Functors

A way of containing values, without changing them, except through `fmap`.
There is no way to create a container.

```haskell
class Functor f where
  fmap :: (a -> b) -> f a -> f b
```

```haskell
instance Functor [] where
  fmap :: (a -> b) -> [a] -> [b]
  fmap _ [] = []
  fmap f (x : xs) = (f x) : (fmap f xs)

instance Functor Maybe where
  fmap :: (a -> b) -> Maybe a -> Maybe b
  fmap _ Nothing = Nothing
  fmap f (Just x) = Just (f x)
```

A valid instance of Functor must satisfy the Functor laws:

1. `fmap id = id`
2. `fmap g . fmap f = fmap (g . f)`

Gives you the same result more efficiently: compiler optimisations!

Can use `deriving Functor`