How can you compile and load a module into the `erl` shell?

```
c(module.erl).
```

Now that module's functions can be accessed with:

```
module:func
```

What do Erlang programs begin with?

```
-module(filename).
```

... for a program defined in filename.erl.

How can you use methods in other modules without module qualification?

```
-import(filename, [func1/arity1, ..., funcN/arityN]).
```

How can you make your module's functions available to other modules?

```
-export(filename, [func1/arity1, ..., funcN/arityN]).
```

or

```
-compile(export_all).
```

What are the Erlang case rules?

Similar to Prolog, and Haskell, functions and atoms begin with a lower case, variables begin with a capital letter or an underscore.

What is an atom?

What are its lexical rules?

An atom is a word that stands for itself.
It begins with a lowercase letter or is enclosed in single quotes.

What are Erlang's primitive data types?

Integers, floats, strings, atoms, lists, tuples, binaries.

How are lists, tuples, and binaries written?

Lists are enclosed in square brackets, tuples in curly brackets, and binaries in double angle brackets.

What's odd about Erlang's Boolean operators?

They don't short-circuit by default. For short-circuiting you have `andalso` and `orelse`.

How are "greater or equal to" and "less than or equal to" expressed?

>= and =<.

not <=.

What are the equality operators?

How are they used?

`==` equal to
`/=` not equal to
`=:=` exactly equal to
`=/=` not exactly equal to

The non-exact versions will coerce its into floats. Otherwise use exact versions to give better hints to the compiler.

What is $=$ in Erlang?

Pattern matching, not assignment. "Assignment" is just a simple case of pattern matching.

What is the syntax of `case` expressions?

```
case Expression of
    Pattern1 [when Guard1] -> Expression_sequence1;
    ...
    PatternN [when GuardN] -> Expression_sequenceN
end
```

What is an expression sequence?

What is its value?

A sequence of expressions separated by commas.

The value of the last expression evaluated.

Give the syntax of `if` expressions.

```
if
    Guard1 -> Expression_sequence1;
    ...
    Guard2 -> Expression_sequence2
end
```

What happens if no guard in an `if` expression is true?

How can this be avoided.

An error will be returned.

One can use the `true` atom as the final guard, although it is usually better to use something more explicit than `true` when possible.

What is the huge restriction on guards?

To enforce no-side-effects, they cannot be user-defined.

You can use type tests, many operators, and a number of built-in functions.

How do you access the length of a list?

Of a tuple?

```
length(List)

size(Tuple)
```

Give the syntax of named functions.

```
name(Patterns1) -> Expression_sequence1;
...
name(PatternsN) -> Expression_sequenceN.
```

Give the syntax of anonymous functions.

What are they often used for?

```
fun(Patterns1) -> Body1;
    (Patterns2) -> Body2;
    ...
    (PatternsN) -> BodyN
end
```

They are often used as parameters to other functions.

Give the syntax of list comprehensions.

```
[Expression || Generator, GuardOrGenerator, ..., GuardOrGenerator]
```

The expression typically makes use of variables defined by a generator.
Guards are simply boolean expressions.
Generators are of the form `El <- List`.

Give a generator that doubles the members of a list.

```
[X * 2 || X <- [1, 2, 3, 4]]
```

Which list functions do not need to be imported or qualified?

`hd` (head), `tl` (tail), `length`

Create a range of integers.

```
lists:seq(From, To)
lists:seq(From, To, Step)
```

Note: `To` is inclusive, unlike in Scala/Python/etc

Get a line from stdin.

Put a line to stdout.

```
Line = io:get_line(Prompt).

io:format(FormatString, ListOfData).
```

How do you write to a file?

```
{ok, Stream} = file:open(FileName, write),
io:format(Stream, FormatString, ListOfData),
file:close(Stream).
```

What are the reserved sequences in format strings?

`~s` a string
`~w` a value in its standard syntax (e.g., strings as lists of integers)
`~p` a value, pretty printed (e.g., strings with quotes around them)
`~n` or `\n` newline

What is the comment syntax?

% causes the rest of the line to be discarded.

`filter` and `map` take which arguments?

First the function to apply to the members of the list, and then the list.

How can you get truncating division?

```
0> 2 rem 3.
2
```

How can you use previously defined functions as parameters to other functions?

Create an anonymous function in-line using the function's full name, *funcName/arity*.
E.g.,

```
get_red() -> filter(fun is_red/1, fruit()).
```

What are strings, really?

Lists of ASCII integer values.

What do `lists:all` and `lists:any` do?

The same as Scala's `forall` and `exists`.

How is Erlang's `zip` more restrictive than other languages'?

The two lists must be of the same length.

Append two lists.

```
lists:append([1, 2], [3, 4])
```

or

```
[1, 2] ++ [3, 4]
```

result in

```
[1, 2, 3, 4]
```

What do `takewhile` and `dropwhile` do?

`takewhile` scans through a list until the predicate fails for the first time, at which point it discards the failed element and everything after.

`dropwhile` scans through the list, discarding everything until the predicate holds for the first time.

What does `partition` do?

It creates a tuple of two lists. The first list holds the elements of the input list for which the predicate held, and the second holds the elements for which the predicate failed.

How can you run code in a new process?

```
Pid = spawn(Function)
```

How can you generate a "return address" to send with your message?

`self()` results in the Pid of the executing process.

Give the syntax of `receive`.

```
receive
    Pattern1 [when Guard1] -> Expression_sequence1;
    ...
    PatternN [when GuardN] -> Expression_sequenceN
after Timeout ->
    TimeoutExpressionSequence
end
```

How is Erlang's `loop()` different from Scala's?

`loop` in Scala is a control structure in the actors library that does the looping for you.
In Erlang `loop` is just the name of a function you're calling so you must make the recursive call at the end of each `receive` case.

What's an idiomatic way to set up an Erlang remote procedure call server?

Provide a function that takes as an argument the Pid of the instance of the module to send the message to.
Send the message there, wait for a response, and then reply.

Ultimately a user may start several processes, but after that point will interact with the blocking rpc function.

Describe Pid registration.

Registering a Pid makes it globally available.

`register(AnAtom, Pid)` gives `Pid` a name

`unregister(AnAtom)`

`registered() -> [AnAtom :: atom()]`

returns a list of all registered processes

`whereis(AnAtom) -> Pid | undefined`

gets Pid of a registered process, or undefined if no such process

What's the difference between `spawn` and `spawn_link`?

`spawn` creates a processes independent from the current one.

`spawn_link` creates a new process linked to the current one, such that if the new process exits non-normally, so will the current one.

How can you introduce/remove links between two running processes?

`link(Pid)` and `unlink(Pid)`

How can you terminate the current process?

How can you send exit signals *without actually exiting*?

`exit(Reason)` exits and sends signals to linked processes.

`exit(Pid, Reason` sends the exit signal to the given process, but doesn't terminate the current process.

How can you link to a process while reserving the right to handle the partner process's exit however you like?

```
process_flag(trap_exit, true)
spawn_link(Function)
```

Now exit signals are converted to regular `{'EXIT', From, Reason}` messages.
However, if the reason is `kill`, exit will still be forced.

Why isn't `try/catch` much used in Erlang?

Crashed processes are normally handled through linking, not in `catch` sequences.

What is a common strategy for making functions tail recursive?

Create a helper function with an additional argument that will serve as an accumulator.