Common errors in writing and running Erlang programs

The aim of this note is to give you some hints about common mistakes made in Erlang.

In the shell

You have defined (correctly!) the `fac/1` function in your module `foo.erl`, but it doesn't work in the shell:

```
1>c(foo).
{ok,foo}
2>fac(6).
** exception error: undefined shell command fac/1
```

That's because you haven't called it in **fully qualified form**:

```
3>foo:fac(6).
** exception error: undefined function foo:fac/1
```

And the problem here is that you haven't **exported** the `fac` function from `foo.erl`.

Syntax

Erlang programs have punctuation that's close to English: expressions within a function body are separated by commas, function clauses are separated by semi-colons and a function definition is ended with a full stop (period). If you get these mixed up, then there will probably be errors.

```
fac(0) ->
    1,
fac(N) ->
    N*fac(N-1).

errors.erl:7: syntax error before: '->'
```

```
fac(0) ->
    1.
fac(N) ->
    N*fac(N-1).

errors.erl:6: function fac/1 already defined
```

```
fac(0) ->
    1;
fac(N) ->
    N*fac(N-1);

errors.erl:7: syntax error before:
errors.erl:2: function fac/1 undefined
```

Syntactic similarities

It's possible to confuse various different things in Erlang:

- Atoms and strings:
  - atoms are enclosed in single quotes 'atom'
  - strings are enclosed in double quotes "string"

- Variables in Erlang begin with a capital letter – Variable – whereas module and function names are atoms, and so begin with small letters: function.
