More with Abstraction

- We can often replace multiple function definitions with a single, more general (more useful) function definition.
- We can often design with abstraction in mind
  - start by considering the general type of operations that will be needed.

map

- Scheme includes the `map` operation:
  - consumes a function and a list, produces a list.
  - map applies the function to each element of the list, and returns a list of the results.

```
(map number? '(1 2 fred 3)) =>
 (list true true false true)
```

Using `map`

- You can define your own functions

```
(define (squared x)
  (* x x))

(map squared '(1 2 3 4 5 6))  =>
(list 1 4 9 16 25 36)
```

`map` implementation

```
(define (map f l)
  (cond
    [(empty? l) empty]
    [else (cons (f (first l))
    (map f (rest l)))]))
```

Exercise: `map2`

- Create a function named `map2` that consumes a function and a list, and produces a list.
- `map2` should apply the function to all adjacent pairs of elements of the list.

```
(map2 + '(1 2 3 4)) => '(3 5 7)
```

Exercise `map2lists`

- Create a function named `map2lists` that consumes a function and two lists, and produces a list.
- `map2lists` should apply the function to the first elements of each list, then the second elements, ...

```
(map2lists + '(1 2 3 4) '(6 2 0 3)) =>
                    '(7 4 3 7)
```

Other abstract functions provided by scheme

`build-list:` builds list by applying a function to integers 0-N.

`filter:` basically filter1 from last class.

`quicksort:` sort a list according to some comparison function.

`andmap, ormap:` determine whether some function (predicate) is true for all or any elements of a list.

`build-list:` builds list by applying a function to integers 0-N.

There are others...

build-list Exercises

```
;; build-list : N (N  ->  X)  ->  (listof X)
;; to construct (list (f 0) ... (f (- n 1)))
(define (build-list n f) ...)
```

Use build-list to create a list of the first 10 squares.

Use build-list to create `'(.1 .01 .001 .0001)`

Quicksort

```
;; quicksort : (listof X) (X X  ->  boolean)
;;    ->  (listof X)
;; to construct a list from all items on alon
;;  in an order according to cmp
(define (quicksort alox cmp) ...)

(quicksort '(1 5 3 8 7 2) <) =>
(list 1 2 3 5 7 8)
```

Quicksort Exercises

- Use quicksort to sort a list of numbers in decreasing order.
- sort a list of posn structures according to how far they are from the point 100,100
  - dist = sqrt($(x-100)^2 + (y-100)^2$)