Where does Coq define booleans and numbers?

What's a "type" in Coq?

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

Name three ways to check that a function works.

How might you create "unit tests"?

Apply negation to the boolean `true`.

How do you fill in a hole in a `Definition`/`Fixpoint'? 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?

What is an enumerated type?

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

What "magic" does Coq provide for natural numbers?

What is the fundamental difference between a data constructor 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?

What is an underscore in the context of `match` 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?

What is the syntax of `intros`?

Describe the `rewrite` tactic.

What does rewriting *left-to-write* mean?

How are are propositions with multiple hypothesis written?

Why doesn't simplification be used to prove all theorems?

Give the syntax of the `destruct` tactic.

Why don't we say `destruct b as [true | false].`?

`destruct` proves a theorem about an enumerated type for each possible ...

Don't confused `case` with ...

What is the syntax of the `induction` tactic?

How can you create sub-theorems without creating a new top-level name?

What is a common non-stylistic reason for using `assert`?