Chapter 3 - Types and Classes

→ A type is a name for a collection of related values

Ex: Bool → contains (True, False) logical values

→ Types help detect errors in your program

Ex: 1 + False gives error

→ If evaluating an expression e would produce a value of type t, then e has type t e :: t → Type signatures

→ All type errors are found at complie time makes them safer and faster no need to do run time checks

→ Basic Types: (Bool, Char, String, Int, Float)

→ List Types

Ex: [False, True, False] :: [Bool]

['a', 'b', 'c', 'd'] :: [Char]

→ Type of a list tells nothing about its length

→ Type of elements is unrestricted can have lists of lists

→ Tuple is a sequence of values of different types

Ex: (False, True) :: (Bool, Bool)

(False, 'a', True) :: (Bool, Char, Bool)

(True, ['a', 'b']) :: (Bool, [Char])

→ Encodes it size

→ Function maps value from one type to another type

Ex: not :: Bool → Bool

even :: Int → Bool

No constraints on types can be anything

```
add :: (Int, Int) → Int
add (x, y) = x + y
```

```
zeroto :: Int → [Int]
zeroto n = [0..n]
```

→ Curried Functions are functions with multiple arguments by returning functions as results

Ex: addPrime :: Int → (Int → Int)
addPrime x y = x + y
↳ One input at a time and waits for the second integer

→ These functions are flexible and useful functions can be made by partially applying curried functions

Ex: addPrime 1 :: Int → Int
↳ single argument because waiting on second input and will increment based on the value passed

→ Bracket to the right avoid extra ()

Ex: Int → Int → Int → Int
↳ Means: Int → (Int → (Int → Int))

→ If no arrows means bracket to left

Ex: mult x y z
↳ Means: ((mult x) y) z

→ A polymorphic function is overloaded if it's type contains one or more class constraints.

Ex: 'a' + 'b' → Since it's not Numeric will give an error

→ Haskell has a number of classes (Num, Eq, Ord)
Num → Numeric
Eq → Equality
Ord → ordered

→ Figure out the type of the function first before defining the function.

Note exercies for this lecture in typesAndClasses Exercises.hs