

Language Modules

## **Modules**

Alloy **modules** are similar to programming languages and act as the namespaces. Alloy comes with a standard library of utility modules.

# **Simple Modules**

```
open util/relation as r
```

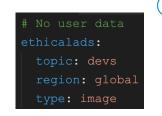
Imports must be at the top of the file. Modules may import new signatures into the spec.

Modules can be imported multiple times under different namespaces.

### **Namespaces**

A module can be namespaced by importing as a name. Name with /. This is also called a **qualified** import.

```
open util/relation as r
-- later
r/dom
```



Al-powered ad network for devs. Get your message in front of the right developers with EthicalAds.

www.ethicalads.io

## **Parameterized Modules**

Ads by EthicalAds

A parameterized module is "generic": its functions and predicates are defined some arbitrary signature. When you import a parameterized module, you in a signature. Its functions and predicates are then specialized to be defined for that

signature.

```
open util/ordering[A]
sig A {}
run {some first} -- returns an A atom
```

Normally ord/first returns an abstract elem. By parameterizing the module with A, the function now returns an A atom.

The input must be a full signature and not a subset of one.

A parameterized module can be imported multiple times using Namespaces.

#### O Note

The following built-in modules are parameterized: ordering, time, graph, and sequence.

# **Creating Modules**

The syntax for a module is

```
module name
```

At the beginning of the file.

### **Private**

Any module predicate, function, and signature can be preceded by <a href="private">private</a>, which means it will not be imported into other modules.

module name
private sig A {}

# **Creating Parameterized Modules**

module name[sig]

-- predicates and functions should use sig