Pre-defined types

- We already saw function types: `a -> b`
- We also saw elementary types: `Int`, `Float`, `Double`, `Char`, and so on
- Tuples group multiple types: `()`, `(a, b)`, `(a, b, c)`, and so on

```
harmonicMeanT ::  (Double, Double) -> Double
harmonicMeanT     (x, y)      =  (2 * x * y)/(x + y)
```

```
harmonicMeanT ::  (Double, Double) -> Double
harmonicMeanT pxy
   =  (2 * (fst pxy) * (snd pxy))/(fst pxy) + (snd pxy))

fst :: (a, b) -> a  -- functions defined in Prelude
snd :: (a, b) -> b
```

Lists

- Lists are either empty: `[]`

Pronounced "nil"

- …or consist of a head and a tail: `x : xs`

Pronounced "cons"

- Lists are homogenous — all elements in one list have the same type
- List are parametric — different lists may contain elements of different type

Some operations on lists

- Length of a list
- Concatenating two lists
- Reversing the elements of a list
- Mapping a function over a list

In Haskell!