U0EUHKVGB : In which case, it will crash on evaluation of the function
U0EUHKVGB : Same goes for if you wrap something in a case statement, or an if statement.
U31AUNK6F : <@U3SJEDR96> ohw nice i did not know that is what strict mend. Useful thanks
U3SJEDR96 : it's also where the subtle difference between `always "foo"` and `(\_ -&gt; "foo")` comes in
U0EUHKVGB : Indeed, see number 4 in this:
<https://medium.com/@eeue56/top-6-ways-to-make-your-elm-app-crash-at-runtime-562b2fa92d70>
U5LFUHH19 : To pattern-match on an empty List, I use "[]." So how do I match (in a "case") an empty Dict?
U23SA861Y : yeah, regex... so infuriating
U0EUHKVGB : <@U5LFUHH19> Use an if statement + `Dict.empty`
U3SJEDR96 : or `case Dict.toList myDict of [] -&gt; "empty!"` (though, if it's not empty, that's a waste)
U23SA861Y : <@U0EUHKVGB> you list function equality as a runtime crash, but shouldn't (wouldn't) the compiler catch that?
U5LFUHH19 : ```case List.length (Dict.keys optionList) of
    0 -&gt; ...
    _ -&gt; ...
```

U3SJEDR96 : could just match on `Dict.size` in that case :slightly_smiling_face:
U3SJEDR96 : <@U23SA861Y> it could, I suppose, if there were another pseudo-typeclass `equatable` or something like that, which would be "any value except functions, json-encode values, and possibly some others".
U5LFUHH19 : <@U3SJEDR96> I didn't know about Dict.size.  Thanks!
U236M9FH9 : If all you need from size is if it's 0 or something else, use `isEmpty` &amp; and if/then like eeue56 said
U236M9FH9 : Otherwise you're traversing the whole dict
U5LFUHH19 : I tried `isEmpty` the other day, and it matched in all cases.  Or something like that.  <@U0EUHKVGB> helped me with it.  In this particular case, I'll need the keys anyway, so I can get them in a "let" .
U5LFUHH19 : The REPL doesn't like this: ```[first, rest] = split "__" "foo__bar_baz"```
U23SA861Y : <@U3SJEDR96>  umm why would you need that. If functions aren't comparable, they aren't comparable. Just don't define (==) for function types
U23SA861Y : nvm, I got what your saying
U5LFUHH19 : Variations with `()` and nothing fail, too.
U23SA861Y : but comparable is a thing?
U3SJEDR96 : yep
U23SA861Y : or is comparable considered to have an ordering
U0LPMPL2U : comparable is if something is greater than or less than something else
U5LFUHH19 : `first :: rest = split "__" "foo__bar_baz"`
U3SJEDR96 : `comparable`, `number` and `appendable`, but they're explained in the faq better than I can do so here :slightly_smiling_face:
```