1. Monads are just applicative functors that support **>>=**
   1. The **>>=** function is called *bind*.
   2. When we have a normal value and a normal function a -> b it is really wasy to feed the value to a function. We just apply the function to the value normally and that is it. However, when dealing with values that come with a context, we need to determine how these **fancy values** are fed to functions and how to account for their behavior.
2. Maybe monad
   1. **>>=** takes a monadic value and a function that takes a normal value.
   2. It returns a monadic value and manages to apply that function to the monadic value. It does this even though the function takes a normal value.
   3. In the case of Maybe, >>= would take a **Maybe a** value and a function of type **a -> Maybe b** and apply the function to the **Maybe a**.
3. Monad Type Class
   1. Monads constitute their own type class
   2. The first function that a Monad class type defines is **return**
      1. This is the same as **pure** from the applicative type class
      2. The type is **(Monad m) => a -> m a**
      3. Return takes a value and puts it in a minimal default context that still holds the value. In other words, return takes something and wraps it in a monad.
      4. Return has nothing to do with ending function execution as is other languages. It takes a normal value and puts it into a context.
   3. Monads can be neatly used in a chain of monadic applications with **>>=**
      1. **>>=** preserves the context of the value to which it applies functions
      2. In this case the context was that the values were values with failure; so when we applied functions to such values the possibility of failure was always taken into account
4. **do** Notation
   1. Monads are so useful they get their own syntactic sugar
   2. do glues together monadic values in sequence
      1. This saves on constantly writing chained **>>=** calls
      2. This allows us to extract values from Monads in a fluent manner.
   3. do expressions are just different syntax for chaining monadic values