Chapter 1 - Introduction

→ Functional programming is a style built on method of computation. Math functions

Ex:

sum [1 .. 10] → Summing the integers from 1 to 10 in Haskell, very straight forward

```
f []     = []
f (x:xs) = f ys ++ [x] ++ f zs
           where
             ys = [a | a <- xs, a <= x]
             zs = [b | b <- xs, b > x]
```

[Quicksort]
Fastest

[ ] → Means sequence of things and same type (in square brackets)

Examples:

[1, 2, 3, 4] → List of ints

[ ] → Empty

[1] [2] [3] → Singleton list

++ → Operators to be performed on lists and appends/concatenation for lists

Ex:

[1, 2, 3] ++ [4, 5] = [1, 2, 3, 4, 5]

: → It's first input is a single value instead of list performs the same operation

Ex:

1: [2, 3, 4] = [1, 2, 3, 4]

$[a \mid a \leftarrow xs, a \leq x]$ → Same meaning (Processing one list to give another)

Similar in Set theory: $\{a \mid a \in S \wedge a \leq x\}$

Means: All values of A which are elements of S and a is less than or equal to x

$f[3, 1, 4, 2]$ → Non-empty list so skip the base case

3 → x

1, 4, 2 → xs

$ys = [1, 2]$

$zs = [4]$

$f\ ys \mathbin{+\!+} [x] \mathbin{+\!+} f\ zs$ → Apply function again

$f[1, 2] \mathbin{+\!+} [3] \mathbin{+\!+} f[4]$ → Always get the singleton list

Empty list ← $= f[\,] \mathbin{+\!+} [1] \mathbin{+\!+} f[2] \mathbin{+\!+} [3] \mathbin{+\!+} [4]$

$= [\,] \mathbin{+\!+} [1] \mathbin{+\!+} [2] \mathbin{+\!+} [3] \mathbin{+\!+} [4]$

$= [1, 2, 3, 4]$ → Run to completion

It sorted and this algorithm is Quick Sort in Haskell

C/C++, Java the algorithm will be longer