Functional Programming in Erlang


futurelearn.com/courses/functional-programming-erlang/1/steps/161070

In this exercise you'll familiarise yourself with the fundamental data types and operations of the rock-paper-scissors game, and define some functions for yourself.

Getting started

Two players choose one of rock, paper and scissors after counting to three and making one of these gestures:

- A clenched fist representing a rock
- A flat hand representing a piece of paper
- Index and middle figure extended representing a pair of scissors

If they choose the same gesture, neither wins; if not, the result is decided this way:

- Rock defeats scissors, because a rock will blunt a pair of scissors
- Paper defeats rock, because a paper can wrap up a rock
- Scissors defeat paper, because scissors cut paper

Suppose that we want to model these three choices in Erlang. One option would be to use integers, characters or strings, but none of these is ideal, for a number of reasons. For example, if we choose integers, what numbers should we associate with `rock`, which with `scissors`?

We'll choose to use the **atoms** `rock` `paper` `scissors` to represent the three choices.

Win and lose

Now we'll define some functions to use here. Let's start by defining beat, which for a move tells us which more beats that one. For instance,

```
beat(rock) = paper
```

Complete the definition of `beat`. What does your definition do if you pass it something which isn't one of the three moves? Can you think of other things you might do in that case?

Next, define the function `lose`, which tells us which moves lose if played against the argument played … for example,

```
lose(rock) = scissors
```

A round

Define a function `result` which when applied to two plays gives the result, from the point of view of the first. For

example:

```
result(rock,paper)
```

should be to lose. In defining this function you should think about how to represent win / lose / draw. Try to think of two alternatives, and discuss the advantages and disadvantages of each. Finally, give a definition of `result` for your chosen representation.

A tournament

A tournament is a series of rounds – each round is a single choice from the two players, which we'll call `left` and `right`. Suppose that the choices are given as two lists; give the `tournament` result as an integer, so that the number counts the difference between the number of wins for left and right. A positive value is an overall win for `left`, a negative for `right`, and zero represents an overall draw. For instance:

```
tournament([rock,rock,paper,paper],[rock,paper,scissors,rock] = -1
```

Which higher-order functions from the `lists` module can you use in computing this solution?

We'll go over a solution to these exercises in the next step, but do use the comments on this page to discuss your solutions, problem solving strategies and observations to take part in the conversation yourselves.
