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 doe 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
```

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)
```

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(S, FormatString, ListOfData),
file:close(S).
```

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.