stream: an abstract model of a sequence of values over time
Implementing streams using lazy lists
A list is either:

- empty
- a value “cons” another list
A lazy list is either:

- empty
- a value “cons” a function that returns a lazy list
(s-cons first rest)

→

(cons first (lambda () rest))
(s-cons first rest)

→

(cons first (lambda () rest))
Streams are a way to decouple the production and consumption of data (in a program)
Consequences

1. Don’t need to access all input data at once.
(define (s-sum-tail numbers acc)
  (if (s-null? numbers)
      acc
      (s-sum-tail (s-rest numbers)
                  (+ (s-first numbers) acc)))))

(define (sum-tail numbers acc)
  (if (null? numbers)
      acc
      (sum-tail (rest numbers)
                (+ (first numbers) acc)))))
(define (s-sum-tail numbers acc)
  (if (s-null? numbers)
      acc
      (s-sum-tail (s-rest numbers)
                  (+ (s-first numbers) acc))))

sumTail numbers !acc =
  if null numbers
  then acc
  else sumTail (tail numbers)
                  (head numbers + acc)
Consequences

2. Facilitate parallelization of stages in a pipeline
Consequences

2. Facilitate parallelization of stages in a pipeline
Consequences

3. Represent **infinite** sequences (in finite time and space!)
Choices and backtracking
the ambiguous operator -<
> ( <- 1 2 3 )
1
> (next)
2
> (next)
3
> (next)
'done