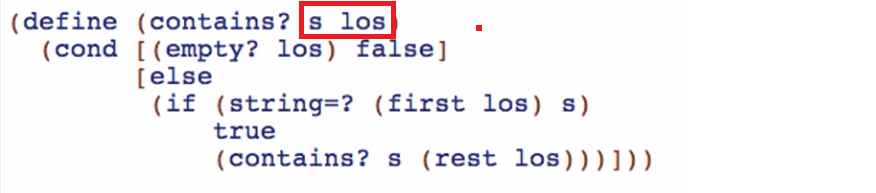
The **final** step of designing an abstract function is to write the signature. Doing this requires three new elements of signature and type notation.

Type inference – other programming languages have this

* Like java, you can determine the type of the object just by hovering at it

Signature for contains? function

1. Check the function

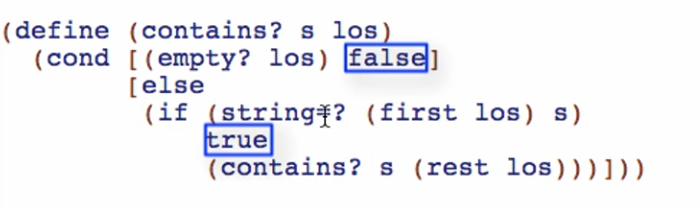


Notice there are 2 arguments that are being consumed, and of course, always 1 type that is produce

So place blanks to the signature for these



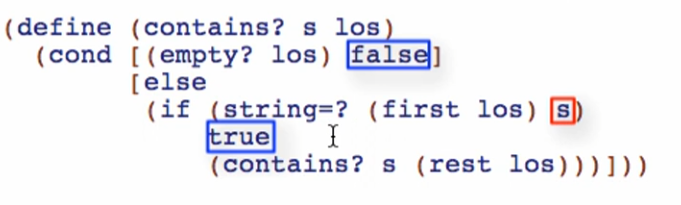
1. Check first all the type that is being produced by the function



In this case, the type of value produced is Boolean



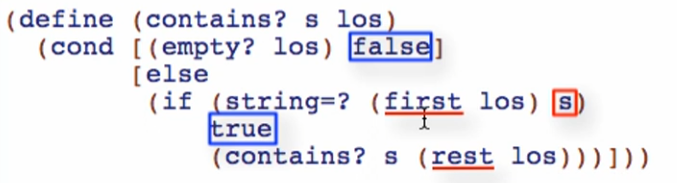
1. Let’s look at the first parameter “s”



string=? takes only Strings as arguments, so s SHOULD be a String

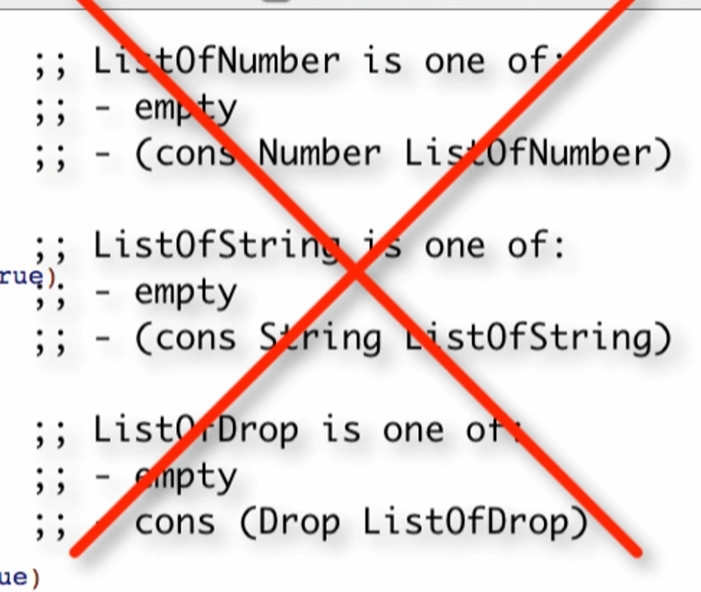


1. Let’s look at the 2nd parameter “los”



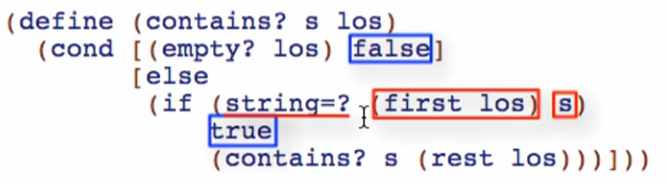
Starting now, we can write lists like this:

 this is just like ListOf\_\_\_\_ from our previous lessons

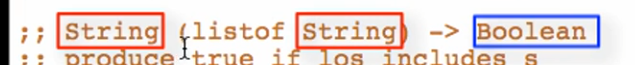


If we are to use this too, we don’t need to have data definitions like this. It will be understood that (listof\_\_\_) is one of empty, and one of (cons \_\_\_ ListOf\_\_\_\_)

We have first and rest primitives, so we know it is a list.

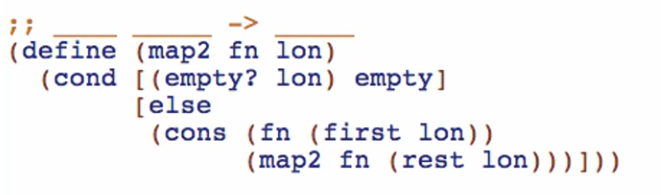


first los is being used as a String here in string=?, so first los SHOULD be a String



Signature for map2 function

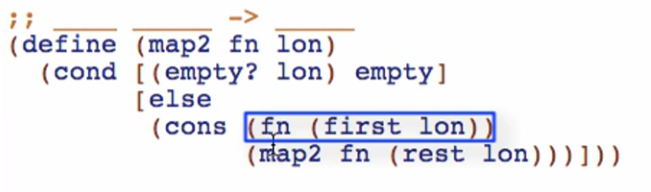
1. Check the function for the number of parameters



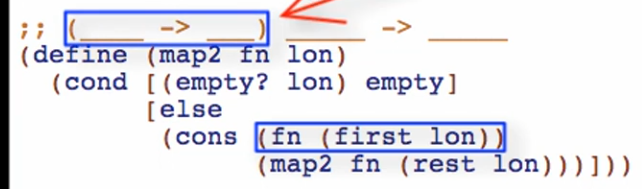
2 parameters = 2 blanks, and always 1 blank for produced

Working through the blanks first:

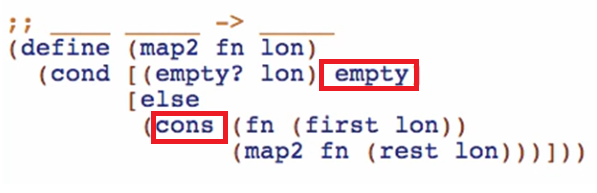
1. Check for the 1st argument: fn



For function types:



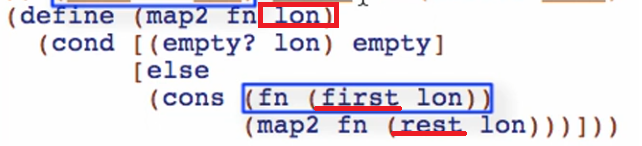
1. Check for the type produced



Empty and cons are keywords that it is producing a list



1. Check for the 2nd parameter

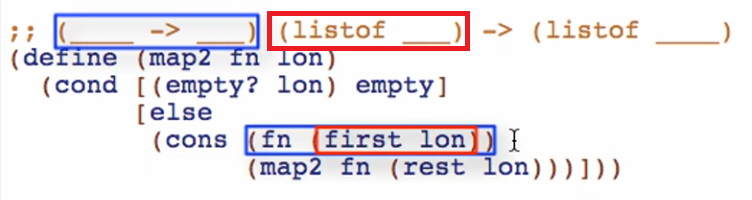


First and rest are primitives for a list!



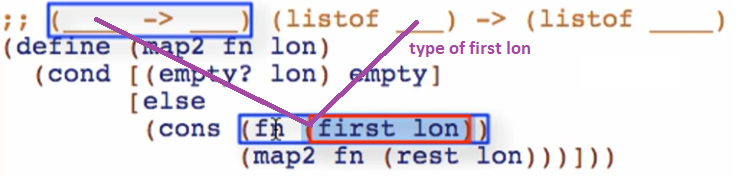
Working through the specific types (not in strict order, you can infer types based on how you view the function)

1. Check the 2nd arg



Notice only “fn” operates on (first lon); map2 never touches first lon

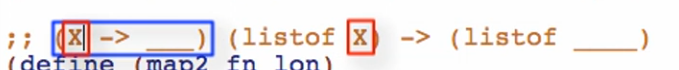
This means that whatever the type of (first lon) is the type of the element of the 2nd parameter, and also the type of the parameter for “fn”



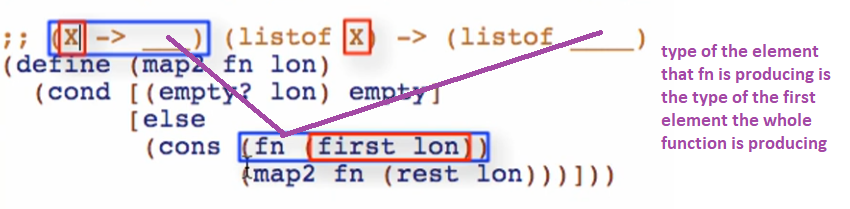
 - means it can accept anything

- this is called a type parameter (by convention, a single upper-case letter)

But “fn” must be prepared to consume X also



1. Checking the produced value



Again we will use type parameter

