Streams

Efficient Sequence Processing

Sequence Operations

Map, filter, and reduce express sequence manipulation using compact expressions

Example: Sum all primes in an interval from $a$ (inclusive) to $b$ (exclusive)

```python
def sum_primes(a, b):
    total = 0
    x = a
    while x < b:
        if is_prime(x):
            total += x
        x += 1
    return total
```

```python
def sum_primes(a, b):
    return sum(filter(is_prime, range(a, b)))
```

Spaces: Constant Also Constant

(Demo)

Streams

Streams are Lazy Scheme Lists

A stream is a list, but the rest of the list is computed only when needed:

```scheme
(car (cons 1 nil)) -> 1
(cdr (cons 1 nil)) -> ()
(car (cons 1 (cons 2 nil))) -> 1
(cdr (cons 1 (cons 2 nil))) -> ()
```

Errors only occur when expressions are evaluated:

```scheme
(car (cons 1 (cons (/ 1 0) nil))) -> ERROR
(car (cons 1 (cons (/ 1 0) nil))) -> (1 . #\[promise (not forced)])
(car (cons 1 (cons (/ 1 0) nil))) -> 1
(cdr (cons 1 (cons (/ 1 0) nil))) -> ERROR
```

(Demo)

Stream Ranges are Implicit

A stream can give on-demand access to each element in order

```scheme
(define (range-stream a b)
    (if (> a b)
        nil
        (cons-stream a (range-stream (+ a 1) b))))
```

```scheme
(define lots (range-stream 0 10000000000000000000))
```

```scheme
(car lots) 1
(cdr lots) 2
```

Infinite Streams
A Stream of Primes

For any prime \( k \), any larger prime must not be divisible by \( k \).

The stream of integers not divisible by any \( k = n \) is:

The stream of integers not divisible by any \( k < n \) filtered to remove any element divisible by \( n \). This recurrence is called the Sieve of Eratosthenes.

\[
\begin{align*}
2, & \ 3, \ 4, \ 5, \ 6, \ 7, \ 8, \ 9, \ 10, \ 11, \ 12, \ 13 \\
\end{align*}
\]