U23SA861Y : well I don't know about the pipeline but using the base decoder package
U23SA861Y : `EntryDecoder = JD.map2 RolodexEntry (JD.field "id" <http://JD.int|JD.int>) (JD.field "name" JD.string) `
U23SA861Y : `CategoryDecoder = JD.map2 RolodexList (JD.field "category" JD.string) (JD.field "list" (JD.list EntryDecoder))`
U23SA861Y : `ListDecoder = JD.list CategoryDecoder`
U23SA861Y : where JD is `import Json.Decode as JD`
U5ABF3BH7 : <@U23SA861Y> Thanks a lot!
U5HM74BD0 : I asked a variant of this question a couple days ago. But it's a little more complicated than what I asked. I have a `List` of records like this: `[{ab = 3, hits=2}, {ab=2, hits=1}]`. I'd like to transform that list into a tuple of records: `({ab=3, hits=2}, {ab=2, hits=1})`. How can I write a function to do that? If the list is longer than 2 records, I can default it out to a dummy tuple: `({ab=0, hits=0}, {ab=0, hits=0})`.
U4872964V : and if the list is shorter?
U4872964V : Sound like you just should do a `case` on the list
U23SA861Y : a tuples size needs to be known at compile time
U5HM74BD0 : When I put this function into the elm-repl, I'm not getting it to compile:```&gt; listToTuple l = \
    case l of \
        {a.ab, a.hits} :: {b.ab, b.hits} :: [] -&gt; \
            ({a.ab, a.hits}, {b.ab, b.hits}) \
        _ -&gt; ({ab=0, hits=0}, {ab = 0, hits=0})```

U23SA861Y : you can't do it for a variable length list
U4872964V : you can't pattern match like that
U23SA861Y : if you know the exact size of the list at compile time you can write a function for it but tuples cannot be dynamically sized at run time
U4872964V : but it looks like you don't care about the record contents anyway
U4872964V : so just do `case l of a :: b :: [] -&gt; (a, b)`
U5HM74BD0 : <@U4872964V> Your version does work for me in the repl. But when i annotate it in my program, I get that elm-make compilation error I mentioned a little while ago. Let me see here...
U23SA861Y : why are you trying to do this, it seems like there should be a better way?
U4872964V : that is always a good question :slightly_smiling_face:
U5HM74BD0 : I am parsing a JSON object. That object comes in as a list, but my model represents them as a tuple.
U23SA861Y : is the list a fixed known size or is it variable
U5HM74BD0 : Maybe I need to rethink the model? I know, though, that there will be exactly two items needed for this piece as far as the application is concerned. I'm representing my 2 sons batting averages in the app: "ab" is at-bats, and "hits" is number of hits for each game....
U23SA861Y : are you going to have another kid?
U5HM74BD0 : :slightly_smiling_face: Haha!!
U23SA861Y : that sound like a many problem, I would leave it as a list
U5EL672TU : zero, one, infinite kids
U5HM74BD0 : Alright, I will rework things. I think you all are right, even though there's *no way* we're going to have any more kids!
U23SA861Y : well, not intentionally anyway
U23SA861Y : :stuck_out_tongue:
U5HM74BD0 : I'm too old for that stuff! Too old to learn a new language like Elm, perhaps, much less too old for having another kid!!
U57KYFW67 : Most of the difficulty isn't in the learning of a functional language. It's in the unlearning of all the object-oriented crap :stuck_out_tongue:
U23SA861Y : Its funny because if you were taught FP first, then imperative would seem absolutely nuts
U57KYFW67 : So often, "How do I do X" has to be answered with "?"
U57KYFW67 : (mu)
U5HM74BD0 : <@U57KYFW67> Friend, I've got to unlearn OO, along with procedural stuff like Basic I learned back in the 80s... Got a lot to unlearn here, folks!
U57KYFW67 : hah
U57KYFW67 : It's a very Zen experience. You have to think less about "Doing" and more about "Being". What things are, rather than how things change.
U23SA861Y : From a mathmatics background it makes alot of sense because nearly all of mathematics is statements
U4872964V : declarative programming
U5HM74BD0 : So, when I come home in the evenings, I need to unlearn OO and such and learn some FP (using Elm), which is great. Except that OO is what I do during the day, so I go and relearn it again! Ha, it's a vicious cycle. One day I'll retire and stroll along the beach with my grandkids and think, "yep, this is what all that madness was about.

Basic-&gt;OO-&gt;FP-&gt;OO-&gt;FP...." Just so that I could enjoy a nice week at the beach with the little ones....
U23SA861Y : Unless you have an OO language that lets you be sneaky and start including some FP stuff in it
U57KYFW67 : It may seem like infinite recursion, but it actually converges on a fixedpoint: a very practical viewpoint where you recognize the advantages and disadvantages of each way of thinking.
U5HM74BD0 : <@U57KYFW67> I don't doubt you at all. But wow, were these the simple days:```10 x=1
20 print "The number is " + x
30 x=x+1
40 goto 20```
And watch the counter scroll off the screen!
To your point, though, FP does feel a lot like mathematics, algebra 2 and such. I'm finding myself coding much quicker in Elm, even if I screw up by representing the model wrong.

U57KYFW67 : I teach math. And one of the things that I find the FP community is a little misleading over is how "mathematical" FP is. I think math tends to signal to people that it's very numeric, but a better word may be "logical".... but of course, the word "logical" is totally wrong too!
U57KYFW67 : It is the logic of mathematics, but with none of the numbers.
U0K7EBT3J : <@U57KYFW67> what about the name "discrete mathematics"?
U57KYFW67 : Which I think most people would find a little more palettable
U0K7EBT3J : i like to think it is "discrete" cause it doesn't have numbers :stuck_out_tongue:
U57KYFW67 : I actually had my current employer ask me (out of curiosity) what Discrete Mathematics was during my interview.
U23SA861Y : when I say it's more mathmatical I mean that you find things like this `f(x) = x+2, g(k,x) = x^k(x)` in math as opposed to `take a number, now increment it , now put it aside, now take another number`
U23SA861Y : and you most definately don't side up with `f(x) = x + y` in math unless y is a constant
U0K7EBT3J : more declarative then imperative then, borrowing some math symbols