CS61A Discussion 10:
Iterators and Streams

TA: **Jerry Chen**
Email: **jerry.c@berkeley.edu**
TA Website: **jerryjrchen.com/cs61a**

Attendance

Form: **tinyurl.com/jerrydisc**

For the weekly question,
- Please submit your quiz answers.
Nice work on last week's quiz!

(Of course, please only check in if you showed up!)

Agenda

1. Week in Review
2. Iterators/Iterables
3. Streams

Week In Review

Lab 11 (Iterators and Generators) - **Due Friday**

Hw7 - **Due Friday**

Proj2 - Due 4/25

- Complete Part 1 of the Scheme project by **Monday** for 1 EC point

Maps Composition - Resubmit by **Friday**

Iterators/Iterables

Hopefully you got the basics from lab & hw…

Iterator

- Steps through a sequence **one item at a time** using `next`
- Implies that calling `next` will **modify some state**

Iterable

- **Returns an iterator** using `iter`

Iterators/Iterables

If something is **iterable**, we can get its **iterator** using `iter` and examine all its elements by repeatedly calling `next` on that iterator.

Keep in mind that iterators are usually **one-time use**. Stepping through a sequence again means calling `iter` again.

Iterators/Iterables

Miscellaneous

- Signal end of an iterator's sequence by raising a `StopIteration` exception

Generators

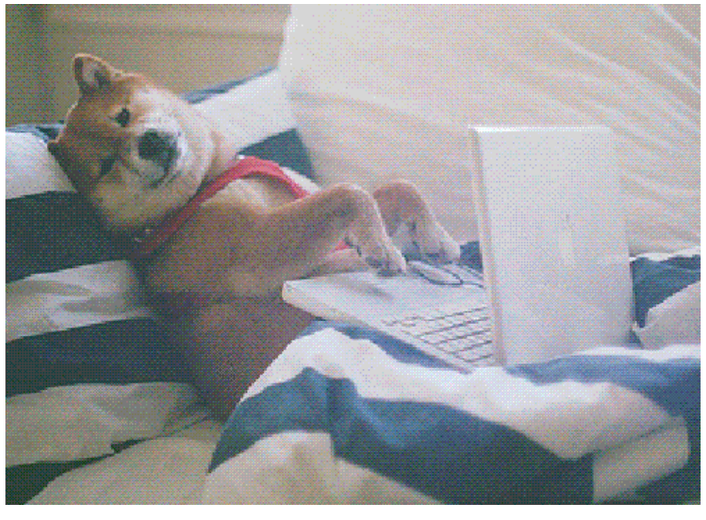
Upon request (they're not in the worksheet)

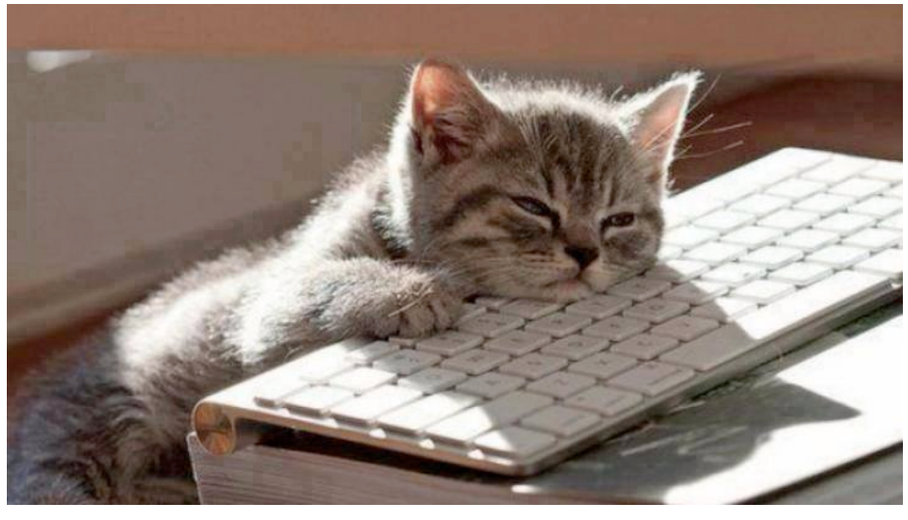
Streams

Like a linked list, except **evaluated lazily**

- Don't make rest **until we ask for it**
- After we ask for it, **remember the result**

- **Rules** (functions) tell us how to create the next element





Streams

Some stuff is the same:

- `car` gets the **front** of a stream
- `nil` is the **empty** stream

Some stuff is different:

- `cons-stream` like cons, but rest is **lazily evaluated**
- `cdr-stream` like cdr, but **tells stream to do the actual computation** if it hasn't already