Discussion 09:
Delayed Expressions

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

Agenda

1. Attendance
2. Announcements
3. Iterators/Iterables (fast)
4. Generators (fast)
5. Streams

Attendance

Sign in at bit.do/jerrydisc

OR

Come to me for check-in

Announcements

Want to talk/listen? EECS community election debrief in **521 Cory from 12 to 2 pm**

Announcements

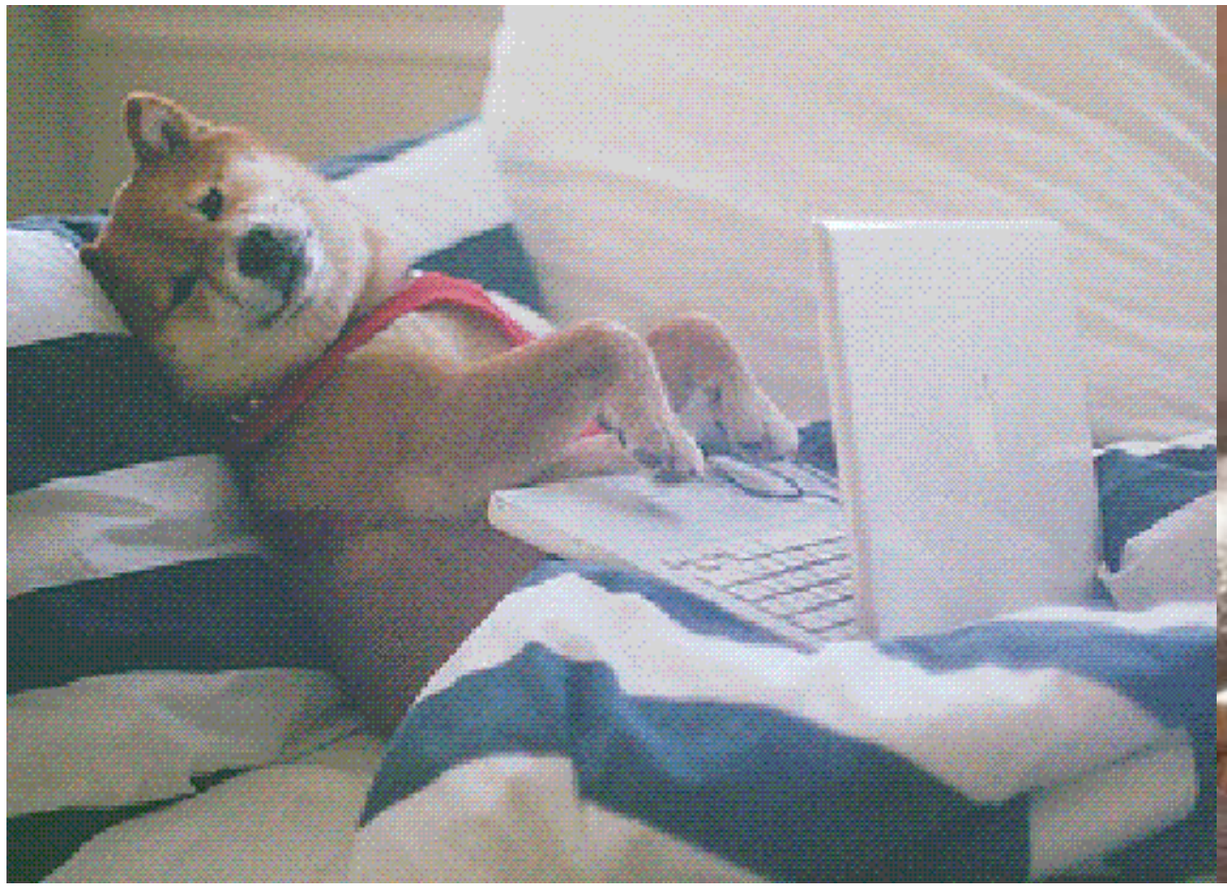
Hw 11 due today, Hw 12 due next Tues

Scheme Proj checkpoint 1 today!

My OH today are 4-5pm only

Delayed Expressions

"Lazy evaluation"



Iterators/Iterables

Iterable

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

Iterator

- Returns the next **item** in sequences using `next()`
- `next()` (probably) will **modify some state**

Iterators/Iterables

"The iterable is a book, and the iterator is a bookmark"

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
- `iter()` of an iterator usually gives you the same iterator back

Generators

Generator functions return a generator -> a special **iterator**

- **next** will cause us to run until the next **yield**
- Return the expression at the yield, and **pause**

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