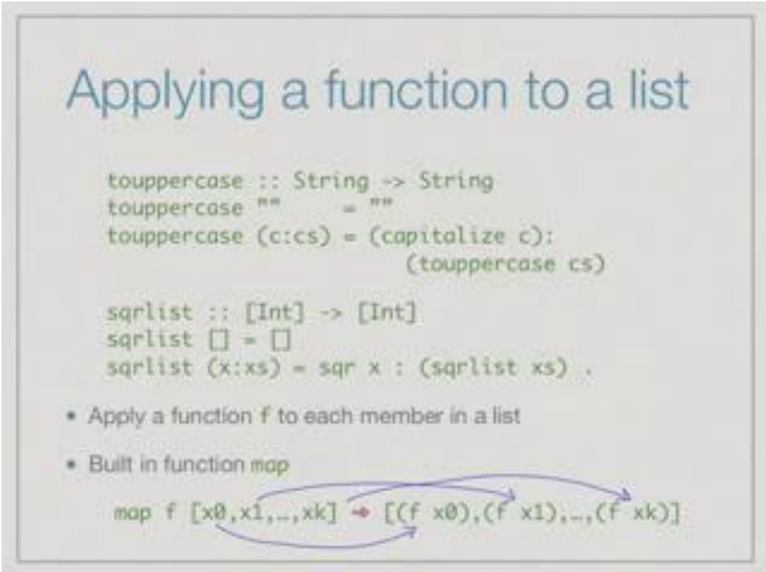
Functional Programming in Haskell
Prof. Madhavan Mukund, S. P Suresh
Chennai Mathematical Institute

Module # 03
Lecture - 03
Lists: Map and Filter

So, we have seen that Haskell support higher order functions, functions that take other functions as arguments and apply them to yet another app. The context of list, map and filter are the two most important higher order functions that one can use.

(Refer Slide Time: 00:19)

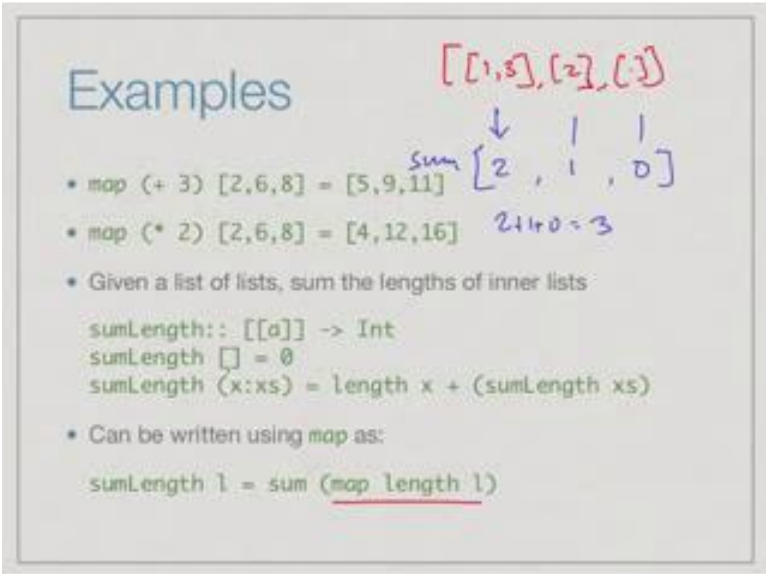


So, to introduce map, let us look at two functions we have seen before, the first function is one that converts entire string into upper case. So, this function is defined inductively using the base function capitalized, which converts a single letter properties. So, to upper case of the empty string is the empty string and if we have an non empty string, then we capitalized the first letter and then recursively upper case the list.

Another function which we have seen is one, which squares all the elements of a list of integers. So, this square of an empty list is an empty list and if we have a non empty list, we square the first number and then recursively square the list. So, in both of these functions we are applying the given function to each element of the list. In the first case we are capitalizing every character in the string, in the second case we are squaring every number in the given list.

So, this is what the built in function map does. So, map in general takes a function f and applies it to every element of the list, so we have a list x 0 to x k, then x 0 will be replaced by f of x 0 and so on. So, we have this kind of function which takes each element and just applies the function one element at a time. So, this produces a new list of exactly the same length is we hold is, in which each exercise we replace it by f of x, so it maps the function to the list.

(Refer Slide Time: 01:49)



So, here is an example, remember we saw last time that if you take an operator like plus, then you can make it a function by using the round bracket. And therefore, if I have plus applied by n m, then the function with one argument consumed plus m, actually adds m to every element that is applied. So, now, if you take the function plus 3, it adds 3 to whatever it is applied to. So, if I map this onto this list, I get plus 3 plus 2, which is 5 6 plus 3 is 9 and 8 plus is 11.

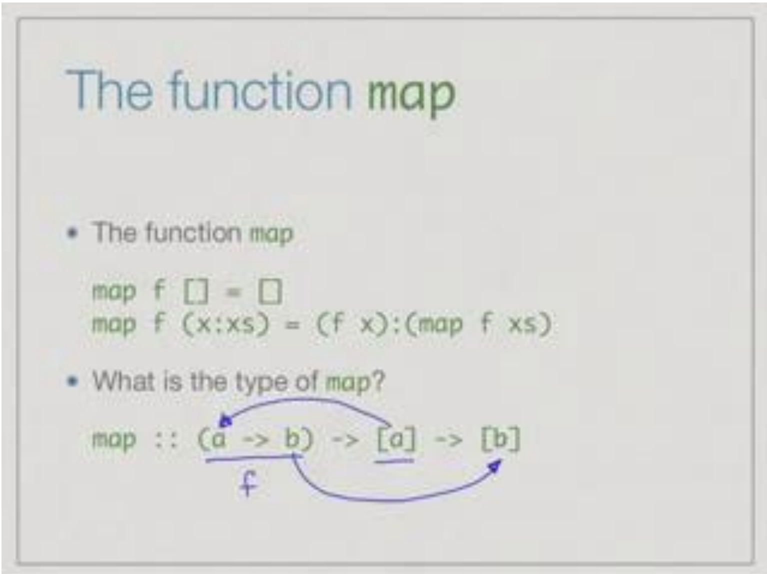
So, this is one easy example of map, here is the similar one is in multiplication. So, now, your 2 times 2 is 4, 6 times 2 is 12, 8 times 2 is 16. So, here is a slightly more involved examples, supposing we have a list of lists, so say we have a list consisting of say 1, 3 and then 5 and then we get an empty list. So, what we want to do is we want to compute the sum of the length of these lists, so the sum of this, the length of this list is 2, the length of this list is 1, the length of this list is 0.

So, for this we want to function that it return 3, this 2 plus 1 plus 0, so it is not the length of the whole list, but the length of the inner list. So, we have a list of any type contain, the list of the list of any type and we want to return an integer and here is the usual inductive definition. We said that the sum of the length of the empty list of the list is 0, because there is no list inside and if I have a non empty collection of list, then I compute the length of the first one and then inductively compute the list.

So, this is the traditional inductive or inductive definition of this function. Now, on the other hand, what we can do is, we can use maps, we say take the list and apply length to each one of them, so list this map. So, this takes for example in the list that we had seen earlier, so the effect of map is to replace this by 2, this by 1 and this by 0. So, now, we have this new list, which is after we map length of l.

And then the built in functions sum adds up all those things, so if I apply sum of these I will get 2 plus 1 plus 0, which is 3. So, sum length can now be successively described in terms of map and sum rather than writing of an inductive definition like this.

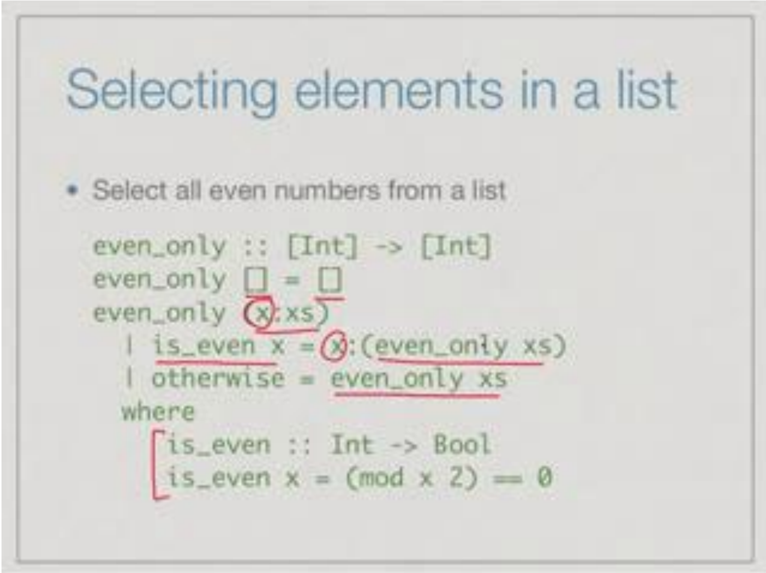
(Refer Slide Time: 04:20)



So, the function map itself can be defined inductively pretty much the way we have been doing the specific cases. So, if you map f to the empty list, then we get the empty list, if you map f to a non empty list, then we apply f to the first element of the list and recursively apply f to all the others. So, what is the type of map? So, map takes of course as input some list, so this is a list of some typing.

And now this is the function that operates an elements of this list, so this function must starts with an input a, but it can produce an output of any type. So, it will be in general of type a to b and the outcome of taking a list of a, now applying a function that goes from a to b is, of course produce a list of p. So, if we ask a type of map it will say, give me a function from some a to some b, give me a list which is compatible with the input type of this function. Because, I want to apply each element of the list and it will intern produce a list, which have the output type of a function.

(Refer Slide Time: 05:37)

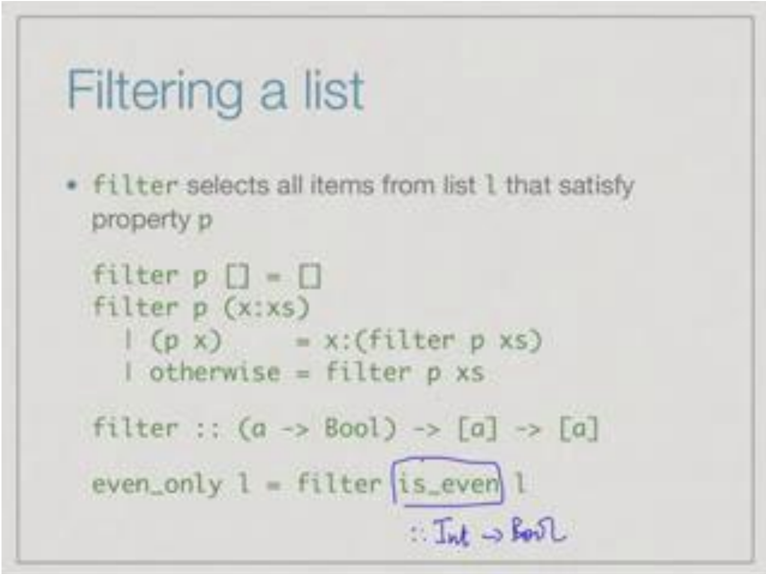


So, map takes the function from a to b, it takes a list of type a and it produces a list of type b, so map is one important higher order function. Now, let us look at the other higher order function, which involves selecting elements in the list. So, here is the specific examples, supposing we have a list of integers and we want to select from the list of integers those that are even. So, inductive definition would say, that if I have the empty list of integers, then I get the empty list of integers.

If I have a non empty list of integers, then I have to decide whether to include the first element or not. So, I have a function here which checks whether a number is even, it just checks the reminder of x divided by 2 is 0, so thus definition of even. So, if x is even, then include x and continue extracting the even numbers in the list, otherwise x is also excluded and just go to the list and this is again a traditional recursive definition of this.

But, we would like to do this in general, so here it says you want to select all the even numbers in general we may have some other property, which is true and false of some elements and pick out all those elements for which the property is true.

(Refer Slide Time: 06:48)



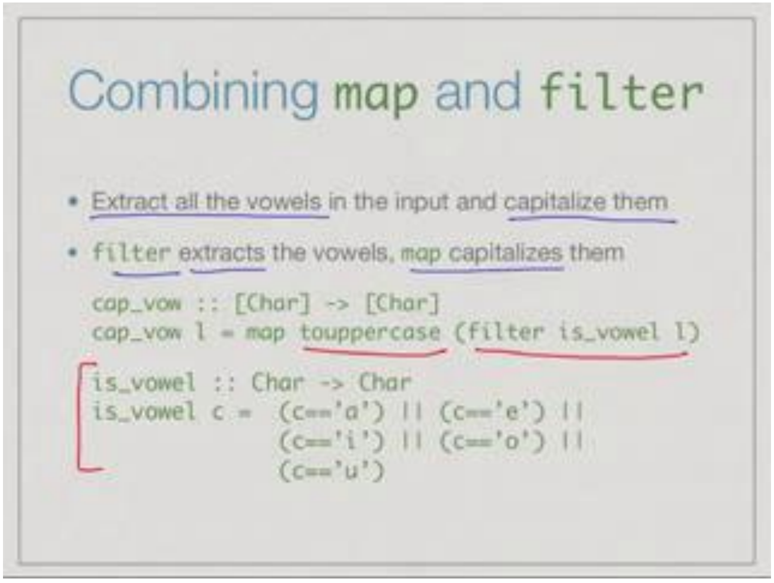
So, this is called filtering, so in filtering we take a property p, so property p is a function that takes a type and maps it to a Boolean value. So, it takes say integers and besides whether they are given or not, take a letters, characters and decide whether they are vowels or not for example. So, filter takes a property and a list and it remove extract exactly those items in the list that satisfy property p. So, filter takes p and if I note a filter an empty list with the property p I get nothing, because no elements are get satisfy the property.

If I have a non empty list, then I check the property on the first element, if it is true I include the first element and continue, otherwise exclude. So, in our previous case the property was is even, but this is the generalization of property. So, as we said a property is just something, which takes the type of the underline list and decides whether or not each element satisfies it is property.

So, it takes a function which maps the type a to Bool, it takes an input list and produces an output list, which is the subset or sub list of the input list that of the same type. It is not like map, which produces a new list, which is the result of transforming the elements to the function. So, in a map each element is transformed their function from a to b, so you get a list of these, in filter you are not transforming anything, you really selecting a sub list, so the output list and the input list have the same type.

Of course, it is possible that none of the elements in the list satisfied p, in which case the output list maybe empty, where it will be of the same type. So, if we go back to our function even only, then the property in this case is the function is even, if you wrote 4. So, it is even remember it takes integers and produces Boolean, just checks whether the remainder is 0 and divided by 2. So, if I filter a list given in this function, then I am extract exactly the even number in the list, so this is a much more subsequent way of writing the same function.

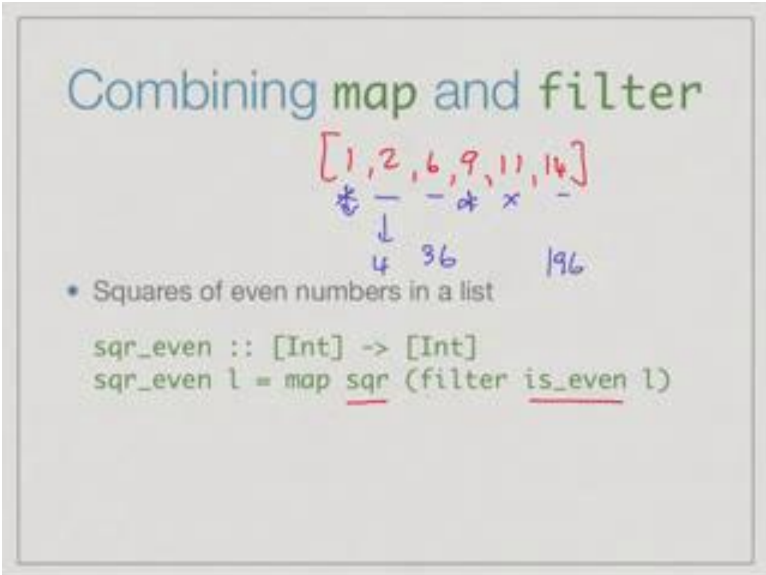
(Refer Slide Time: 08:59)



So, very often map and filter occur in conjunctions, so very often what we want to do is to take a given list, extract some elements from that list, which satisfy given property and then transform those elements to some new elements. So, here is an example supposing we want to extract all the vowels. So, this is the filter and then we want to capitalize these letters, so that is a map. So, filter extracts a vowels and map capitalize them, so we have first a filter function called this vowels similar is even, we just checks whether the character is in a e i o u.

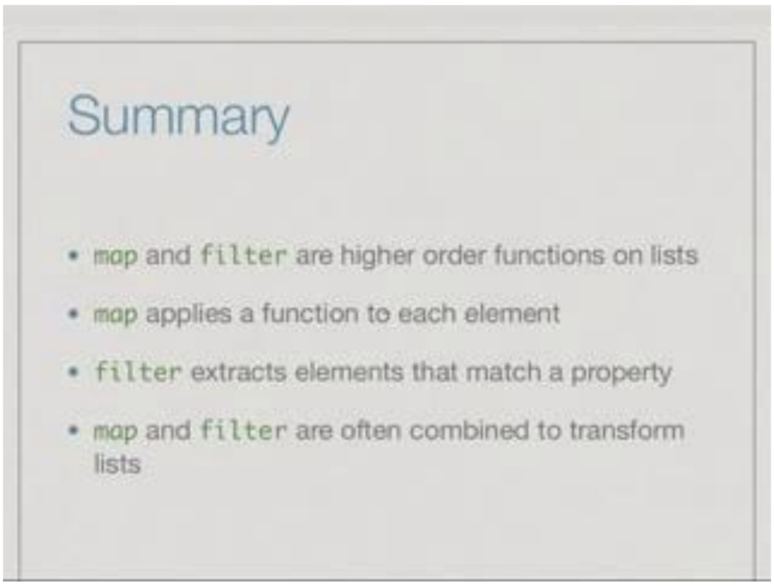
So, we first apply filter to the list with these vowels as the property, this will result in a list, which has exactly those characters, which are vowels in the regional list. And then, we use our earlier function to upper case which capitalizes every element in a string to apply to this list, which contains these vowels. So, filter followed by map is a very common form that we will find a many functions that we be used.

(Refer Slide Time: 10:06)



So here, is another simple example of filter and maps supposing we want to square or the even numbers. So, we first filter the list I get in the even numbers of, then we maps the square function, which is the list. So, this will square each individual element of that list and give us to squares of the even numbers, so if we have list 1 2 6 9 11 14, then first filter will give us that this and this and this are executed. So, I get 2 6 and 14 and then map will give me 436 and whatever this is 196.

(Refer Slide Time: 10:49)



So, to summarize map and filter are extremely useful higher order functions of this. So, maps take the function and a list and applies the function each element of the list. Whereas, filter takes a property that evaluates true or false, next track elements on the list that match the property, that is those elements for which the property is true and we often use these an combination, so to extract the sub list and then apply a function to that list.