U29JSAR9S : makes sense, and good advice - cheers
U29JSAR9S : since I've got someone who's obviously a few ElmFu belts ahead of me - is there a better way of writing something like this:
```
hexToDec : String -&gt; Maybe Int
hexToDec hex =
    String.split "" hex
        |&gt; List.map hexCharToDec
        |&gt; \decimals -&gt;
            if List.any isNothing decimals then
                Nothing
            else
                List.map (withDefault 0) decimals
                    |&gt; List.reverse
                    |&gt; List.indexedMap toActualValue
                    |&gt; List.sum
                    |&gt; Just
```

??

U48AEBJQ3 : @dans <http://package.elm-lang.org/packages/rtfeldman/hex/1.0.0/Hex#fromString> ?
U5ABF3BH7 : <@U0CLDU8UB> Sorry for the delay, yes, I need something that opens a list with all options without typing and something that I can also filter it by typing.
U48AEBJQ3 : er, <@U29JSAR9S>
U29JSAR9S : <@U48AEBJQ3> I was doing it as a learning exercise, figured it was a solved problem but seemed like an interesting one to solve again
U48AEBJQ3 : Well, then: <https://github.com/rtfeldman/hex/blob/1.0.0/src/Hex.elm#L14>
U0LPMPL2U : In your implementation, I would probably try a combination of `List.foldl` and `Maybe.andThen` rather than the conditional + `List.map` + `List.sum`
U29JSAR9S : cheers, I'll have a look into the `andThen` function
U0LPMPL2U : Actually, summing Maybes should be done with `Maybe.map2 (+)`, not `Maybe.andThen`
U0LPMPL2U : ```sumMaybes : List (Maybe Int) -> Maybe Int
sumMaybes digits =
  List.foldl (Maybe.map2 (+)) (Just 0) digits
```

U29JSAR9S : <@U0LPMPL2U> yep, that works - I had to change my `toActualValue` function to handle Maybes but after that I was able to just use your `sumMaybes`
U48AEBJQ3 : <@U29JSAR9S> The original `toActualValue` was `Int -&gt; Int -&gt; Int`? You should be able to go `Maybe.map &amp;lt;| toActualValue`
U29JSAR9S : I should ask more questions in this channel, getting lots of good suggestions :slightly_smiling_face:
U29JSAR9S : in this case though I tried that and I think the function is cleaner moving the Maybe.map inside the `toActualValue` function
U48AEBJQ3 : I would suggest against that. If you want to clean it up, make another helper function to give it a cleaner name.
U29JSAR9S : the function name for `toActualValue` is a bit rubbish though, I know :slightly_smiling_face:
U48AEBJQ3 : Regardless of that, you are mixing concerns by having it deal with `Maybe`. It's good practice to write the function that doesn't need to know about how to take the `Maybe` apart and then inject that information using `map` or `andThen`
U48AEBJQ3 : Also, I think that you can go `Bitwise.shiftLeftBy &amp;lt;| (*) 4` to raise something to the appropriate power.
U29JSAR9S : that makes sense
U23SA861Y : instead of indexed map I'd probably use a fold there
U23SA861Y : `foldr (\v (p,a) -&gt; (p*16,a+p*v)) (1,0) digits`
U23SA861Y : likewise a maybe list function can be constructed```
maybeList: List (Maybe a) -> Maybe (List a)
maybeList l =
  let
   mapped = List.filterMap identity
```

```
  in
   if List.length mapped = List.length l then
      Just mapped
   else
     Nothing
```

U29JSAR9S : so you're suggesting something like:```
hexToDec : String -> Maybe Int
hexToDec hex =
    String.split "" hex
        |> List.map hexCharToDec
        |> maybeList
        |> (Maybe.map << Tuple.second << foldr (\v (p,a) -> (p*16,a+p*v)) (1,0))

```
?

U23SA861Y : it would do the thing, I'd probably split out the `Tuple.second &lt;&lt; foldr` into a named function because it's a bit dense and give it a nice name
U23SA861Y : accumlate list
U23SA861Y : or have it be accumulateWithStride and have the 16 come in as a parameter
U23SA861Y : The only thing not specified would be what happens to the empty list
U48AEBJQ3 : I think I would like `sumDigits : Int -&gt; List Int -&gt; Int` better.
U48AEBJQ3 : But names are hard.
U29JSAR9S : just trying to test it out as I'm not convinced it does actually solve it (I think it might miss the case when you don't multiply by a power at all for the first hex digit) - getting:
```
the argument to function `length` is causing a mismatch.

120| List.length mapped
                        ^^^^^^

Function `length` is expecting the argument to be:

    List b

But it is:

    List (Maybe b) -> List b
```

for the maybeList function - and I'm too tired to figure out whats up myself :slightly_smiling_face:

U23SA861Y : ahh the line should be `List.filterMap identity l`
```