Table of Contents

cmp

This package provides utility functions for comparing values.

Generic Comparison Functions

The library provides generic comparison functions that work with any type implem
enting the Compare trait:

```

test "generic comparison" {

  inspect(@cmp maximum(3, 4), content="4")
  inspect(@cmp minimum(3, 4), content="3")
}
```

Comparison by Key

With @cmp.maximum_by_key() and @cmp.minimum_by_key(), it is possible to comp
are values based on arbitrary keys derived from the them. This is particularly u
seful when you need to compare complex objects based on some specific aspect or
field.

```

test "cmp_by_key" {
  struct Person {
    name : String
    age : Int
  } derive(Show)


  let s1 = "hello"
  let s2 = "hi"
  let longer = @cmp maximum_by_key(s1, s2, String::length)
  inspect(longer, content="hello")


  let alice = { name: "Alice", age: 25 }
  let bob = { name: "Bob", age: 30 }
  let younger = @cmp minimum_by_key(alice, bob, p => p age)
  inspect(younger, content="{name: \"Alice\", age: 25}")


  let p1 = ("first", 1)
  let p2 = ("second", 1)
  let snd = (p : (_, _)) => p 1
  assert_eq(@cmp minimum_by_key(p1, p2, snd), p1)
  assert_eq(@cmp maximum_by_key(p1, p2, snd), p2)
}
```