In the standard library.

A set of data values.

```
Inductive bool : Type :=
  | true : bool
  | false : bool.
```

```
Definition negb (b : bool) : bool :=
  match b with
  | true => false
  | false => true
  end.
```

```
Definition andb (b1 : bool) (b2 : bool) : bool :=
  match b1 with
  | true => b2
  | false => false
  end.
```

Argument and result types.

```
Example test_negation:
   (negb true) = false.
Proof. simpl. reflexivity. Qed.
```

- Use `Eval` on a test case and observe the result.
- Use `Example` to record expected result, then as Coq to verify.
- "extract" function `Definition` to OCaml, Scheme, or Haskell.

```
Eval simpl in (negb true).
```

Surround the fragments with square brackets.

```
Definition admit {T : Type} : T. Admitted.
```

It fills in the hole in an incomplete Definitions.

`admit` fills in holes in `Definitions`.
`Admitted` fills in holes in `Examples`.

```
bool->bool->bool
```

It causes Coq to print the type of an expression.