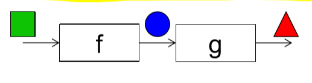
Haskell Basics

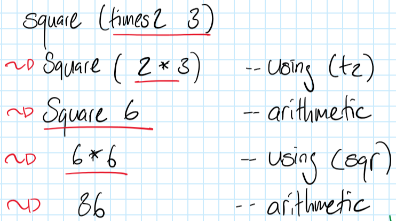
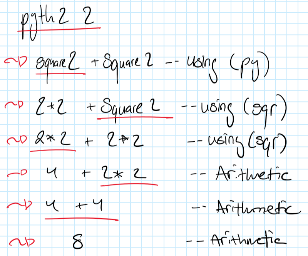
Function Application Rules: No side Effects allowed 🡪 The Ouput depends on inputs only !

Haskell Functions are pure 🡪 For every specific Input, a function always computes exactly the same output.

**Models of Computation:**

* ****Imperative : Step by Step instructions 🡪 Zeit spielt eine grosse Rolle, da man am Schluss nicht weiss, wann welche Variable geändert wurde
* ****Functional: Applying functions to arguments 🡪 Applying functions to arguments

Haskell ist eine lazy-funktionalle Sprache = Das Ergebnis des auszuwertenden Ausdrucks wird nur so weit berechnet wie gerade nötig. 🡪 Call-by-Need Strategie

Schritweise Evaluation von ausdrücken:

Types 🡪 Geben Bits eine Bedeutung

Values are represented by data = usually as bits 0 | 1 🡪 Without interpretation => useless

A Type is a set of related Values 🡪 Verträge/ Schnittstellen

Haskell has a strong Typesystem 🡪 Stricter typing rules at compile time

Basic Types:

**Bool:** The two logical values ‘True’ and ‘False’

**Char:** This Type contains all single characters that are available from a normal Keyboard ‘a’, ‘3’

**Int:** Fixe Grösse im Speicher. Uses a fix amount of memory, CPU supported, fast

**Integer:** Contains all integers, with as much memory as needed, slower computation

**Double:** Floating point numbers, Uses fix amount of memory

Defining your own Enumeration Types

**data** Color = Red | Yellow | Green **deriving** (Show)

**data** ToDo = Stop | Wait | Go **deriving** (Show)

Tuples

A Tuple is a finite sequence of components of possibly different type

(False,8,»Hallo») // tripple 🡪 fasst drei Werte zusammen

# TODO: Slide 02\_LE\_Types 15