CS61A Discussion 9

Iterators, Generators, Streams

Iterators vs. Iterables

- Iterable (ex. [1,2,3])
  - Has an iter method, which can return an iterator object
- Iterator
  - Has as iter method (which in good practice returns itself), and a next method which returns the next element of your sequence
- All Iterators are also Iterables!
- Important Distinction:
  - next() will change future calls to next()
  - An iterator spawned from an iterable (that is not an iterator) will not affect it’s ‘parent’

Use of Iterators

- You have already been using them!
  - For, zip, range..
- Can use this to lazily evaluate values (foreshadowing!)

Generator Functions

Making your own Iterator

- We can make iterators with generator functions
- Use yield rather than return
  - Special statement!
  - Frame stays open and waits for another next call
  - Allows interpreter to understand it is a generator rather than a normal function
- Call to the generator function acts as a call to iter()
- When the generator function finishes, subsequent calls to next() will raise StopIteration

Try Except Statements

Streams

Streams

- Objects built in scheme which take advantage of the concept of ‘lazy evaluation’
  - Don’t do any work unless you are asked to
  - Functions look similar to regular functions except with cons-stream and cdr-stream
- Promises