

Haskell:

- Type System, Lazy Evaluation, Pure Functional

Y = 1 :: Int

- We can declare the type of symbol or expression

Type Signature:

- NameFunction :: Integer -> Integer

Functions of multiple arguments that can be applied to their arguments one at a time are called **curried functions**

- average :: Float-> Float-> Float
- average 3.0 4.0 is equivalent to (average 3.0) 4.0

Infix notation vs Prefix:

- 3.0 `average` 4.0 = average 3.0 4.0
- (+) 3 4 = 3+4

Lazy – definition of symbols are evaluated when needed

Parameterized types

- Here is an overview of some frequently used type classes, and some overloaded operations on these type classes.
- ❑ Typeclass **Show**
 - functions: `show :: Show a => a -> String`: convert the given value into a string.
 - member types: almost all predefined types, excluding function types.
- ❑ Typeclass **Eq**
 - functions: `(==), (/=) :: Eq a => a -> a -> Bool`: equality and inequality.
 - member types: almost all predefined types, excluding function types.
- ❑ Typeclass **Ord**
 - functions: `(<), (>), (<=), (>=) :: Ord a => a -> a -> Bool`: less than, greater than, less or equal, greater or equal
 - member types: almost all predefined types, excluding function types.
 - all types in **Ord** are already in **Eq**, so if you are using both `==` and `<` on a value, it is sufficient to require it to be in **Ord**.
- ❑ Typeclass **Num**
 - functions: `(+), (-), (*) :: Num a => a -> a -> a`: arithmetic operations.
 - member types: **Float**, **Double**, **Int**, **Integer**
- ❑ Typeclass **Integral**
 - functions: `div, mod :: Integral a => a -> a -> a`: division.
 - member types: **Int** (fixed precision), **Integer** (arbitrary precision)
- ❑ Typeclass **Fractional**
 - functions: `(/) :: Fractional a => a -> a -> a`: division.
 - member types: **Float**, **Double**

Tuples can have many types

`["red", "green", "blue"] : "yellow" ⇒ Error!`

`++` list append, `!!` get element at index, `head [list]` = get head