What is the basic syntax of the `data` keyword?

What are value constructors, really?

What aren't they?

Literals can be thought of as what?

How can you make your typeclass inherit another one?

When there's only one value constructor for a type, it's common to do what?

How do you export types from a module?

What happens if you don't export any value constructors?

Give an example of a standard library type like this.

What is record syntax?

Give its syntax.

What use-site convenience does record syntax provide?

Define `Maybe`.

When should you use type parameters?

When should you not?

Why is there a strong convention against putting typeclass constraints in data declarations?

What's the catch when deriving `Eq`?

What is the default interpretation of deriving `Ord`?

What is a value constructor with no parameters called?

What syntactic convenience to `Enums` have?

Describe type synonyms?

What abilities do type synonyms have over Scala's type aliases?

What is a concrete type?

Describe `Either`.

What is it used for.

Demonstarte recursive types by defining `List` with standards syntax.

Do the same with record syntax.

What is a fixity declaration?

Give its syntax.

Give the `data` definition of a tree.

Give the basic typeclass syntax.

Manually create an instance of `Eq`.

How can a typeclass implement all its functions yet require that instances implement some of them?

What is *subclassing* in Haskell?

Given an example.

How can you find information about names that aren't expressions?

What's the `Functor` typeclass?

Give a partial implementation.

What is a *kind*?

How can the be found using `ghci`?