204: Swift Functional Programming

Part 1: Overview

raywenderlich.com

Functional Programming

- A *style* of programming

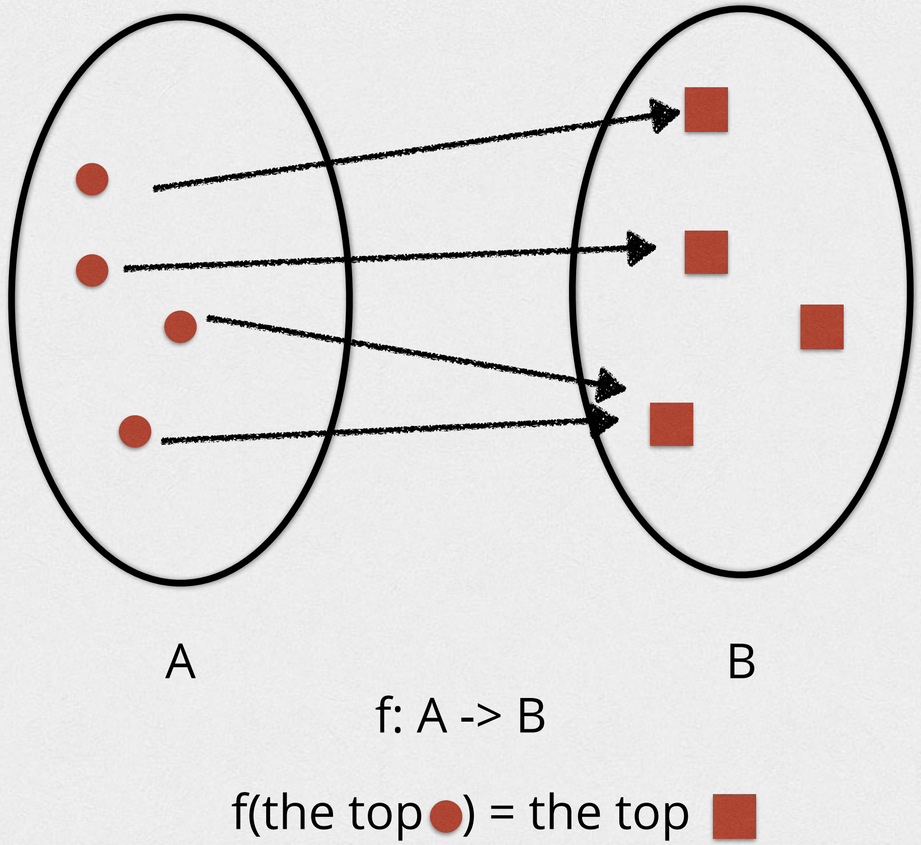
  (like OOP is a style) ...

- ... treating the *mathematical function* as the primary unit of abstraction

  (instead of an object, or a procedure)

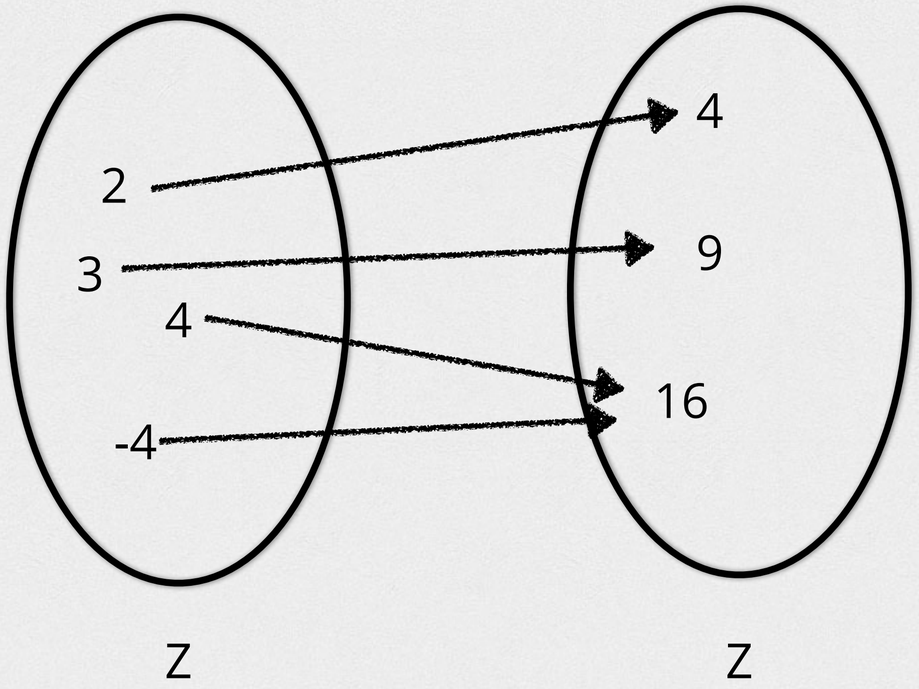
Mathematical function

- mathematical function
  - map values to values
  - a value is just an element in a set (.e.g, set of integers, of strings, etc.)



Mathematical function

- mathematical function
  - map values to values
  - a value is just an element in a set (.e.g, set of integers, of strings, etc.)



f: Z -> Z

f(3) = 9

f(x) = x * x

mathematical functions
establish true relations
which remain true

computational functions
do things

$$t_0 = 5$$

```
NSNumber * t0 = @5;
```

$$f(t_0) = 50$$

```
NSNumber * pos = f(t0);
pos; // => 50
```

$$f(t_0) = 50$$

```
pos = f(t0);
pos; //=> 100;
```

(surprise!)

mathematical variables
are names
we give to values

$$t_0 = 5$$

$$t_0 = 5$$

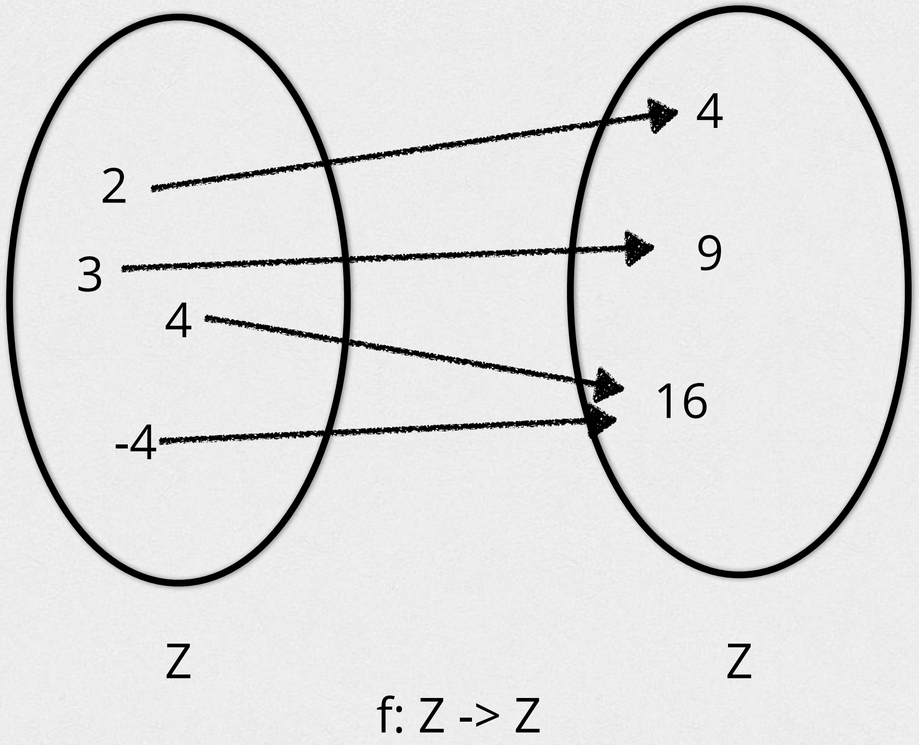
$$t_0 = 6$$

computational variables
are like names for places,
whose contents can change

```
NSNumber * five = @5;

five = @6; // ?!

five = @7;

[five setIntegerValue:8]; // ?! ?!
```

Mathematical function

- mathematical function
  - map values to values
  - a value is just an element in a set



f(x) = x * x

Elements of FP style

- Key practices to make programs “mathier":
  - immutable variables
  - pure functions (no side-effects)
  - higher-order functions

Swift's support for FP style

- Nicer functions:
  - consistent syntax for functions, closures, methods
- Can create more "mathy" variables:
  - Immutability attributes (Immutable, *just* like numbers)
  - Value types (Unaliased, a *bit* like numbers)
- Etc: complex enums, enum case patterns, generics, type inference, ...

Today's agenda

- Values:
  - immutable vs mutable problem modeling
- Functions:
  - elementary higher order functions (map,filter,reduce)