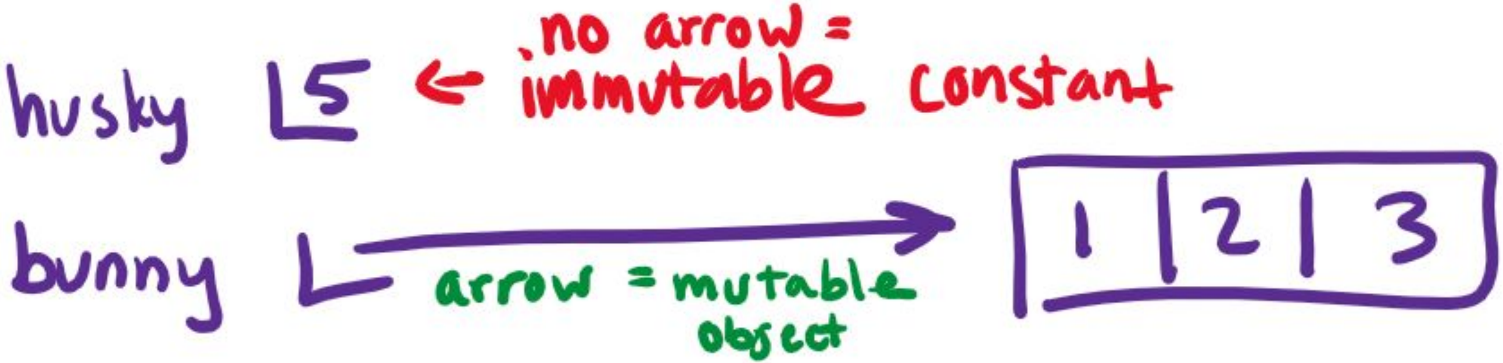
```
>> a = [1, 2, 3]
>> reduce(lambda x,y: x+y, a)
6
```

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?”

append vs extend

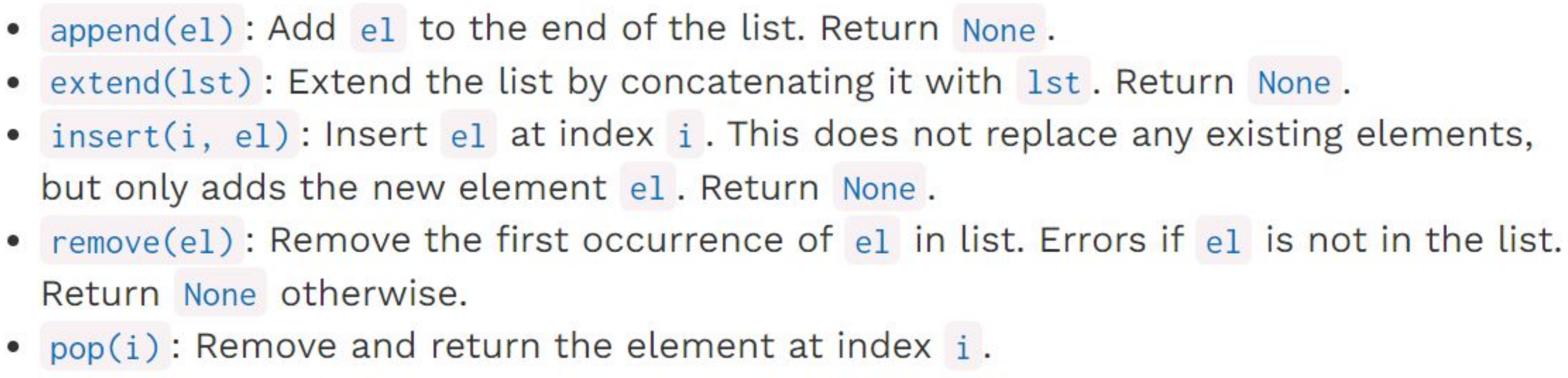
Append puts one element onto the back of a list

```
    [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]
```

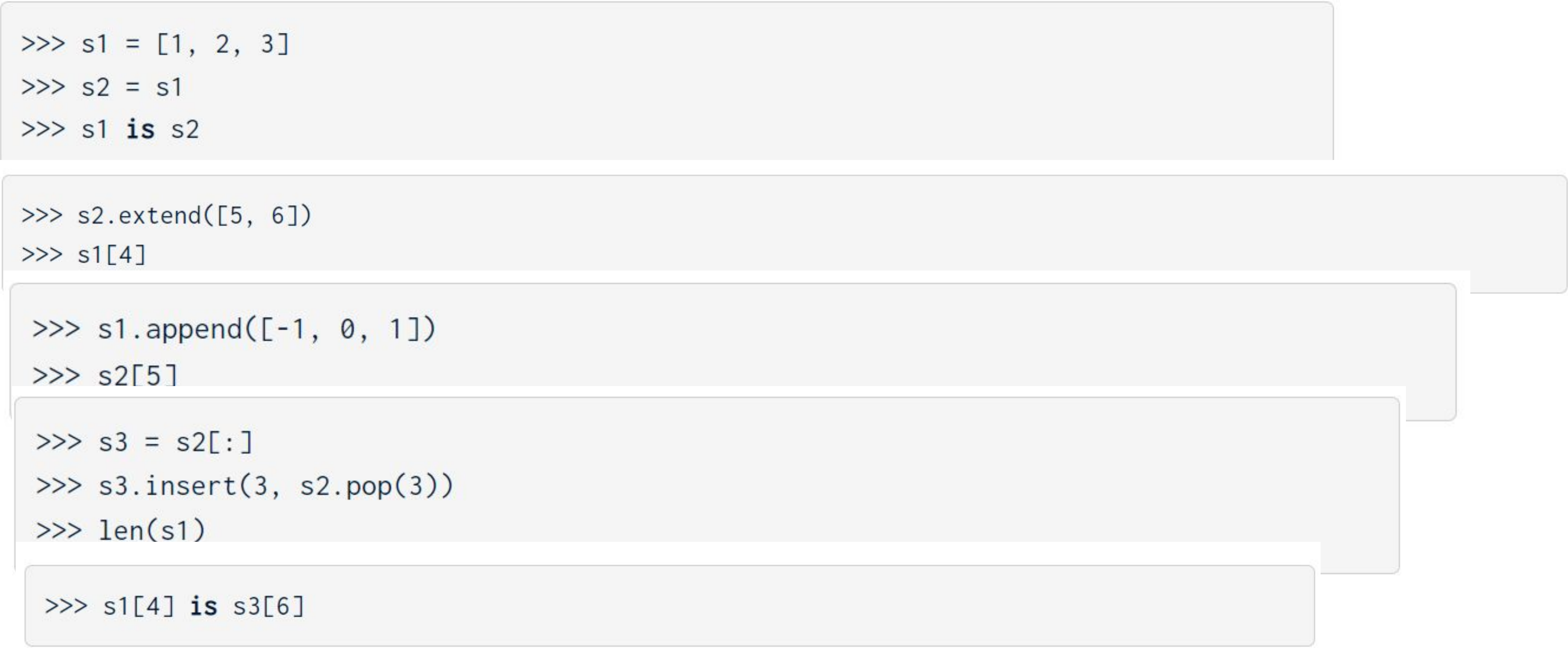
list mutation functions (summary)



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

Q2: WWPD: Mutability

What would Python display? In addition to giving the output, draw the box and pointer diagrams for each list to the right.



```
>>> s1 = [1, 2, 3]
>>> s2 = s1
>>> s1 is s2
```

```
>>> s2.extend([5, 6])
>>> s1[4]
```

```
>>> s1.append([-1, 0, 1])
>>> s2[5]
```

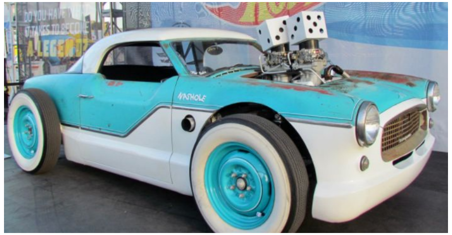
```
>>> s3 = s2[:]
>>> s3.insert(3, s2.pop(3))
>>> len(s1)
```

```
>>> s1[4] is s3[6]
```

Object Oriented Programming

Some OOP vocab

- **Class:** a blueprint for making objects
- **Instance:** one of those objects



- Class attribute: shared by **all** objects of that type (# wheels, model)
- Instance Variable: specific to **your** object (gas, mileage)

Dot notation

```
class SomeClass:
  def do_stuff(self, param):
    # DO STUFF

a = SomeClass()
SomeClass.do_stuff(a, 1)
a.do_stuff(1) # Same effect as line above!
```

Let’s do some WWPD…

Go to discussion worksheet!!

(pythontutor)

Keyboard

- A keyboard consists of buttons:
  - pos (int): ID
  - key (string): what the button outputs
  - times_pressed (int): # times it's been pressed

Keyboard Example

```
b_c = Button(0, “c”)
b_v = Button(1, “v”)
k = Keyboard(b_c, b_v)
k.typing([0, 0, 0, 1, 0, 1, 1, 0, 1])
```

```
cccvcvvcv
```