Notes from first lecture

Variables

Variables assigned like this:

y :: Int y = 2

Functions

```
functionName :: Integer -> Integer
functionName 0 = 1
```

The first 'Integer' is the input (what it takes as input) and the second Integer is the return type.

If statements

```
functionName n
    | n < 10 = doSomething
    | otherwise = doSomethigElse
```

Pairs

You can make a pair of anything

```
p :: (Int, Char)
p = (3, 'x')

--Or Using it in functions

sumPair :: (Int, Int) -> Int
sumPair (x, y) = x + y
```

Lists

Lists are one of the most basic data types

```
nums :: [Integer]
nums = [1,2,3,4]

-- Or

range = [2,4..100]
```

Cons

The con symbol : is important. It takes an element and list and makes a new list e.g.

```
a = 1 : []
b = 3 : (1 : [])
c = 2 : 3 : 4 : []
```

Functions on lists

Lists can be utilised in functions, they can make operations or additions e.g.

```
intListLength :: [Integer] -> Integer
intListLength [] = 0
intListLength (x:xs) = 1 + intListLength xs
```

Important notes:

Sum Function

- The sum function `sum [list]` computes the sum of the numbers of a structure.
- The concatmap function `concatMap` maps a function over all the elements of a container and concatenate the resulting lists.