1 bbox to Functor and Applicative

1.1 Functor

What is Applicative in Haskell?
Functor - it takes the value from a bbox and compute it and wrap the result with a bbox

```
class Functor f where
    return:: a
    fmap::(a -> b) -> (f a -> f b)
instance Functor f where
    return ...
    fmap ...
```

Morphism from Category to Category, convert object to object
and convert arrow/function to arrow/function
Functor satifies some laws:
There exists Identity and satify associative laws
Functor has identity, or left and right identities
and satisfies associative laws

In Haskell, List is Functor: The morphism is fmap
objects are the elements in List
function is just lambda function $(->)$

```
([])::x = [x]
fmap::(a->b) ->(f a -> f b)
```

([]) is morphism that converts object to object
fmap is morphism that converts function to function

1.2 Applicative

If you know Functor, then you will know what is Applicative
Applicative - is just take two values from two boxes and compute them
and wrap the result into a box

It sounds complicated, but it is nothing more than a box
Functor and Applicative are pretty much like a bbox
All you need to do is wrap and unwrap the boxes,wrap and unwrap your presents like in Chrismas

1.3 Examples

List is like a bbox contains zero or more items
Maybe is like a bbox contains Nothing or one item