



---

# Programming Languages

**CS 251**  
*Fall 2021*

---

*Carolyn Anderson*

Recap

# Terminology

---

**First-class functions:** functions that are treated just like other values in the language, including being able to appear in all syntactic environments.

**Higher-order functions:** functions that take functions as arguments.

# Functions returning functions

---

# Properties of map

---

- ◆ Input items and return items do not need to be of the same type
- ◆ Preserves the length of the original list

# Properties of filter

---

- ◆ Function given as argument must return a boolean
- ◆ Does not preserve the length of list
- ◆ Returns copies of items from the original list

# Warm-up: filter out even numbers

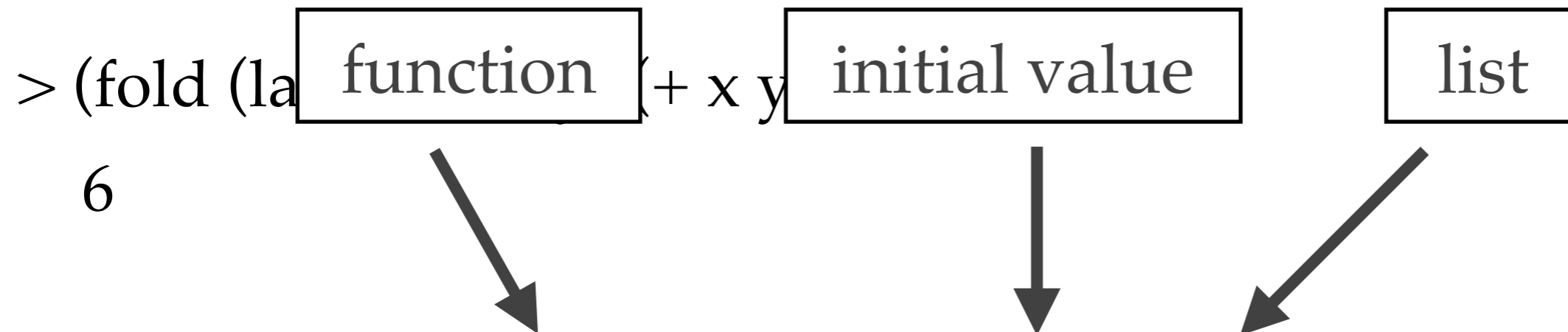
---

Using filter, write a function that returns all odd numbers from a list of numbers.

# Fold: returning a single value

---

**Fold** is a higher-order function that takes a list and returns a single value. It is also known as **reduce**.





# Fold: returning a single value

---

```
(define (add x y) (+ x y))
```

```
(fold add 0 (list 1 2 3) )
```

```
(fold add (+ 1 0) (list 2 3))
```

```
(fold add (+ 2 1) (list 3))
```

```
(fold add (+ 3 3) (list ))
```

# Foldl and foldr

---

```
(define (add x y) (+ x y))
```

```
(foldl add 0 (list 1 2 3))  
(foldl add (+ 1 0) (list 2 3))  
(foldl add (+ 2 1) (list 3))  
(foldl add (+ 3 3) (list ))
```

```
(foldr add 0 (list 1 2 3))  
(foldr add (+ 3 0) (list 2 3))  
(foldr add (+ 2 3) (list 3))  
(foldr add (+ 1 5) (list ))
```

# Properties of fold

---

- ◆ Returns a single value of any type
- ◆ Takes an initial value as an argument, as well as the list and the function to apply
- ◆ Function supplied must have two arguments

# Fold's initial value argument

---

- ◆ What **return type** do you want?
- ◆ What **initial value** do you need?

# Exercise: list and

---

Write a version of `and` that takes a list.

Return true if all items in the list are true and false otherwise.

Use one of the built-in higher-order functions that we have discussed.

# Exercise: list xor

---

Write a function that returns true if and only if 1 item in the list is true.

Use one of the built-in higher-order functions that we have discussed.

# Properties of map and fold

One property of map is that mapping function  $f$  over list  $l$ , and then mapping function  $g$  over the result, is equivalent to mapping the composition of  $f$  and  $g$  over  $l$ .

```
(define (add-5 x) (+ x 5))
```

```
(define (multiply-by-10 x) (* x 10))
```

```
(define numbers (list 1 2 3))
```

```
> (map multiply-by-10  
   (map add-5 numbers))  
(60 70 80)
```

```
> (map (lambda (x)  
        (multiply-by-10 (add-5 x)))  
       numbers)  
(60 70 80)
```

# Properties of map and fold

Similarly, mapping function  $f$  over list  $l$  and then folding function  $g$  over the result is equivalent to folding the composition of  $f$  and  $g$  over  $l$ .

```
(define (add-5 x) (+ x 5))  
(define (sum x y) (+ x y))  
(define numbers (list 1 2 3))
```

```
>(fold sum  
  0  
  (map add-5 numbers))
```

21

```
> (fold (lambda (x y)  
        (+ (add-5 x) y))  
  numbers
```

21