```
Require Import ModuleName
```

It's a well-formed expression.

*Expressions* are terms that can be thought of as programs.
*Types* are terms that determine if a type is well-formed and obeys accompanying constraints.

A period.

... opened and closed, with the most recently opened scopes taking precedence.

```
Open Scope scope.
Close Scope scope.
```

```
Locate "something or other".
```

Use underscores for identifiers in the string.
The bottom interpretation takes precedence.

A name associated with a scope used to explicitly provide the interpretation scope for a term.

```
(term)%key
```

```
Print Scope scope
```

Determining if a term is well-formed and what its type is.

```
Check term.
```

In `nat_scope`, whose delimiting key is `nat`.

In `Z_scope` whose delimiting key is `Z`.
It is provided in `ZArith`.

Coq has no type coercion.

There is however the possibility of defining such a conversion using implicit coercions.

- *Atomic types*, represented by single identifiers.
- *Arrow types*, of the form `Type1 -> Type2`.

... either interpretation scope (a collection of notations) or a scope in the usual sense (the textual area where a binding is active).

```
(ident : SomeType)
(ident := term : someType)
```

Global scopes are active for the remainder of development.
Local scopes are active only in the current *section*.

The environment contains global bindings while the context contains the local ones.

`Reset Initial`

It brings you to an empty context and the initial environment.

`Reset identifier`

Brings you back to the state just after the given identifier was introduced.