Where does Coq define booleans and numbers?

What's a "type" in Coq?

As an example of an enumerated type, define the boolean type.

Define a boolean negation function.

Define a function for boolean conjunction.

Make a named assertion that `~true` is `false`, then prove it.

Name three ways to check that a function works.

Apply negation to the boolean `true`.

What are Coq's names for boolean and/or/not?

How do you fill in a hole in a `Definition`? In an `Example`?

How does Coq write the type of a boolean conjunction function?

What does the `Check` command do?

How will we use the module system?

As an example of a type with a sum constructor, define `nat`.

When we use `Inductive` to define a type, we should see it as what?

What is the fundamental difference between a data constructor and and functions?

Name some keywords that can introduce a function.

What kind of recursion does Coq allow?

What notational convenience does Coq provide for multiple parameters of the same type?

How does one match on *multiple* expressions?

How is "language support" introduced for some definitions?

Name two kinds of language support available.

How can one choose between multiple notation interpretations for an expression.

Which tactic is like `simpl` "on steroids"?

The `reflexivity` tactic implicitly does what?

What's the difference between the simplification of `simpl` and that of `reflexivity`?

What does the `intros` tactic do?