# 1 Arguments and parameters

A value that can be used as an argument to a function is a *first-class* value.

Haskell is *lexically scoped*.

Incomplete pattern matches applied to data they don't handle will return *bottom*, a non-value used to denote that the program cannot return a value or result.

# 2 Pattern matching against data constructors

Some data constructors have parameters, and pattern matching can let us expose and make use of the data in their arguments.

**Definition : newtype**

newtype is a special case of **data** declaration. It permits only 1 constructor and only one field.

# 3 Higher-order function

**Definition : Higher-order function**

*Higher-order functions* are functions that accept functions as arguments. Functions are values.

Guards always evaluate sequentially, so it should be ordered from the case that is most restricive to the one that is least restrictive.

## Function composition and "pointfree" style

**Definition : Currying**

*Currying* is the process of transforming a function that takes multiple arguments into a series of functions which each take 1 argument and return one result.

And don't use error.